User Guide

AWS Deadline Cloud

Version latest

AWS Deadline Cloud: User Guide

Table of Contents

What is AWS Deadline Cloud?

Deadline Cloud is an AWS service you can use to create and manage compute-intensive workloads on Amazon Elastic Compute Cloud (Amazon EC2) instances. Deadline Cloud orchestrates workloads ranging from 3D rendering and visual effects to scientific simulations, financial modeling, machine learning model training, and data processing—any task that benefits from distributing work across a fleet of workers.

Deadline Cloud provides console interfaces, local applications, command line tools, and an API. With Deadline Cloud, you can create, manage, and monitor farms, fleets, jobs, user groups, and storage. You can also specify hardware capabilities, create environments for specific workloads, and integrate the applications and tools that your workflow requires into your Deadline Cloud pipeline.

Deadline Cloud provides a unified interface to manage all of your rendering projects in one place. You can manage users, assign projects to them, and grant permissions for job roles.

Topics

Features of Deadline Cloud

Here are some of the key ways Deadline Cloud can help you run and manage compute-intensive workloads:

- Quickly create your farms, queues, and fleets. Monitor their status, and gain insights into the operation of your farm and jobs.
- Centrally manage Deadline Cloud users and groups, and assign permissions.

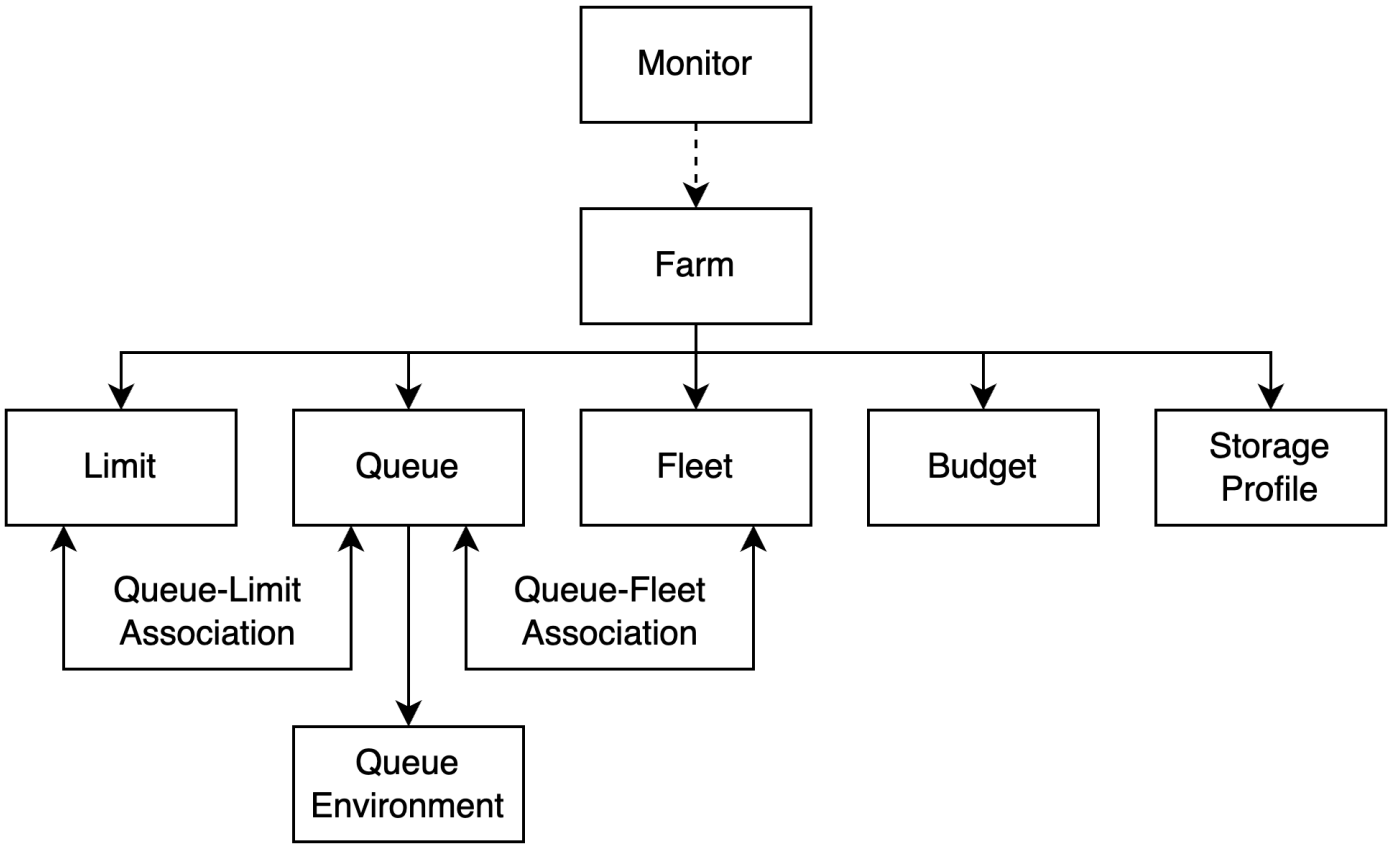
- Manage sign-in security for project users and external identity providers with AWS IAM Identity Center.
- Securely manage access to project resources with AWS Identity and Access Management (IAM) policies and roles.
- Use tags to organize and quickly find project resources.
- Manage project resource usage and estimated costs for your project.
- Provide flexible compute management options to support rendering in the cloud or in person.

Concepts and terminology for Deadline Cloud

To help you get started with AWS Deadline Cloud, this topic explains some of its key concepts and terminology.

Farm resources

This diagram shows how Deadline Cloud farm resources work together.



Farm

A farm contains all other resources related to submitting and running jobs. Farms are independent from each other making them useful for separating production environments.

Queue

A queue holds jobs for scheduling on associated fleets. Users can submit jobs to a queue and manage their priority and status inside the queue. A queue must be associated with a fleet with a queue-fleet association for its jobs to be run, and queues can be associated with multiple fleets.

Fleet

A fleet contains compute capacity for running jobs. Fleets can be service-managed or customer-managed. Service-managed fleets run in Deadline Cloud and include built-in functionality like autoscaling, licensing, and software access. Customer-managed fleets run on your own compute resources like Amazon EC2 instances or on-premises servers.

Budget

A budget sets spending thresholds for your job activity and allows you to take actions when thresholds are reached, such as stopping job scheduling.

Queue environment

A queue environment defines scripts that run on each worker to set up or tear down the workload environment. They are useful for setting environment variables, installing software, and configuring asset storage.

Storage profile

A storage profile is a configuration for a group of hosts and workstations, that tells where data is located on the file system. Deadline Cloud uses storage profiles to map paths when running jobs on differently configured hosts, such as a job submitted from Windows and running on Linux.

Limit

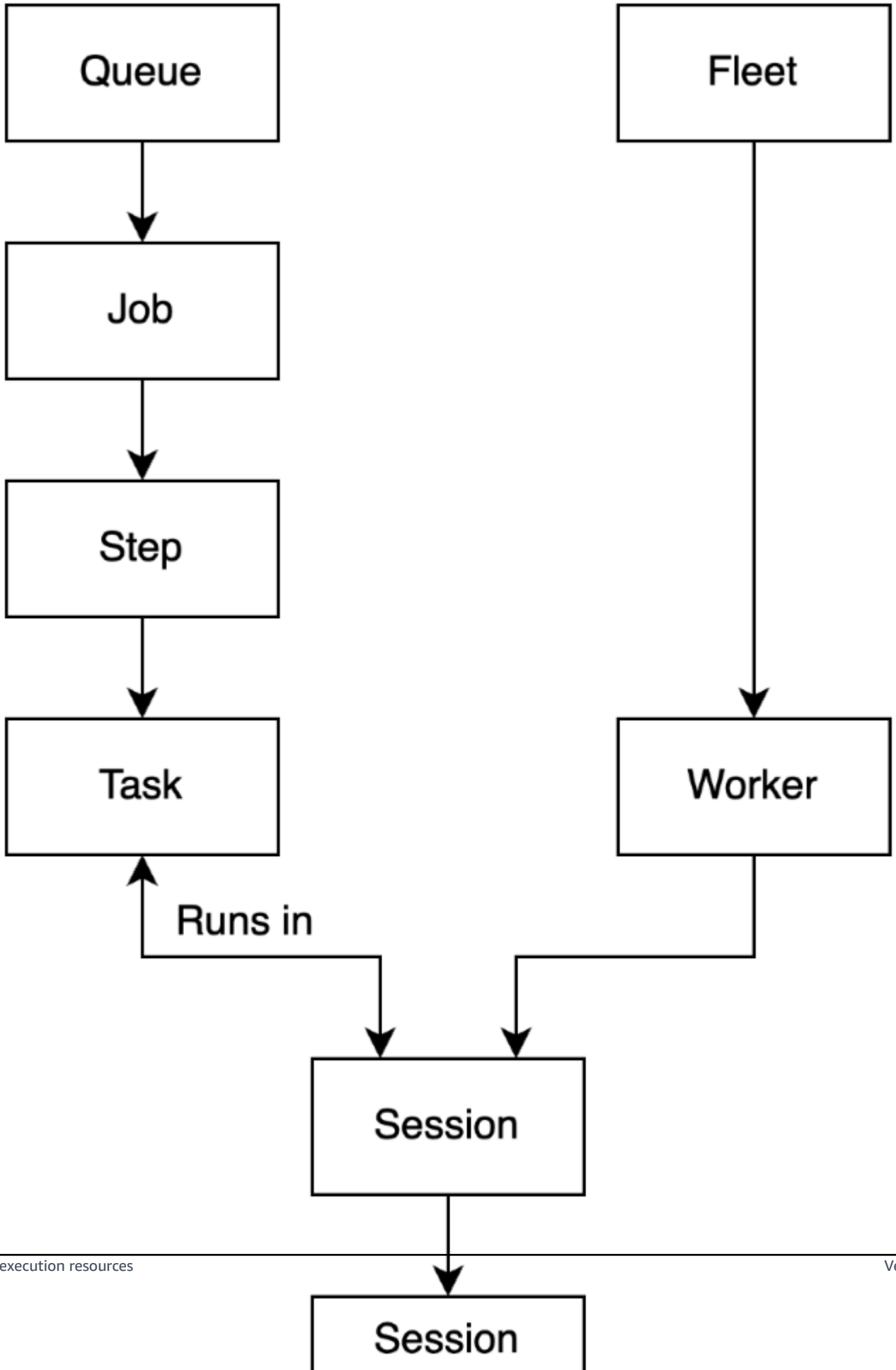
A limit allows you to track usage of shared resources such as floating licenses and control how they are allocated between jobs. Limits are associated with queues with queue-limit associations.

Monitor

The monitor configures the URL for the Deadline Cloud monitor web application, allowing end users to monitor and manage jobs. It can be accessed in a browser or through the Deadline Cloud monitor desktop application.

Job execution resources

This diagram shows how Deadline Cloud job resources work together.


Queue
Fleet
Job
Step
Task
Worker
Runs in
Session
Session

Job

A job is a set of work that a user submits to Deadline Cloud to be scheduled and run on available workers. A job may render a 3D scene or run a simulation. Jobs are created from reusable job templates, which define the runtime environment and processes, and job-specific parameters. Jobs contain steps and tasks that define the work to be performed, and they can be configured with priorities, maximum worker counts, and retry settings.

Job priority

Job priority is the approximate order that Deadline Cloud processes a job in a queue. You can set the job priority between 0 and 100, jobs with a higher number priority are generally processed first. Jobs with the same priority are processed in the order received.

Job properties

Job properties are settings that you define when submitting a render job. Some examples include frame range, output path, job attachments, renderable camera, and more. The properties vary based on the DCC that the render is submitted from.

Step

A step is part of a job that provides a template for running many tasks that are identical except for the task parameter values. Steps can have dependencies on other steps, allowing you to create complex workflows with sequential or parallel execution paths. In rendering jobs, a step often defines the command for rendering a frame and uses the frame number as the task parameter.

Task

A task is the smallest unit of work in Deadline Cloud. Tasks are part of steps and are executed by workers, representing individual operations that need to be performed as part of a job. Tasks can be configured with specific parameters and are assigned to workers based on their capabilities and availability. In rendering jobs, a task often renders a single frame.

Worker

Workers are part of a fleet and execute tasks from jobs. Workers can be configured with specific capabilities such as GPU accelerators, CPU architecture, and operating system. In service-managed fleets, workers are created automatically as the fleet scales out and in.

Instance

Fleets use instances for CPU resources. An instance is an Amazon EC2 performance instance. Deadline Cloud uses On-Demand and Spot instances.

On-Demand instance

On-Demand instances are priced by the second, have no long-term commitment, and will not be interrupted.

Spot instance

Spot instances are unreserved capacity that you can use at a discounted price, but may be interrupted by On-Demand requests.

Wait and Save

The Wait and Save feature provides delayed job scheduling for lower cost and can be interrupted by On-Demand and Spot requests. Wait and Save is only available within Deadline Cloud service-managed fleets.

Wait and Save is for managing the execution of visual computing workloads in AWS Deadline Cloud. See AWS service terms for details.

Session

A session represents a worker's sequence of work on a job. During a single session, a worker may be assigned multiple tasks which it runs one after another. Sessions often have setup actions which configure environments and load assets before running the task actions.

Session action

A session action represents specific operations performed during a session such as setting up the environment, running a task, and syncing assets.

Other important concepts and terminology

Usage explorer

Usage explorer is a feature of Deadline Cloud monitor. It provides an approximate estimate of your costs and usage.

Budget manager

Budget manager is part of the Deadline Cloud monitor. Use the budget manager to create and manage budgets. You can also use it to limit activities to stay within budget.

Deadline Cloud client library

The open-source client library includes a command line interface and library for managing Deadline Cloud. Functionality includes submitting job bundles based on the Open Job Description specification to Deadline Cloud, downloading job attachment outputs, and monitoring your farm using the command line interface (CLI).

Digital content creation application (DCC)

Digital content creation applications (DCCs) are third-party products where you create digital content. Deadline Cloud has built-in integrations with many DCCs such as Autodesk Maya, Blender, and Maxon Cinema 4D allowing you to submit jobs from within the DCC and render on service-managed fleets with pre-configured software and licensing.

Job attachments

Job attachments are a Deadline Cloud feature that you upload and download assets as part of a job such as textures, 3D models, and lighting rigs. Job attachments are stored in Amazon S3 and avoid the need for shared network storage.

Job template

A job template defines the runtime environment and all processes that run as part of a Deadline Cloud job.

Deadline Cloud submitter

A Deadline Cloud submitter is a plugin for a DCC that allows users to easily submit jobs from within the DCC.

License endpoint

A license endpoint makes Deadline Cloud's usage-based licensing for third-party products available inside your VPC. This model is pay as you go, and you are charged for the number of hours and minutes that you use. License endpoints are not connected to farms and can be used independently.

Tags

A tag is a label that you can assign to an AWS resource. Each tag consists of a key and an optional value that you define. With tags, you can categorize your AWS resources in different ways, such as by purpose, owner, or environment.

Usage-based licensing (UBL)

Usage-based licensing (UBL) is an on-demand licensing model that is available for select third-party products. This model is pay as your go, and you are charged for the number of hours and minutes that you use.

Getting started with Deadline Cloud

Use Deadline Cloud to quickly create a render farm with default settings and resources, such as Amazon EC2 instance configuration and Amazon Simple Storage Service (Amazon S3) buckets.

You can also define the settings and resources when you create a render farm. This method takes more time than using the default settings and resources but gives you more control.

After you're familiar with Deadline Cloud Concepts and terminology, see Getting started for step-by-step instructions for creating your farm, adding users, and links to helpful information.

For information about migrating from Deadline 10, including a concept mapping, see Migrate from Deadline 10 to AWS Deadline Cloud in the *Deadline Cloud Developer Guide*.

Accessing Deadline Cloud

You can access Deadline Cloud in any of the following ways:

- **Deadline Cloud console**– Access the console in a browser to create a farm and its resources, and manage user access. For more information, see Getting started.
- **Deadline Cloud monitor**– Manage your render jobs, including updating priorities and job statuses. Monitor your farm and view logs and job status. For users with Owner permissions, the Deadline Cloud monitor also provides access to explore usage and create budgets. The Deadline Cloud monitor is available as both a web browser and a desktop application.
- **AWS SDK and AWS CLI**– Use the AWS Command Line Interface (AWS CLI) to call the Deadline Cloud API operations from the command line on your local system. For more information, see Set up a developer workstation.

Deadline Cloud documentation

Deadline Cloud documentation is split across two guides:

- **This user guide** is for artists who submit and monitor jobs from their digital content creation (DCC) applications, and for studio administrators who set up farms, queues, users, and budgets using the console and the Deadline Cloud monitor.
- The Deadline Cloud Developer Guide is for pipeline developers who build job bundles, submitters, and integrations with the AWS Command Line Interface (AWS CLI) and API. It is also for infrastructure engineers who manage customer-managed fleets, images, networking, and licensing.

Each topic is documented once, in the guide for its audience. When a topic in one guide depends on a topic that the other guide owns, you will find a short summary and a link instead of a second copy.

Related services

Deadline Cloud works with the following AWS services:

- **Amazon CloudWatch**– With CloudWatch, you can monitor your projects and associated AWS resources. For more information, see Monitoring with CloudWatch in the *Deadline Cloud Developer Guide*.
- **Amazon EC2**–This AWS service provides virtual servers that run your applications in the cloud. You can configure your projects to use Amazon EC2 instances for your workloads. For more information, see Amazon EC2 instances.
- **Amazon EC2 Auto Scaling**– With Auto Scaling, you can automatically increase or decrease the number of Amazon EC2 instances in a customer-managed fleet. Deadline Cloud publishes fleet size recommendations that you can use to scale your fleet based on the work available in your queues. With service-managed fleets, you don't need to configure scaling. For more information, see Create fleet infrastructure with an Amazon EC2 Auto Scaling group in the *Deadline Cloud Developer Guide*.
- **AWS PrivateLink**– AWS PrivateLink provides private connectivity between virtual private clouds (VPCs), AWS services, and your on-premises networks, without exposing your traffic to the public internet. AWS PrivateLink makes it easy to connect services across different accounts and VPCs. For more information, see AWS PrivateLink.
- **Amazon S3**– Amazon S3 is an object storage service. Deadline Cloud uses Amazon S3 buckets to store job attachments. For more information, see the Amazon S3 User Guide.

- **IAM Identity Center**– IAM Identity Center is an AWS service where you can provide users with single sign-on access to all their assigned accounts and applications from one place. You can also centrally manage multi-account access and user permissions to all of your accounts in AWS Organizations. For more information, see AWS IAM Identity Center FAQs.

How Deadline Cloud works

With Deadline Cloud, you can create and manage rendering projects and jobs directly from digital content creation (DCC) pipelines and workstations.

You submit jobs to Deadline Cloud using the AWS SDK, AWS Command Line Interface (AWS CLI), or Deadline Cloud job submitters. Deadline Cloud supports the Open Job Description (OpenJD) for job template specification. For more information, see Open Job Description on the GitHub website.

Deadline Cloud provides job submitters. A *job submitter* is a DCC plugin for submitting render jobs from a third-party DCC interface, such as Maya or Nuke. With a submitter, artists can submit rendering jobs from a third-party interface to Deadline Cloud where project resources are managed and jobs are monitored, all in one location.

Most DCC integrations also include an adaptor. An *adaptor* is a program installed on the worker hosts that runs the DCC application for a job. The adaptor keeps the application and scene loaded between tasks so that consecutive tasks don't repeat application startup and scene loading, reports render progress and status to the job's logs, and remaps file paths in the scene to their locations on the worker. The submitter and the adaptor are the two parts of a DCC integration: the submitter runs on the artist's workstation, and the adaptor runs on the worker hosts. For the list of applications with submitters and adaptors, see Supported Software.

With a Deadline Cloud farm, you can create queues and fleets, manage users, and manage project resource usage and costs. A *farm* consists of queues and fleets. A *queue* is where submitted jobs are located and scheduled to be rendered. A *fleet* is a group of worker nodes that run tasks to complete jobs. A queue must be associated with a fleet so that the jobs can render. A single fleet can support multiple queues and a queue can be supported by multiple fleets.

Jobs consist of steps, and each step consists of specific tasks. With the Deadline Cloud monitor, you can access statuses, logs, and other troubleshooting metrics for jobs, steps, and tasks.

Permissions in Deadline Cloud

Deadline Cloud supports the following:

- Managing access to its API operations using AWS Identity and Access Management (IAM)
- Managing access of workforce users using an integration with AWS IAM Identity Center

Before anyone can work on a project, they must have access to that project and the associated farm. Deadline Cloud is integrated with IAM Identity Center to manage workforce authentication and authorization. Users can be added directly to IAM Identity Center, or permission can be connected to your existing identity provider (IdP) such as Okta or Active Directory. IT administrators can grant access permissions to users and groups at different levels. Each subsequent level includes the permissions for the previous levels. The following list describes the four access levels from the lowest level to the highest level:

- **Viewer**– Permission to see resources in the farms, queues, fleets, and jobs they have access to. A viewer can't submit or make changes to jobs.
- **Contributor**– Same as a viewer, but with permission to submit jobs to a queue or farm.
- **Manager**– Same as contributor, but with permission to edit jobs in queues they have access to, and grant permissions on resources that they have access to.
- **Owner**– Same as manager, but can view and create budgets and see usage.

> **Note**
>
> These permissions don't give users access to the AWS Management Console or permission to modify Deadline Cloud infrastructure.

Users must have access to a farm before they can access the associated queues and fleets. User access is assigned to queues and fleets separately within a farm.

You can add users as individuals or as part of a group. Adding groups to a farm, fleet, or queue can make it easier to manage access permissions for large groups of people. For example, if you have a team that is working on a specific project, you can add each of the team members to a group. Then, you can grant access permissions to the entire group for the corresponding farm, fleet, or queue.

Software support with Deadline Cloud

Deadline Cloud works with any software application that can be run from a command line interface and controlled by using parameter values. Deadline Cloud supports the OpenJD specification for

describing work as **jobs** with software script **steps** that are parameterized (such as across a frame range) into **tasks**. Assemble OpenJD job instructions into job bundles with Deadline Cloud tools and features to create, run, and license the steps from a third-party software application.

Jobs need licensing to render. Deadline Cloud offers usage-based-licensing (UBL) for a selection of software application licenses that is billed by the hour in minute increments based on usage. With Deadline Cloud, you can also use your own software licenses if you like. If a job can't access a license, it doesn't render and produces an error that displays in the task log in the Deadline Cloud monitor.

Integrate Deadline Cloud into your pipeline

You can integrate your existing rendering pipelines with AWS Deadline Cloud to streamline your workflow management and job submission processes.

While the example below uses a visual effects pipeline, the same integration pattern applies to any compute-intensive workflow—scientific simulations, financial modeling, machine learning training, or data processing. In each case, you configure how data reaches worker hosts, which applications run on those hosts, how operators submit jobs, and how you monitor progress and control costs.

What is pipeline integration?

A pipeline integration of Deadline Cloud refers to how a Deadline Cloud farm provides batch processing for your interactive and automated workflows. This example uses a visual effects pipeline that you can adapt to the applications and processes your operators use in their workflows.

A visual effects pipeline consists of the stages of post-production to process input footage, 3D models, animation, textures, lighting, rendered images, and more. It prescribes how different departments exchange assets to perform the tasks they are responsible for. A well-designed pipeline facilitates efficient creation of final images for a television show or similar.

By integrating a Deadline Cloud farm into your pipeline, you can offload long-running jobs onto a queue, and prioritize how Deadline Cloud schedules them on fleets of worker hosts. You can use fleets managed by the service, and you can create your own fleets on-premises or on AWS.

For creating your pipeline integration, consider the following factors:

- Where is your asset data stored, and how will you provide them to worker hosts in the farm?

- Which applications and plugins do your jobs need, and how will you provision them onto worker hosts in the farm?
- When artists or other operators have jobs to run, how will they submit them to the farm?
- Who will monitor the progress and status of jobs, and how will you control costs and optimize the utilization of worker hosts?

Example of an on-premises studio with a farm on AWS

This example focuses on a pipeline where artists work together on-premises and submit jobs to a farm on AWS for rendering. The approach presented here is quick to onboard onto Deadline Cloud and provides a flexible starting point for customization.

Here are the factors for pipeline integration of this example studio:

- Asset data is stored on a NAS shared file system in their on-premises office.
  - On Windows, projects are mounted to the P: drive and utilities are mounted to X:.
  - On macOS, projects are mounted to /Volumes/Projects and utilities are mounted to /Volumes/Utilities.
- They use Maya for 3D modeling, Arnold for rendering, and Nuke for compositing. No custom plugins are installed in these applications.
- They want to use the default submission experience.
- Artists will monitor their own jobs and producers will monitor costs and adjust priorities when needed.

The pipeline integration for this studio uses job attachments to transfer data from the studio premises to and from AWS, as it can be easy to get started with and can scale to large fleet sizes. The job attachments S3 bucket configured on the queue acts as a cache tier between the on-premises NAS and worker hosts on AWS.

When artists submit jobs from Maya or Nuke, the Deadline Cloud integrated submitter scans the scene to identify the files needed for the job to run, and then attaches them to the job by uploading them to S3. A high performance hash is used to identify files that were previously uploaded by any artist in the studio. This way, when an artist is iteratively submitting new versions of the same shot, or one artist hands a shot off to another, only new or modified files need to be uploaded in the process of submitting the job.

The studio uses both Windows and macOS workstations, so they configure storage profiles with file system locations of local type for both their projects and utilities drive. See the Storage profiles for job attachments topic for more details about how this supports the path mapping necessary when jobs run on a different operating system than they are submitted from. They also configure a Linux host on their network to automatically download the output of all tasks of jobs in the queue when they complete. To learn how to set it up, see Automatic downloads for job attachments.

The farm contains two Linux service-managed fleets with vCPUs and RAM requirements set to ranges starting from a minimum specification the studio needs for their jobs. One of the fleets is configured to provide a small number of spot instances to provide consistent render capacity during work hours, and the other fleet is configured as wait and save to render more jobs during off-peak hours at a lower cost. All of Maya, the Maya for Arnold plugin, and Nuke are provided for Linux service-managed fleets from the deadline-cloud conda channel, alongside usage-based licensing. In order to save overhead from application installation, they replace the default conda environment configured for the queue in the Deadline Cloud console with the conda queue environment with improved caching on the GitHub website.

To support job submission, they set up Deadline Cloud submitters on each workstation, selecting the Maya and Nuke integrations. With Deadline Cloud monitor, they can log into the farm, monitor progress of jobs, and view log outputs for diagnosing issues. Both the Maya and Nuke submitters feature integrated dialogs to submit jobs from within the application interface.

When configuring user access levels in the farm (see How permissions work in Deadline Cloud), they give Contributor access to artists so they can submit jobs, view all jobs, and modify properties of their own jobs. They give Manager access to render wranglers so they can modify properties of all the jobs. They give Owner access to producers, so they can track spending and usage by creating budgets and exploring usage costs.

Getting started with Deadline Cloud

To create a farm in AWS Deadline Cloud, you can use either the Deadline Cloud console or the AWS Command Line Interface (AWS CLI). Use the console for a guided experience creating the farm, including queues and fleets. Use the AWS CLI to work directly with the service, or for developing your own tools that work with Deadline Cloud.

To create a farm and use the Deadline Cloud monitor, set up your account for Deadline Cloud. You only need to set up the Deadline Cloud monitor infrastructure once per account. From your farm, you can manage your project, including user access to your farm and its resources.

To create a farm with minimal resources to accept jobs, select **Quickstart** in the console home page. **Set up the Deadline Cloud monitor** walks you through those steps. These farms start with a queue and a fleet that are automatically associated. This approach is a convenient way to create sandbox style farms to experiment in.

Topics

Set up your AWS account

Sign up for an AWS account

To get started with AWS, you need an AWS account. For information about creating an AWS account, see Getting started with an AWS account in the *AWS Account Management Reference Guide*.

Set up the Deadline Cloud monitor

To get started, you'll need to create your Deadline Cloud farm infrastructure, including a monitor, queue, and fleet. You can also perform additional, optional steps including adding groups and users, choosing a service role, and adding tags to your resources.

Step 1: Create your monitor

The Deadline Cloud monitor uses AWS IAM Identity Center to authorize users. By default, the IAM Identity Center instance that you use for Deadline Cloud must be in the same AWS Region as the monitor. However, if you have Multi-Region support enabled in IAM Identity Center, you can create a monitor in a different Region. For more information, see What is AWS IAM Identity Center. If your console is using a different Region when you create the monitor, you'll get a reminder to change to the IAM Identity Center Region.

Your monitor's infrastructure consists of the following components:

- **Monitor name**: The **Monitor name** is how you can identify your monitor — for example *AnyCompany monitor*. Your monitor's name also determines your **monitor URL**.
- **Monitor URL**: You can access your monitor by using the **Monitor URL**. The URL is based on the **Monitor name** — for example *https://anycompanymonitor.awsapps.com*.
- **AWS Region**: The **AWS Region** is the physical location for a collection of AWS data centers. When you set up your monitor, the Region defaults to the closest location to you. We recommend changing the Region so it is located closest to your users. This placement reduces lag and improves data transfer speeds. By default, AWS IAM Identity Center must be enabled in the same AWS Region as Deadline Cloud, unless you have Multi-Region support enabled in IAM Identity Center. For more information, see What is AWS IAM Identity Center.

> **Important**
>
> You can't change your Region after you finish setting up Deadline Cloud.

Complete the tasks in this section to configure your monitor's infrastructure.

To configure your monitor's infrastructure

1. Sign in to the **AWS Management Console** to start the Welcome to Deadline Cloud setup, then choose **Next**.
2. Enter the **Monitor name** — for example **`AnyCompany Monitor`**.
3. (Optional) To change the **Monitor URL**, choose **Edit URL**.
4. (Optional) To change the **AWS Region** so it's closest to your users, choose **Change Region**.
   a. Select the Region closest to your users.

b. Choose **Apply Region**.

5. (Optional) To further customize your monitor setup, select **Additional settings**.
6. If you are ready for Step 2: Define farm details, choose **Next**.

Additional settings

Deadline Cloud setup includes additional settings. With these settings, you can view all the changes Deadline Cloud setup makes to your AWS account, configure your monitor user role, and change your encryption key type.

AWS IAM Identity Center

AWS IAM Identity Center is a cloud-based single sign-on service for managing users and groups. IAM Identity Center can also be integrated with your enterprise single sign-on (SSO) provider so that users can sign in with their company account.

Deadline Cloud enables IAM Identity Center by default, and it is required to set up and use Deadline Cloud. By default, the IAM Identity Center instance that you use for Deadline Cloud must be in the same AWS Region as the monitor. However, if you have Multi-Region support enabled in IAM Identity Center, you can create a monitor in a different Region. For more information, see What is AWS IAM Identity Center.

Configure service access role

An AWS service can assume a service role to perform actions on your behalf. Deadline Cloud requires a monitor user role for it to give users access to resources in your monitor.

You can attach AWS Identity and Access Management (IAM) managed policies to the monitor user role. The policies give users permissions to perform certain actions, such as creating jobs in a specific Deadline Cloud application. Because applications depend on specific conditions in the managed policy, if you don't use the managed policies, the application might not perform as expected.

You can change the monitor user role after you complete setup, at any time. For more information about user roles, see IAM Roles.

The following tabs contain instructions for two different use cases. To create and use a new service role, choose the **New service role** tab. To use an existing service role, choose the **Existing service role** tab.

New service role

To create and use a new service role

1. Select **Create and use a new service role**.
2. (Optional) Enter a **Service user role** name.
3. Choose **View permission details** for more information about the role.

Existing service role

To use an existing service role

1. Select **Use an existing service role**.
2. Open the dropdown list to choose an existing service role.
3. (Optional) Choose **View in IAM console** for more information about the role.

Step 2: Define farm details

Back on the Deadline Cloud console, complete the following steps to define the farm details.

1. In **Farm details**, add a **Name** for the farm.
2. For **Description**, enter the farm description. A description can help you identify your farm's purpose.
3. Create a group and add uses for your farm. After you set up your farm, you can use the Deadline Cloud management console to add or change groups and users.
4. (Optional) Choose **Additional farm settings**.

    a. (Optional) By default, your data is encrypted with a key that AWS owns and manages for your security. You can choose **Customize encryption settings (advanced)** to use an existing key or to create a new one that you manage.

    If you choose to customize encryption settings using the checkbox, enter a AWS KMS ARN, or create a new AWS KMS by choosing **Create new KMS key**.

    b. (Optional) Choose **Add new tag** to add one or more tags to your farm.
5. Choose one of the following options:

    - Select **Skip to Review and Create** to review and create your farm.

- Select **Next** to proceed to additional, optional steps.

(Optional) Step 3: Define queue details

The queue is responsible for tracking progress and scheduling work for your jobs.

1. Starting in **Queue details,** provide a **Name** for the queue.
2. For **Description**, enter the queue description. A clear description can help you quickly identify your queue's purpose.
3. For **Job attachments**, you can either create a new Amazon S3 bucket or choose an existing Amazon S3 bucket. If you don't have an existing Amazon S3 bucket, you'll need to create one.

    a. To create a new Amazon S3 bucket, select **Create new job bucket**. You can define the name of the job bucket in the **Root prefix** field. We recommend calling the bucket **`deadlinecloud-job-attachments-[QUEUENAME]`**.

    You can only use lowercase letters and dashes. No spaces or special characters.

    b. To search for and select an existing Amazon S3 bucket, select **Choose from existing Amazon S3 bucket**. Then, search for an existing bucket by choosing **Browse S3**. When the list of your available Amazon S3 buckets display, select the Amazon S3 bucket you want to use for your queue.
4. (Optional) Choose **Additional farm settings**.

    a. If you are using customer-managed fleets, select **Enable association with customer-managed fleets**.

        i. For customer-managed fleets, add a **Queue-configured user**, and then set the POSIX and/or Windows credentials. Alternatively, you can bypass the run-as functionality by selecting the checkbox.

        ii. If you want to set a budget for a queue, choose **Require a budget for this queue**. If you require a budget, you must create the budget using the Deadline Cloud console to schedule jobs in the queue.

    b. Your queue requires permission to access Amazon S3 on your behalf. We recommend you create a new service role for every queue.

        i. For a new role, complete the following steps.

A. Select **Create and use a new service role**.

B. Enter a **Role name** for your queue role or use the provided role name.

C. (Optional) Add a queue role **Description**.

D. You can view the IAM permissions for the queue role by choosing **View permission details**.

ii. Alternatively, you can select an existing service role.

c. (Optional) Add environment variables for the queue environment using name and value pairs.

d. (Optional) Add tags for the queue using key and value pairs.

Choose one of the following options:

- Select **Skip to Review and Create** to review and create your farm.
- Select **Next** to proceed to additional, optional steps.

(Optional) Step 4: Define fleet details

A fleet allocates workers to execute your rendering tasks. If you need a fleet for your rendering tasks, check the box for **Create fleet**.

1. **Fleet details**

   a. Provide both a **Name** and optional **Description** for your fleet.

   b. Review the fleet type and operating system for awareness.

2. In the **Instance market type** section, choose either **Spot**, **On-demand**, or **Wait and Save Instance**. Amazon EC2 On-demand instances provide faster availability and Amazon EC2 Spot and Wait and Save instances are better for cost saving efforts.
3. For **Auto scaling** the number of instances in your fleet, choose both a **Minimum** number of instances and a **Maximum** number of instances.

   We strongly recommend to always set the minimum number of instances to **0** to avoid incurring extra costs.
4. Review the worker capabilities for awareness.
5. (optional) Choose **Additional fleet settings**

a. Your fleet requires permission to write to CloudWatch on your behalf. We recommend you create a new service role for every fleet.

   i. For a new role, complete the following steps.

      A. Select **Create and use a new service role**.

      B. Enter a **Role name** for your fleet role or use the provided role name.

      C. (Optional) Add a fleet role **Description**.

      D. To view the IAM permissions for the fleet role, choose **View permission details**.

   ii. Alternatively, you can use an existing service role.

b. (Optional) Add tags for the fleet using key and value pairs.

After you enter all the fleet details, choose **Next**.

Step 7: Review and create

Review the information entered to create your farm. When you're ready, choose **Create farm**.

The progress of your farm's creation is displayed on the **Farms** page. A success message displays when your farm is ready for use.

After your farm is ready, give your team access to it. For the common steps to onboard artists, teams, or vendors, see Onboard users to your farm.

Set up your workstation

Set up a workstation so that you can submit jobs to AWS Deadline Cloud from your digital content creation (DCC) application. A Deadline Cloud *submitter* is a DCC plugin. You use it to submit jobs from a third-party DCC interface that you're familiar with. You can follow these steps yourself, or an administrator can install the software in advance. You sign in with your own user account.

Before you begin, you need the following:

- The monitor URL for your farm, which looks like `https://MY-MONITOR.REGION.deadlinecloud.amazonaws.com/`. Your administrator shares it with you. If you're the administrator, see Share the Deadline Cloud monitor URL.

- A user that can sign in to the monitor. If your organization uses single sign-on, sign in with your existing company account. If your administrator creates a user for you in AWS IAM Identity Center, accept the emailed invitation first. If you're the administrator, see Managing users in Deadline Cloud.
- The DCC installed on the workstation. For example, if you want to download the Deadline Cloud submitter for Blender, you need to have Blender already installed on your workstation.

Complete this process on every workstation that you use to submit renders.

Topics

Step 1: Install the Deadline Cloud submitter

Download the submitter installer for your operating system:

Download for Windows	Download for Linux	Download for MacOS (arm64)

With the installer, you can install the following submitters:

Software	Supported versions	Windows installer	Linux installer	MacOS (arm64) installer
Adobe After Effects	2024 - 2026	Included	Not included	Included
Autodesk 3ds Max	2024 - 2027	Included	Not included	Not included
Autodesk Arnold for Cinema 4D	4.8.4.1	Included	Not included	Included

Software	Supported versions	Windows installer	Linux installer	MacOS (arm64) installer
Autodesk Arnold for Maya	7.1 - 7.5	Included	Included	Included
Autodesk Maya	2023 - 2027	Included	Included	Included
Autodesk VRED	2025 - 2026	Included	Not included	Not included
Blender	3.6 - 5.1	Included	Included	Included
Chaos V-Ray for Maya	6 - 7	Included	Included	Included
Foundry Nuke	15 - 17	Included	Included	Included
KeyShot Studio	2023 - 2025	Included	Not included	Included
Maxon Cinema 4D	2024 - 2026	Included	Not included	Included
Maxon Redshift for Maya	2025-2026	Included	Included	Included
SideFX Houdini	19.5 - 22.0	Included	Included	Included

The standard installer doesn't include the Unreal Engine submitter, which has a separate setup process. For installation instructions, see the Unreal Engine Submitter Setup Guide on the GitHub website.

Windows

1. In a file browser, navigate to the folder where the installer downloaded, and then select `DeadlineCloudSubmitter-windows-x64-installer.exe`.
   a. If a **Windows protected your PC** pop-up displays, choose **More info**.
   b. Choose **Run anyway**.
2. After the AWS Deadline Cloud Submitter Setup Wizard opens, choose **Next**.

3. Choose the installation scope by completing one of the following steps:
   - To install for only the current user, choose **User**.
   - To install for all users, choose **System**.

     If you choose **System**, you must exit the installer and re-run it as an administrator by completing the following steps:

     a. Right-click on **`DeadlineCloudSubmitter-windows-x64-installer.exe`**, and then choose **Run as administrator**.

     b. Enter your administrator credentials, and then choose **Yes**.

     c. Choose **System** for the installation scope.
4. After selecting the installation scope, choose **Next**.
5. Choose **Next** again to accept the installation directory.
6. Select **Integrated submitter for Nuke**, or whichever submitter you want to install.
7. Choose **Next**.
8. Review the installation, and choose **Next**.
9. Choose **Next** again, and then choose **Finish**.

Linux

> **Note**
>
> The Deadline Cloud integrated Nuke installer for Linux and Deadline Cloud monitor can only be installed on Linux distributions with at least GLIBC 2.31.

1. Open a terminal window.
2. To do a system install of the installer, enter the command **`sudo -i`** and press **Enter** to become root.
3. Navigate to the location where you downloaded the installer.

   For example, **`cd /home/`*`USER`*`/Downloads`**.
4. To make the installer executable, enter **`chmod +x DeadlineCloudSubmitter-linux-x64-installer.run`**.

5. To run the Deadline Cloud submitter installer, enter **`./DeadlineCloudSubmitter-linux-x64-installer.run`**.
6. When the installer opens, follow the prompts on your screen to complete the Setup Wizard.

macOS (arm64)

1. In a file browser, navigate to the folder where the installer downloaded, and then select the file.
2. After the AWS Deadline Cloud Submitter Setup Wizard opens, choose **Next**.
3. Choose **Next** again to accept the installation directory.
4. Select **Integrated submitter for Maya**, or whichever submitter you want to install.
5. Choose **Next**.
6. Review the installation, and choose **Next**.
7. Choose **Next** again, and then choose **Finish**.

Step 2: Install and set up Deadline Cloud monitor

You can install the Deadline Cloud monitor desktop application with Windows, Linux, or macOS.

Windows

1. Download the Deadline Cloud monitor installer for Windows:

   Download Deadline Cloud monitor for Windows
2. Run the downloaded installer and follow the prompts to complete the installation.

   To perform a silent install, use the following command:

```
DeadlineCloudMonitor_x64-setup.exe /S
```

   By default the monitor is installed in `C:\Users{username}\AppData\Local\DeadlineCloudMonitor`. To change the installation directory, use this command instead:

```
DeadlineCloudMonitor_x64-setup.exe /S /D={InstallDirectory}
```

Linux (AppImage)

To install Deadline Cloud monitor AppImage on RPM or Debian distros

> **Note**
>
> Deadline Cloud monitor requires GLIBC 2.34 or later. On Ubuntu 22 machines, install the Debian package instead of the AppImage. For instructions, see the **Linux (Debian)** tab.

1. Download the Deadline Cloud monitor AppImage:

   Download Deadline Cloud monitor (AppImage)

2. On RHEL-family systems such as Red Hat Linux, Rocky Linux, and Alma Linux, create a symbolic link to the CA bundle. RHEL-family systems keep the CA bundle at `/etc/ssl/certs/ca-bundle.crt`, but Deadline Cloud monitor expects `/etc/ssl/certs/ca-certificates.crt` and blocks profile creation with the error `Couldn't load TLS file database` until that file exists. To create the link, enter:

   ```
   sudo ln -sf /etc/ssl/certs/ca-bundle.crt /etc/ssl/certs/ca-certificates.crt
   ```

3. To make the AppImage executable, enter:

   ```
   chmod a+x deadline-cloud-monitor_amd64.AppImage
   ```

Linux (Debian)

To install Deadline Cloud monitor Debian package on Debian distros

> **Note**
>
> Deadline Cloud monitor requires GLIBC 2.34 or later.

1. Download the Deadline Cloud monitor Debian package:

   Download Deadline Cloud monitor (.deb)

2. To install the Deadline Cloud monitor Debian package, enter:

```
sudo apt update
sudo apt install ./deadline-cloud-monitor_amd64.deb
```

3. If the install fails on packages that have unmet dependencies, fix the broken packages and then run the following commands.

```
sudo apt --fix-missing update
sudo apt update
sudo apt install -f
```

Linux (RPM)

To install Deadline Cloud monitor RPM on Red Hat Linux 10, Rocky Linux 10, or later

> **Note**
>
> Deadline Cloud monitor requires GLIBC 2.34 or later and `webkit2gtk-4.1`, which isn't available on Red Hat Linux 9 or Rocky Linux 9. On those systems, install the AppImage instead. For instructions, see the **Linux (AppImage)** tab.

1. Download the Deadline Cloud monitor RPM:

   Download Deadline Cloud monitor (.rpm)

2. Install the Deadline Cloud monitor:

```
sudo dnf install deadline-cloud-monitor.x86_64.rpm
```

macOS (arm64)

1. Download the Deadline Cloud monitor installer for macOS:

   Download Deadline Cloud monitor for macOS (arm64)

2. Open the downloaded file. When the window displays, select and drag the Deadline Cloud monitor icon into the **Applications** folder.

After you complete the download, you can verify the authenticity of the downloaded software. You might want to do this to ensure no one has tampered with the files during or after the download process. See Verify the authenticity of downloaded software.

After downloading Deadline Cloud monitor, use the following procedure to set up the Deadline Cloud monitor.

To set up Deadline Cloud monitor

1. Open **Deadline Cloud monitor**.
2. When prompted to create a new profile, complete the following steps.
   a. Enter your monitor URL into the URL input, which looks like **`https://MY-MONITOR.REGION.deadlinecloud.amazonaws.com/`**
   b. Enter a **Profile** name.
   c. Choose **Create Profile**.

      Your profile is created and your credentials are now shared with any software that uses the profile name that you created.
3. After you create the Deadline Cloud monitor profile, you can't change the profile name or the studio URL. If you need to make changes, do the following instead:
   a. Delete the profile. In the left navigation pane, choose **Deadline Cloud monitor** > **Settings** > **Delete**.
   b. Create a new profile with the changes that you want.
4. From the left navigation pane, use the **>Deadline Cloud monitor** option to do the following:
   - Change the Deadline Cloud monitor profile to log in to a different monitor.
   - Enable **Autologin** so you don't have to enter your monitor URL on subsequent opens of Deadline Cloud monitor.
5. Close the Deadline Cloud monitor window. It continues to run in the background and enable other Deadline Cloud tools to access your render farm.
6. For each digital content creation (DCC) application that you plan to use for your rendering projects, complete the following steps:
   a. From your Deadline Cloud submitter, open the Deadline Cloud workstation configuration.

b. In the workstation configuration, select the profile that you created in the Deadline Cloud monitor. Your Deadline Cloud credentials are now shared with this DCC and your tools should work as expected.

Step 3: Launch the Deadline Cloud submitter

The steps to load and launch the submitter are unique to each DCC. For the instructions for your DCC, see Supported Software.

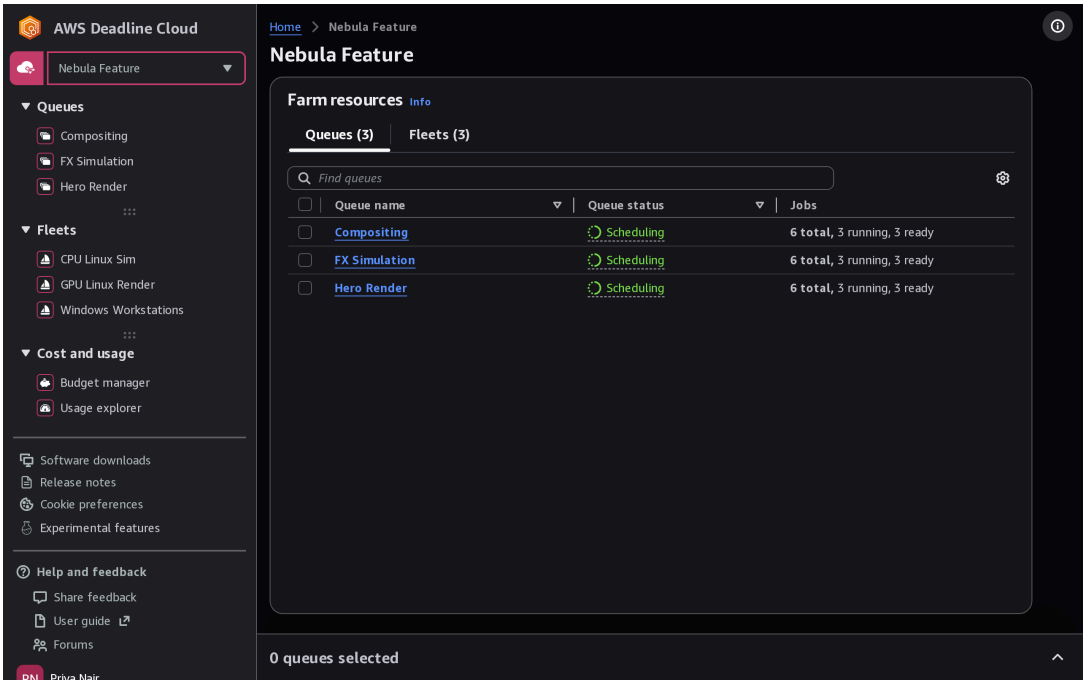
If you want a free DCC to test your setup with, Blender is a good choice. See the Blender Installation instructions.

After you submit a job from your DCC, your Deadline Cloud farm receives it and a compatible fleet processes it. To verify your setup, open the monitor and confirm that your job appears and completes. For information on how to view job progress in the monitor, see Using the Monitor.

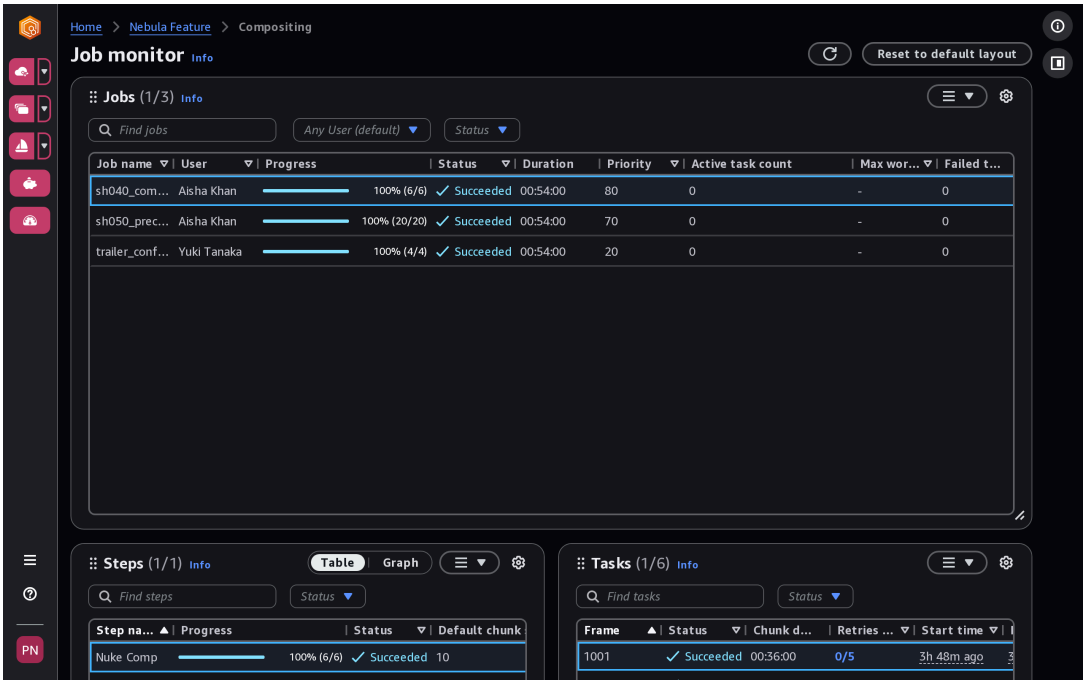
Using the Deadline Cloud monitor

The AWS Deadline Cloud monitor provides you with an overall view of your visual compute jobs. You can use it to monitor and manage jobs, view worker activity on fleets, track budgets and usage, and to download a job's results.

When you open a farm, the monitor lists its queues and fleets. The navigation pane provides access to each queue's job monitor and to the cost and usage tools.



Each queue has a job monitor that shows you the status of jobs, steps, and tasks. The monitor provides ways to manage jobs directly from the monitor. You can make prioritization changes, cancel jobs, requeue jobs, and resubmit jobs.



The Deadline Cloud monitor has a table that shows summary status for a job, or you can select a job to see detailed task logs that help troubleshoot issues with a job.

You can use the Deadline Cloud monitor to download the results to the location on your workstation that was specified when the job was created.

The Deadline Cloud monitor also helps you monitor usage and manage costs. For more information, see Track spending and usage for Deadline Cloud farms.

Topics

- Monitors and farms in multiple Regions
- Share the Deadline Cloud monitor URL
- Open the Deadline Cloud monitor
- Submit a job bundle
- View queue and fleet details in Deadline Cloud
- Manage jobs, steps, and tasks in Deadline Cloud
- View and manage job details in Deadline Cloud
- Customize the panels in the job monitor
- View a step in Deadline Cloud
- View a task in Deadline Cloud

- View session and worker logs in Deadline Cloud
- View worker details in the worker dashboard
- Download finished output in Deadline Cloud
- View output download status in Deadline Cloud
- Browsing job attachments in Deadline Cloud
- Automate Deadline Cloud monitor desktop deployment and workflows
- Manage cookie preferences in the Deadline Cloud monitor

Monitors and farms in multiple Regions

The Deadline Cloud monitor displays farms from all AWS Regions where AWS Deadline Cloud (Deadline Cloud) is available. When you open the monitor, you see farms in the same Region as the monitor and farms in other Regions that do not use a customer managed key. In most cases, a single monitor is all you need, even if you have farms in multiple Regions.

Managing users across Regions

You can add users and groups to farms in any Region from the Deadline Cloud console. The console handles cross-Region AWS IAM Identity Center (IAM Identity Center) membership assignment. Your IAM Identity Center instance doesn't need to be in the same Region as the farm: a membership references the IAM Identity Center user or group by its identifier, so you can assign the same users and groups to farms in any Region. If you assign memberships with the API or the AWS CLI, for example with `AssociateMemberToQueue`, send the request to the Region that contains the farm. For more information about managing farm membership, see Managing users in Deadline Cloud.

When to create additional monitors

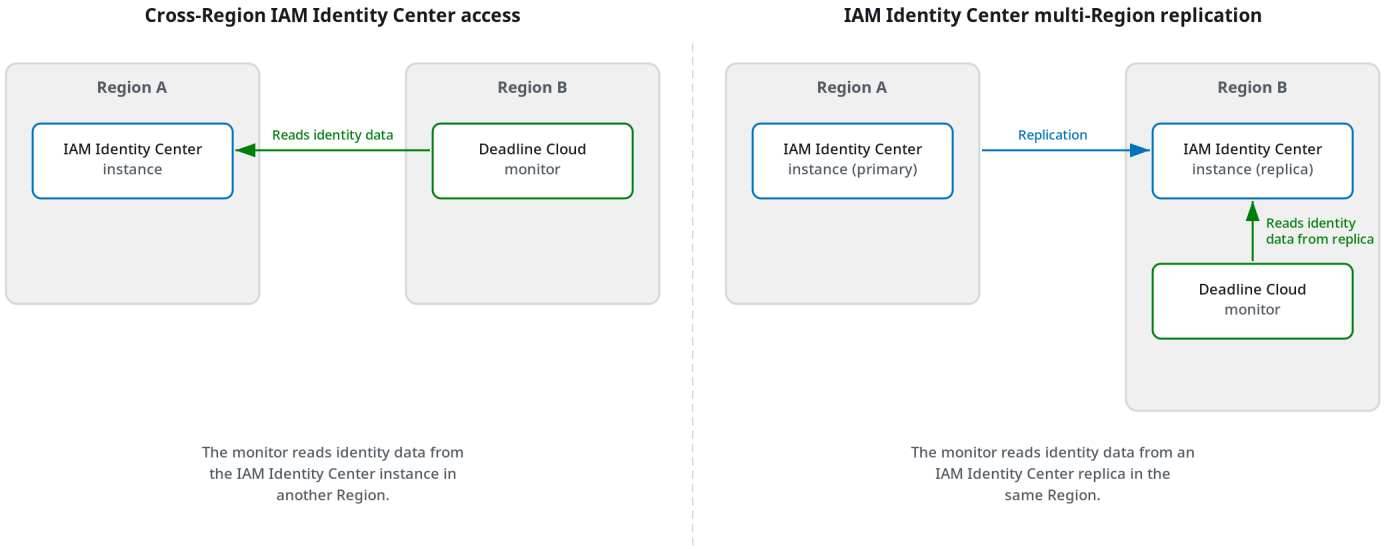
Because a single monitor shows farms in all Regions, you typically need only one monitor. You might want additional monitors in the following situations:

- **Customer managed keys in multiple Regions** – A monitor cannot display farms in other Regions that use a customer managed key. If you have farms with customer managed keys in multiple Regions, you need a monitor in each of those Regions.
- **Regional availability** – You want all Deadline Cloud resources, including the monitor, IAM Identity Center instance, and farms, to remain within a single Region. This configuration ensures that an outage in one Region does not affect your ability to manage resources in another Region.

- **Multiple IAM Identity Center instances** – Each monitor connects to one IAM Identity Center instance. If your organization uses separate IAM Identity Center instances for different teams or business units, you need a monitor for each instance.

The following diagram and table compare the two approaches for connecting additional monitors to your IAM Identity Center instance across Regions.

Comparison of multi-monitor approaches



Comparison of multi-monitor approaches

Consideration	Cross-Region IAM Identity Center access	IAM Identity Center multi-Region replication
Setup requirements	No additional IAM Identity Center setup required	Requires configuring IAM Identity Center replication
Identity data location	Remains in the IAM Identity Center Region only	Replicated to each configured Region
Latency	Depends on distance to the IAM Identity Center Region	Lower latency when an IAM Identity Center replica is in the same Region

Consideration	Cross-Region IAM Identity Center access	IAM Identity Center multi-Region replication
Regional availability	Depends on IAM Identity Center Region availability	Continues to work if the IAM Identity Center primary Region is unavailable

Cross-Region IAM Identity Center access

With cross-Region IAM Identity Center access, you create an Deadline Cloud monitor in a different Region than your IAM Identity Center instance. Deadline Cloud reads IAM Identity Center identity data from the Region where your IAM Identity Center instance is located.

When you create a monitor using the Deadline Cloud console, the console automatically detects your IAM Identity Center instance and connects the monitor to it, even if the instance is in a different Region. When you create a monitor using an AWS SDK, specify the Region where your IAM Identity Center instance is located.

Considerations

- Cross-Region IAM Identity Center access requires your IAM Identity Center instance to be in a commercial AWS Region. IAM Identity Center instances in opt-in Regions aren't supported.
- You can't change the IAM Identity Center Region after you create the monitor.

IAM Identity Center multi-Region replication

IAM Identity Center multi-Region replication synchronizes your IAM Identity Center identity store data, including users, groups, and group memberships, to additional AWS Regions. After you enable replication to a Region, you can connect your monitor in that Region to the IAM Identity Center replica.

Multi-Region replication is useful in the following scenarios:

- You need lower latency for users closer to the replicated Region.
- You need monitors that continue to work if the IAM Identity Center primary Region is unavailable.

To enable multi-Region replication, see Using IAM Identity Center across multiple AWS Regions in the *IAM Identity Center User Guide*. After you enable replication for a Region, you can create Deadline Cloud monitors there by using the console or an AWS SDK.

Share the Deadline Cloud monitor URL

When you set up the Deadline Cloud service, by default you create a URL that opens the Deadline Cloud monitor for your account. Use this URL to open the monitor in your browser or on your desktop. Share the URL with other users so that they can access the Deadline Cloud monitor.

Before a user can open the Deadline Cloud monitor, you must grant the user access. To grant access, either add the user to the list of authorized users for the monitor or add them to a group with access to the monitor. For more information, see Managing users in Deadline Cloud.

To share the monitor URL

1. Open the Deadline Cloud console.
2. From **Get started**, choose **Go to Deadline Cloud dashboard**.
3. On the navigation pane, choose **Dashboard**.
4. In the **Account overview** section, choose **Account details**.
5. Copy and then securely send the **URL** to anyone who needs to access the Deadline Cloud monitor.

Open the Deadline Cloud monitor

You can open the Deadline Cloud monitor by any of the following ways:

- **Console** – Sign in to the AWS Management Console and open the Deadline Cloud console.
- **Web** – Go to the monitor URL that you created when you set up Deadline Cloud.
- **Monitor** – Use the desktop Deadline Cloud monitor.

When you use the console, you must be able to sign in to AWS using an AWS Identity and Access Management identity, and then sign in to the monitor with AWS IAM Identity Center credentials. If you only have IAM Identity Center credentials, you must sign in using the monitor URL or the desktop application.

To open the Deadline Cloud monitor (web)

1. Using a browser, open the monitor URL that you created when you set up Deadline Cloud.
2. Sign in with your user credentials.

To open the Deadline Cloud monitor (console)

1. Open the Deadline Cloud console.
2. In the navigation pane, select **Farms**.
3. Select a farm, then choose **Manage jobs** to open the **Deadline Cloud monitor** page.
4. Sign in with your user credentials.

To open the Deadline Cloud monitor (desktop)

1. Open the Deadline Cloud console.

   -or-

   Open the Deadline Cloud monitor - web from the monitor URL.

2. - On the Deadline Cloud console, do the following:
     1. In the monitor, choose **Go to Deadline Cloud dashboard**, and then choose **Downloads** from the left menu.
     2. From **Deadline Cloud monitor**, choose the monitor version for your desktop.
     3. Choose **Download**.
   - On the Deadline Cloud monitor - web, do the following:
     - From the left menu, choose **Workstation setup**. If the **Workstation setup** item isn't visible, use the arrow to open the left menu.
     - Choose **Download**.
     - From **Select an OS**, choose your operating system.
3. Download the Deadline Cloud monitor - desktop.
4. After you download and install the monitor, open it on your computer.

   - The first time you open the Deadline Cloud monitor, you must provide the monitor URL and create a profile name. Next you sign in to the monitor with your Deadline Cloud credentials.

- After you create a profile, you open the monitor by selecting a profile. You might need to enter your Deadline Cloud credentials.

Change your language settings

After you create and open your Deadline Cloud monitor, you can change your language settings. By default, the monitor language is set to your system's language settings.

To change your language settings from Deadline Cloud monitor (desktop)

1. From your user profile, select **Settings**, then choose **Language**.
2. From the dropdown menu, select one of the available languages.
3. Confirm that your chosen language is the listed option, then choose **Confirm and apply** to apply the change.

   After the monitor refreshes, it displays in the chosen language.

   After you change the language setting, it is the default upon opening and remains the default until you change it again or uninstall the desktop application.

To change the Deadline Cloud monitor language on the web, change the preferred language in your browser settings.

> **Note**
>
> If your browser or operating system is set to a language that is not supported by Deadline Cloud, English becomes the default language for Deadline Cloud monitor.

Submit a job bundle

You can submit a job bundle directly from the AWS Deadline Cloud monitor desktop application. A job bundle is a directory that contains the files and information needed to submit a job to Deadline Cloud. For sample job bundles, see deadline-cloud-samples on the GitHub website.

To submit a job bundle

- In the Deadline Cloud monitor desktop application, choose **File**, **Submit Job Bundle**. This feature is not available in the Linux AppImage or MacOS x64 builds.

View queue and fleet details in Deadline Cloud

You can use the Deadline Cloud monitor to view the configuration of the queues and fleets in your farm. You can also use the monitor to see a list of the jobs in a queue or the workers in a fleet.

You must have `VIEWING` permission to view queue and fleet details. If the details don't display, contact your administrator to get the correct permissions.

To view queue details

1. Open the Deadline Cloud monitor.
2. From the list of farms, choose the farm that contains the queue that you're interested in.
3. In the list of queues, choose a queue to display its details. To compare the configuration of two or more queues, select more than one check box.
4. To see a list of jobs in the queue, choose the queue name from the list of queues or from the details panel.

If the monitor is already open, you can select the queue from the **Queues** list in the left navigation pane.

To view fleet details

1. Open the Deadline Cloud monitor.
2. From the list of farms, choose the farm that contains the fleet that you're interested in.
3. In **Farm resources**, choose **Fleets**.
4. In the list of fleets, choose a fleet to display its details. To compare the configuration of two or more fleets, select more than one check box.
5. To see a list of workers in the fleet, choose the fleet name from the list of fleets or from the details panel.

If the monitor is already open, you can select the fleet from the **Fleets** list in the left navigation pane.

Manage jobs, steps, and tasks in Deadline Cloud

When you select a queue, the job monitor section of the Deadline Cloud monitor shows you the jobs in that queue, the steps in the job, and the tasks in each step. When you select a job, step, or task, you can use the **Actions** menu to manage each.

To open the job monitor, follow the steps to view a queue in View queue and fleet details in Deadline Cloud, then select the job, step, or task to work with.

For jobs, steps, and tasks, you can do the following:

- Change the status to **Requeued**, **Succeeded**, **Failed**, or **Canceled**.
- Download the processed output from the job, step, or task.
- Copy the ID of the job, step, or task.

For the selected job, you can:

- Archive the job.
- Modify the job properties, including name, description, priority, or max worker count.
- View step to step dependencies.
- View additional details using the job's parameters.
- Resubmit the job.

For for more information, see View and manage job details in Deadline Cloud.

For each step, you can:

- View the dependencies for the step. The dependencies for a step must be completed before the step runs.

For details, see View a step in Deadline Cloud.

For each task, you can:

- View logs for the task.
- View task parameters.

For more information, see View a task in Deadline Cloud.

View and manage job details in Deadline Cloud

The **Job monitor** page in the Deadline Cloud monitor provides you with the following:

- An overall view of the progress of a job.
- A view of the steps and tasks that make up the job.

Choose a job from the list to view a list of steps for the job, and then choose a step from the list of steps to view the tasks for the job. After you choose an item, you can use the **Actions** menu for that item to view details.

To view job details

1. Follow the steps to view a queue in View queue and fleet details in Deadline Cloud.
2. In the navigation pane, select the queue where you submitted your job.
3. Select a job using one of the following methods:

    a. From the **Jobs** list, select a job to view its details.

    b. From the **search** field, enter any text associated with the job, such as the job name or user that created the job. From the results that display, select the job you want to view.

The details of a job include the steps in the job and the tasks in each step. You can use the **Actions** menu to do the following:

- Change the status of the job. You can requeue, resume, suspend, mark as succeeded, mark as failed, or cancel a job.
- Resubmit the job with different properties or settings.
- Archive the job. For more information, see Archive a job.
- View and modify the properties of a job.
    - You can view the dependencies between steps in the job.
    - You can change the priority of the job in a queue. Jobs with higher number priority are processed before jobs with lower number priority. Jobs can have a priority between 1 and 100. When two jobs have the same priority, the oldest job is scheduled first.

  - You can change the maximum worker count, maximum failed tasks count, and maximum retries per task.
- View the parameters for the job that were set when the job was submitted.
- Download the output of a job. When you download the output of a job, it contains all of the output generated by the steps and tasks in the job.
- Browse the input and output file attachments for the job. For more information, see Browsing job attachments in Deadline Cloud.

Archive a job

To archive a job, it must be in a terminal state, `FAILED`, `SUCCEEDED`, `SUSPENDED`, or `CANCELED`. The `ARCHIVED` state is final. After a job is archived, it can't be requeued or modified.

The job's data is not affected by archiving the job. The data is deleted when the inactivity timeout is reached, or when the queue containing the job is deleted.

Other things that happen to archived jobs:

- Archived jobs are hidden in the Deadline Cloud monitor.
- Archived jobs are visible in a read-only state form the Deadline Cloud CLI for 120 days before deletion.

Requeue a job

When you requeue a job, all of the tasks without step dependencies switch to `READY`. The status of steps with dependencies switch to `READY` or `PENDING` as they are restored.

- All jobs, steps, and tasks switch to `PENDING`.
- If a step doesn't have a dependency, it switches to `READY`.

Resubmit a job

There might be times when you want to run a job again, but with different properties and settings. For example, you might submit a job to render a subset of testing frames, verify the output, then run the job again with the full frame range. To do this, resubmit the job.

When you resubmit a job, new tasks without dependencies become `READY`. New tasks with dependencies become `PENDING`.

- All new jobs, steps, and tasks become `PENDING`.
- If a new step doesn't have a dependency, it becomes `READY`.

When you resubmit a job, you can only change properties that were defined as configurable when the job was first created. For example, if the name of a job is not defined as a configurable property of the job when first submitted, then the name cannot be edited on resubmission.

Customize the panels in the job monitor

The **Job monitor** page is built from panels. The **Jobs**, **Steps**, and **Tasks** panels are shown by default. Other panels are available to add when you need them, and you can move and resize any panel so that the information you use most is where you want it.

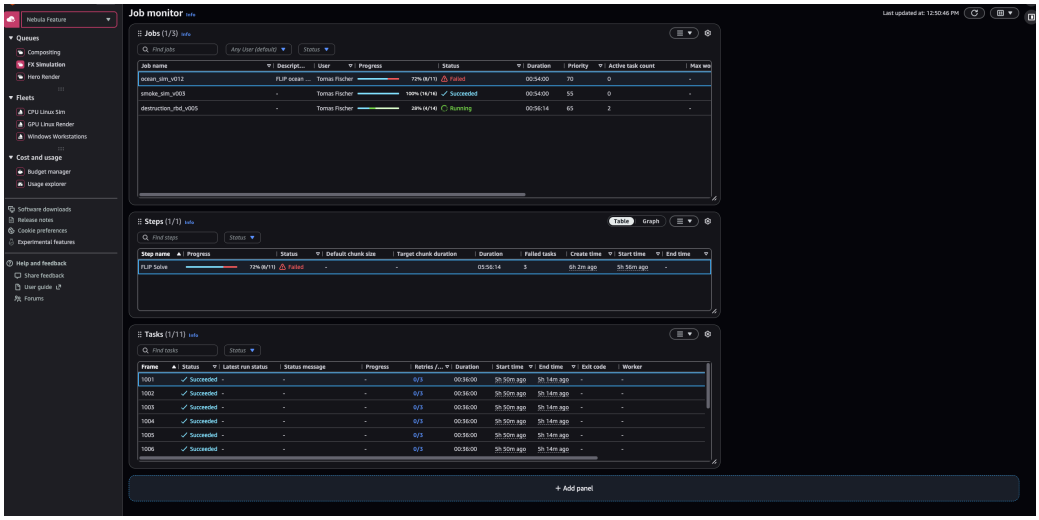
The monitor saves your arrangement in your browser and restores it the next time you open the job monitor.

The following panels are available to add:

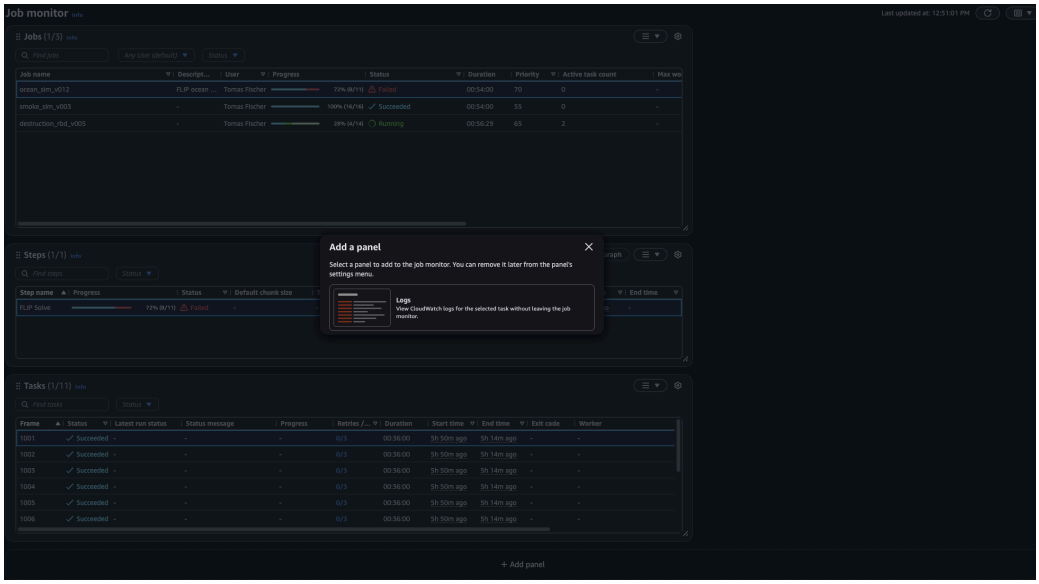
- **Logs** shows the Amazon CloudWatch logs for the selected task, so that you can read them without leaving the job monitor. For more information, see View session and worker logs in Deadline Cloud.

To add a panel

1. Follow the steps in View and manage job details in Deadline Cloud to view a list of jobs.
2. Choose **Add panel**, the outlined tile that follows the last panel on the page. You can also choose **Add panel** from the actions menu at the top of the page.



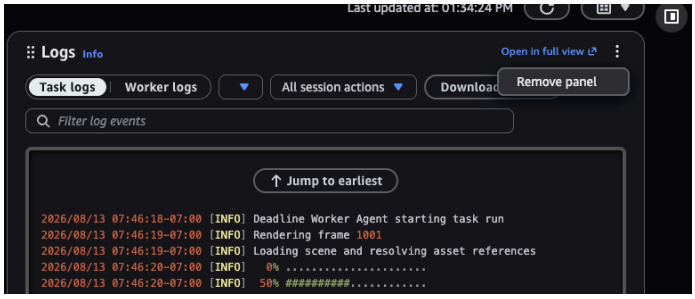
3. In **Add a panel**, choose the panel that you want. The panel is added after the last panel on the page.



The **Add panel** tile appears only when panels remain to add. After the page shows every available panel, the tile no longer appears.

To remove a panel

1. In the header of the panel that you want to remove, choose the **Panel settings** icon.
2. Choose **Remove panel**.



To move a panel, use its drag handle. To resize a panel, use its resize handle. With either handle, select the handle, press Space or Enter, use the arrow keys to move or resize the panel, and then press Space or Enter to confirm the change or Esc to discard it. You can also select and hold either handle to move or resize the panel.

To return the page to the panels and arrangement that it started with, choose **Reset to default layout** from the actions menu at the top of the page. Resetting discards your customizations, including any panel that you added that isn't shown by default.

View a step in Deadline Cloud

Use the AWS Deadline Cloud monitor to view the steps in your processing jobs. In the **Job monitor**, the **Steps** list shows the list of steps that make up the selected job. When you select a step, the **Tasks** list shows the tasks in the step.

To view a step

1. Follow the steps in View and manage job details in Deadline Cloud to view a list of jobs.
2. Select a job from the **Jobs** list.
3. Select a step from the **Steps** list.

You can use the **Actions** menu to do the following:

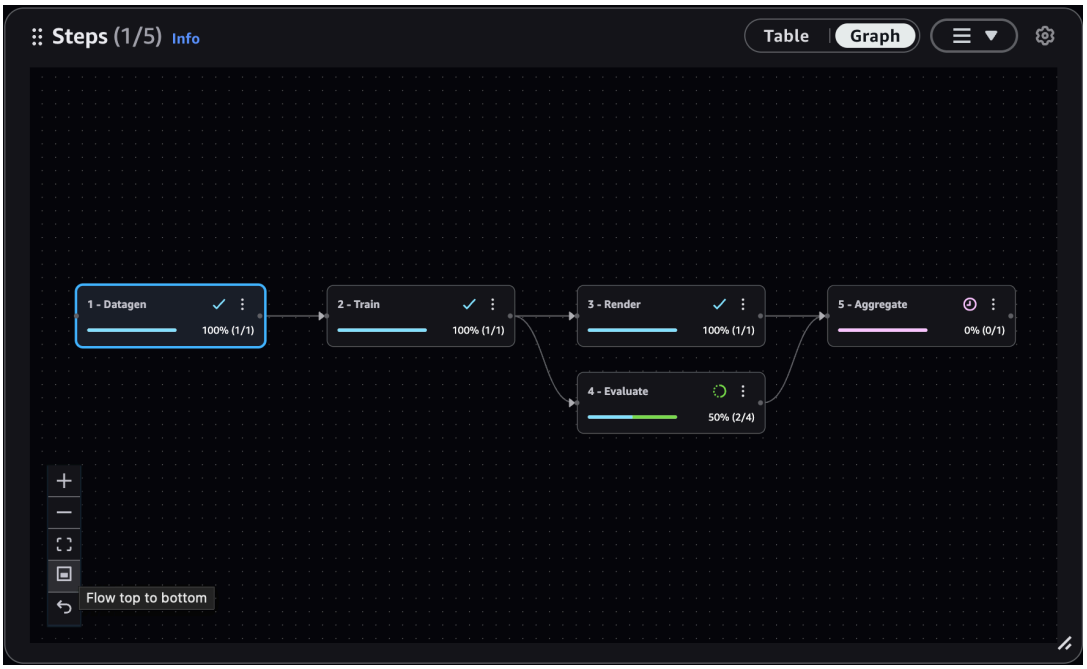
- Change the status of the step. You can requeue, resume, suspend, mark as succeeded, mark as failed, or cancel a step.
- Download the output of the step. When you download the output of a step, it contains all of the output generated by the tasks in the step.
- Browse the input and output file attachments for the step. For more information, see Browsing job attachments in Deadline Cloud.

- View the dependencies of a step. The dependencies table shows a list of steps that must be complete before the selected step starts, and a list of steps that are waiting for this step to complete.

View step dependencies as a graph

The **Steps** list can show your steps as a table or as a dependency graph. The graph shows each step as a node and each dependency as an arrow that points from a step to the step that depends on it, so you can see the order in which your steps run. Each node shows the step's name and run status.

Jobs with dependencies are created by workflows that submit dependent steps, such as a job that renders frames and then publishes the result. If a job has no step dependencies, the graph reports that none were found.



To view the step dependency graph

1. Select a job from the **Jobs** list.
2. At the top of the **Steps** list, choose **Graph**. To return to the table, choose **Table**.

In the graph, you can do the following:

- Choose a node to select its step. The **Tasks** list then shows the tasks in that step.

- On a node, choose the actions menu, then choose **View step dependencies** to filter the panel to the steps that the selected step depends on and the steps that depend on it.
- Move a node to make a crowded part of the graph easier to read. To move a node with the keyboard, select the node, and then use the arrow keys. To move it in larger increments, hold down Shift. To return every node to the position that the graph computed for it, choose **Reset layout**.
- Change the direction that dependencies flow. The graph flows from left to right by default. To flow it from top to bottom instead, choose **Flow top to bottom**. To flow it from left to right again, choose **Flow left to right**. Changing the direction returns any node that you moved to its computed position. The Deadline Cloud monitor remembers the direction that you choose and uses it the next time that you view a graph.

Graph controls

The graph controls are in the lower-left corner of the graph. To see what a control does, point to it or move the keyboard focus to it. The graph provides the following controls.

Zoom in

Magnifies the graph so that you can read the steps in a crowded area.

Zoom out

Shrinks the graph so that you can see more of it at once.

Fit view

Resizes and centers the graph so that every step fits in the panel. Choose this control to return to a full view of the graph after you move or zoom it.

Flow top to bottom, Flow left to right

Changes the direction that dependencies flow. This control names the direction that it changes the graph to, so it shows **Flow top to bottom** while the graph flows from left to right.

Reset layout

Returns every node to the position that the graph computed for it, which discards the position of any node that you moved.

View a task in Deadline Cloud

Use the AWS Deadline Cloud monitor to view the tasks in your processing jobs. In the **Job monitor**, the **Tasks** list shows the tasks that make up the step selected in the **Steps** list.

To view a task

1. Follow the steps in View and manage job details in Deadline Cloud to view a list of jobs.
2. Select a job from the **Jobs** list.
3. Select a step from the **Steps** list.
4. Select a task from the **Tasks** list.

You can use the **Actions** menu to do the following:

- Change the status of the task. You can requeue, suspend, mark as succeeded, mark as failed, or cancel a task.
- View task runs to see the history of attempts for the task.
- View session logs and worker logs. You can also add the **Logs** panel to see the logs for the selected task in the job monitor. For more information, see View session and worker logs in Deadline Cloud.
- View the worker dashboard for the worker that processed the task. For more information, see View worker details in the worker dashboard.
- View the parameters that were set when the task was created.
- Download the output of the task. When you download the output of a task, it contains only the output generated by the selected task.
- Browse the input and output file attachments for the task. For more information, see Browsing job attachments in Deadline Cloud.

Viewing a task's latest run

The **Tasks** list shows details from each task's most recent run. The columns include **Latest run status**, **Status message**, **Progress**, **Exit code**, **Worker**, **Fleet name**, and **Instance type**. Use these columns to find a failed or retried task without opening the task runs view for each task. For steps that use task chunking, the list also shows a **Chunk range** column.

To show more details from the latest run, open the table preferences and select additional columns. Additional columns include **Latest run start time**, **Latest run end time**, **Latest run duration**, **Host name**, **IPv4 addresses**, **vCPU**, **Memory**, **Operating system**, **GPU chip name**, and **GPU count**.

To open the worker dashboard for the worker that processed a task, choose the worker ID in the **Worker** column. To open the fleet details, choose the fleet name in the **Fleet name** column. For more information, see View worker details in the worker dashboard.

A task's status can differ from its latest run status. For example, a failed task shows a latest run status of **Never attempted** when the service didn't start the run. The two statuses can also differ for a few seconds after a run finishes, while the monitor refreshes.

View session and worker logs in Deadline Cloud

Logs provide you with detailed information about the status and processing of tasks. In the AWS Deadline Cloud monitor, you can see the following two types of logs:

- *Session logs* detail the timeline of actions, including:
  - Setup actions, such as attachment syncing and loading the software environment
  - Running a task or set of tasks
  - Closure actions, such as shutting down the environment on a worker

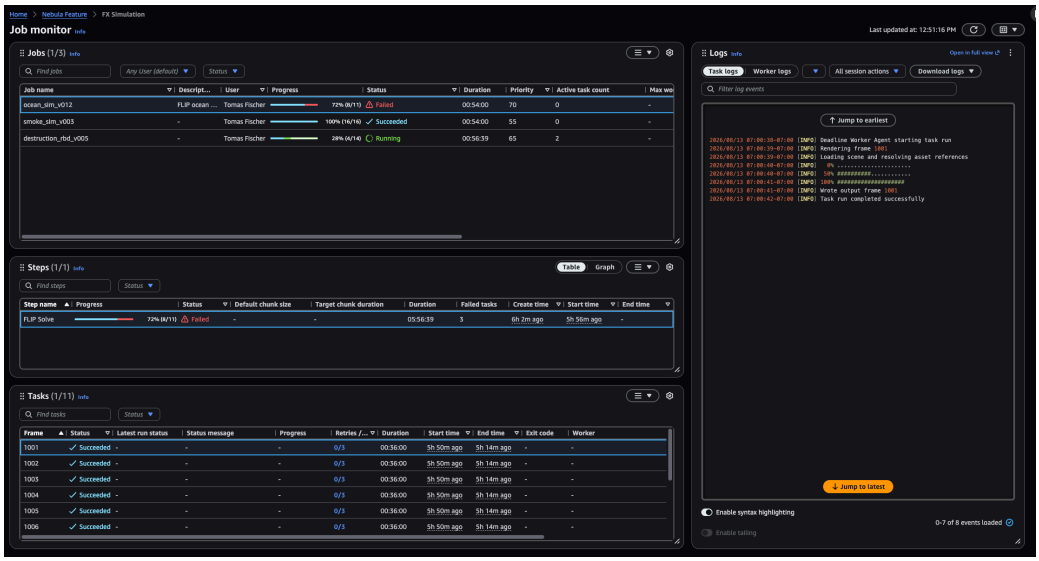
  A session includes processing of at least one task, and can include multiple tasks. Session logs also show information about Amazon Elastic Compute Cloud (Amazon EC2) instance type, vCPU, and memory. Session logs also include a link to the log for the worker used in the session.
- *Worker logs* provide details for the timeline of actions that a worker processes during its lifecycle. Worker logs can contain information about multiple sessions.

You can download session and worker logs so that you can examine them offline.

You can read a task's logs in two places. **View logs** opens the logs on their own page. The **Logs** panel shows the logs for the task that you select on the **Job monitor** page, so you can move between tasks without leaving the job, step, and task lists. The panel isn't shown until you add it. For more information, see Customize the panels in the job monitor.

To view task logs in the job monitor

1. Add the **Logs** panel to the **Job monitor** page. For more information, see Customize the panels in the job monitor.
2. Select a job, select a step, and then select a task. The **Logs** panel shows the logs for the selected task.



3. (Optional) Choose **Task logs** or **Worker logs**.
4. (Optional) Choose **All session actions**, and then choose a single action to move the log to that part of the session.
5. (Optional) To find text in the log, use **Filter log events**. To save the log, choose **Download logs**.
6. (Optional) To open the logs on their own page, choose **Open in full view**.

> **Note**
>
> When the **Logs** panel is on the page, the task **Actions** menu doesn't show the **View logs** and **View worker logs** items that open logs in the monitor, because the panel already shows the logs for the selected task. The items that open logs in a new tab are still available.

To view session logs

1. Follow the steps in View and manage job details in Deadline Cloud to view a list of jobs.

2. Select a job from the **Jobs** list.
3. Select a step from the **Steps** list.
4. Select a task from the **Tasks** list.
5. From the **Actions** menu, choose **View logs**.

The **Timelines** section shows a summary of the actions for the task. To see more tasks run in the session and to see the shutdown actions for the session, choose **View logs for all tasks**.

To view worker logs from a task

1. Follow the steps in View and manage job details in Deadline Cloud to view a list of jobs.
2. Select a job from the **Jobs** list.
3. Select a step from the **Steps** list.
4. Select a task from the **Tasks** list.
5. From the **Actions** menu, choose **View logs**.
6. Choose **Session info**.
7. Choose **View worker log**.

To view worker logs from fleet details

1. Follow the steps in View queue and fleet details in Deadline Cloud to view a fleet.
2. Select a **Worker ID** from the **Workers** list.
3. From the **Actions** menu, choose **View worker logs**.

View worker details in the worker dashboard

The *worker dashboard* provides details for the worker that processes a task. You can see:

- Metadata, such as the instance type, for the worker.
- The session actions that the worker performed.
- Worker performance, including CPU, memory, and disk usage. The dashboard also shows GPU utilization for GPU-accelerated instances.
- A graph of the CPU, memory, and disk usage over time. The graph also includes GPU utilization for GPU-accelerated instances.

- A graph of the disk speed over time.
- The worker log for the task.

To view the worker dashboard from a task

1. Follow the steps in View and manage job details in Deadline Cloud to view a list of jobs.
2. Select a job from the **Jobs** list.
3. Select a step from the **Steps** list.
4. Select a task from the **Tasks** list.
5. In the task table, from the **Actions** menu, choose **View worker dashboard**.

To view the worker dashboard from fleet details

1. Follow the steps in View queue and fleet details in Deadline Cloud to view a fleet.
2. Select a **Worker** from the **Workers** list.
3. From the **Actions** menu, choose **View worker dashboard**.

Use cases

Detecting under-provisioned instances

When renders take longer than expected, the worker dashboard can help determine if your instances are adequately sized for your workloads. While 100% vCPU utilization is normal for many renderers, consistently high memory usage near maximum capacity and elevated disk space utilization may indicate that your instances are under-provisioned. For GPU-accelerated workloads, consistently high GPU utilization might also indicate that you need additional GPU capacity. In such cases, upgrading your fleet's instance configuration can reduce render errors and significantly improve render times. However, it's important to continue monitoring worker performance after upgrading to ensure you've found the optimal balance - upgrading too aggressively can lead to unnecessary costs through over-provisioning.

Detecting over-provisioned instances

Even when tasks are completing successfully, there may be opportunities to optimize your costs. The worker dashboard can reveal if you're paying for more compute power than your workloads

require. If you see that the worker has low average vCPU usage, minimal memory utilization, and excess unused disk space, you can downsize the instance configuration of your fleet.

Troubleshooting failed tasks

When investigating failed tasks, the worker dashboard serves as a valuable diagnostic tool. Pay particular attention to peak memory usage and disk space utilization - if these metrics approach or reach 100%, they're likely the root cause of your task failures. Such resource exhaustion indicates that your current instances lack the capacity to handle your workloads effectively. In these cases, provisioning instances with increased memory or disk space will help ensure successful task completion.

Optimal instance utilization rate

vCPU Utilization

Target range: 70–90%

- **Below 70%**: Likely underutilizing compute resources, meaning you're paying for more CPU than your workload needs
- **70–90%**: Optimal range where you're efficiently using resources without hitting bottlenecks
- **Consistently at 100%**: Could indicate CPU bottlenecks that might slow down renders

Some render tasks will naturally be more CPU-intensive than others, and 100% vCPU usage may not be an issue. Real-time visualization tasks might show more consistent CPU utilization, while tasks with changing computational requirements might have varying patterns.

Memory Utilization

Target range: 70–85%

- **Below 50%**: Potentially oversized instances for your workload
- **70–85%**: Optimal utilization with enough headroom for spikes
- **Above 90%**: Risk of performance degradation or out-of-memory errors

Memory requirements can vary significantly depending on scene complexity, texture resolution, and simulation data. Monitoring memory trends over time is important to identify if your workloads are growing in memory requirements.

Disk Space Utilization

Target range: 60–80%

- **Below 40%**: Likely over-provisioned storage
- **60–85%**: Good utilization with room for temporary files and caches
- **Above 85%**: Risk of running out of space during large renders

Remember that disk I/O performance can be just as important as capacity, especially for workloads that read/write large texture or cache files during rendering.

Download finished output in Deadline Cloud

After a job is finished, you can use the AWS Deadline Cloud monitor to download the results to your workstation. The output file is stored with the name and location that you specified when you created the job.

Output files are stored indefinitely. To reduce storage costs, consider creating an S3 Lifecycle configuration for your queue's Amazon S3 bucket. For more information, see Managing your storage lifecycle in the *Amazon Simple Storage Service User Guide*.

To download the finished output of a job, step, or task

1. Follow the steps in View and manage job details in Deadline Cloud to view a list of jobs.
2. Select the job, step, or task that you want to download the output for.
   - If you select a job, you can download all of the output for all of the tasks in all of the steps for that job.
   - If you select a step, you can download all of the output for all of the tasks in that step.
   - If you select a task, you can download the output for that individual task.
3. From the **Actions** menu, choose **Download output**.
4. The output will be downloaded to the location set when the job was submitted.

> **Note**
>
> Downloading output using the menu is currently only supported for Windows and Linux. If you have a Mac and you choose the **Download output** menu item, a window shows the AWS CLI command that you can use to download the rendered output.

To browse and selectively download individual files instead of all output, see Browsing job attachments in Deadline Cloud.

View output download status in Deadline Cloud

The **Download status** column in the AWS Deadline Cloud monitor shows, for each job and each task, whether automatic downloads copied the output files to the download location. A job's render status and its download status answer different questions. A status of SUCCEEDED means the tasks rendered. The **Download status** column tells you whether the resulting files are on your file system yet.

The column reports one specific transfer: the automatic downloads that the `deadline queue sync-output` command performs for a queue. A studio administrator usually schedules that command once for the whole queue, so finished output lands on the shared storage everyone works from without anyone downloading files by hand. Deadline Cloud moves output files in two other ways, and the column doesn't report either of them.

Ways output files move, and where each one is reported

Transfer	What it moves	Who sets it up	Where you see it
Automatic downloads	Output from every job that finished in a queue, to the shared download location.	A studio administrator, once per queue.	The **Download status** column, described on this page.
Download finished output	Output from one job you pick, to the computer you're sitting at, when you ask for it.	Nobody. The action is in the monitor.	The download progress in the monitor. See Download finished

Transfer	What it moves	Who sets it up	Where you see it
			output in Deadline Cloud.
Job attachment transfer during a session	Input files onto the worker before a task runs, and output files back to Amazon S3 when it finishes.	Nobody. The transfer is part of any job that uses job attachments.	The session actions in the job's logs. See View session and worker logs in Deadline Cloud.

Because the two other transfers are reported elsewhere, a job can show **Downloaded** in the column while a task's output is still uploading to Amazon S3 in a session, or show a dash after you already downloaded the output to your own computer.

If you watch jobs in the monitor, you don't set up anything to see the column. You need the Deadline Cloud monitor desktop application and access to the location where the download command records the status, which is the shared storage that holds your outputs. If you administer the queue, scheduling the download command is what makes the column work for everyone who watches the queue. For more information, see Set up scheduled downloads.

Availability and prerequisites

The **Download status** column is available in the Deadline Cloud monitor desktop application version 1.2.0 or later. It doesn't appear in the web version of the monitor. There is no setting to turn the feature on. The column shows data when the following are in place:

- Automatic downloads are running for the queue with the `deadline queue sync-output` command, using version 0.60.4 or later of the Deadline CLI. That version started recording the download status that the monitor reads. For more information, see Set up scheduled downloads and Configuring AWS credentials.
- Jobs in the queue use job attachments to record their output files. For more information, see Job attachments in Deadline Cloud.
- The download command knows where output files belong on the machine that runs it. Most studios run the command on one machine that writes to shared storage, which needs a storage profile to map the paths from the submitting machine to the downloading one. If you run the command on the machine you submit jobs from, the submitted paths are already correct there,

so you can pass `--ignore-storage-profiles` instead of configuring a profile. Download status is recorded either way. For more information, see Storage profiles for job attachments.

- The machine running the monitor can reach the location where the download command recorded the status. With a storage profile, the `deadline queue sync-output` command writes the status file into the profile's file system location, in a `.deadline` subfolder on the same shared storage that holds the downloaded output. That location must be mounted on the monitor's machine, and the column shows **Unavailable from this machine** when it isn't. With `--ignore-storage-profiles`, the command records the status on its own machine, where the monitor reads it without a shared mount.

The column fills in after the `deadline queue sync-output` command completes its first run for the queue. Until then, jobs show a dash.

The column is shown by default on both the jobs table and the tasks table. To hide or show it, open the table preferences and select **Download status** under **Column preferences**. Each table remembers the setting separately.

Download status for a job

Each job in the jobs table shows one of the following download statuses:

Downloaded

Every available output for the job is in the download location.

In progress

A progress bar, such as 3/4, showing that outputs for some tasks are downloaded and more are still to come. The remaining outputs arrive on a later download run. A job whose tasks are done can also show a bar when some of its tasks had problems: a red segment marks tasks that failed to render, or whose download failed. Open the job's tasks table to see which.

An error name, such as **Permission denied**

The download failed. The tasks rendered, but copying the files to the download location didn't complete. The cell names the reason: **Permission denied**, **Disk full**, **Path not found**, or **Network error**. A failure the monitor can't classify shows **Failed**. For the full list of reasons and how to resolve them, see Download error codes.

No attachments or **Missing storage profile**

The downloader skipped the job, and the cell names the reason. A job without job attachments has no output to download and needs no action. A job that is missing a storage profile downloads nothing until the profile is configured, so that reason is actionable. For more information, see Why was my job skipped?.

- (a dash)

There is no download information for the job yet. The most common cause is timing: the job finished after the last download run, so the downloader hasn't seen it yet. The status fills in on the next run. A job where no task succeeded also shows a dash, because it produced no output to download. The job's own status column already reports that failure.

Unavailable from this machine

A download status record exists, but the monitor can't read it from your computer. Check that the shared drive that holds your outputs is mounted. For where the status record lives and why the mount matters, see Availability and prerequisites.

Download status for a task

Open a job to see the same column for each task in the tasks table. The per-task view shows which outputs landed. Each task shows one of the following:

Downloaded

The task's output is in the download location.

An error name, such as **Permission denied**

The download of the task's files failed. Choose the status to see the error detail and a **Troubleshoot with AI** button. For more information, see Download error codes.

No outputs

The task produced no file to download. The task either finished successfully without writing any files, which is normal for setup or validation work, or it failed to render on the farm. The task's own run status column tells you which. For more information, see Is it a download problem or a render problem?.

No attachments or **Missing storage profile**

The job was skipped, and the reason shows on each task as well. **No attachments** means the job isn't set up to produce downloadable output, so none of its tasks have anything to download, even tasks that rendered successfully. Every task in the job shows the label, and it needs no action. In the tasks table, **Missing storage profile** is a link that opens an explanation, a link to the storage profile documentation, and a **Troubleshoot with AI** button.

- (a dash)

The download run hasn't looked at the task yet, or nothing is recorded for it. The task's output may still be coming down. Wait for the next run.

A task that was requeued while its earlier output is still on disk keeps showing **Downloaded** with a blue info icon. Choose the icon to see the explanation: the files in the download location are from the previous run, and a newer run is in flight. The icon clears after the task finishes and the next download run completes.

Outputs last synced indicator

Every value in the **Download status** column is a snapshot from the last time the download command ran. The **Outputs last synced** indicator in the page header tells you when that was, so you know how current the column is. Check the indicator before acting on a download status.

Green, such as "Outputs last synced 5 minutes ago"

The last download run succeeded recently. The column is current.

Yellow, "Outputs may be stale"

No download run has completed in over 30 minutes, so the statuses in the column may be out of date. Select the indicator for details. The usual cause is that the scheduled download command stopped running on the machine that syncs the queue.

Red, "Output sync failed"

The last download run failed. Select the indicator to see the failures grouped by reason, along with a **Troubleshoot with AI** button that helps you diagnose the specific failure. For more information, see Troubleshoot with AI.

The column refreshes on its own about every 60 seconds. It also refreshes as soon as you return focus to the monitor window, so clicking away and back pulls the latest download run immediately. You don't need to restart the monitor to pick up a new run. Refreshing rereads what the download command already recorded. If the indicator shows the outputs may be stale, the download command isn't running, and refreshing won't change the column.

Browsing job attachments in Deadline Cloud

Use the job attachments browser in the Deadline Cloud monitor to see the file structure of your job's input and output attachments. You can selectively download files, navigate the folder hierarchy, and filter by input or output. For supported file types, you can also preview content inline without downloading.

The attachments browser displays both input files (submitted with the job) and output files (produced by workers during processing). You can filter the view to show only inputs, only outputs, or all files together.

Opening the attachments browser

You can open the attachments browser from a job, step, or task. The scope of files displayed depends on the level from which you open the browser.

To browse attachments for a job

1. Follow the steps in View and manage job details in Deadline Cloud to view a list of jobs.
2. Select the job that you want to browse attachments for.
3. From the **Actions** menu, choose **Browse attachments**.

To view attachments for a specific step or task, select the step or task, and then choose **Browse attachments** from the **Actions** menu. When you browse at the step or task level, the browser shows only the output files that the step or task produced.

You can also open the context menu on a job, step, or task and choose **Browse attachments**.

Navigating and filtering files

The attachments browser displays files in an expandable tree structure organized by their original file paths. Use the following controls to navigate:

- **Segment control** – Switch between viewing **All**, **Output**, or **Input** files.
- **Text filter** – Filter the file list by name to find specific files.
- **Previewable files only** – Turn on the **Previewable files only** toggle to show only files that support inline preview. Use this toggle when a folder mixes previewable files, such as JPG images, with formats that don't support preview, such as EXR images. Stepping through previews then skips the files without one.
- **Folder expansion** – Choose a folder to expand or collapse its contents.

Downloading files

You can download individual files or a selection of files from the attachments browser.

To download selected files

1. In the attachments browser, select the checkboxes next to the files or folders that you want to download. Selecting a folder selects all files within that folder.
2. Choose **Download**.
3. Choose a download method:
   - **AWS Command Line Interface (AWS CLI)** – Downloads files to your storage profile paths, preserving the original directory structure. Use this method for large files or when you need files in their original locations.
   - **Browser download** – Downloads the selected files as a ZIP archive directly in your browser. Use this method for quick access to a small number of files. A warning appears when the total size exceeds 500 MB.

> **Note**
>
> On the Deadline Cloud monitor desktop application (Windows, macOS, and Linux), the AWS CLI download runs directly without requiring you to copy a command. The desktop application also supports downloading files to their original storage profile paths.

Previewing files

For supported file types, the attachments browser can display an inline preview without downloading the file.

To preview a file

1. In the attachments browser, select a file name in the table.
2. View the details panel that opens on the side. The panel shows metadata such as file size, path, and modification date. For supported file types, an inline preview loads automatically.
3. (Optional) To open a full-screen preview, choose the expand icon that appears when you hover over a file row.

The following file types support inline preview:

- Images: PNG, JPG, JPEG, GIF, BMP, WebP, SVG, ICO
- Text: TXT, LOG, JSON, XML, YAML, YML, CSV, MD, INI, CFG, CONF, SH, BAT, PY, JS, TS, TSX, JSX, HTML, CSS, TOML
- Video: MP4, WebM, OGG
- Audio: MP3, WAV, OGG, FLAC, AAC, WebM

Automate Deadline Cloud monitor desktop deployment and workflows

The AWS Deadline Cloud monitor desktop application includes a command-line interface (CLI) that administrators can use to set up profiles for users and that artists and developers can use to integrate the monitor into automated workflows on their workstations.

Finding the Deadline Cloud monitor executable

To use the CLI commands, run the Deadline Cloud monitor executable from a terminal. The default installation location depends on your operating system and installation method.

Windows

```
%LOCALAPPDATA%\DeadlineCloudMonitor\DeadlineCloudMonitor.exe
```

macOS

```
/Applications/DeadlineCloudMonitor.app/Contents/MacOS/DeadlineCloudMonitor
```

Linux (deb or RPM package)

```
/usr/bin/deadline-cloud-monitor
```

Linux (AppImage)

Run the AppImage file directly from the location where you downloaded it.

In the following examples, replace `DeadlineCloudMonitor` with the full path to the executable for your operating system.

Setting up a profile for streamlined user access

Administrators use the `create-profile` command to create Deadline Cloud monitor profiles for users. This command configures a profile so that users can open the monitor, log in, and start working without additional configuration or profile selection.

The `create-profile` command accepts the following flags:

- `--enable-auto-login` – Configures the monitor to automatically log in with the most recently used profile when the application starts.
- `--set-as-deadline-default` – Sets the profile as the default for Deadline Cloud tools, including the Deadline Cloud submitter, the Deadline CLI, and the Deadline Cloud GUI applications. This flag does not affect the AWS Command Line Interface (AWS CLI).

When both flags are enabled, users open the monitor and are logged in automatically with no other configuration or profile selection required.

To create a profile

Run the following command, replacing the placeholder values with your monitor details.

```
DeadlineCloudMonitor create-profile \
    --profile profile-name \
    --monitor-id monitor-id \
```

```
    --monitor-url https://monitorName.region.deadlinecloud.amazonaws.com \
    --enable-auto-login \
    --set-as-deadline-default
```

The command creates the profile and writes the configuration to the Deadline Cloud configuration files on the user's workstation. The monitor URL must be in the format `https://monitorName.region.deadlinecloud.amazonaws.com`.

> **Note**
>
> The `create-profile` command exits after creating the profile. To open the monitor with the new profile, run the `login` command or open the Deadline Cloud monitor desktop application.

Integrating the Deadline Cloud monitor into your workflows

Use the `login`, `logout`, and `handle-url` commands to integrate the Deadline Cloud monitor into scripts and automated workflows on your workstation.

Logging in and logging out

Use the `login` and `logout` commands to control authentication as part of a workflow. For example, a script that submits jobs can use the `login` command to ensure the user is authenticated before submission begins.

When you use the `login` command, the monitor opens directly to the specified profile, skipping the profile selection screen. After authentication completes, the monitor minimizes to the system tray so that your workflow can continue. If the monitor is already running for the specified profile, the existing window comes to the foreground instead of starting a new instance.

To log in to a profile

Run the following command, replacing *profile-name* with the name of your Deadline Cloud monitor profile.

```
DeadlineCloudMonitor login --profile profile-name
```

To log out of a profile

Run the following command to clear the credentials for a profile and signal any running monitor instance for that profile to exit.

```
DeadlineCloudMonitor logout --profile profile-name
```

Opening the monitor to a specific page

Use the `handle-url` command to open the Deadline Cloud monitor to a specific page. This command is useful when a script performs an action, such as creating a job, and you want to automatically open the monitor to show the result. For example, after a script submits a job, the script can call `handle-url` to open the monitor directly to the job details page.

You can also use the `deadline-cloud-monitor://` URL as a link on company websites, wikis, or task trackers to let users open the monitor directly to a specific page.

The URL uses the `deadline-cloud-monitor://` protocol scheme with a `launch` command. The URL includes the profile name and the monitor page URL to open.

To open the monitor to a specific page

Run the following command, replacing *`monitor-page-url`* with the URL-encoded monitor page URL and *`profile-name`* with your profile name.

```
DeadlineCloudMonitor handle-url --url "deadline-cloud-monitor://launch?url=monitor-page-url&profile=profile-name"
```

Manage cookie preferences in the Deadline Cloud monitor

The AWS Deadline Cloud monitor uses browser storage to save your UI preferences, such as table column layouts, your active farm selection, dashboard layouts, and your download location. You can control whether the monitor stores these preferences by managing your cookie preferences.

If you are in the European Union, a consent banner prompts you to choose your preferences the first time you open the monitor. In other locations, all cookie categories are active by default and no banner appears. You can review and change your choices at any time using the following steps.

Topics

- Cookie categories
- Withdrawing functional cookie consent

To change your cookie preferences

1. Open the Deadline Cloud monitor. For more information, see Open the Deadline Cloud monitor.
2. In the left navigation, choose **Cookie preferences**.

   If the navigation is collapsed, choose **Menu**, then choose **Cookie preferences**.
3. In the dialog, select the categories of cookies that you want to allow.
4. Choose **Save preferences**.

> **Note**
>
> The monitor saves your cookie preference for each browser on the web and for each profile in the desktop application. It does not follow your AWS account. If you use a different browser or create a new desktop profile, you must set your preferences again. Cookie preferences expire after one year, at which point the monitor prompts you to choose again.

Cookie categories

The cookie preferences dialog shows the following categories:

Essential

Essential cookies are required for the monitor to function, including authentication and session management. You cannot turn off essential cookies.

Functional

With functional cookies, the monitor remembers your UI preferences between sessions, such as your table column layouts, active farm selection, and dashboard arrangements.

Performance

Performance cookies collect anonymous statistics about how you navigate the site. The monitor does not use performance cookies at this time.

Advertising

Advertising cookies help deliver relevant marketing content. The monitor does not use advertising cookies at this time.

Withdrawing functional cookie consent

Important

If you decline or withdraw consent for functional cookies, the monitor deletes your saved preferences. The next time you open or refresh the monitor, it uses the default settings. Affected settings include table column layouts, your active farm selection, job monitor board layout, your download location, and unsaved job submission content.

The monitor deletes preferences because it cannot retain UI settings in your browser without functional cookie consent. If you later re-enable functional cookies, the monitor begins saving your preferences again, but it does not restore your previous settings.

Organize your farms, queues, and fleets

You can start with a single farm that contains one queue and one fleet. As the number of users and the variety of workloads in your AWS Deadline Cloud (Deadline Cloud) farm grow, it helps to organize the work more deliberately: a queue for each team or project, a fleet for each hardware type, or a separate farm for work that must remain isolated.

Three questions drive the design:

- **Who can see, submit, and manage which jobs?** Queues are the natural boundary for permissions and budgets.
- **Which hardware runs which jobs?** Fleets group workers by capability, and each job's host requirements route it to a fleet that can run it.
- **Does any of the work need to be strictly isolated from the rest?** For work that must not mix, such as work for different clients, create a separate farm for each workload. Separate farms are a hard security boundary.

How queues, fleets, and farms divide the work

Queues and fleets have a many-to-many relationship. You can associate one queue with several fleets, and one fleet with several queues. When a fleet serves multiple queues, it divides its workers among them. For more information, see Associate a queue and fleet.

When a queue has more than one fleet, each job's host requirements (such as GPUs, vCPUs, memory, and operating system) determine which fleets can run it. You do not need separate queues for GPU jobs and CPU jobs: associate one queue with a GPU fleet and a CPU fleet, and a render that requires a GPU is scheduled only to the GPU fleet, while jobs without GPU requirements can run on the CPU fleet. The Deadline Cloud integrated submitters include host requirements settings, so artists can describe the hardware a job needs without knowing which fleet provides it.

Queues are also where you control access and spending. You grant users an access level on each queue separately, so a user can submit jobs to one queue while only viewing another. You can also set a budget for each queue. For more information, see Managing users in Deadline Cloud and Control costs with a budget.

Queue permissions control who can see and manage jobs, but they don't fully isolate the jobs themselves. Jobs from different queues that share a fleet run on the same worker hosts, where they can share common resources such as caches of packages and files. Separate farms are a hard security boundary: farms share no Deadline Cloud resources with each other, unless you weaken the boundary yourself by sharing an external resource such as an Amazon Simple Storage Service (Amazon S3) bucket. For more information, see Isolate workloads with farms, fleets, and queues.

Divide with	Best for separating	Considerations
Queues	Teams, shows, or departmen ts that need their own job lists, permissions, and budgets	Jobs from queues that share a fleet run on the same worker hosts
Fleets	Hardware and software environments, such as GPU workers, CPU workers, or a customer-managed fleet	A fleet that serves multiple queues divides its workers among them
Farms	Workloads that need a hard isolation boundary, such as different vendors or clients	Farms share nothing, so each farm needs its own queues, fleets, and user grants

Common patterns

The following layouts cover most organizations. You can combine them, for example a shared farm for your own teams alongside a separate farm for each vendor.

One shared farm, one queue per team

Within a single organization, a shared farm with a queue for each team or workload (such as a show at a studio, or a machine learning training pipeline) is usually the right starting layout. Everyone sees the whole farm in one view. You grant each team the contributor level on its own queue and the viewer level elsewhere, so teams can follow all the work without changing each other's jobs. Budgets track spending for each queue.

Associate the queues with shared fleets. Shared fleets keep utilization high and keep workers warm, because a worker that finishes one team's job can pick up another team's job without

waiting for a new instance to start. The tradeoff is that the teams share worker hosts, which is normally acceptable within one organization's trust boundary. To add a queue, see Create a queue.

A farm for each vendor or client

When you work with outside vendors, or run work for multiple clients that must not mix, create a separate farm for each one. Users granted access to one farm can't see the others, and jobs in one farm can't run on another farm's fleets or read its job attachments. Keep the boundary intact: don't share job attachment Amazon S3 buckets, fleets, or other resources across farms. To add a farm, see Create a farm. For the security details, see Isolate workloads with farms, fleets, and queues. For the end-to-end steps to bring a vendor onto your farm, see Onboard users to your farm.

A restricted fleet for sensitive content

To keep work under a security or privacy restriction on dedicated workers, create a queue and a fleet for that work, and associate the fleet only with that queue. Jobs are scheduled only to fleets associated with their queue, so a fleet associated with a single queue runs that queue's jobs and no others. See Associate a queue and fleet. For work that must not share anything with the rest of the studio, use a separate farm instead.

Complete the boundary around the restricted queue:

- Grant access only to approved users and groups. See Managing users in Deadline Cloud.
- Use a dedicated job attachments Amazon S3 bucket or root prefix, with a queue role scoped to it. See the section called “Job attachment buckets”.
- Run jobs as an operating system user that no other queue uses. See Run jobs as dedicated OS users.
- Limit who can change the fleet's queue–fleet associations. See the section called “Policy to manage queue–fleet associations for a specific fleet”.

For more information about the security considerations behind these steps, see Isolate workloads with farms, fleets, and queues.

Separate queues for service-managed and customer-managed fleets

If your farm has both service-managed and customer-managed fleets, in most cases create a separate queue for each fleet type. Jobs written for the two fleet types usually assume different run environments. A queue for service-managed fleets typically uses the conda queue environment

to install applications for each job, while jobs for a customer-managed fleet expect the software that you preinstalled on your worker hosts. A job configured for one environment often fails on the other. Keeping the queues separate also keeps each queue's environment settings simple. For more information, see Choose between service-managed and customer-managed fleets and Default conda queue environment.

Deadline Cloud farms

With a Deadline Cloud farm, you can manage users and project resources. A *farm* is a where your project resources are located. Your farm consists of queues and fleets. A *queue* is where submitted jobs are located and scheduled to be rendered. A *fleet* is a group of worker nodes that run tasks to complete jobs. After you create a farm, you can create queues and fleets to meet your project's needs.

Create a farm

1. From the Deadline Cloud console, choose **Go to Dashboard**.
2. In the Farms section of the Deadline Cloud dashboard, choose **Actions** → **Create farm**.
   - Alternatively, in the left side panel choose **Farms and other resources**, then choose **Create Farm**.
3. Add a **Name** for your farm.
4. For **Description**, enter the farm description. A clear description can help you quickly identify your farm's purpose.
5. (Optional) By default, your data is encrypted with a key that AWS owns and manages for your security. You can choose **Customize encryption settings (advanced)** to use an existing key or to create a new one that you manage.

   If you choose to customize encryption settings using the checkbox, enter a AWS KMS ARN, or create a new AWS KMS by choosing **Create new KMS key**.
6. (Optional) For **Cost scale factor**, enter a value to adjust how costs are displayed in the usage explorer and budget manager. Values less than 1 represent discounts, values greater than 1 represent premiums, and 1 (the default) leaves costs unchanged. For more information, see Cost scale factor.
7. (Optional) Choose **Add new tag** to add one or more tags to your farm.
8. Choose **Create farm**. After creation, your farm displays.

Deadline Cloud queues

A queue is a farm resource that manages and processes jobs.

To work with queues, you should already have a monitor and farm set up.

Topics

Create a queue

1. From the Deadline Cloud console dashboard, select the farm that you want to create a queue for.
   - Alternatively, in the left side panel choose **Farms and other resources**, then select the farm you want to create a queue for.
2. In the **Queues** tab, choose **Create queue**.
3. Enter a name for your queue.
4. For **Description**, enter the queue description. A description helps you identify your queue's purpose.
5. For **Job attachments**, you can either create a new Amazon S3 bucket or choose an existing Amazon S3 bucket.
   a. To create a new Amazon S3 bucket
      i. Select **Create new job bucket**.
      ii. Enter a name for the bucket. We recommend naming the bucket `deadlinecloud-job-attachments-[MONITORNAME]`.
      iii. Enter a **Root prefix** to define or change your queue's root location.
   b. To choose an existing Amazon S3 bucket
      i. Select **Choose an existing S3 bucket** > **Browse S3**.
      ii. Select the S3 bucket for your queue from the list of available buckets.

6. (Optional) To associate your queue with a customer-managed fleet, select **Enable association with customer-managed fleets**.
7. If you enable association with customer-managed fleets, you must complete the following steps.

   > **Important**
   >
   > We strongly recommend specifying users and groups for run-as functionality. If you don't, it will degrade your farm's security posture because the jobs can then do everything the worker's agent can do. For more information about the potential security risks, see Run jobs as dedicated OS users.

   a. For Run as user:

      To provide credentials for the queue's jobs, select **Queue-configured user**.

      Or, to opt out of setting your own credentials and run jobs as the worker agent user, select **Worker agent user**.

   b. (Optional) For Run as user credentials, enter a user name and group name to provide credentials for the queue's jobs.

      If you are using a Windows fleet, you must create an AWS Secrets Manager secret that contains the password for the Run as user. If you don't have an existing secret with the password, choose **Create secret** to open the Secrets Manager console to create a secret. For more information, see Manage access to Windows job user secrets in the *Deadline Cloud Developer Guide*.
8. Requiring a budget helps manage costs for your queue. Select either **Don't require a budget** or **Require a budget**.
9. Your queue requires permission to access Amazon S3 on your behalf. You can create a new service role or use an existing service role. If you don't have an existing service role, create and use a new service role.

   a. To use an existing service role, select **Choose a service role**, and then select a role from the dropdown.

   b. To create a new service role, select **Create and use a new service role**, and then enter a role name and description.

10. (Optional) To add environment variables for the queue environment, choose **Add new environment variable**, and then enter a name and value for each variable you add.
11. (Optional) Choose **Add new tag** to add one or more tags to your queue.
12. To create a default conda queue environment, keep the checkbox selected. To learn more about queue environments, see Create a queue environment. If you are creating a queue for a customer-managed fleet, clear the checkbox.
13. Choose **Create queue**.

Create a queue environment

A queue environment is a set of environment variables and commands that set up fleet workers. You can use queue environments to provide software applications, environment variables, and other resources to jobs in the queue.

When you create a queue, you have the option of creating a default conda queue environment. This environment provides service-managed fleets access to packages for partner DCC applications and renderers. The default environment For more information, see Default conda queue environment.

You can add queue environments using the console, or by editing the json or YAML template directly. This procedure describes how to create an environment with the console.

1. To add a queue environment to a queue, navigate to the queue and select the **Queue environments tab**.
2. Choose **Actions**, then **Create new with form**.
3. Enter a name and description for the queue environment.
4. Choose **Add new environment variable**, and then enter a name and value for each variable you add.
5. (Optional) Enter a priority for the queue environment. The priority indicates the order that this queue environment will run on the worker. Higher priority queue environments will run first.
6. Choose **Create queue environment**.

Default conda queue environment

When you create a queue associated with a service-managed fleet, you have the option of adding a default queue environment that supports conda to download and install packages in a virtual environment for your jobs. For more information, see conda documentation on the conda website.

If you add a default queue environment with the Deadline Cloud console, the environment is created for you. If you add a queue another way, such as the AWS CLI or with CloudFormation, you'll need to create the queue environment yourself. To ensure you have the correct contents for the environment, you can refer to the queue environment template YAML files. For the contents of the default queue environment, see the default queue environment YAML file on the GitHub website.

There are other queue environment templates available on the GitHub website that you can use as a starting point for your own needs.

Conda provides packages from *channels*. A channel is a location where packages are stored. Deadline Cloud provides a channel, `deadline-cloud`, that hosts conda packages that support partner DCC applications and renderers. Select each tab below to view the available packages for Linux or Windows.

Linux

- Autodesk Arnold for Cinema 4D
  - `cinema4d-c4dtoa=2025`
- Autodesk Arnold for Maya
  - `maya-mtoa=2024.5.3`
  - `maya-mtoa=2025.5.4`
  - `maya-mtoa=2026.5.5`
  - `maya-mtoa=2027.5.6`
- Autodesk Maya
  - `maya=2024`
  - `maya=2025`
  - `maya=2026`
  - `maya=2027`
  - `maya-openjd`

- Autodesk VRED
  - `vredcore=2025`
  - `vredcore=2026`
- Blender
  - `blender=3.6`
  - `blender=4.2`
  - `blender=4.5`
  - `blender=5.0`
  - `blender=5.1`
  - `blender-openjd`
- Chaos V-Ray for Maya
  - `maya-vray=2025.7`
  - `maya-vray=2026.7`
  - `maya-vray=2027.7`
- Foundry Nuke
  - `nuke=15`
  - `nuke=16`
  - `nuke=17`
  - `nuke-openjd`
- Maxon Cinema 4D
  - `cinema4d=2025`
  - `cinema4d=2026`
  - `cinema4d-openjd`
- Maxon Redshift for Maya
  - `maya-redshift=2025.4`
  - `maya-redshift=2026.8`
- SideFX Houdini
  - `houdini=19.5`
  - `houdini=20.0`
  - `houdini=20.5`

- houdini=21.0
- houdini=22.0
- houdini-openjd

Windows

- Adobe After Effects
  - aftereffects=24.6
  - aftereffects=25.1
  - aftereffects=25.2
  - aftereffects=25.6
  - aftereffects=26.0
- Autodesk Arnold for Cinema 4D
  - cinema4d-c4dtoa=2025
  - cinema4d-c4dtoa=2026
- Maxon Cinema 4D
  - cinema4d=2024
  - cinema4d=2025
  - cinema4d=2026
  - cinema4d-openjd
- Unreal Engine
  - unrealengine=5.4
  - unrealengine=5.5
  - unrealengine=5.6
  - unrealengine=5.7
  - unrealengine=5.8
  - unrealengine-openjd

Note

For **Cinema 4D**, the Linux conda package does not support substance 3D materials. Jobs with this material fail with one of the following errors:

```
Commandline: ./modules/io_substance/source/substance_framework/src/details/
detailsengine.cpp:794:
 SubstanceAir::Details::Engine::Context::Context(SubstanceAir::Details::Engine&,
 SubstanceAir::RenderCallbacks*): Assertion `res==0' failed.
```

```
/home/job-user/.conda/envs/<hash>/Lib/deadline/cinema4d_adaptor/Cinema4DAdaptor/
adaptor.sh: line 44: 10832 Segmentation fault      (core dumped) $C4DEXE
 ${ARGS[*]}
```

We recommend that you submit jobs with substance materials to Windows instead.
In Cinema 4D 2025.3.3 on Linux, globalized asset paths can cause segmentation faults. Therefore, the Linux conda package contains Cinema 4D 2025.3.1 with Redshift 2025.6.0 instead. If you need features or bug fixes from Cinema 4D 2025.3.3, we recommend two options: upgrade to Cinema 4D 2026 or submit those jobs to Windows instead.
For **Cinema 4D OpenJD,** to prevent any timeout issues, we recommend you set task run timeouts to double their expected render time, instead of using the default 2 day timeout.

When you submit a job to a queue with the default conda environment, the environment adds two parameters to the job. These parameters specify the conda packages and channels to use to configure the job's environment before tasks are processed. The parameters are:

- `CondaPackages` – a space-separated list of package match specifications, such as `blender=3.6` or `numpy>1.22`. The default is empty to skip creating a virtual environment. For more information, see Package match specifications on the conda website.
- `CondaChannels` – a space separated list of conda channels such as `deadline-cloud`, `conda-forge`, or `s3://amzn-s3-demo-bucket/conda/channel`. The default is `deadline-cloud`, a channel available to service-managed fleets that provides partner DCC applications and renderers. For more information, see Channels on the conda website.

When you use an integrated submitter to send a job to Deadline Cloud from your DCC, the submitter populates the value of the `CondaPackages` parameter based on the DCC application

and submitter. For example, if you are using Blender the `CondaPackage` parameter is set to `blender=3.6.* blender-openjd=0.4.*`.

We recommend you pin any submissions to only the versions listed in the table above, for example blender=3.6. Pinning to the major.minor version is recommended because patch releases affect the available packages. For example, when we release Blender 3.6.17, we will no longer distribute Blender 3.6.16. Any submissions pinned to blender=3.6.16 will fail. If you pin to blender=3.6, then you will get the latest distributed patch version and jobs will not be impacted. By default, the DCC submitters pin to the current versions listed in the table above, excluding the patch number, such as blender=3.6.

Associate a queue and fleet

To process jobs, you must associate a queue with a fleet. You can associate a single fleet with multiple queues and a single queue with multiple fleets. When you associate a fleet with multiple queues, it divides its workers evenly among them. Similarly, when you associate a queue with multiple fleets, it distributes jobs evenly across those fleets.

> **Note**
>
> To use wait and save, we recommend you associate your queue only with a fleet that uses wait and save instance types. If you associate your queue with more than one fleet, and any of those fleets use spot or on-demand instance types, your fleet might not process your jobs with wait and save instances.

To associate an existing queue with an existing fleet, complete the following steps:

1. From your Deadline Cloud farm, select the **Queue** you want to associate with a fleet. The queue displays.
2. To select a fleet to associate with your queue, choose **Associate fleets**.
3. Choose the **Select fleets** dropdown. A list of available fleets displays.
4. From the list of available fleets, select the **checkbox** next to the fleet or fleets you want to associate with your queue.
5. Choose **Associate**. The fleet association status should now be **Active**.

Stop a queue fleet association

To stop a queue fleet association, complete the following steps:

1. From your queue, select the **Associated fleets** tab.
2. Select the checkbox for the fleet you want to stop associating with the queue.
3. From the Actions dropdown, select **Eventual stop** or **Immediate stop**.

   To finish processing jobs before the association stops, select Eventual stop. To immediately stop processing jobs, select Immediate stop.
4. In the confirmation window, enter **`confirm`** and then choose **Stop**.
5. (Optional) To disassociate the fleet from the queue, complete the following steps:

   a. Wait for the association status to change to **Stopped**.
   b. After the association has stopped, if you haven't already, select the checkbox for the fleet.
   c. From the Actions dropdown, select **Disassociate fleet**.
   d. In the confirmation window, choose **Disassociate**.

Reactivate a queue fleet association

To reactivate a queue fleet association, complete the following steps:

1. From your queue, select the **Associated fleets** tab.
2. Select the checkbox for the fleet you want to reactivate the queue fleet association.
3. From the Actions dropdown, choose **Start**. The association status changes to Active.

Deadline Cloud fleets

This section explains how to manage service-managed fleets and customer-managed fleets (CMF) for Deadline Cloud.

You can set up two types of Deadline Cloud fleets:

- Service-managed fleets are fleets of workers that have default settings provided by Deadline Cloud. These default settings are designed to be efficient and cost effective.
- Customer-managed fleets (CMFs) provide you with full control over your processing pipeline. A CMF can reside within AWS infrastructure, on premises, or in a co-located data center. CMFs include provisioning, operations, management, and decommissioning workers in the fleet.

When you associate a fleet with multiple queues, it divides its workers evenly among those queues.

For more information about choosing between the two fleet types, see Choose between service-managed and customer-managed fleets.

Topics

Choose between service-managed and customer-managed fleets

When you set up compute for your farm, the first decision is who manages the workers: Deadline Cloud or you. A *service-managed fleet* (SMF) is a fleet of Amazon Elastic Compute Cloud (Amazon EC2) workers that Deadline Cloud provisions, scales, and maintains for you. A *customer-managed fleet* (CMF) is a fleet of workers that you provision and operate yourself. Workers can be Amazon EC2 instances, on-premises machines, or hardware in a co-located data center.

The following table compares the fleet types. The sections after the table link to detailed instructions for each.

Consideration	Service-managed fleet	Customer-managed fleet
Worker management	Deadline Cloud provisions, scales, and decommissions Amazon EC2 workers for you	You provision, operate, and decommission the workers
Where workers run	Amazon EC2 instances that Deadline Cloud manages	Amazon EC2 instances, on-premises machines, or a co-located data center
Operating systems	Linux (AL2023) or Windows Server 2022	Linux, Windows, or macOS
Software environment	Worker image maintained by Deadline Cloud; add software with the conda queue environment or host configuration scripts	You build the worker image and install the software and drivers you need
Scaling	Automatic, based on your fleet's auto scaling configuration	You set up auto scaling for Amazon EC2 workers; on-premises capacity is fixed
Licensing	For supported software, usage-based licensing (UBL) activates automatically; you can also connect to your own license server	Bring your own licenses, or set up a license endpoint for UBL
Storage	Job attachments, persistent storage volumes, and VPC resource endpoints to reach shared storage in your VPC	Any storage you configure, such as your existing network file system
Cost model	Pay for workers only while they process jobs	You pay for your own compute (Amazon EC2 instances or on-premises hardware). Deadline Cloud also charges for each hour a worker is connected to the farm

You don't have to choose only one fleet type. A farm can contain both, for example on-premises workers in a CMF plus cloud capacity in an SMF for peak demand. For more information, see Extend your on-premises render farm to the cloud in the *Deadline Cloud Developer Guide*.

Service-managed fleets

With a service-managed fleet, you choose the instance capabilities, operating system, and market option (spot, on-demand, or wait-and-save), and Deadline Cloud handles the rest. Associate the fleet with a queue that uses the default conda queue environment. Deadline Cloud then configures the workers with packages for supported digital content creation (DCC) applications and renderers.

For more information, see the following:

- Service-managed fleets
- Software licensing for service-managed fleets
- Persistent storage for service-managed fleets
- Understand the cost model for service-managed fleets
- Configure and use Deadline Cloud service-managed fleets in the *Deadline Cloud Developer Guide*

Customer-managed fleets

With a customer-managed fleet, you run the workers and Deadline Cloud acts as the repository and scheduler for your jobs. Each worker runs the Deadline Cloud worker agent and needs access to the Deadline Cloud service endpoint.

For more information, see the following:

- Create and use Deadline Cloud customer-managed fleets in the *Deadline Cloud Developer Guide*
- Set up auto scaling for customer-managed fleets in the *Deadline Cloud Developer Guide*
- Connect customer-managed fleets to a license endpoint in the *Deadline Cloud Developer Guide*

Service-managed fleets

A service-managed fleet (SMF) is a fleet of workers that have default settings provided by Deadline Cloud. These default settings are designed to be efficient and cost-effective.

Some of the default settings limit the amount of time that workers and tasks can run. A worker can only run for seven days and a task can only run for five days. When the limit is reached, the task or worker stops. If this stop happens, you might lose work that worker or task was running. To avoid

this, monitor your workers and tasks to ensure they don't exceed the maximum duration limits. To learn more about monitoring your workers, see Using the Deadline Cloud monitor.

Create a service-managed fleet

There are 3 types of instance options you can choose for your service-managed fleet; spot, on-demand, and wait-and-save. Spot instances are unreserved capacity that you can use at a discounted price, but might be interrupted by on-demand requests. On-demand instances are priced by the second, have no long-term commitment, and will not be interrupted. Wait-and-save provides delayed job scheduling for reduced cost and can be interrupted by on-demand and spot requests.

1. From the Deadline Cloud console, navigate to the farm you want to create the fleet in.
2. Select the **Fleets** tab, and then choose **Create fleet**.
3. Enter a **Name** for your fleet.
4. (Optional) Enter a **Description**. A clear description can help you quickly identify your fleet's purpose.
5. Select **Service-managed** fleet type.
6. Choose the **Spot**, **On-demand**, or **Wait and Save** instance market option for your fleet. By default, fleets use the Spot option.
7. For service access for your fleet, select an existing role or create a new role. A service role provides credentials to instances in the fleet, granting them permission to process jobs, and to users in the monitor so that they can read log information.
8. Choose **Next**.
9. Choose between CPU only instances or GPU accelerated instances. GPU accelerated instances may be able to process your jobs faster, but can be more expensive.
10. Select the operating system for your workers. You can leave the default, **Linux** or choose **Windows**.
11. (Optional) If you selected GPU accelerated instances, set the maximum and minimum number of GPUs in each instance. For testing purposes you are limited to one GPU. To request more for your production workloads, see Requesting a quota increase in the *Service Quotas User Guide*.
12. Enter the minimum and maximum **vCPUs** that you require for you fleet.
13. Enter the minimum and maximum **memory** that you require for you fleet.
14. (Optional) You can choose to allow or exclude specific instance types from your fleet to ensure only those instance types are used for this fleet.

15. (Optional) Set the maximum number of instances to scale the fleet so that capacity is available for the jobs in the queue. We recommend that you leave the minimum number of instances at **0** to ensure the fleet releases all instances when no jobs are queued.
16. Choose **Next**.
17. Under **Storage capabilities**, choose a **Storage mode** for your fleet:

    - **Persistent storage** (Recommended) – Preserves cached data across worker lifecycle events, eliminating cold-start delays by maintaining application caches, packages, and workspaces. Additional Amazon Elastic Block Store (Amazon EBS) storage charges apply.
    - **Root storage only** – Does not cache data across worker lifecycle events. Best for jobs with minimal dependencies or fast startup times.

    Configure the **Root storage** settings (Size, IOPS, and Throughput) for the boot volume. If you chose **Persistent storage**, also configure the persistent volume settings (Size, Mount path, Throughput, Max idle time, and IOPS). For more information about persistent storage, see Persistent storage for service-managed fleets.
18. Choose **Next**.
19. (Optional) Define custom worker capabilities that define features of this fleet that can be combined with custom host capabilities specified on job submissions. One example is a particular license type if you plan to connect your fleet to your own license server.
20. Choose **Next**.
21. (Optional) To associate your fleet with a queue, select a **queue** from the dropdown. If the queue is set up with the default conda queue environment, your fleet is automatically provided with packages that support partner DCC applications and renderers. For a list of provided packages, see Default conda queue environment.
22. Choose **Next**.
23. (Optional) To add a tag to your fleet, choose **Add new tag**, and then enter the **key** and **value** for that tag.
24. Choose **Next**.
25. Review your fleet settings, and then choose **Create fleet**.

Update a fleet configuration

You can change a fleet's configuration from the Deadline Cloud console. For example, you can change the fleet's instance types and worker capabilities. You can also use the UpdateFleet API operation.

An update doesn't replace running workers. Workers that are already running when you update the fleet keep their original instance type and capabilities. They keep this configuration until they scale in, and idle workers scale in when the fleet is above its minimum worker count. Deadline Cloud can schedule jobs that you submit shortly after an update on these existing workers. As a result, the new configuration might not take effect immediately.

To make sure that all workers use the new configuration, drain the fleet first:

1. Set the fleet's maximum worker count to 0.
2. Use the `ListWorkers` API operation to confirm that the fleet has no workers.
3. Restore the fleet's maximum worker count. New workers launch with the updated configuration.

Use a GPU accelerator

You can configure worker hosts in your service-managed fleets to use one or more GPUs to accelerate processing your jobs. Using an accelerator can reduce the time that it takes to process a job, but can increase the cost of each worker instance. You should test your workloads to understand the trade offs between a fleet using GPU accelerators and fleets that don't.

GPUs are not available for fleets with wait-and-save instances.

> **Note**
>
> For testing purposes you are limited to one GPU. To request more for your production workloads, see Requesting a quota increase in the *Service Quotas User Guide*.

You decide whether your fleet will use GPU accelerators when you specify the worker instance capabilities. If you decide to use GPUs, you can specify the minimum and maximum number of GPUs for each instance, the types of GPU chips to use, and the runtime driver for the GPUs.

Instance capabilities are also how you guarantee the amount of memory each worker has. A step's `amount.worker.memory` host requirement selects compatible fleets and doesn't launch larger instances. For more information about how host requirements and fleet configuration interact, see Host requirements and fleet capabilities in the *Deadline Cloud Developer Guide*.

The available GPU accelerators are:

- `T4` - NVIDIA T4 Tensor Core GPU
- `A10G` - NVIDIA A10G Tensor Core GPU
- `L4` - NVIDIA L4 Tensor Core GPU
- `L40s` - NVIDIA L40S Tensor Core GPU
- `RTX PRO Server 6000` - NVIDIA RTX PRO 6000 Blackwell Server Edition GPU

You can choose from the following runtime drivers:

- `Latest` - Use the latest runtime available for the chip. If you specify `latest` and a new version of the runtime is released, the new version of the runtime is used.
- `grid:r580` - NVIDIA vGPU software 19 on the NVIDIA website (Long-Term Support).
- `grid:r570` - NVIDIA vGPU software 18 on the NVIDIA website. Scheduled for removal on July 12, 2026.
- `grid:r535` - NVIDIA vGPU software 16 on the NVIDIA website (Long-Term Support). Scheduled for removal on August 5, 2026.
- `grid:r550` (deprecated) - NVIDIA vGPU software 17 on the NVIDIA website.

Deadline Cloud uses `latest` as the default runtime driver for all accelerators. If you specify a runtime driver for some accelerators, you must specify one for all of them. Mixing explicit values with blank values returns an error.

GPU driver lifecycle

Deadline Cloud supports NVIDIA Long-Term Support (LTS) branch drivers for service-managed fleets. Deadline Cloud plans to continue supporting LTS branches as NVIDIA releases them. For more information about the NVIDIA vGPU software lifecycle, see the NVIDIA vGPU Software Lifecycle Policy on the NVIDIA website.

When NVIDIA reaches end-of-life for a driver branch, Deadline Cloud deprecates that driver and eventually removes it. If you use a pinned driver version, migrate your fleets to `latest` or to a newer supported driver before the removal date.

The following table shows the current driver support status.

GPU runtime driver lifecycle

Runtime driver	NVIDIA vGPU software	Branch type	Status	NVIDIA end-of-life
`grid:r580`	vGPU 19	Long-Term Support	Active	July 2028
`grid:r570`	vGPU 18	Production	Deprecating; scheduled for removal July 12, 2026	March 2026 (reached)
`grid:r535`	vGPU 16	Long-Term Support	Deprecating; scheduled for removal August 5, 2026	July 2026
`grid:r550`	vGPU 17	Production	Deprecated	Not available

Persistent storage for service-managed fleets

AWS Deadline Cloud (Deadline Cloud) persistent storage provides dedicated Amazon Elastic Block Store (Amazon EBS) gp3 volumes, separate from the root boot volume, for service-managed fleet (SMF) workers. These volumes preserve data across worker lifecycle events. With persistent storage, conda package installations, application caches, and asset files remain available when workers are replaced during routine maintenance or scaling operations.

How persistent storage works

When you enable persistent storage on a service-managed fleet, Deadline Cloud automatically manages the lifecycle of Amazon EBS volumes for your workers:

1. When a worker launches, Deadline Cloud creates or reuses an available Amazon EBS volume in the same Availability Zone and attaches the volume to the worker.
2. Deadline Cloud formats the volume (if new) and mounts it at the path you specify in the fleet configuration.
3. When the worker terminates or is replaced, Deadline Cloud detaches the volume and makes it available for reuse by a future worker in the same fleet and Availability Zone.

Because volumes are reused within the same fleet and Availability Zone, subsequent workers benefit from data that was previously written to the volume. The volume provides dedicated bandwidth and IOPS without contention between workers.

> **Note**
>
> Persistent storage is available only for service-managed fleets. For customer-managed fleets, you manage your own storage infrastructure.

Benefits of persistent storage

Persistent storage provides the following benefits for service-managed fleet workloads:

- **Faster job startup** – Conda package installations, compiled shaders, and processed assets persist across worker replacement, eliminating repeated downloads and installations.
- **Dedicated performance** – Each worker receives its own Amazon EBS volume with dedicated IOPS and throughput, avoiding the contention that occurs with shared network storage.
- **Automatic management** – Deadline Cloud handles volume creation, attachment, formatting, mounting, and cleanup without requiring manual intervention.
- **Runtime integration** – Supported runtime consumers such as conda queue environments and the virtual file system (VFS) immutable cache automatically use persistent storage when available, without requiring changes to your job configuration.
- **Cost control** – Configure a time-to-live (TTL) to automatically clean up unused volumes and reduce storage costs during idle periods.

When to use persistent storage

Consider enabling persistent storage for your service-managed fleet in the following scenarios:

- Your jobs use conda packages that require significant download and installation time.
- Your rendering workloads compile shaders or process assets that can be reused across subsequent renders.
- You use Perforce or other version control systems where workspace sync state reduces data transfer on subsequent updates.
- Your jobs use the virtual file system (VFS) and would benefit from a persistent immutable asset cache.
- You want dedicated storage performance without the operational overhead of managing shared network file systems.
- You install custom renderers or other software on workers through host configuration scripts and want those installations cached to the persistent volume.

Configuring persistent storage for a fleet

You can configure persistent storage when you create a new service-managed fleet or update an existing fleet.

Configuring persistent storage (console)

Before you begin, you must have an existing farm with at least one service-managed fleet, or be ready to create a new fleet.

To configure persistent storage for a fleet

1. Sign in to the AWS Management Console and open the Deadline Cloud console.
2. In the navigation pane, choose **Farms**, and then select your farm.
3. Choose the **Fleets** tab, and then choose **Create fleet**, or select an existing service-managed fleet and choose **Edit**.
4. Under **Storage capabilities**, for **Storage mode**, choose **Persistent storage**.
5. Configure the **Root storage** settings for the boot volume (Size, IOPS, and Throughput).
6. Under **Persistent storage**, configure the following settings:

    - **Size** – The size of the persistent volume. The valid range is 1–65,536 GiB. Verify that the default size is suitable for your render workloads, and consider increasing the volume size for workflows that use larger assets or caches.

- **Mount path** – The absolute path where the volume is mounted on the worker (for example, `/mnt/persistent` for Linux). For Windows workers, specify a drive letter such as `D:`.
- **Throughput** – The provisioned throughput for the volume. Valid range is 125–2,000 MiB/s.
- **Max idle time** – How long an available volume can sit idle before it is deleted. Select a value from the dropdown (for example, 12 hours).
- **IOPS** – The provisioned IOPS for the volume. Valid range is 3,000–80,000 IOPS. IOPS must be at least 4× throughput.

7. Complete the remaining fleet configuration steps and choose **Create fleet** or **Save changes**.

Configuring persistent storage (AWS CLI)

To configure persistent storage using the AWS Command Line Interface (AWS CLI), include the `persistentVolumeConfiguration` object in the `serviceManagedEc2FleetConfiguration` when you call `CreateFleet` or `UpdateFleet`.

The `persistentVolumeConfiguration` object accepts the following parameters. For valid ranges and default values, see CreateFleet in the *Deadline Cloud API Reference*.

Parameter	Type	Required	Description
`sizeGiB`	Integer	No	The size of the persistent volume in GiB.
`iops`	Integer	No	The provisioned IOPS for the gp3 volume.
`throughputMiB`	Integer	No	The provisioned throughput in MiB/s for the gp3 volume.
`mountPath`	String	Yes	The mount location on the worker. For Linux, specify an absolute path (for example, `/mnt/persistent`). For Windows, specify a drive letter (for example, `D:`).

Parameter	Type	Required	Description
`lastUsedTtlHours`	Integer	No	The number of hours after a volume was last used before automatic cleanup.

The following AWS CLI example creates a service-managed fleet with persistent storage enabled:

```
aws deadline create-fleet \
    --farm-id farm-0123456789abcdef0 \
    --display-name "Rendering Fleet" \
    --max-worker-count 20 \
    --configuration '{
      "serviceManagedEc2FleetConfiguration": {
        "instanceCapabilities": {
          "vCpuCount": {"min": 4, "max": 16},
          "memoryMiB": {"min": 16384, "max": 65536},
          "osFamily": "LINUX",
          "rootEbsVolume": {"sizeGiB": 250}
        },
        "instanceMarketOptions": {"type": "spot"},
        "persistentVolumeConfiguration": {
          "sizeGiB": 2048,
          "iops": 16000,
          "throughputMiB": 500,
          "mountPath": "/mnt/persistent",
          "lastUsedTtlHours": 168
        }
      }
    }'
```

To add persistent storage to an existing fleet, include the same `persistentVolumeConfiguration` object in an `update-fleet` call. To disable persistent storage, omit the `persistentVolumeConfiguration` object from the fleet configuration in an `UpdateFleet` call. Deadline Cloud automatically cleans up existing volumes when they are no longer attached to a worker.

Runtime integration

When persistent storage is successfully mounted on a worker, Deadline Cloud sets the `DEADLINE_PERSISTENT_MOUNT` environment variable to the configured mount path. This environment variable is available to all job processes running on the worker, including host configuration scripts and job template actions. The following runtime consumers automatically use persistent storage when the environment variable is present:

- **Conda queue environments** – Package installations are stored on the persistent volume, so subsequent workers reuse previously installed packages instead of downloading and installing them again.
- **Virtual file system (VFS) immutable cache** – The VFS stores its immutable asset cache on the persistent volume, so previously downloaded assets are available without re-downloading from Amazon Simple Storage Service (Amazon S3).

You can also reference the `DEADLINE_PERSISTENT_MOUNT` environment variable in your own Open Job Description job templates and scripts to store data that should persist across worker lifecycle events. The following example shows a step that writes output to persistent storage:

```
steps:
  - name: ProcessAssets
    script:
      actions:
        onRun:
          command: bash
          args:
            - "-c"
            - |
              CACHE_DIR="${DEADLINE_PERSISTENT_MOUNT}/my-app-cache"
              mkdir -p "$CACHE_DIR"
              # Your processing logic here
```

Managing persistent volumes

You can view and manage persistent volumes for your fleet using the Deadline Cloud console, AWS CLI, or API. The following operations are available:

- **List volumes** – View all persistent volumes associated with a fleet, including their state, size, and Availability Zone.

- **Get volume details** – Retrieve detailed information about a specific volume, including its current state, attachment status, and configuration.
- **Delete a volume** – Permanently delete an unattached persistent volume that is no longer needed. You cannot delete a volume that is currently attached to a worker.

The following AWS CLI example retrieves the details of a persistent volume:

```
aws deadline get-volume \
    --farm-id farm-0123456789abcdef0 \
    --fleet-id fleet-0123456789abcdef0 \
    --volume-id volume-0123456789abcdef0
```

The `state` field in the response indicates the current volume state. Possible values are:

- `AVAILABLE` – The volume is detached and ready for attachment.
- `IN_USE` – The volume is currently attached to a worker.
- `PENDING_CREATION` – The volume is being created.
- `PENDING_ATTACHMENT` – The volume is reserved for attachment to a worker.
- `PENDING_DELETION` – The volume is marked for deletion.

Updating persistent storage configuration

You can update the persistent storage configuration on an existing fleet. The following table describes how configuration changes affect existing persistent volumes:

Change	Behavior
Increase or decrease IOPS or throughput	Deadline Cloud applies changes before the next volume attachment.
Increase volume size	Deadline Cloud enlarges the volume before the next attachment and expands the file system automatically during worker startup.
Decrease volume size	Not supported. Amazon EBS volumes cannot be reduced in size.

Change	Behavior
Change mount path	Applies only to new workers. Existing workers retain their current mount path.
Remove persistent storage	Deadline Cloud marks existing volumes for deletion and cleans them up when they are no longer attached to a worker.

Important

Configuration changes do not affect existing workers. Changes apply only to new workers that launch after the update.

Encryption

Persistent volumes use the encryption settings configured at the farm level. If you configured a customer-managed AWS Key Management Service (AWS KMS) key for your farm, persistent volumes are encrypted with that key. Otherwise, persistent volumes are encrypted with a service-owned key.

Considerations

Keep the following considerations in mind when using persistent storage:

- Persistent volumes are a caching optimization, not durable primary storage. Use persistent volumes only for data that you can recreate, such as package installations, compiled shaders, and asset caches. Deadline Cloud can replace a volume at any time, and you can't access persistent volumes directly.
- Deadline Cloud configures the worker's home directory to use the persistent volume. Software that stores data in the home directory (such as conda packages and application caches) automatically benefits from persistence. If your software writes to paths outside the home directory, you must reconfigure it to use the persistent mount path, or those files do not persist across worker lifecycle events.
- Persistent volumes are not attached to multiple workers simultaneously. Each volume serves one worker at a time, but is reused by different workers across lifecycle events.

- Volumes are scoped to a specific fleet and Availability Zone. A volume created in one Availability Zone cannot be reused by a worker in a different Availability Zone.
- A specific worker isn't guaranteed to receive the same volume it used previously. Any available volume in the same fleet and Availability Zone can be assigned.
- If persistent storage cannot be provisioned (for example, due to quota limits), the job fails. Workers do not fall back to running without persistent storage.
- You are billed for persistent storage based on the number of active volumes and their configuration. To control costs during idle periods, configure a TTL or remove the persistent storage configuration from your fleet.

Software licensing for service-managed fleets

Deadline Cloud provides usage-based licensing (UBL) for commonly used software packages. Supported software packages are automatically licensed when they run on a service-managed fleet. You don't need to configure or maintain a software license server. Licenses scale so you won't run out for larger jobs.

You can install software packages that support UBL using the built-in Deadline Cloud conda channel, or you can use your own packages. For more information about the conda channel, see Create a queue environment.

For a list of supported software packages and information about pricing for UBL, see AWS Deadline Cloud pricing.

Bring your own license with service-managed fleets

With Deadline Cloud usage-based licensing (UBL), you don't need to manage separate license agreements with software vendors. However, if you have existing licenses or need to use software that isn't available through UBL, you can use your own software licenses with your Deadline Cloud service-managed fleets. Workers connect to your license server through a VPC resource endpoint, through port forwarding to an instance in your account, or over the internet. The license server can be in another VPC or AWS account, as long as the endpoint or forwarding instance has network access to it.

You can also combine both methods so workers use your existing licenses first and fall back to UBL when they run out.

For a comparison of the licensing options and setup instructions, see Using software licenses with Deadline Cloud in the *Deadline Cloud Developer Guide*.

VFX Reference Platform compatibility

The VFX Reference Platform is a common target platform for the VFX industry. To use the standard service-managed fleet Amazon EC2 instance running Amazon Linux 2023 with software that supports the VFX Reference Platform, review the following considerations when using a service-managed fleet.

The VFX Reference Platform is updated annually. These considerations for using an AL2023 including Deadline Cloud service-managed fleets are based on the calendar year (CY) 2022 through 2024 Reference Platforms. For more information, see VFX Reference Platform on the VFX Reference Platform website.

> **Note**
>
> If you are creating a custom Amazon Machine Image (AMI) for a customer-managed fleet, you can add these requirements when you prepare the Amazon EC2 instance.

To use VFX Reference Platform supported software on an AL2023 Amazon EC2 instance, consider the following:

- The glibc version installed with AL2023 is compatible for runtime use, but not for building software compatible with the VFX Reference Platform CY2024 or earlier.
- Python 3.9 and 3.11 are provided with the service-managed fleet making it compatible with VFX Reference Platform CY2022 and CY2024. Python 3.7 and 3.10 are not provided in the service-managed fleet. Software requiring them must provide the Python installation in the queue or job environment.
- Some Boost library components provided in the service-managed fleet are version 1.75, which is not compatible with the VFX Reference Platform. If your application uses Boost, you must provide your own version of the library for compatibility.
- Intel TBB update 3 is provided in the service-managed fleet. This version is compatible with VFX Reference Platform CY2022, CY2023, and CY2024.
- Other libraries with versions specified by the VFX Reference Platform are not provided by the service-managed fleet. You must provide the library with any application used on a service-

managed fleet. For a list of libraries, see the reference platform on the VFX Reference Platform website.

Worker AMI software contents

This section provides information on software installed on AWS Deadline Cloud service-managed worker Amazon Machine Images (AMIs).

AWS Deadline Cloud service-managed worker AMIs are based on both Windows Server 2022 and Amazon Linux 2023, and include additional software specifically installed to support rendering workloads. These AMIs are continuously updated to maintain functionality.

The software on these AMIs is organized into one of the following support categories:

Service-provided software packages

Software specifically installed and maintained for rendering workloads

Additional system software

All other software that might change without notice

Service-provided software packages

These software packages are installed to support rendering workloads and are maintained for compatibility. You can safely take dependencies on these packages.

Development Tools & Languages

Linux (AL2023):

- Python 3.11
- Git

Windows (Server 2022):

- Python 3.11
- Git for Windows

AWS tools

Both platforms:

- AWS Command Line Interface v2 (AWS CLI v2)

System libraries & utilities

Linux:

- FUSE and FUSE3 libraries for filesystem operations
- Image Libraries
  - libpng
  - libjpeg
  - libtiff
- OpenGL Libraries
  - mesa-libGLU
  - mesa-libGL
  - mesa-libEGL
  - libglvnd-opengl
- Development Libraries:
  - json-c (JSON parsing)
  - libnsl (network services library)
  - libxcrypt-compat (encryption compatibility)
- X Window Libraries
  - libXmu
  - libXpm
  - libXinerama
  - libXcomposite
  - libXrender
  - libXrandr
  - libXcursor
  - libXi

  - libxdamage
  - libXtst
  - libxkbcommon
  - libSM
- Network and system utilities
  - tcsh

GPU accelerated fleets

- Nvidia grid drivers

Package managers

Linux:

- conda/Mamba package manager (installed in `/opt/conda`)
- DNF package manager (system packages)
- pip (Python package installer)

Windows:

- conda/Mamba package manager (installed in `C:\ProgramData\conda`)
- pip (Python package installer)

Additional system software

All other software on the AMI can be updated, removed, or changed without notice. Do not take dependencies on any software not explicitly listed in the *Supported Software Packages* section above. This restriction includes but is not limited to:

- Operating system packages and libraries
- Service management components
- Base AMI software and drivers
- Software dependencies and runtime libraries
- System configuration tools and utilities

Additional system software examples

Linux: System packages such as systemd, kernel modules, hardware drivers, networking components, and the supporting libraries installed as part of the base AL2023 distribution.

Windows: Windows system components, Microsoft Edge, Amazon EC2 service software, hardware drivers, and Windows runtime components.

Best practices

Dependency management: Only take dependencies on software listed in the *Supported software packages* section.

Package Versions: For specific software versions, install specific packages using package managers (such as pip, conda, and more.) rather than relying on AMI-provided versions.

Environment Isolation: Use virtual environments (such as Python venv, and conda environments) to isolate your specific dependencies.

AMI update model

Note the following information about how the worker AMI updates.

- Worker AMIs are continuously updated with no versioning system.
- Updates occur automatically as part of the service operation.
- No advance notification system is provided for AMI updates.

Customer-managed fleets

When you want to use a fleet of workers that you manage, you can create a customer-managed fleet (CMF) that Deadline Cloud uses to process your jobs. Use a CMF when:

- You have existing on-premises workers to integrate with Deadline Cloud.
- You have workers in a co-located data center.
- You want direct control of Amazon Elastic Compute Cloud (Amazon EC2) workers.

When you use a CMF, you have full control over and responsibility for the fleet. This includes provisioning, operations, management, and decommissioning workers in the fleet.

For more information, see Create and use Deadline Cloud customer-managed fleets in the *Deadline Cloud Developer Guide*.

Auto scaling configuration

Deadline Cloud provides auto scaling configuration options that allow you to customize how your fleet scales workers up and down. These settings help you balance job processing speed with cost efficiency based on your workflow requirements.

You can configure the following auto scaling settings for your fleet:

- **Minimum worker count** – Specifies the minimum number of workers maintained in the fleet at all times.
- **Maximum worker count** – Limits how many workers can run simultaneously.
- **Scale out rate** – Controls how quickly workers are added to your fleet.
- **Worker idle duration** – Controls how long workers wait for new work before shutting down.
- **Standby worker count** – Maintains a warm standby pool of idle workers to start jobs fast.

How auto scaling works depends on your fleet type:

- **Service-managed fleets** – Deadline Cloud automatically implements auto scaling based on your configuration. You configure the settings and the service handles worker provisioning.
- **Customer-managed fleets** – If you have completed the auto scaling setup for your customer-managed fleet, the auto scaling configuration works the same as for service-managed fleets. The service uses the configuration to calculate desired capacity and sends recommended fleet size events to your fleet. For more information, see Set up auto scaling for customer-managed fleets in the *Deadline Cloud Developer Guide*.

Scale out rate

The **scale out rate** (`scaleOutWorkersPerMinute`) setting controls how many workers start launching per minute when your fleet scales out. The scale out rate determines your fleet's scale-up speed when a large job arrives. Workers take several minutes to launch. To maintain a warm pool of workers that can start jobs immediately, set the standby worker count. For more information, see Standby worker count.

Consider the following when configuring the scale out rate:

- A higher rate launches more workers quickly, which can reduce job completion time for large jobs.
- A higher rate may launch more workers than necessary for short-lived tasks, increasing costs.
- A lower rate can help detect job failures earlier and reduce costs from wasted compute on failing jobs.
- For short-lived tasks, a conservative scaling approach can be more cost-effective because workers spend less time loading environments relative to actual task execution.

> **Note**
>
> The scale out rate is a best-effort setting. Actual scaling speed may vary based on instance availability and other system factors. In rare conditions, the actual rate may briefly exceed the configured value.

Worker idle duration

The **worker idle duration** (`workerIdleDurationSeconds`) setting controls how long a worker remains available after it finishes processing a job, measured in seconds. The default value is 300 seconds (5 minutes).

This setting is useful for iterative workflows where artists frequently revise and resubmit jobs. By keeping workers available longer, subsequent job submissions can start processing immediately without waiting for new workers to launch.

Consider the following when configuring worker idle duration:

- A longer duration keeps workers available for rapid iteration, reducing wait times between job submissions. However, longer durations increase costs because idle workers continue to incur charges.
- A shorter duration reduces costs by shutting down idle workers more quickly.
- For service-managed fleets, the maximum value is 86,400 seconds (24 hours) because workers are refreshed every 24 hours. If a worker has been running for 23 hours and you set an idle duration of 10 hours, the worker shuts down after 1 hour when it reaches the 24-hour limit.

Standby worker count

The **standby worker count** (`standbyWorkerCount`) setting specifies the number of idle workers to maintain as a warm standby pool. These workers can process new jobs without the delay of launching new instances.

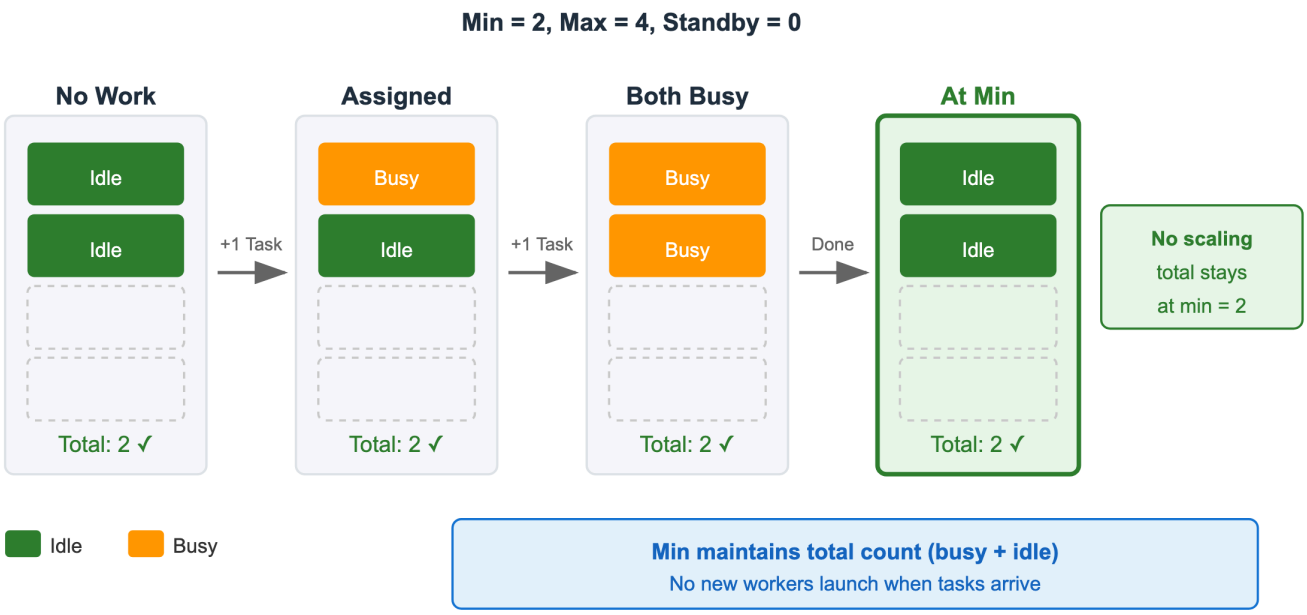
This setting is useful when you want to reduce job start latency. For example, standby workers are helpful when rendering with Windows instances, when using host configuration scripts that install local dependencies, or when workers require significant setup time. The fleet attempts to maintain the configured number of idle workers, but the idle count may temporarily drop while replacement workers are launching.

Consider the following when configuring standby worker count:

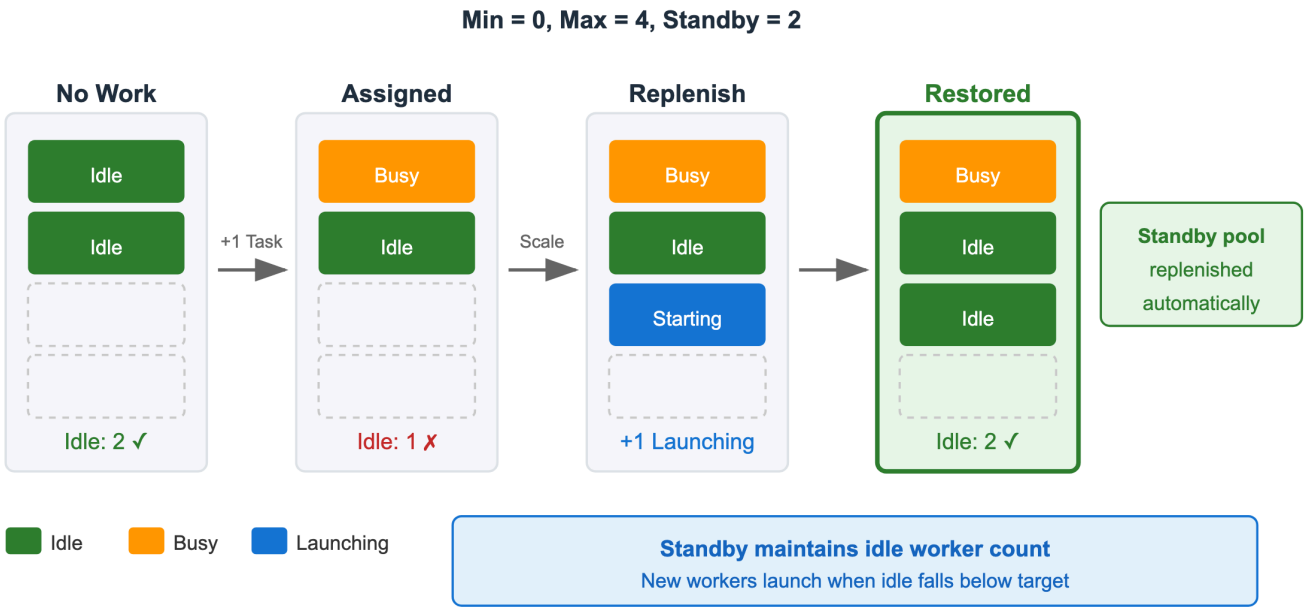
- Standby workers incur costs even when not processing jobs. Balance the number of standby workers against your budget and job start latency requirements.
- When the fleet reaches its maximum worker count, the standby pool may not be fully maintained. For example, if all workers are busy and the fleet is at its maximum size, no additional idle workers are launched.
- When the standby worker count exceeds the minimum worker count, the minimum worker count is effectively overridden. For example, with a minimum of 1 and a standby of 2, the fleet keeps 2 idle workers when no work is available, making the minimum setting redundant.

The following diagrams show how minimum worker count and standby worker count affect fleet scaling behavior. Choose a tab to view each scenario.

Minimum worker count



Standby worker count



To automatically adjust your standby worker count on a schedule, use the sample AWS CloudFormation (CloudFormation) template at fleet_standby_scheduling on the GitHub website.

Configuring auto scaling settings

You can configure auto scaling settings when you create a fleet or update an existing fleet.

To configure auto scaling settings

1. Open the Deadline Cloud console.
2. Navigate to the farm that contains your fleet.
3. Choose the **Fleets** tab.
4. Select the fleet you want to configure, then choose **Edit**.
5. In the **Auto scaling** section, configure the following settings:

   - **Minimum worker count** – Enter the minimum number of workers to maintain.
   - **Maximum worker count** – Enter the maximum number of workers allowed.
   - **Scale out rate** – Enter the number of workers to launch per minute.
   - **Worker idle duration** – Enter the number of seconds that workers remain idle before shutting down.
   - **Standby worker count** – Enter the number of standby workers to maintain.
6. Choose **Save changes**.

Managing users in Deadline Cloud

AWS Deadline Cloud uses AWS IAM Identity Center to manage users and groups. IAM Identity Center is a cloud-based single sign-on service that can be integrated with your enterprise single-sign on (SSO) provider. With integration, users can sign in with their company account.

Deadline Cloud enables IAM Identity Center by default, and signing in to the Deadline Cloud monitor requires it. You can also use Deadline Cloud without IAM Identity Center by calling the API or CLI with IAM credentials; see How permissions work in Deadline Cloud. An organization owner for your AWS Organizations is responsible for managing the users and groups that have access to your Deadline Cloud monitor. For more information, see What is AWS Organizations.

Where you create and manage users depends on your IAM Identity Center identity source: with the default IAM Identity Center directory, you create users in the Deadline Cloud console, and with an external identity provider such as Okta or Microsoft Entra ID, you create them in that system. For more information, see Understanding your identity source.

To bring a new artist, team, or vendor onto your farm, start with Onboard users to your farm.

Topics

How permissions work in Deadline Cloud

AWS Deadline Cloud (Deadline Cloud) controls access with two systems. *Access levels* control what you can see and do when you sign in to the Deadline Cloud monitor or submit jobs from your workstation. *AWS Identity and Access Management (IAM) policies* control what AWS credentials can do, such as administering farms in the Deadline Cloud console or calling the API from a pipeline script.

When you sign in to the monitor, AWS IAM Identity Center (IAM Identity Center) authenticates you, and your access levels govern what you can do. Your requests also carry your identity, so Deadline Cloud records who created or last updated each job. IAM policies govern programmatic access, such as pipeline automation and worker hosts. You can also use IAM credentials, for example to administer farms in the Deadline Cloud console, but Deadline Cloud attributes those requests to the IAM role or user, so jobs submitted with a shared role all show the same identity.

Who	Signs in with	Access controlled by	Where you grant it
People using the monitor or the integrated submitters, such as artists and researchers	The monitor URL, through IAM Identity Center	Access levels on farms, queues, and fleets	The Deadline Cloud console, on each resource's **Access management** tab
Administrators working in the Deadline Cloud console	The AWS Management Console	IAM policies	IAM
Pipeline scripts and automation that call the Deadline Cloud API or CLI with AWS credentials	An IAM role or user	IAM policies	IAM
Worker hosts in your fleets	The fleet's IAM role	Service roles on the fleet and queue	IAM, when you create the fleet or queue

Where you grant access levels

You grant each user or group one of four access levels: viewer, contributor, manager, and owner. Each level includes everything from the levels before it. For the exact permissions at each level, see the section called “What each access level allows” at the end of this topic.

You assign access levels on farms, queues, and fleets in the Deadline Cloud console. The Deadline Cloud documentation and API call each assignment a *membership*. A grant on the farm applies to every queue and fleet in the farm. A grant on a queue or fleet applies only to that resource, so you can give a team the contributor level on its own queue and the viewer level everywhere else.

A grant on a group applies to whoever is in the group when they make a request. When you add a user to the group in your identity source, that user gets the group's access without any change to the grant, and when you remove a user from the group, that user loses the group's access the same way. The group's membership on the farm, queue, or fleet stays in place; you don't update or refresh it as the group's members change.

When a user has grants at more than one level, Deadline Cloud applies the highest one. For example, a user with the viewer level on the farm and the manager level on one queue has manager permissions on that queue and viewer permissions everywhere else in the farm. A lower-level grant can't reduce a farm-level grant.

Access levels control what you can see and manage. Access levels don't decide where jobs run: the service schedules jobs based on queue–fleet associations and each job's host requirements. For example, a grant on a fleet lets users see the fleet and its workers; it doesn't route their jobs to that fleet. To divide your farm so that permissions and hardware both land where you want them, see Organize your farms, queues, and fleets.

If you have no membership on a farm, queue, or fleet, you can't see that resource in the monitor, even though you can sign in. If your only membership is on a queue or fleet, you don't see the farm in your farm list, but you can still work with that queue or fleet. To also control who can sign in to the monitor at all, see Restricting which users can access the monitor.

How access levels relate to IAM

Access levels are enforced through IAM, but you don't write IAM policies to use them. When you sign in to the monitor, IAM Identity Center authenticates you, and the monitor makes AWS requests on your behalf using the *monitor role*, an IAM role in your account. The AWS managed policies attached to the monitor role allow each operation based on your access level on the farm, queue, or fleet involved. Although these requests use an IAM role, they keep your identity, so jobs still record which person created or updated them. The monitor desktop application also shares these credentials with the Deadline Cloud CLI and the integrated submitters, so the same access levels apply there.

The credentials that the monitor issues are temporary. When you remove a user's access or disable the user in your identity source, the user can't sign in again or get new credentials, but credentials already issued keep working until they expire, at most 15 minutes later.

If you sign in through the monitor, you don't need an IAM user or policies of your own. To change what an access level allows, for example to let contributors cancel their own jobs, add a policy to the monitor role. For more information, see Monitor role in the *Deadline Cloud Developer Guide*.

When to grant IAM permissions instead

If you use AWS credentials directly, for example in a pipeline script or in the Deadline Cloud console, your permissions come from IAM policies, and access levels don't apply. Grant IAM permissions for:

- **Console administration** – Creating farms, queues, and fleets, and assigning access levels, are Deadline Cloud console operations that require IAM permissions. See the section called "Policy to access the console".
- **Pipeline automation** – A submission service or CI system calls the Deadline Cloud API with an IAM role scoped to the operations it needs. For example policies, see Identity-based policy examples for Deadline Cloud.
- **Worker hosts** – Workers use the fleet role and queue roles that you configure when creating those resources. For more information, see Service roles for Deadline Cloud in the *Deadline Cloud Developer Guide*.

For the full IAM reference for Deadline Cloud, including supported actions, resources, and condition keys, see Identity and Access Management in Deadline Cloud.

What each access level allows

The following tables list the permissions at each access level for each resource type that a grant can be placed on, with the default AWS managed policies. A grant on a farm also confers the queue and fleet permissions on every queue and fleet in that farm. Budgets and usage data appear only in the farm table. The owner level on a queue or fleet doesn't include them; you grant them with the owner level on the farm. To change what a level allows, see the section called "How access levels relate to IAM". To grant an access level, see the section called "Assign permissions".

Farm permissions by access level

Permission	Viewer	Contributor	Manager	Owner
View farm details	Yes	Yes	Yes	Yes

Permission	Viewer	Contributor	Manager	Owner
View queues and fleets	Yes	Yes	Yes	Yes
Submit jobs	No	Yes	Yes	Yes
Manage user access	No	No	Yes	Yes
View and create budgets	No	No	No	Yes
View usage data	No	No	No	Yes

Queue permissions by access level

Permission	Viewer	Contributor	Manager	Owner
View queue details	Yes	Yes	Yes	Yes
View jobs in queue	Yes	Yes	Yes	Yes
Submit jobs to queue	No	Yes	Yes	Yes
Edit and cancel jobs	No	No	Yes	Yes
Manage queue user access	No	No	Yes	Yes

Fleet permissions by access level

Permission	Viewer	Contributor	Manager	Owner
View fleet details	Yes	Yes	Yes	Yes
View workers in fleet	Yes	Yes	Yes	Yes
Manage fleet user access	No	No	Yes	Yes

Onboard users to your farm

Onboarding gives a new group of users access to a farm: members of your own team, an outside vendor or contractor, or another team at your organization. Every setup differs in the details, but the common steps are the same. The answers to three questions decide how the checklist applies:

- **Who manages their identity?** You create users in the Deadline Cloud console, or they already exist in your organization's identity provider.
- **What should they see?** The whole farm, one queue on a shared farm, or a separate farm of their own.
- **Who sets up their workstations?** Each user sets up their own workstation, or you prepare the workstations in advance.

Work through the following steps for each group that you onboard:

1. **Decide where their work goes** – Add a queue to a shared farm for a team inside your organization, or create a separate farm when the work must stay isolated, such as for a vendor or client. For how to choose, see Organize your farms, queues, and fleets. Associate each new queue with the fleets that will run its work. See Associate a queue and fleet.
2. **Create their sign-ins** – Monitor users come from AWS IAM Identity Center (IAM Identity Center). If your identity source is the IAM Identity Center directory, create the users from the Deadline Cloud console. If you use an external identity provider such as Okta or Microsoft Entra ID, create the users there instead. See Create and manage users with IAM Identity Center directory and Manage users with an external identity provider.

   > **Note**
   >
   > Monitor sign-ins cover the monitor and the integrated submitters. To onboard a pipeline developer or a tool that calls the Deadline Cloud API or CLI, provide AWS credentials instead. See Getting started with Deadline Cloud resources in the *Deadline Cloud Developer Guide*.

3. **Grant access** – Assign each user or group an access level on the farms, queues, and fleets that they'll use. Grants can be as open or as scoped as your organization works: many organizations grant everyone broad access across a shared farm, while others limit each group to its own queue. See How permissions work in Deadline Cloud.

4. **Set up their workstations** – Send each user who submits from a digital content creation (DCC) application a link to Set up your workstation. On that page, they install the submitter and the monitor desktop application, sign in with the monitor URL, and connect their DCC. If you use shared storage, create storage profiles so that file paths map correctly between workstations and workers. See Storage profiles in Deadline Cloud. For guidance on securing workstations, see Security best practices - workstations.
5. **Verify the access** – Have one user sign in and submit a test job. Confirm that the job completes and that the user sees the farms and queues that you intend.
6. **Plan the offboarding** – Decide up front how access ends. When you remove a user or group from IAM Identity Center, they can no longer sign in to the monitor or access farm resources. See Create and manage users with IAM Identity Center directory.

Onboard your own artists

You and your artists follow the checklist as written. Create a group for each department, show, or project, grant the group contributor access on its queue, and send each artist the link to Set up your workstation so they can set up their own workstation. The same steps apply to any group at your organization that submits work from workstations. If your organization manages workstations centrally, an administrator can install the submitter and the monitor on each workstation in advance. The monitor installer supports a silent install on Windows. Each user still signs in with their own account.

Onboard a vendor or contractor

For people outside your organization, isolation and lifecycle matter most. Create a separate farm for each vendor so that their users can't see the rest of your organization's work. See A farm for each vendor or client. For the security details of the farm boundary, see Isolate workloads with farms, fleets, and queues. To keep unassigned users from signing in to the monitor at all, enable the require assignments setting. See Restricting which users can access the monitor.

If your identity source is an external identity provider, vendor sign-ins are guest accounts in that provider, so coordinate account creation and removal with the team that manages it. Move files with job attachments instead of granting vendors access to your shared storage. See Job attachments in Deadline Cloud. To cap what a vendor's work can spend, set a budget on their queue. See Control costs with a budget. Agree on an end date for the engagement, and remove the vendor's users and groups when it passes.

Onboard an internal team

A team inside your organization usually doesn't need the vendor isolation steps. You can add a queue for the team to your shared farm and grant an existing group from your identity source contributor access on that queue and viewer access elsewhere. See One shared farm, one queue per team. If you track spending by team, set a budget on the team's queue.

Onboard other kinds of users

Some users don't need the full checklist:

- To onboard someone who manages the farm for you, such as a technical director, grant them manager or owner access so that they can grant permissions and create budgets. See How permissions work in Deadline Cloud.
- To onboard someone who watches and manages work without submitting it, such as a coordinator or supervisor, create a sign-in, grant viewer or manager access, and share the monitor URL. The web monitor requires no installation. See Share the Deadline Cloud monitor URL.
- To onboard a pipeline developer who builds job bundles, submitters, and integrations, provide AWS credentials and the CLI. See Getting started with Deadline Cloud resources in the *Deadline Cloud Developer Guide*.
- If you're building rendering into a product for your own end users, see Deadline Cloud Architecture Guidance in the *Deadline Cloud Developer Guide*.

Understanding your identity source

IAM Identity Center uses an identity source to define where users are managed. There are two types of identity sources:

IAM Identity Center directory

> This option is the default identity source. Users are created and managed directly within IAM Identity Center. You can create users through the Deadline Cloud console or the IAM Identity Center console. Users receive email invitations to join your organization, and passwords are managed within IAM Identity Center.

External identity provider (IdP)

Users are federated from an external system such as Okta, Microsoft Entra ID, or other SAML 2.0 identity providers. Users must be created in the external system first. The Deadline Cloud console cannot create users when an external IdP is configured, but you can assign permissions to existing users. Passwords are managed by the external IdP.

To check your identity source configuration or change it, see Manage your identity source in the IAM Identity Center User Guide.

Create and manage users with IAM Identity Center directory

If your identity source is set to IAM Identity Center directory, you can create and manage users and groups directly through the Deadline Cloud console. Users created in the console will receive email invitations from IAM Identity Center. After accepting the invitation, users can access the Deadline Cloud monitor.

> **Note**
>
> If your IAM Identity Center is connected to an external identity provider, you cannot create users through the Deadline Cloud console. See the section called "Manage users with external IdP" for information about managing users with an external IdP.

1. Sign in to the AWS Management Console and open the Deadline Cloud console. From the main page, in the **Get started** section, choose **Set up Deadline Cloud** or **Go to dashboard**.
2. In the left navigation pane, choose **User management**. By default, the **Groups** tab is selected.

Depending on the action to take, choose either the **Groups** tab or **Users** tab.

Groups

To create a group

1. Choose **Create group**.
2. Enter a group name. The name must be unique among groups in your IAM Identity Center organization.

To remove a group

1. Select the group to remove.
2. Choose **Remove**.
3. In the confirmation dialog, choose **Remove group**.

> **Note**
>
> You are removing the group from IAM Identity Center. Group members can no longer sign in to the Deadline Cloud or access farm resources.

Users

To add users

1. Choose the **Users** tab.
2. Choose **Add users**.
3. Enter the name, email address, and username for the new user.
4. (Optional) Choose one or more IAM Identity Center groups to add the new user to.
5. Choose **Send invite** to send the new user an email with instructions for joining your IAM Identity Center organization.

To remove a user

1. Select the user you to remove.
2. Choose **Remove**.
3. In the confirmation dialog, choose **Remove user**.

> **Note**
>
> You are removing the user from IAM Identity Center. The user can no longer sign in to the Deadline Cloud monitor or access farm resources.

Manage users with an external identity provider

If your IAM Identity Center is connected to an external identity provider (IdP) such as Okta or Microsoft Entra ID, users must be created and managed in that external system. The Deadline Cloud console cannot create new users when an external IdP is configured.

After users are created in your external IdP and synchronized to IAM Identity Center, you can assign them permissions to Deadline Cloud resources. See the section called “Assign permissions” for information about assigning permissions at the farm, queue, and fleet level.

For information about managing your external identity provider configuration, see Manage your identity source in the IAM Identity Center User Guide.

Restricting which users can access the monitor

To sign in to the Deadline Cloud monitor, a user needs two levels of access:

- **AWS IAM Identity Center (IAM Identity Center) application access** – Permission to sign in to the Deadline Cloud monitor application.
- **Deadline Cloud resource access** – Permission to view farms, queues, and other resources after signing in.

By default, IAM Identity Center application access is open to all users in your identity store. Deadline Cloud resource access is the primary access control layer. However, you can add a second layer of control by restricting who can sign in to the monitor application.

To restrict monitor sign-in, enable the **Require assignments** setting on the Deadline Cloud monitor application in IAM Identity Center. After you enable this setting, only users and groups that you explicitly assign to the application can sign in to the monitor.

To enable Require assignments for the monitor application

1. Open the IAM Identity Center console at https://console.aws.amazon.com/singlesignon.
2. In the navigation pane, choose **Applications**.
3. Select the Deadline Cloud monitor application.
4. In the application's assignment configuration, enable **Require assignments**.
5. Assign the specific users and groups who need access to the application.

Users and groups that you don't assign to the application can't sign in to the monitor, even if they exist in your IAM Identity Center identity store.

For more information about application assignments, see Assign user access to applications in the *IAM Identity Center User Guide*.

Assign permissions to users and groups

Use the Deadline Cloud console to assign access levels to users and groups at the farm, queue, or fleet level.

> **Note**
>
> Changes to access permissions might take up to 10 minutes to reflect in the system.

To navigate to access management

1. Sign in to the AWS Management Console and open the Deadline Cloud console.
2. In the left navigation pane, choose **Farms and other resources**.
3. Select the farm to manage. Choose the farm name to open the details page. You can search for the farm using the search bar.
4. (Optional) To manage a queue or fleet instead of the farm, choose the **Queues** or **Fleets** tab, and then choose the queue or fleet to manage.
5. Choose the **Access management** tab.

Depending on the action to take, choose either the **Groups** tab or **Users** tab.

Groups

To add groups

1. Select the **Groups** toggle.
2. Choose **Add group**.
3. From the dropdown, select the groups to add.
4. For the group access level, choose one of the following options:

- **Viewer**
- **Contributor**
- **Manager**
- **Owner**

5. Choose **Add**.

To remove groups

1. Select the groups to remove.
2. Choose **Remove**.
3. In the confirmation dialog, choose **Remove group**.

Users

To add users

1. To add a user, choose **Add user**.
2. From the dropdown, select the users to add.
3. For the user access level, choose one of the following options:

   - **Viewer**
   - **Contributor**
   - **Manager**
   - **Owner**

4. Choose **Add**.

To remove users

1. Select the user to remove.
2. Choose **Remove**.
3. In the confirmation dialog, choose **Remove user**.

Deadline Cloud jobs

A *job* is a set of instructions that AWS Deadline Cloud uses to schedule and run work on available workers. When you create a job, you choose the farm and queue to send the job to.

A *submitter* is a plugin for your digital content creation (DCC) application that manages creating a job in the interface of your DCC application. After you create the job, you use the submitter to send it to Deadline Cloud for processing.

The submitter creates an Open Job Specification (OpenJD) template that describes the job. At the same time it uploads your asset files to an Amazon Simple Storage Service (Amazon S3) bucket. To reduce upload time, the submitter only sends files that have changed since the last upload to Amazon S3. For more information about OpenJD, see the OpenJD specifications on the GitHub website.

You can also create a job in the following ways.

- From a terminal – for users submitting a job that are comfortable using the command line.
- From a script – for customizing and automating workloads.
- From an application – for when the user's work is in an application, or when an application's context is important.

For more information, see How to submit a job to Deadline Cloud in the *Deadline Cloud Developer Guide*.

A job consists of:

- *Priority* – The approximate order that Deadline Cloud processes a job in a queue. You can set the job priority between 0 and 100, jobs with a higher number priority are generally processed first. Jobs with the same priority are processed in the order received.
- *Steps* – Defines the script to run on workers. Steps can have requirements such as minimum worker memory or other steps that need to complete first. For example, a job can have one step that renders the frames of a scene, and a second step that uploads thumbnails of the finished frames to your project tracker after the render step completes. Each step has one or more tasks.
- *Tasks* – A unit of work sent to a worker to perform. A task is a combination of a step's script and parameters, such as a frame number, that are used in the script. The job is complete when all tasks are complete for all steps.

- *Environment* – Set up and tear down instructions shared by multiple steps or tasks.

Topics

Using a Deadline Cloud submitter

A *submitter* is a tool that integrates with your digital content creation (DCC) application so that you can send render jobs directly to Deadline Cloud without switching between applications or manually transferring files.

Submitters are available for many popular DCC applications. Installing a submitter adds Deadline Cloud specific options to your application's interface, typically in the render settings or export menu.

With a Deadline Cloud submitter you can:

- Configure render job parameters in your familiar DCC environment
- Submit jobs to Deadline Cloud without leaving your application
- Reduce the potential for errors associated with manual file transfers
- Save time by switching between applications less often

To find a submitter for your DCC application and instructions for installing it, see the Set up your workstation page.

If your application doesn't have a supported submitter, you can still run jobs for your application. There might be a sample job bundle available for it, or you can construct a simple submitter for the application's render CLI command. For more information, see Open Job Description (OpenJD) templates for Deadline Cloud in the *Deadline Cloud Developer Guide*.

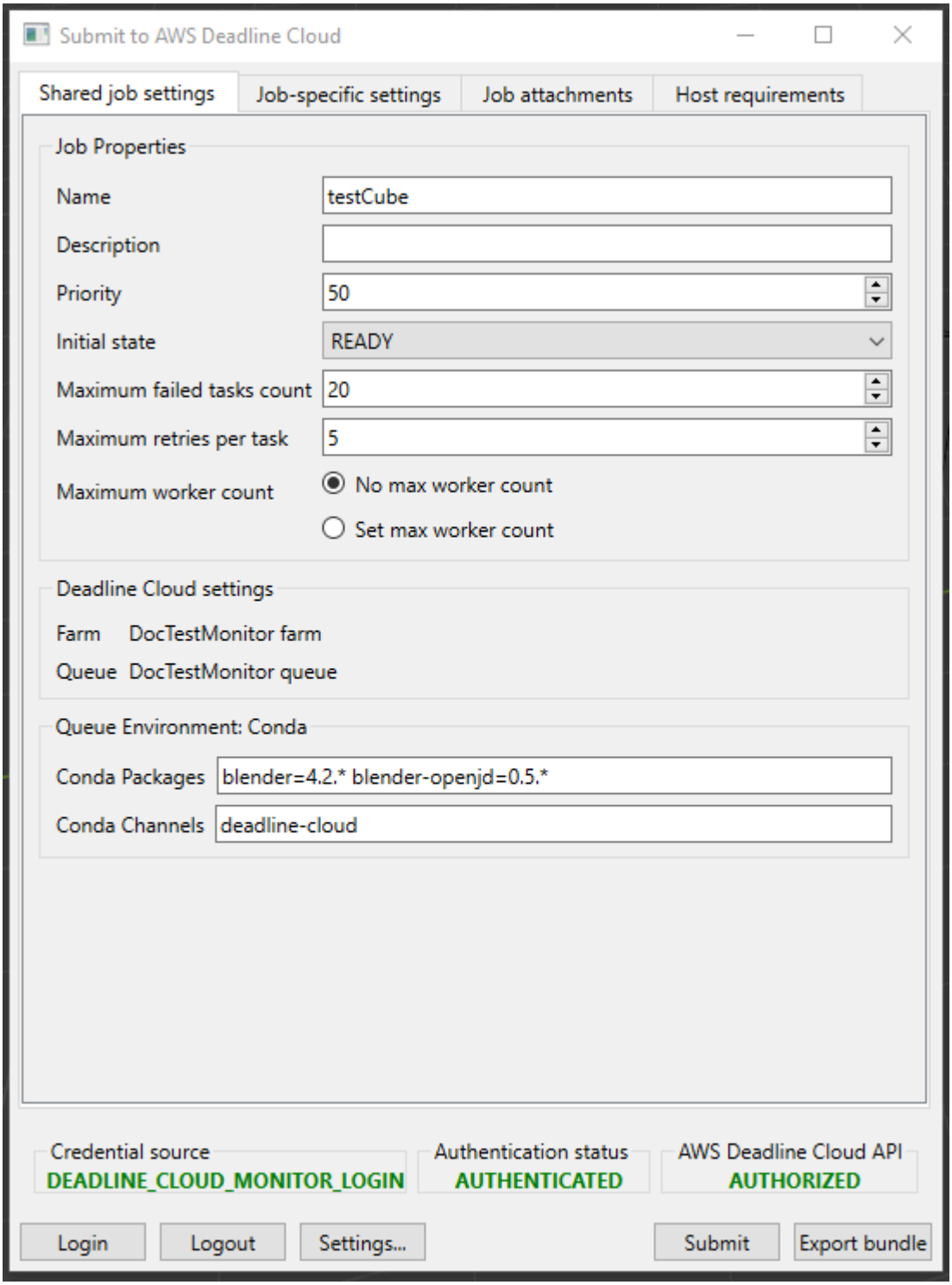
The examples in this topic use the Blender submitter, but the steps for using other submitters are similar.

> **Note**
>
> To use a submitter, you must be signed in to the Deadline Cloud monitor.

The submitter has four tabs.

Topics

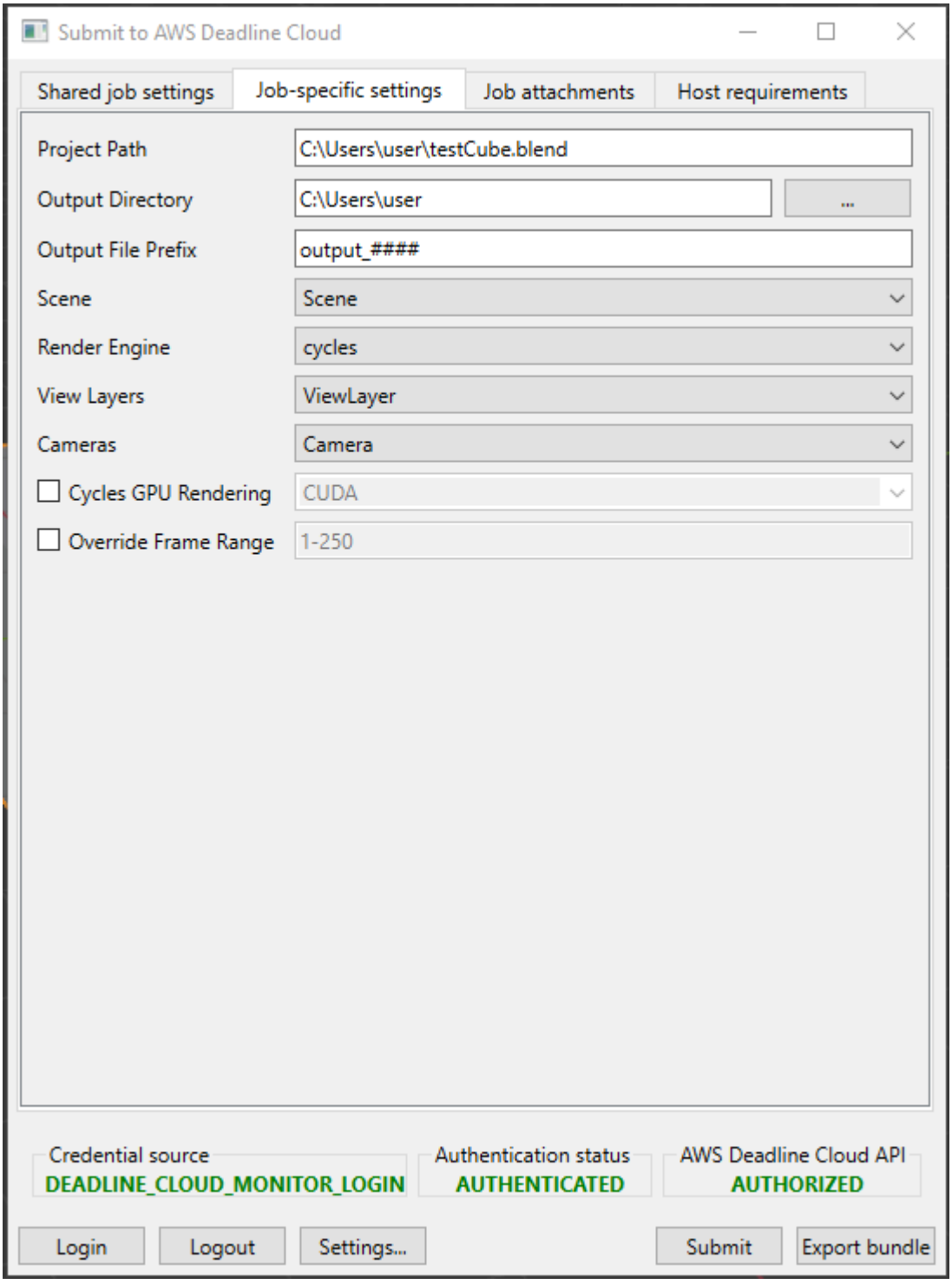
Shared job settings tab



The shared job settings tab contains the settings that are common to all jobs sent to Deadline Cloud using the submitter. The three sections are:

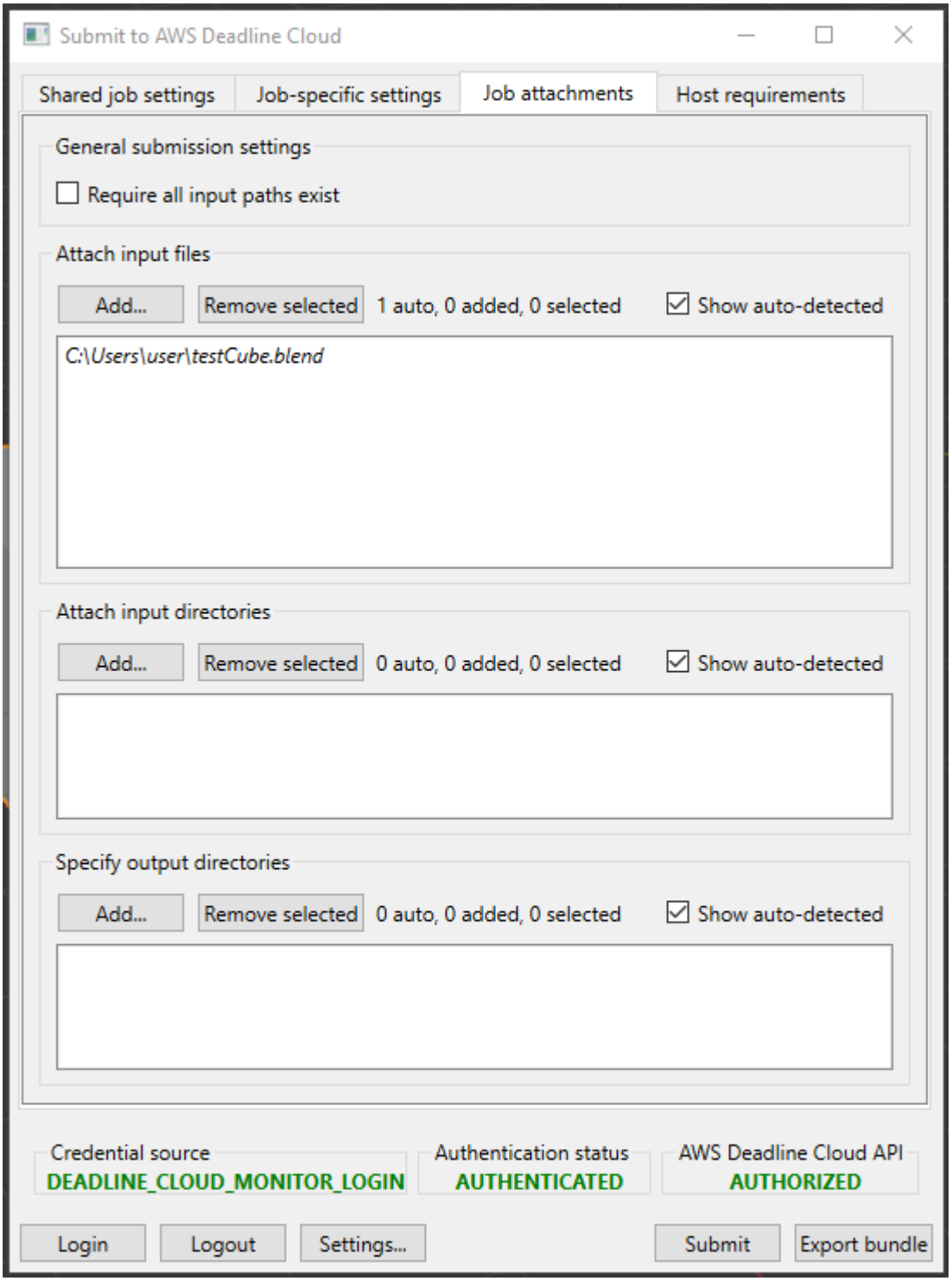
- *Job properties* – Sets the overall properties of the job. These properties are present in submitters for all DCC applications.
- *Deadline Cloud settings* – Shows the farm and queue that the job is sent to. To change the farm and queue, use the **Settings...** button at the bottom of the submitter.
- *Queue environment* – Sets the parameter values defined in the queue environment. Deadline Cloud adds the default parameter values for your DCC application, you can add additional values if necessary.

Job-specific settings tab



The job-specific settings tab contains the settings specific to your DCC application. Specify these settings based on the options available in your application.

Job attachments tab

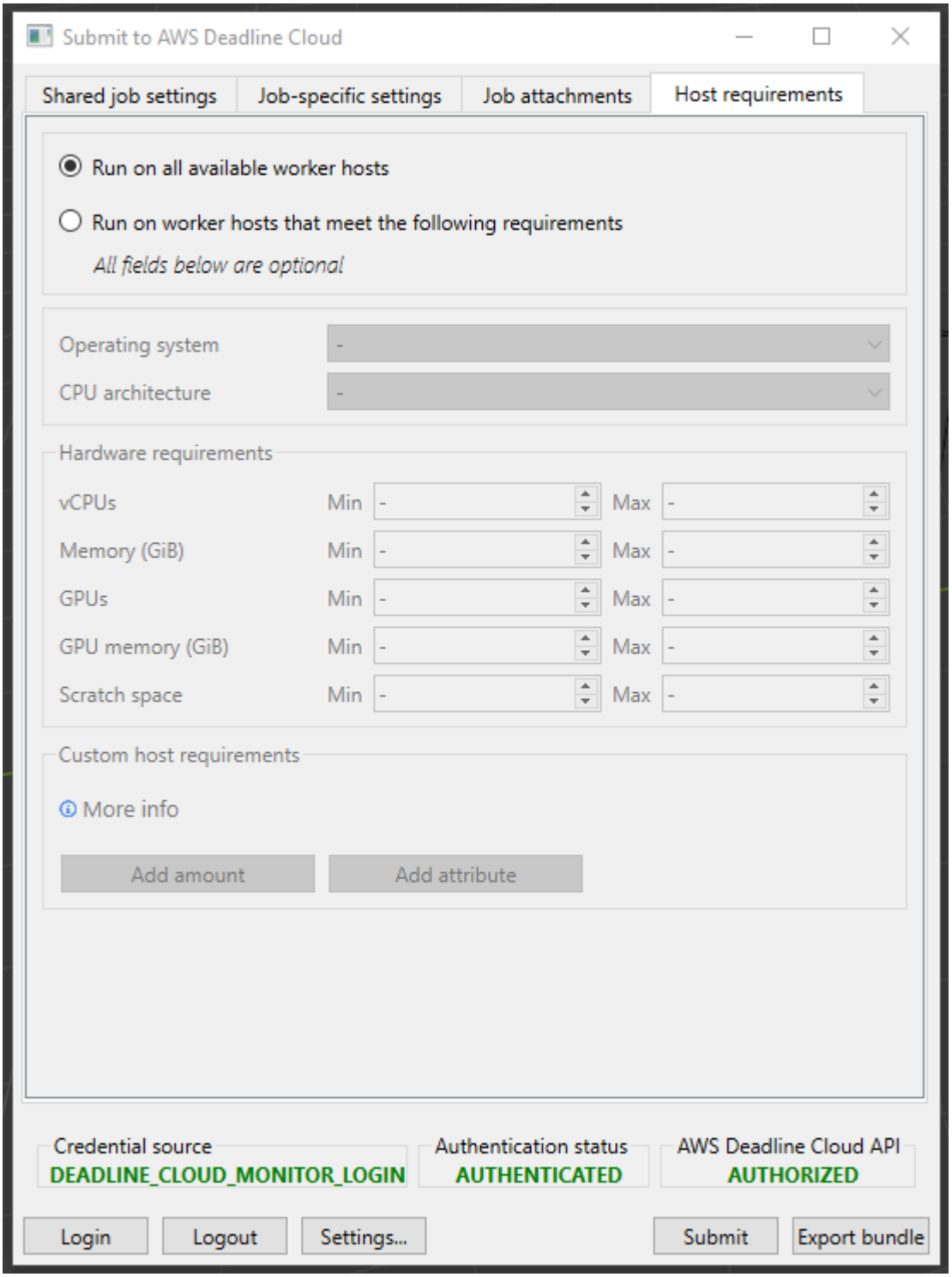


The job attachments tab shows all of the files needed to complete a render. The submitter tries to find all of the files required for the render. The files that it identifies appear in the lists in italics.

You can add additional input files and directories that contain other assets required for the render that were not automatically detected.

If your job writes files to multiple output directories, you must specify the directories here so that they are part of the job download.

Host requirements tab



The host requirements tab sets the fleet capabilities required to process the job. Capabilities are specified for the entire fleet, not individual workers in the fleet.

If your queue has associated resource limits, use the **Add amount** button to specify the limit. For more information, see Create resource limits for jobs

Load and submit shared job bundles

With AWS Deadline Cloud, you can share ready-to-use job bundles on a queue. If you can submit jobs to the queue, you can browse and preview the bundles shared on it. You can also submit them as jobs. You use the job bundle browser for each of these tasks. The job bundle browser is a graphical tool that you open from the Deadline Cloud command line. You don't need to know where the bundle files are stored or copy them manually.

The job bundle browser shows bundles from three sources:

- **Queue** – Bundles your team shared on the selected queue.
- **Local** – Bundles in folders on your workstation.
- **History** – Bundles from jobs you submitted before, so you can submit a previous job again with different settings.

For information about how to share a job bundle on your queue, see Share job bundles on your queue in the *Deadline Cloud Developer Guide*.

Prerequisites

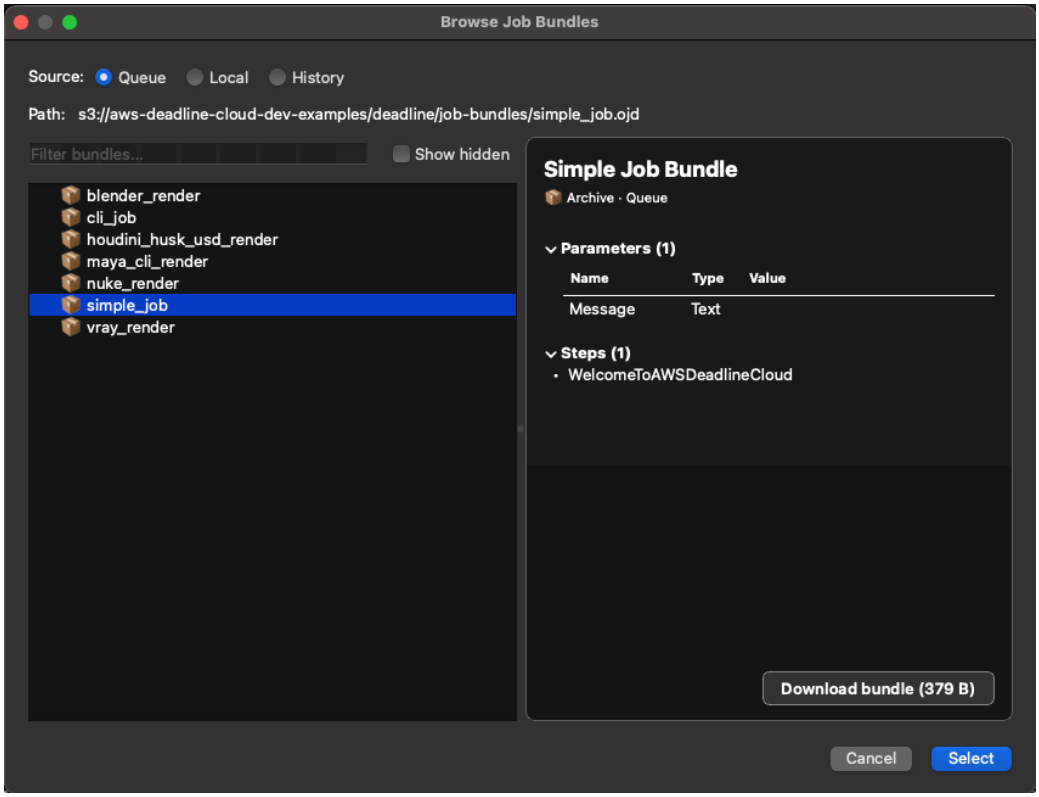
Before you begin, make sure that you have the Deadline Cloud command line interface (CLI) installed and configured, including its graphical components. You open the job bundle browser from the CLI. For installation and configuration steps, see Getting started in the *Deadline Cloud Developer Guide*.

Submit a shared job bundle

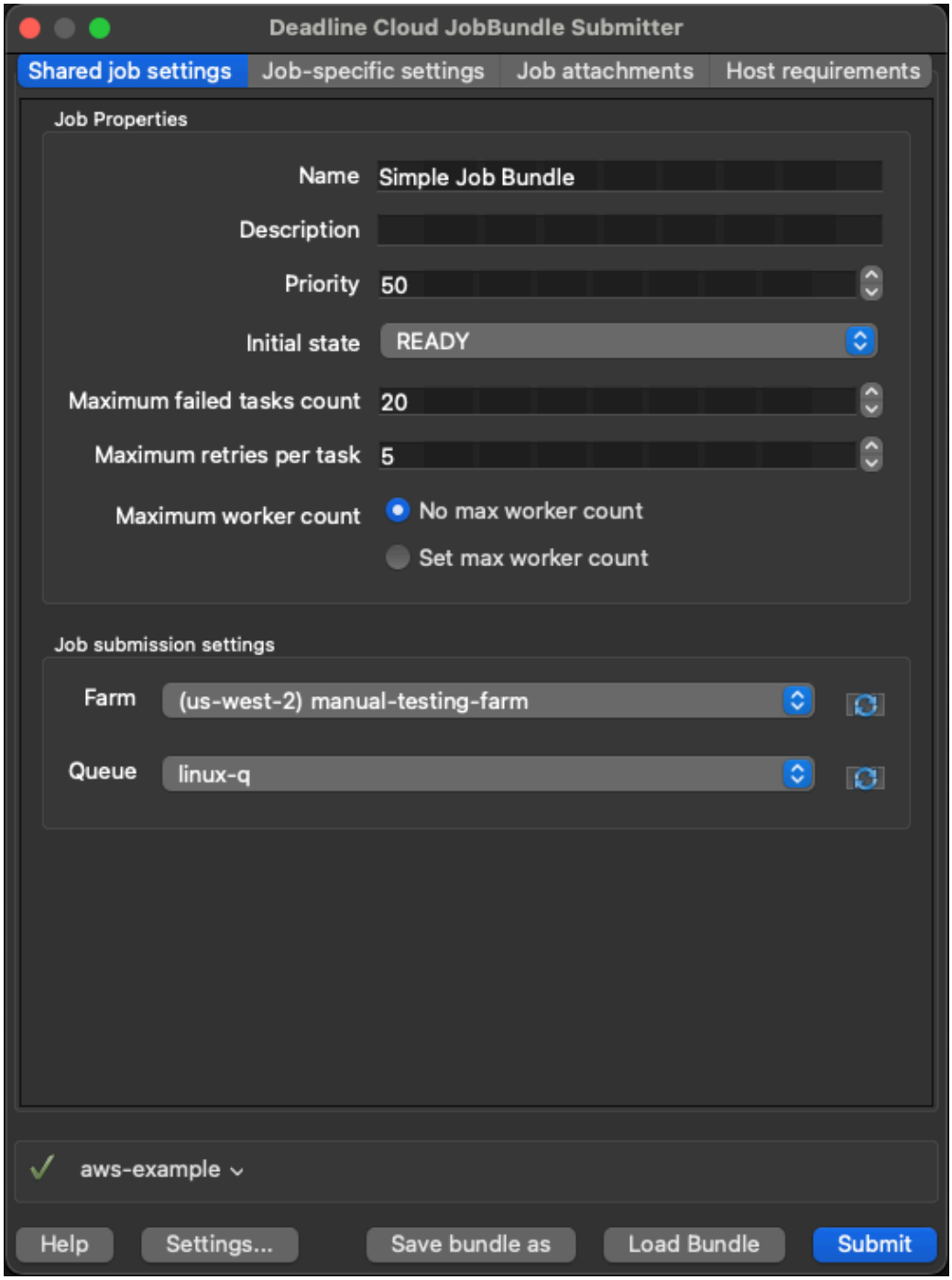
1. From a terminal, run the following command to open the job bundle browser:

   ```
   deadline bundle gui-submit --browse
   ```

   The job bundle browser opens, as shown in the following image:



2. For **Source**, choose **Queue**.
3. Select a bundle from the list. The preview shows the bundle's description, its parameters, and the steps it runs, so you can confirm it's the right one. To narrow a long list, enter part of a name in the filter box.
4. Choose **Select**. The bundle downloads and opens in the job submission dialog.

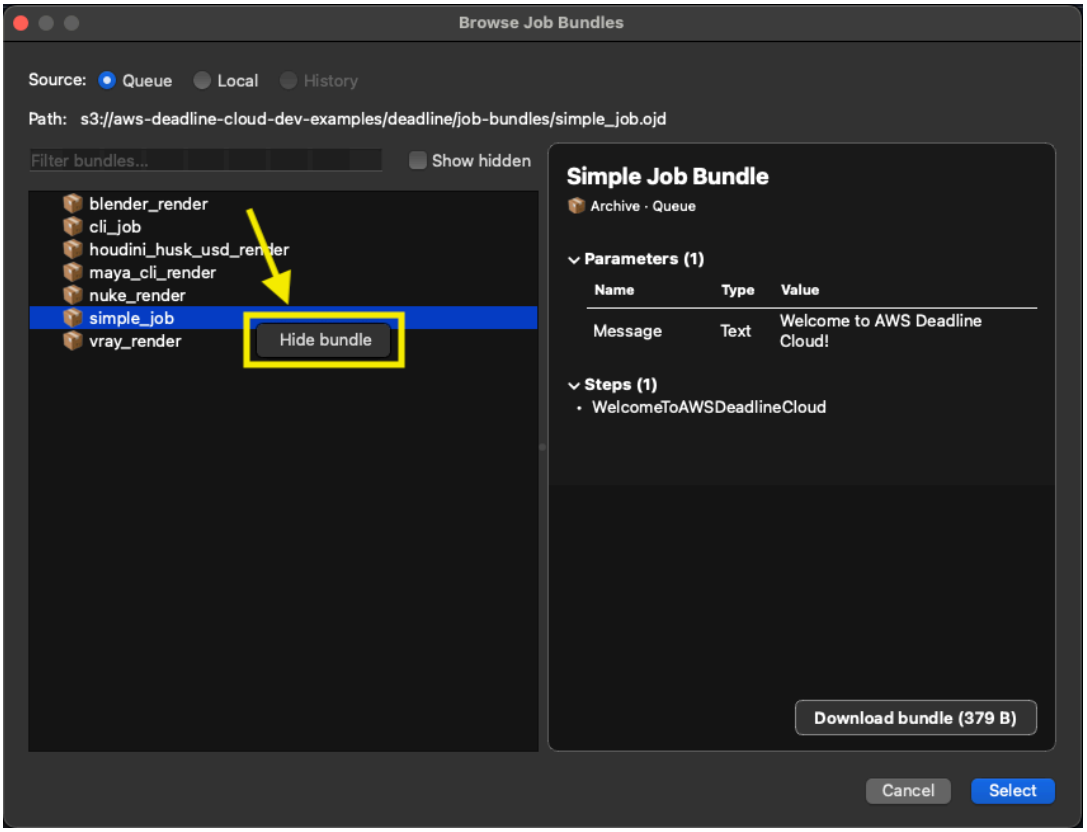


5. Set the parameter values for your job, then choose **Submit**. To go back and pick a different bundle instead, choose **Load Bundle**.

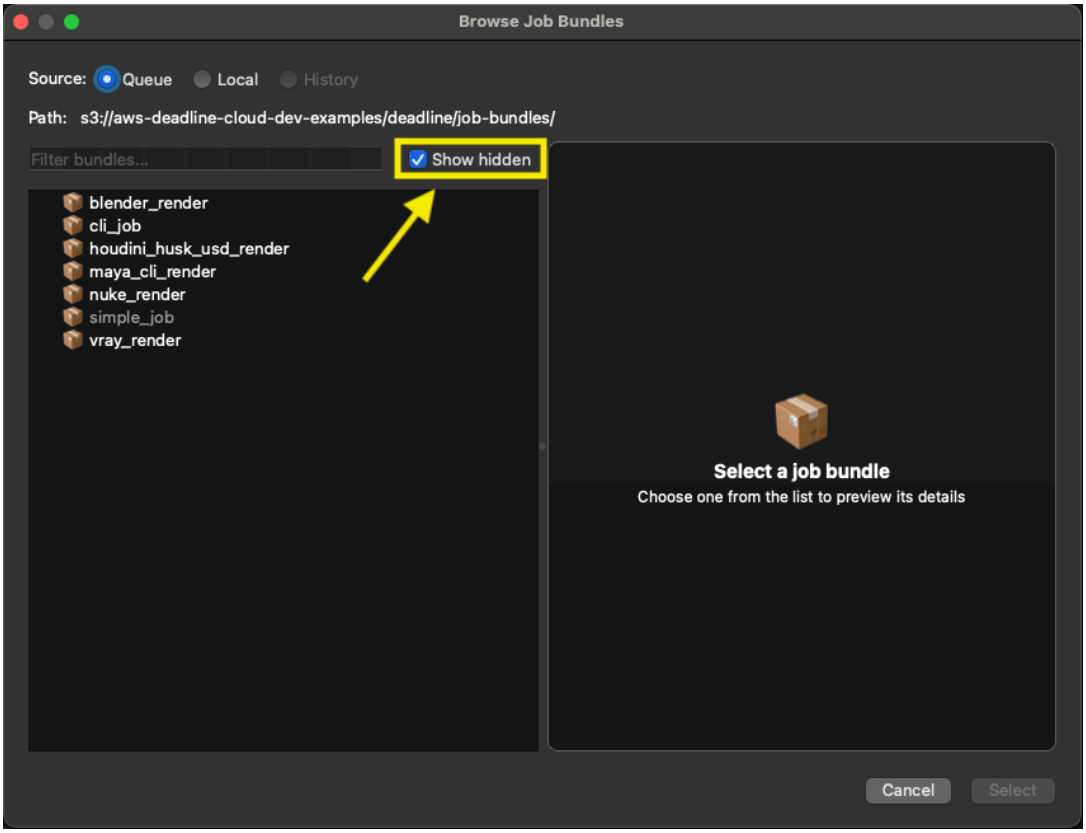
To look inside a bundle without submitting it, choose **Download bundle** in the preview. The bundle downloads and opens in your file explorer so that you can inspect its files.

Hide shared bundles you don't use

As a queue accumulates shared bundles, you can hide the ones you don't use. Open the context menu for a bundle, then choose **Hide bundle**. Hiding a bundle only changes your own view on your workstation. The bundle stays on the queue and your teammates still see it.



To bring a bundle back, select the **Show hidden** checkbox. Then open the context menu for the bundle and choose **Unhide bundle**.



Processing Deadline Cloud jobs

When a job enters a queue, Deadline Cloud schedules it on one or more fleets associated with the queue. Deadline Cloud chooses the fleet based on the capabilities configured for the fleet and the host requirements of the step. If a job has a requirement that no associated fleet can meet, Deadline Cloud sets the job's status to `NOT_COMPATIBLE` and cancels the remaining steps.

Workers process a job's tasks in sessions. A session sets up the environment for a step, runs one or more tasks, and then tears the environment down. The software required for the step must be available on the worker for the job to run. If the fleet's scaling settings allow, Deadline Cloud opens sessions on multiple workers at the same time.

After all tasks in each step are finished, the job is complete and the output is ready to download to your workstation. Even if the job didn't finish, the output from each step and task that finished is available to download.

> **Note**
>
> Deadline Cloud removes jobs 120 days after they were submitted. When a job is removed, all of the steps and tasks associated with the job are also removed. If you need to re-run the job, submit the OpenJD template for the job again.

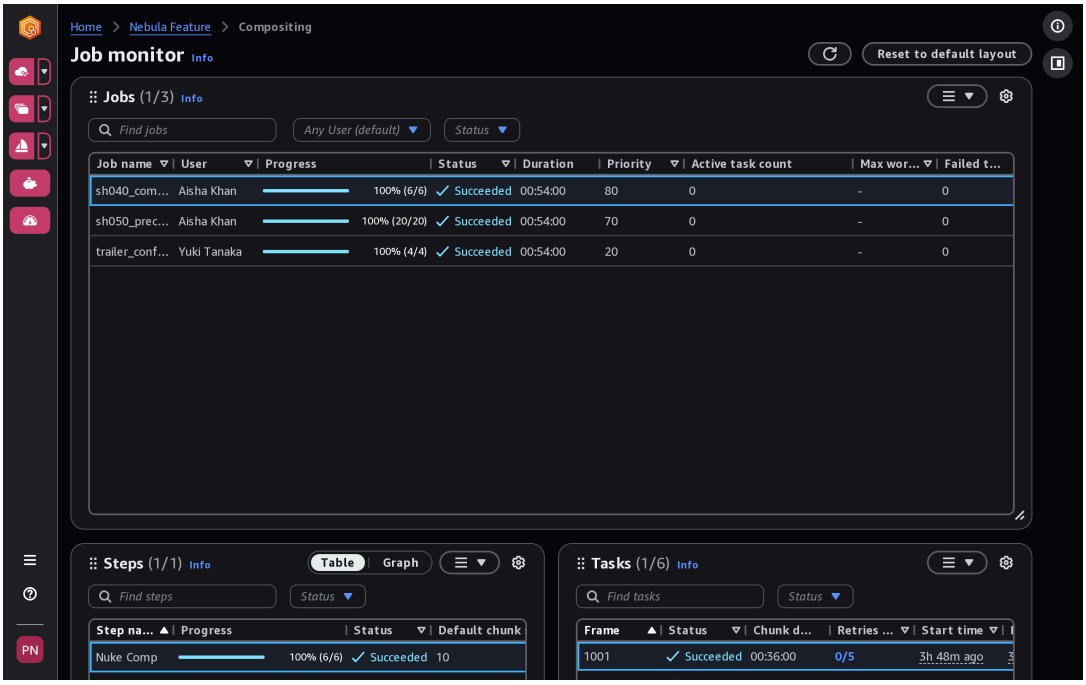
For the details of scheduling – scheduling configurations for a queue, how fleet compatibility is determined, sessions, and step dependencies – see Schedule jobs in Deadline Cloud in the *Deadline Cloud Developer Guide*.

Monitoring Deadline Cloud jobs

Use the AWS Deadline Cloud monitor to get an overall view of your jobs, including:

- Monitor and manage jobs
- View worker activity on fleets
- Track budgets and usage
- Download a job's results

To monitor a specific job, select the farm and queue containing the job, then select the job from the list. You can use the search box to locate a specific job or jobs in the queue.



Open the context menu for a job, step, or task. You can:

- Change the status
- Suspend and resume the item
- Requeue the item
- Download the output
- For jobs: Modify job properties like the name, description, priority, or max worker count
- For tasks: View task and worker logs

For more information, see Using the Deadline Cloud monitor. To modify a job with the AWS Command Line Interface (AWS CLI) or the Deadline Cloud API, see Modify a job in Deadline Cloud in the *Deadline Cloud Developer Guide*.

If your team uses automatic downloads, the jobs and tasks tables in the Deadline Cloud monitor desktop application also show a **Download status** column that reports whether each job's outputs reached your file system. For more information, see View output download status in Deadline Cloud.

Each task in a job or step has a status. The status of a job or step depends on the status of its tasks. The status is determined by tasks that have these statuses, in order. Step statuses are determined the same as the job status.

The following list describes the statuses:

`NOT_COMPATIBLE`

The job is not compatible with the farm because there are no fleets that can complete one of the tasks in the job.

`RUNNING`

One or more workers are running tasks from the job. As long as there is at least one running task, the job is marked `RUNNING`.

`ASSIGNED`

One or more workers are assigned tasks in the job as their next action. The environment, if any, is set up.

`STARTING`

One or more workers are setting up the environment for running tasks.

`SCHEDULED`

Tasks for the job are scheduled on one or more workers as the worker's next action.

`READY`

At least one task for the job is ready to be processed.

`INTERRUPTING`

At least one task in the job is being interrupted. An interruption can happen when you manually update the job's status, or when Amazon Elastic Compute Cloud (Amazon EC2) reclaims a Spot Instance.

`FAILED`

One or more tasks in the job didn't complete successfully.

`CANCELED`

One or more tasks in the job have been canceled.

`SUSPENDED`

At least one task in the job has been suspended.

`PENDING`

A task in the job is waiting on the availability of another resource.

`SUCCEEDED`

All tasks in the job were successfully processed.

Supported Software

Deadline Cloud supports many digital content creation (DCC) applications for 3D rendering, animation, visual effects, and compositing. The applications in the following table are preconfigured to work out of the box with some or all of the following:

- **Submitter** – An integrated plug-in for submitting jobs directly from the application. For more information, see Set up your workstation.
- **Service-managed fleet conda packages** – Prebuilt packages in the `deadline-cloud` conda channel that install the application on workers automatically, with no image to build or maintain. For more information, see Creating a queue environment.
- **Usage-based licensing (UBL)** – Pay-as-you-go licensing on service-managed fleets, so you don't need to bring your own license server. Applications that don't require a license to render are marked as *Not needed*. For more information, see Software licensing for service-managed fleets.

You can use custom plugins and add-ons with supported applications. To compare delivery methods and find setup instructions, see Use custom plugins with Deadline Cloud.

You aren't limited to the applications in the table. You can run almost any application or plugin on Deadline Cloud by packaging it yourself. For more information, see Software that isn't listed.

Software support summary

The following table summarizes support for each application. Select an application for versions, submitter installation steps, renderers, plugins, and conda package details.

Software	Supported versions	Submitter	Service-managed fleet conda packages	Usage-based licensing
Adobe After Effects	2024 - 2026	Windows, macOS	Windows	No
Autodesk 3ds Max	2024 - 2027	Windows	No	Yes

Software	Supported versions	Submitter	Service-managed fleet conda packages	Usage-based licensing
Autodesk Arnold for Cinema 4D	4.8.4.1	Windows, macOS	Windows, Linux	Yes
Autodesk Arnold for Maya	7.1 - 7.5	Windows, macOS, Linux	Linux	Yes
Autodesk Maya	2023 - 2027	Windows, macOS, Linux	Linux	Yes
Autodesk VRED	2025 - 2026	Windows	Linux	No
Blender	3.6 - 5.1	Windows, macOS, Linux	Linux	Not needed
Chaos V-Ray for Maya	6 - 7	Windows, macOS, Linux	Linux	Yes
Foundry Nuke	15 - 17	Windows, macOS, Linux	Linux	Yes
KeyShot Studio	2023 - 2025	Windows, macOS	No	No
Maxon Cinema 4D	2024 - 2026	Windows, macOS	Windows, Linux	Yes
Maxon Redshift for Maya	2025-2026	Windows, macOS, Linux	Linux	Yes
SideFX Houdini	19.5 - 22.0	Windows, macOS, Linux	Linux	Yes
Unreal Engine	5.4 - 5.8	Windows	Windows	Not needed

Software that isn't listed

If an application, plugin, or version that you need isn't in the table, you can still run it on Deadline Cloud. Service-managed fleets install software from any conda channel that you configure, so a customer-managed fleet isn't required. You have the following options:

- Build a conda package for the application or plugin and host it in your own channel. Workers on service-managed and customer-managed fleets install it the same way as packages from the `deadline-cloud` channel. Examples of software you can add as packages include plugins such as MayaUSD, custom shaders, and OCIO configurations. For more information, see Create a conda package for an application or plugin in the *Deadline Cloud Developer Guide*.
- Install the software directly on the worker hosts of a customer-managed fleet, for example through a custom Amazon Machine Image (AMI). For more information, see Install and configure software required for jobs in the *Deadline Cloud Developer Guide*.

For an overview of all the ways to provide software to your jobs, see Provide applications for your jobs in the *Deadline Cloud Developer Guide*.

Beyond digital content creation

Deadline Cloud also supports general-purpose compute-intensive workloads including scientific simulations, financial modeling, machine learning model training and evaluation, autonomous driving simulation, and data processing. You can run any workload that benefits from distributed parallel processing by creating custom job bundles. For examples that span these domains, see Code examples in the *Deadline Cloud Developer Guide*.

Adobe After Effects

Adobe After Effects is a professional digital visual effects, motion graphics, and compositing application. After Effects is fully supported by Deadline Cloud with comprehensive integration including submitters and conda packages for increased rendering performance. This guide provides step-by-step instructions for using AWS Deadline Cloud with After Effects to render your projects faster by distributing rendering tasks across multiple machines.

Support overview

After Effects is supported by the following components:

- **Submitter**: Integrated submitter for direct job submission from After Effects with automatic scene and asset detection.
- **Conda packages**: Deadline Cloud for automatic installation on service-managed fleets.
- **Cross-platform compatibility**: Submitter support for Windows and macOS with worker support for Windows.

After Effects version compatibility

The following table shows current support levels for After Effects versions:

Major Version	Submitter Support	Conda Support
2024	Windows, macOS	Windows
2025	Windows, macOS	Windows
2026	Windows, macOS	Windows

Deadline Cloud Conda Channel

The following table lists all conda packages applicable to After Effects available to Service-managed fleets in the deadline-cloud conda channel:

OS	Package	Version
Windows	aftereffects	24.6
Windows	aftereffects	25.1
Windows	aftereffects	25.2
Windows	aftereffects	25.6
Windows	aftereffects	26.0

Getting started

Complete the following steps to set up After Effects with Deadline Cloud. You will install the required submitter and monitor on your workstation and begin submitting render jobs to your queue.

1. Create a service-managed fleet and associate it with a queue. Your queue must be set up with a queue environment that supports the deadline-cloud conda channel. For more information, see Creating a queue environment.
2. Install the Deadline Cloud monitor on your artist workstation using the Deadline Cloud monitor Installers. For more information, see Set up your workstation.
3. Install the Deadline Cloud After Effects submitter on your artist workstation using the Deadline Cloud Submitter Installers. When you install the submitter, you can choose between User Install (no admin required) or System Install (Windows only, requires admin). macOS users must use User Install.
   - **User Install**: Installs to user directory without admin privileges. The submitter will be a standalone window rather than a dockable panel.
     - Windows: `C:\Users\<user>\DeadlineCloudSubmitter\Submitters\AfterEffects\AE<version>`
     - macOS: `/Users/<user>/Library/Preferences/Adobe/After Effects/<version>/Scripts/ScriptUI Panels`
   - **System Install** (Windows only): Installs to Adobe After Effects installation directory as a dockable panel.
     - Windows: `C:\Program Files\Adobe\Adobe After Effects <version>\Support Files\Scripts\Script UI Panels`

Installation

To install the Deadline Cloud submitter for After Effects, prepare the following environment:

- Windows or macOS workstation.
- Adobe After Effects 24, 25, or 26 installation.
- Python 3.9 or higher.
- Access to an Deadline Cloud farm with either:
  - A service-managed fleet with After Effects available (via custom conda package).

- A customer-managed fleet with After Effects and licensing set up.

Prerequisites

Before installing the submitter, complete the following steps:

1. Install the Deadline Cloud CLI and Deadline Cloud monitor by running the Deadline Cloud submitter and monitor installers from the downloads section of the Deadline Cloud service in your AWS console.
2. Enable script permissions in After Effects. The submitter requires the ability to write files and send communication over the network to function properly. By default, After Effects scripts are not allowed to perform these actions. For more information, see Adobe's scripts reference on the Adobe website.

   To allow scripts to write files or send communication over a network, complete the steps for your operating system:

   - **Windows**: Choose **Edit** > **Preferences** > **Scripting & Expressions** > select **Allow Scripts To Write Files And Access Network**.
   - **macOS**: Choose **After Effects** > **Settings** > **Scripting & Expressions** > select **Allow Scripts To Write Files And Access Network**.

Installing the submitter

The After Effects submitter extension allows you to submit jobs to Deadline Cloud directly from within After Effects.

> **Note**
>
> During the installer process, you choose between User Install or System Install. macOS users default to User Install, because automatic System Install is currently not supported on macOS (manual System Install is still possible for admin users). Read only the section below that matches your selection.

User install

1. Download the Deadline Cloud submitter installer by following Step 1: Install the Deadline Cloud Submitter.
2. Run the installer (no admin required).
3. Follow the prompts and select the After Effects submitter and choose the major version (for example, 24, 25, or 26).
4. The submitter is installed based on your operating system:
   - **macOS**: The installer automatically places `DeadlineCloudSubmitter(User).jsx` and `DeadlineCloudSubmitter_Assets` into your After Effects user preferences directory for all minor versions of the selected major version:

     ```
     /Users/<user>/Library/Preferences/Adobe/After Effects/<version>/Scripts/ScriptUI
      Panels
     ```

     The submitter appears in the **Window** menu in After Effects.
   - **Windows**: The submitter is installed to a standalone location by default:

     ```
     C:\Users\<user>\DeadlineCloudSubmitter\Submitters\AfterEffects\AE<version>
     ```

     You will need to run the script manually using **File** > **Scripts** > **Run Script File**.

System install

> **Note**
>
> Automatic macOS system install is not yet supported by the submitter installer. macOS admin users can perform a manual system install by copying files to the application directory (see Manual installation (alternative)).

1. Download the Deadline Cloud submitter installer by following Step 1: Install the Deadline Cloud Submitter.
2. **Windows only**: Right-click the installer and choose **Run as Admin**.
3. Follow the prompts and select the After Effects submitter. It is installed to:

- **Windows**: `C:\Program Files\Adobe\Adobe After Effects <version>\Support Files\Scripts\Script UI Panels`
- **macOS** (manual): `/Applications/Adobe After Effects <version>/Scripts/ScriptUI Panels`

Final setup steps

1. After installing using the submitter installer, ensure you log into your Deadline Cloud user profile by running `deadline auth login` in the Terminal/PowerShell or logging in using Deadline Cloud monitor.
2. To install the necessary dependencies used by the After Effects submitter, run the following in your local Terminal or Command Prompt:

```
pip install fonttools
```

3. Restart After Effects if it was open.

Manual installation (alternative)

If you prefer to install manually, you can copy the submitter files directly:

1. Locate the `DeadlineCloudSubmitter.jsx` file and the `DeadlineCloudSubmitter_Assets` folder in the `dist` folder of the deadline-cloud-for-after-effects repository on the GitHub website.
2. Copy both to the **ScriptUI Panels** folder within your After Effects installation:
   - **Windows**: `Program Files\Adobe\Adobe After Effects <version>\Support Files\Scripts\Script UI Panels`
   - **macOS**: `Applications/Adobe After Effects <version>/Scripts/Script UI Panels`
3. Restart After Effects if it was open.

Setting up After Effects with your Deadline Cloud farm

After Effects conda packages are available in Deadline Cloud service-managed fleets. For the list of supported versions, see the Deadline Cloud user guide. If you want to build a conda channel that

contains a different After Effects conda package, follow the instructions for creating custom conda channels.

You can also use the After Effects conda recipe in the deadline-cloud-samples package on the GitHub website as a reference when building the package.

Jobs created by this submitter require the `aerender` executable to be available on the PATH of the user that runs your jobs. Alternatively, you can set the `AERENDER_EXECUTABLE` environment variable to point to the aerender executable.

Updating the submitter

To update the submitter to the latest version, download and run the latest submitter installer.

Using the After Effects submitter

To use the Deadline Cloud submitter for After Effects, ensure your farm is configured with an After Effects-capable fleet (see Setting up After Effects with your Deadline Cloud farm) and have the submitter installed. Log into Deadline Cloud monitor or provide AWS credentials using a configuration profile for Deadline Cloud access.

Submit a job

> **Note**
>
> Follow the instructions below that match the install type you selected during installation.

User install - macOS

1. Launch Adobe After Effects.
2. Open the submitter by choosing **Window** > **DeadlineCloudSubmitter(User).jsx**.
3. Follow the steps in Submitting your job below.

User install - Windows

1. Launch Adobe After Effects.
2. Open the submitter by choosing **File** > **Scripts** > **Run Script File**. Navigate to the `DeadlineCloudSubmitter.jsx` file and select it to run the submitter. If you recently closed

the submitter, you can reopen it by choosing **File** > **Scripts** > **Recent Script Files** and selecting the `DeadlineCloudSubmitter.jsx` file.

3. Follow the steps in Submitting your job below.

System install

1. Launch After Effects as Admin (Windows) or with appropriate permissions (macOS).
2. Open the submitter by choosing **Window** > **DeadlineCloudSubmitter.jsx**.
3. Follow the steps in Submitting your job below.

Submitting your job

1. Choose **Open Render Queue** on the submitter. Add any composition to your render queue and set up your render settings, output module, and output path.
2. Choose **Refresh** on the submitter to see your composition in the composition list.
3. Select your composition you want to render and choose **Submit** to submit a render job. For details on available settings, see After Effects-specific settings below.
4. If you see a warning popup window with "You are about to run the script contained in file", you can suppress the warning by following the instruction in the popup.
5. Install any Python libraries if prompted and choose the **Login** button in the bottom left if you are not logged in.
6. Set the farm and queue you are submitting to using the **Settings** button, and choose **Submit**.

> **Note**
>
> The After Effects submitter calls the Deadline Cloud GUI submitter to complete job submission. If you encounter any issues on the GUI submitter, see the deadline-cloud library on the GitHub website for help.

Shared job settings

The following settings apply to the entire job:

- **Farm Selection** - Choose which farm your job will render on.

- **Queue Selection** - Select the specific queue within your chosen farm.
- **Job Name** - Give your render job a descriptive name.
- **Job Description** - Add optional details about your render job.
- **Priority** - Set job priority for queue management.
- **Initial State** - Control whether the job starts immediately or remains paused.
- **Max Failed Tasks Count** - Set the maximum number of tasks that can fail before the job is marked as failed.
- **Max Retries Per Task** - Set the number of times a failed task is retried.
- **Max Worker Count** - Set the maximum number of workers that can work on this job simultaneously.

After Effects-specific settings

The following settings are specific to After Effects rendering:

- **Composition Selection** - Select which composition from your render queue to submit.
- **Frames Per Task** - (For image sequences) Specify how many frames each task should render. By default, Deadline Cloud creates one task for every frame. Increasing this value can speed up rendering in jobs where each frame renders quickly.
- **Multi-Frame Rendering** - Enable After Effects multi-frame rendering. You can also specify the max percentage of CPU usage to allocate towards rendering. For more information, see Adobe's multi-frame rendering documentation on the Adobe website.
- **Timeout** - (Optional) Configure timeout settings (days, hours, and minutes) to handle unexpected cases where a task may become unresponsive. The default timeout is 2 days.
- **Output Type** - Automatically detected based on your render queue settings (image sequence or video).
- **Output Path** - Directory where rendered files will be saved (from your render queue settings).

For information about the other submitter tabs, see the Deadline Cloud guide for using a submitter.

Font attachment system

Fonts used in the submitted composition are detected by the submitter and are automatically added as job attachments on submission. Fonts are installed on the worker before the render starts and are removed when the job ends.

Supported font types:

- OpenType (`.otf`).
- TrueType (`.ttf`).
- TrueType Collection (`.ttc`) - majority supported.
- Adobe Fonts on the Adobe website.
- Windows bitmap fonts (`.fon`) - only supported on Windows machines.

Troubleshooting fonts:

If fonts are missing at render time:

1. Check that fonts are installed on your system:
   - **Windows**: Open **Settings** > **Personalization** > **Fonts** to view installed fonts.
   - **macOS**: Open Font Book application to view installed fonts.
2. Verify fonts are included in job attachments: After choosing **Submit** in the submitter, check the **Job Attachments** tab in the submission dialog to confirm your fonts are listed.

Adobe Creative Cloud Fonts:

For Deadline Cloud to use fonts distributed through Adobe Creative Cloud, they must be made available for all non-Adobe apps on your workstation.

To install fonts for non-Adobe apps in Creative Cloud:

1. Open Adobe Creative Cloud Desktop.
2. Choose **Adobe Fonts** on the account sidebar under **Your plan** to show the Adobe Fonts panel.
3. Choose **Added fonts** on the **Adobe Fonts** sidebar to show your added fonts.
4. Choose **Install family** next to the fonts you would like to make available for non-Adobe apps.

Viewing the job bundle

To submit a job, the submitter first generates a job bundle, and then uses functionality from the deadline-cloud package to submit the job bundle to your render farm to run.

If you want to see the job that will be submitted to your farm:

1. Use the **Export Bundle** button in the submitter to export the job bundle in the job history directory (default: `~/.deadline/job_history`).
2. If you want to submit the job from the export, rather than through the submitter, you can use the Deadline Cloud application to submit that bundle to your farm.

Troubleshooting

The following sections address common issues and questions about using the After Effects submitter with Deadline Cloud.

Rendering questions

Q: Can I run custom ExtendScript (.jsx) scripts on Deadline Cloud workers?

A: Yes, with limitations. The After Effects conda package on service-managed fleets packages the renderer and its dependencies, not a full After Effects installation. You can run custom `.jsx` scripts on workers using host configurations. However, scripts that make networking calls will not work on workers, because the "Allow Scripts To Write Files And Access Network" preference is a local user setting that cannot be configured without a full installation.

Troubleshooting questions

Q: My After Effects job is timing out or I'm seeing PermissionDenied errors during session cleanup. What's going on?

A: This issue is typically caused by After Effects hanging due to a UI dialog (for example, `alert()`, `confirm()`) that cannot be dismissed in headless mode. The symptoms are:

- **Timeout errors**: The process hangs waiting for user interaction that will never come, resulting in a `subprocess.TimeoutExpired` error.
- **PermissionDenied errors during session cleanup**: The hung After Effects process holds locks on Adobe DLLs (`AdobeXMP.dll`, `dvacore.dll`, `dynamiclink.dll`, etc.). When the worker attempts session cleanup, it produces `Access to the path '<file>' is denied` errors.

Fixing the root cause (the hung process) resolves both the timeout and the cleanup errors. Remove all UI calls from scripts intended for farm execution.

Troubleshooting tip: Try running the same workflow on a Windows workstation with the After Effects UI to eliminate any non-Deadline Cloud-specific issues.

Q: I'm getting "aerender Error: Could not read from source" when my job runs. What's wrong?

A: This error typically means the output path was not set in your composition's render queue before submitting the job. The After Effects submitter requires that you add your composition to the render queue and set up your render settings, output module, and output path before submission. See Using the After Effects submitter for the full submission workflow.

Q: My job fails at "Install Fonts to Worker" with `AddFontResource failed to load`. What do I do?

A: Some fonts are not installable on certain operating systems due to OS-specific limitations. When the font manager attempts to install an unsupported font on a worker, it fails with an error like:

```
OSError: AddFontResource failed to load "...\tempFonts\HelveticaNeue-
CondensedBlack.ttc"
```

To resolve this issue, try the following steps:

1. Check whether the font is actually used in your composition - it may have been uploaded by mistake.
2. If the font is not needed, remove it from the job attachments in the submitter before resubmitting.
3. If the font is needed, try substituting a different font that is compatible with Windows (Deadline Cloud only runs After Effects on Windows workers).
4. As a workaround, you can convert text layers to shapes in After Effects. For more information, see Creating shapes from text on the Adobe website. Note that conversion removes the font dependency entirely and is one-way. You will no longer be able to edit the text or use text-specific features.
5. If none of the preceding options work, create an issue in the repository on the GitHub website for further investigation.

Tip: You can check if a font is installable on Windows by double-clicking the font file on a Windows machine. If it opens and shows an "Install" option, the font is supported.

Error: Couldn't find Python 3 or higher on your PATH

For macOS:

1. Open Terminal and run the following commands: `where python` and `where python3`. If you're not getting any results, you need to install Python for your workstation.
2. If you do not have `python3` CLI installed but you have `python`, run `python --version` to check if you have Python 3.9 or higher. If not, install Python 3.9 or higher first and add it to your path.
3. After you get results from the version check and where check, then check which Python executable is actively being used. Run `which python` and `which python3`. If you're not getting any results, you need to add your Python CLI to your zsh `$PATH`.
4. You must also ensure you're adding the Python that was used to install the Deadline Cloud CLI. Run `python -m pip list` and/or `python3 -m pip list` to verify, and add to `$PATH` whichever Python applies.
5. Add the `bin` folder containing the Python CLI to your path by editing `~/.zshrc` (and `~/.bashrc` if applicable) and updating the `$PATH`. For example, if your Python and Deadline Cloud CLI are under `/Library/Frameworks/Python.framework/Versions/3.13/bin`, add the following line to your `~/.zshrc` file at the end of the file so it gets final priority when the `$PATH` is evaluated:

   ```
   export PATH=$PATH:/Library/Frameworks/Python.framework/Versions/3.13/bin
   ```

6. Save and exit, then run `source ~/.zshrc`.
7. Now if you run `python --version` and `deadline --version`, you have access to both executables and After Effects does too. Retry job submission. Your output should follow a pattern similar to:

   ```
   user@7cf34df03377 ~ % which python3
   /Library/Frameworks/Python.framework/Versions/3.13/bin/python3
   user@7cf34df03377 ~ % where deadline
   /Library/Frameworks/Python.framework/Versions/3.13/bin/deadline
   ```

For Windows:

1. Open Command Prompt (or PowerShell).
2. Run `where python`, `where python3`, `where py`, and `where deadline` to locate the executables. In PowerShell, use `Get-Command python` instead of `where python`.
3. Press Windows + S and search for "Edit the system environment variables". This step requires admin access. Choose **Environment Variables...**.
4. Depending on whether the Deadline Cloud CLI was a user installation or system installation, select the **Path** variable under either **User variables** or **System variables**, then choose **Edit**.
5. In the Edit window, choose **New** and paste the path to the folder containing your Python and Deadline Cloud executables.
6. Ensure that the binary folders containing deadline and your Python CLI have been added. Deadline will be located in the DeadlineCloudSubmitter folder when installed from the submitter installer or under a Python folder if managed by pip. If you're managing the Deadline Cloud CLI with pip, run `python -m pip list` and/or `python3 -m pip list` to ensure you add the Python containing the Deadline Cloud CLI to your path.

Error: Deadline Not Found

1. Open your Terminal or Command Prompt and run `deadline --version` to verify installation.
2. Then follow the troubleshooting steps above for Python for your operating system and verify that deadline is on your `$PATH`.
3. If you have multiple Python installations and manage the Deadline Cloud CLI using pip, verify that the Python on your `$PATH` is the Python that managed your Deadline Cloud CLI installation. Run `python -m pip list` and `python3 -m pip list` to verify.

Error: Missing job template

Example error: `Missing job template at /Applications/Adobe After Effects 2025/Scripts/ScriptUI Panels/DeadlineCloudSubmitter_Assets/JobTemplate`.

This error occurs when the `DeadlineCloudSubmitter_Assets` folder is missing or not located next to the `DeadlineCloudSubmitter.jsx` script file.

1. Locate where your `DeadlineCloudSubmitter.jsx` script is installed or moved to.
2. Ensure the `DeadlineCloudSubmitter_Assets` folder is in the same directory as the script.

3. If the assets folder is missing, copy it from your original installation source to the same location as the JSX script.
4. Restart After Effects and try running the submitter again.

Warning: Unsupported After Effects version detected

This warning means you are using an After Effects version that is not available in the deadline-cloud conda channel. For example, if you're using After Effects 24.3, but the channel supports only 24.6.

If you continue, the default major version in use locally is filled in the CondaPackages field, but your job submission may fail unless you have created a custom conda channel with your specific After Effects version and included this channel in the CondaChannels parameter.

To resolve this, you can either:

1. Switch to a supported After Effects version.
2. Acknowledge the warning and proceed (at your own risk).
3. Create a custom conda channel with your desired After Effects version.

Command prompt flashes on Windows after submission

Go to the Windows Start menu and search for "Manage app execution aliases". Then disable the `python3.exe` and `python.exe` aliases manually and retry submission.

Font with an unsupported extension was found

See Font attachment system for supported font types.

After Effects plugins

You can make After Effects plugins available on service-managed fleet workers by using either a conda package or a host configuration script. Plugins that are packaged as a conda package install into the After Effects `Plug-ins` directory when the conda environment activates. Plugins that require a full installer instead use a host configuration script that runs when each worker launches. To package several `.aex` plugins together, use the aftereffects-plugin-bundle conda recipe on the GitHub website.

Saber

Saber is a visual effects plugin for After Effects. You can build a conda package that installs the `Saber.aex` plugin on Windows workers with the aftereffects-saber conda recipe, and then add it to a custom conda channel. You can also use the recipe as a template for packaging other `.aex` plugins.

Conda recipe: aftereffects-saber conda recipe on the GitHub website.

Red Giant

You can install Maxon Red Giant on your workers with a host configuration script that fetches and silently installs the Red Giant installer when each worker launches. The script has been tested and verified on Windows GPU service-managed fleets. If a persistent volume is attached to the fleet, the script installs the software once and restores it on later boots to reduce worker startup time.

Host configuration script: aftereffects_redgiant on the GitHub website.

Advanced configurations

Using unsupported versions

Deadline Cloud only supports and tests the workstation and worker software versions in the table above. When using the submitter, the worker will attempt to install the same version as used on the workstation. This installation fails if the workstation version of After Effects does not appear in the version table above.

If you require an unsupported version of After Effects, you have the following options:

- When submitting the job from After Effects, you may override the CondaPackages queue parameter to specify a supported version to use on the worker (for example, `aftereffects=2025`). This override might or might not work, depending on the features used by your scene and how After Effects works with scenes from your workstation version.
- You may build a custom conda recipe and channel for your desired version to be installed on the worker. Use the conda recipe for a supported version linked below as a starting point, and package your desired version in a custom conda channel. For more information about creating custom conda channels, see Creating custom conda channels.

Open source resources

The submitter and standalone job bundle are open source and available on the GitHub website:

- Deadline Cloud for After Effects
- Standalone After Effects job bundle

Autodesk 3ds Max

> **Note**
>
> When using Autodesk 3ds Max with AWS Deadline Cloud, you can use Autodesk Cloud Rights included with your subscription. For more information about Cloud Rights and subscription benefits, see Subscription Benefits FAQ: Cloud Rights on the Autodesk website.

Autodesk 3ds Max is a professional 3D computer graphics program for creating 3D animations, models, games, and images. Deadline Cloud provides comprehensive support for 3ds Max with integrated submitters, host configuration scripts, usage-based licensing, and adaptors for increased rendering performance. This guide provides step-by-step instructions for using AWS Deadline Cloud with 3ds Max to render your projects faster by distributing rendering tasks across multiple machines.

Support overview

3ds Max is supported by the following components:

- **Submitter**: Integrated submitter for direct job submission from 3ds Max with automatic scene and asset detection.
- **Host Configuration Script**: Example host configuration script to install 3ds Max.
- **Adaptor**: A program on worker hosts that keeps the application loaded between tasks for faster rendering and reports render progress.
- **Cross-platform compatibility**: Submitter support for Windows with worker support for Windows and automatic path mapping.
- **Usage-based Licensing**: Pay-as-you-go licensing for 3ds Max and Corona.

3ds Max version compatibility

The following table shows current support levels for 3ds Max versions:

Major Version	Submitter Support	Host Configuration Support
2024	Windows	Windows
2025	Windows	Windows
2026	Windows	Windows
2027	Windows	Windows

3ds Max differences from other digital content creation tools

In Deadline Cloud, 3ds Max is installed using host configuration scripts instead of conda packages. This differs from most other DCCs in Deadline Cloud due to unique requirements of the 3ds Max installation process, as the application must be installed by a system administrator.

Getting started

To use 3ds Max with Deadline Cloud:

1. Create a service-managed fleet and associate it with a queue. Configure the fleet with GPU support if you intend to use GPU-accelerated rendering features. The fleet must be configured with a host configuration script that installs 3ds Max. For more information, see 3ds Max Host Configuration script setup and the 3ds Max Host Config example on the GitHub website.
2. Install the Deadline Cloud monitor and 3ds Max submitter on your artist workstation using the Deadline Cloud Submitter and monitor Installers. For more information, see Set up your workstation.
3. Submit your job directly from 3ds Max using the integrated submitter to the queue.
4. Monitor the job and download the output using the Deadline Cloud monitor.

Fleet host configuration

Before setting up the 3ds Max submitter, configure the Deadline Cloud fleet as follows.

3ds Max is a popular digital content creation tool provided by Autodesk. 3ds Max runs on Windows, and requires administrative access to install onto a host. Because of the administrative requirement, Deadline Cloud recommends installing 3ds Max onto the worker host using host configuration scripts.

Custom fleet host configuration scripts allow you to perform administrative tasks, such as software installation, on your service-managed fleet workers. These scripts run with elevated privileges, giving you the flexibility to configure your workers for your system.

Examples

Examples are available for 3ds Max 2024-2027 in the 3ds Max host configuration scripts folder on the GitHub website. The examples cover the V-Ray, Corona, and Arnold renderers, and the tyFlow, Pencil+, and AEC plugins. To request additional examples, suggest ideas in the 3ds Max discussion forum on the GitHub website.

> **Note**
>
> Although the examples install specific 3ds Max versions, the Deadline Cloud submitter supports 3ds Max 2025, 2026, and 2027 as well. The installation script should work equivalently for 3ds Max 2025, 2026, and 2027.

Installation

To install the Deadline Cloud submitter for Autodesk 3ds Max, prepare the following environment:

- Windows workstation.
- Autodesk 3ds Max 2024, 2025, 2026, or 2027 installation.
- Optional: V-Ray 6 or 7 for 3ds Max installation.
- Access to an Deadline Cloud farm with either:
  - A Windows service-managed fleet with Autodesk 3ds Max host configuration.
  - A customer-managed fleet with Autodesk 3ds Max and licensing set up.

Installing the submitter

The Autodesk 3ds Max submitter extension allows you to submit jobs to Deadline Cloud directly from within 3ds Max. To install the submitter:

1. Download the Deadline Cloud submitter installer.
2. Run the installer and follow the on-screen instructions.
3. Launch 3ds Max after installation.

Updating the submitter

To update the submitter to the latest version, download and run the latest submitter installer.

Using the Autodesk 3ds Max submitter

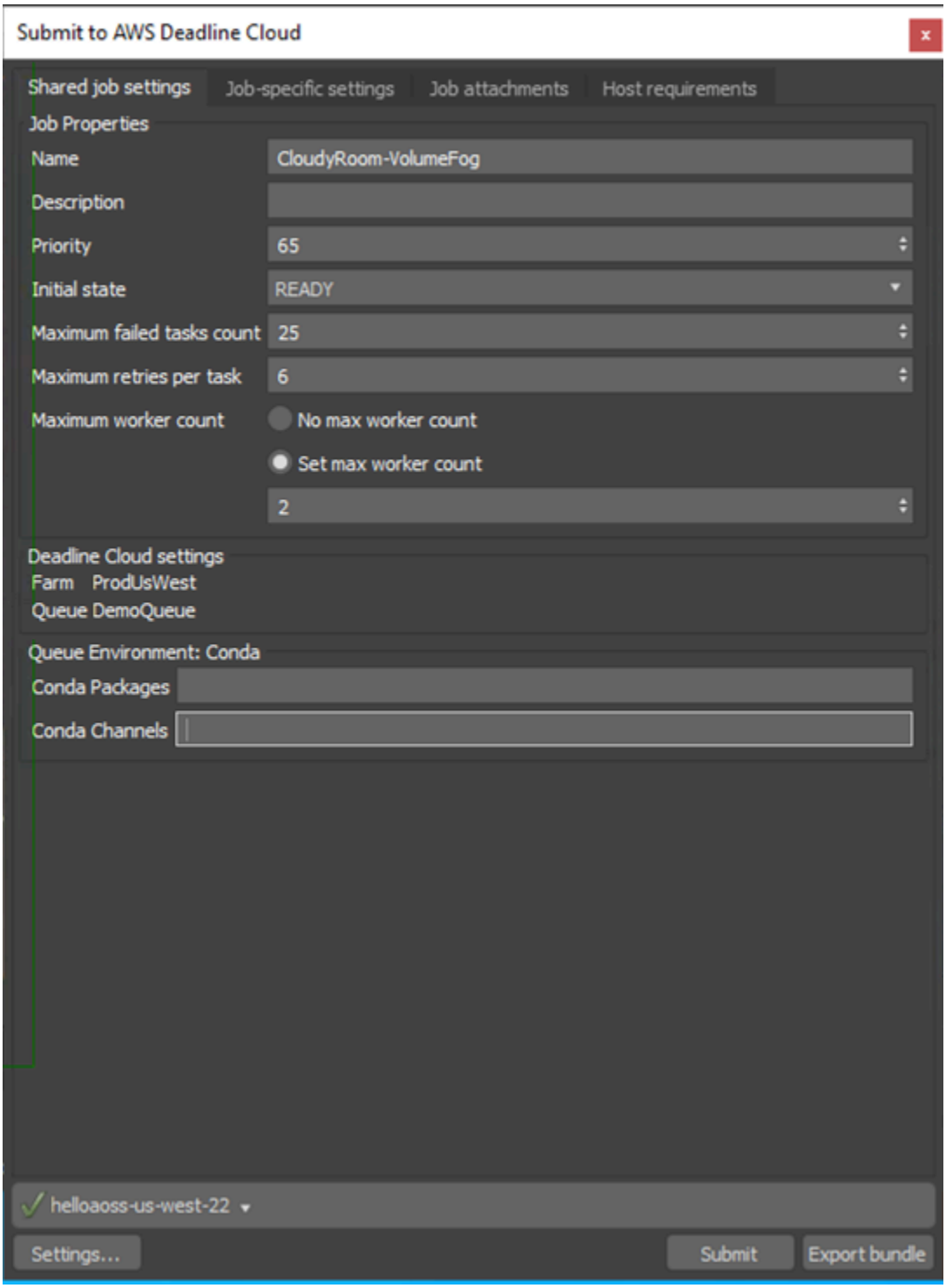
To use the Deadline Cloud submitter for 3ds Max, make sure that your farm is configured with a 3ds Max-capable fleet, and that the submitter is installed. Log into Deadline Cloud monitor or provide AWS credentials using a configuration profile for Deadline Cloud access.

Submitting a job

To submit a job from 3ds Max to Deadline Cloud:

1. Save your 3ds Max file.
2. On the 3ds Max menu bar, choose **Deadline Cloud**.
3. Use the tabs in the dialog to customize your job.
4. (Optional) To export a job's associated files to your job history directory without submitting it, choose **Export bundle**.
5. Choose **Submit** and follow the prompts to send your job to Deadline Cloud.

Shared job settings

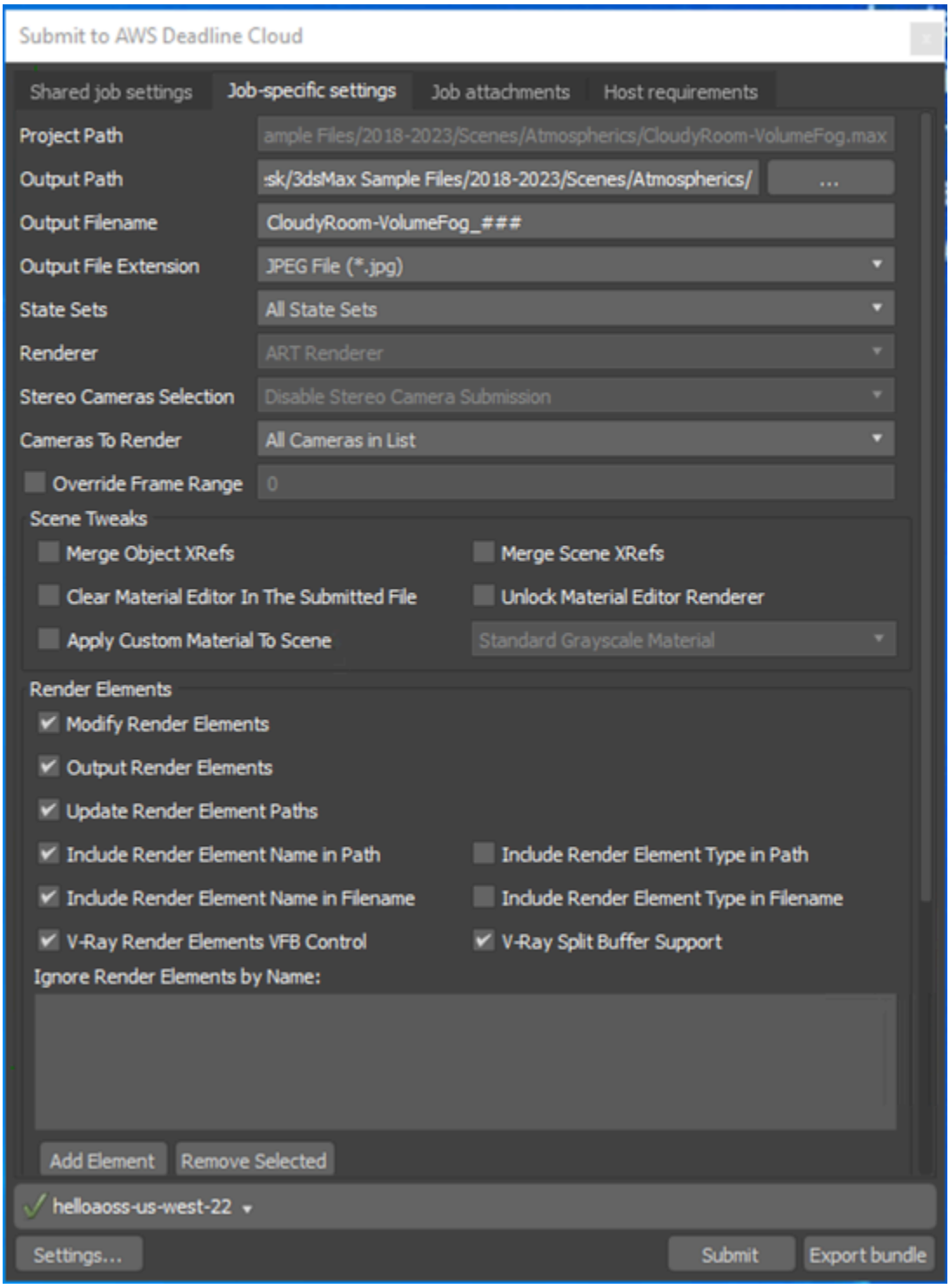


The following settings apply to the entire job:

- **Farm Selection** - Choose which farm your job will render on.
- **Queue Selection** - Select the specific queue within your chosen farm.

- **Job Name** - Give your render job a descriptive name.
- **Job Description** - Add optional details about your render job.
- **Priority** - Set job priority for queue management.
- **Initial State** - Control whether the job starts immediately or remains paused.
- **Max Failed Tasks Count** - Maximum number of tasks that can fail before the job is marked as failed.
- **Max Retries Per Task** - Number of times a failed task will be retried.
- **Max Worker Count** - Maximum number of workers that can work on this job simultaneously.
- **Conda Packages** - This setting must be empty as 3ds Max does not use conda.
- **Conda Channels** - This setting must be empty as 3ds Max does not use conda.

3ds Max-specific settings



The following settings are specific to 3ds Max rendering:

- **Project Path** - The 3ds Max project path (automatically detected).
- **Output Path** - Directory where rendered images will be saved.

- **Output Filename** - Base name for rendered image files. Use ### to represent the frame number.
- **Output File Extension** - File format for rendered images (for example, .exr, .png, .jpg).
- **State Sets** - Select which 3ds Max state set to use for rendering.
- **Renderer** - Current renderer from 3ds Max render settings (read-only).
- **Stereo Cameras Selection** - Choose stereo camera rendering options if a stereo plugin is available.
- **Cameras To Render** - Select specific cameras or render all cameras.
- **Override Frame Range** - Optionally override the scene's frame range with custom values.

Scene tweaks

The following options modify the scene during submission:

- **Merge Object XRefs** - Merge external object references into the scene.
- **Merge Scene XRefs** - Merge external scene references into the scene.
- **Clear Material Editor In The Submitted File** - Remove materials from the material editor.
- **Unlock Material Editor Renderer** - Unlock the material editor renderer.
- **Apply Custom Material To Scene** - Apply a custom material to all scene objects.

Render elements

Render elements in 3ds Max are specialized output passes. They separate different aspects of the rendered image into individual components for advanced compositing and post-production workflows. These elements allow artists to isolate specific rendering components, such as diffuse color, specular highlights, shadows, reflections, and material properties. Artists can then precisely control and adjust these components in post-production without re-rendering the entire scene. Deadline Cloud for 3ds Max provides comprehensive render elements support with advanced path management, V-Ray integration, and automatic configuration during rendering.

The submitter provides render elements support with the following options:

- **Modify Render Elements** - Enables any changes to render element settings for this scene. If selected, the following options are applied at render time.
- **Output Render Elements** - Control enable/disable render elements output.
- **Update Render Element Paths** - Automatically update output paths during submission.

- **Include Name/Type in Path** - Add render element names or types to output directory paths.
- **Include Name/Type in Filename** - Add render element names or types to output filenames.
- **V-Ray Specific Settings** - VFB control and split buffer support for V-Ray render elements.
- **Ignore Render Elements by Name** - Exclude specific render elements from output.

For information about the other submitter tabs, see the Deadline Cloud guide for using a submitter.

Known limitations

Maximum number of state sets / batch views per job

The Open Job Description (OpenJD) specification limits a job to a maximum of 50 job parameters. Because the submitter creates per-step parameters for each state set or batch view, this places a practical ceiling on how many can be included in a single job submission. For more information, see the OpenJD Template Schemas on the GitHub website.

The submitter uses a fixed set of global parameters, plus per-step parameters that scale with the number of state sets or batch views:

Parameter group	Count
Base parameters (scene file, error checking)	2
Camera parameter (when a specific camera is selected)	0 or 1
Render element parameters (when scene has render elements)	up to 10
Per state set in Default mode (frames, output path/name/format, resolution)	6 each
Per batch view in Batch Render mode (frames, output path/name/format, resolution, camera, scene state, preset, pixel aspect)	10 each

The practical limits are:

Submission mode	Render elements	Specific camera	Max per job
Default	No	No	8 state sets
Default	No	Yes	7 state sets
Default	Yes	No	6 state sets
Default	Yes	Yes	6 state sets
Batch Render	No	N/A	4 batch views
Batch Render	Yes	N/A	3 batch views

Submitting a job that exceeds 50 parameters will fail with a validation error. If you need to render more state sets or batch views than the limit allows, split them across multiple job submissions.

Command-line rendering with 3dsmaxcmd

Use the optional command-line render workflow to render your scene with `3dsmaxcmd.exe`, the 3ds Max command-line render server, instead of the standard adaptor. Because `3dsmaxcmd.exe` registers as a render server, network-licensed plugins such as PSOFT Pencil+ 4 NTR render without a watermark. The workflow is opt-in. Unless you enable it, the submitter uses its default behavior.

When to use the command-line render workflow

Network-licensed plugins, such as PSOFT Pencil+ 4 NTR, add a watermark to output rendered through the standard adaptor. The watermark appears because `3dsmaxbatch.exe` does not register as a render server. Use the command-line render workflow when your scene uses a plugin that requires 3ds Max to run as a render server to produce watermark-free output.

For most other scenes, use the standard submitter workflow. The standard adaptor provides sticky sessions and additional monitoring that the command-line render workflow does not.

> **Note**
>
> The command-line render workflow does not redirect V-Ray Frame Buffer raw output, such as raw and split-channel files. For V-Ray, use the standard **Output Filename** field or the

standard adaptor workflow. Otherwise, V-Ray writes that output outside the captured job output.

How the command-line render workflow works

When you enable the workflow, the submitter builds a job bundle that renders the scene with `3dsmaxcmd.exe`. On the worker, the task reads the session's Deadline Cloud path mapping rules. It then generates a pre-render MAXScript that remaps the scene's asset and output paths to their session locations. The task then invokes `3dsmaxcmd` once per frame. The job uses the name and description from **Shared job settings**. It also uses the **Job attachments** and **Host requirements** from the other submitter tabs.

Submitting a command-line render job

To submit a job that renders with 3dsmaxcmd:

1. Save your 3ds Max file.
2. On the 3ds Max menu bar, choose **Deadline Cloud**.
3. On the **3dsmaxcmd Render** tab, select **Enable 3dsmaxcmd Command-Line Render**.
4. Configure the render options for the job.
5. Choose **Submit** and follow the prompts to send your job to Deadline Cloud.

The tab provides the following options:

- **Frame List** – Frame range to render, for example `1-100` or `1,5,10-20`.
- **Output Path** – Directory where rendered images are saved. The output path is required so that the render output goes to a captured location instead of the local output path baked into the scene.
- **Output Filename** – Output file name with extension, for example `render.exr`. Leave blank to use the scene's Render Setup output.
- **Camera** – Named camera to render. Leave blank to use the scene's active view.
- **3dsmaxcmd Executable** – Path to `3dsmaxcmd.exe` on the worker. Defaults to `3dsmaxcmd`, resolved on the worker's PATH. Set a full path if the executable is not on the PATH.
- **Override Task Run Timeout** – When selected, a frame render that exceeds the timeout is cancelled. Leave unselected to allow renders to run until complete.

Setting up a fleet for Pencil+

Install Pencil+ 4 on service-managed fleet workers with a host configuration script, the same way you install 3ds Max. Example scripts that install 3ds Max and Pencil+ 4 are available for 3ds Max 2025 and 2027 on the GitHub website:

- 3dsmax-2025-and-pencilplus-4.ps1
- 3dsmax-2027-and-pencilplus-4.ps1

The scripts install the Pencil+ 4 NTR edition, which renders watermark-free without consuming a license when 3ds Max runs as a render server. Because `3dsmaxcmd.exe` is a render server, you don't need to configure a license server for the command-line render path. For the general host configuration setup steps, see Fleet host configuration.

V-Ray standalone tile rendering

For advanced V-Ray users, you can export V-Ray scene files (`.vrscene`) locally within 3ds Max and submit them as standalone job bundles with tile rendering support. This workflow is particularly useful for large resolution renders where tiling can reduce memory footprint and optimize render times.

When to use this workflow

Tile rendering with V-Ray Standalone on Linux workers is beneficial for:

- Large resolution renders (outdoor advertising, high-resolution entertainment content).
- Scenes with high memory requirements that benefit from processing smaller regions.
- Optimizing render resources by splitting images into evenly sized regions rendered in parallel.
- Minimizing rendering time through parallel processing.
- Reducing infrastructure costs by using Linux Amazon Elastic Compute Cloud (Amazon EC2) workers instead of Windows workers (Linux Amazon EC2 instances typically have lower hourly rates than equivalent Windows instances).

Exporting V-Ray scene files

V-Ray for 3ds Max includes a Scene Exporter that creates `.vrscene` files containing all scene information (geometry, lights, shaders) that can be rendered with V-Ray Standalone.

To export a V-Ray scene file:

1. In 3ds Max, configure your V-Ray render settings as needed.
2. Use the V-Ray Scene Exporter to export your scene as a `.vrscene` file.

The exported file is a text-based format that contains complete scene data.

Submitting tile rendering jobs

Once you have exported your `.vrscene` file, you can use the standalone tile rendering job bundle to submit optimized rendering jobs to Deadline Cloud.

For general information about creating and submitting job bundles, see Open Job Description templates for Deadline Cloud in the Deadline Cloud Developer Guide.

Reference implementation:

The tile_render_with_vray_linux sample on the GitHub website demonstrates:

- How to split large images into tiles.
- Parallel rendering of tiles on Linux workers.
- Automatic tile assembly after rendering completes.

You can submit this job bundle using the Deadline Cloud CLI:

```
deadline bundle submit <path-to-job-bundle>
```

Or use the GUI submitter:

```
deadline bundle gui-submit <path-to-job-bundle>
```

Benefits of this approach:

- Reduced memory usage per render task.
- Parallel processing of tiles for faster overall render times.
- Better resource utilization across your Deadline Cloud farm.
- Flexibility to customize tile dimensions based on your scene requirements.

- Cost savings by using Linux workers instead of Windows workers (Linux Amazon EC2 instances typically cost less than equivalent Windows instances).

Job bundle structure

The tile rendering job bundle uses Open Job Description templates to define:

- Job parameters for specifying the number of horizontal and vertical tiles.
- Task parameters that create individual tasks for each tile.
- A rendering step that processes each tile in parallel.
- An assembly step that stitches tiles together after rendering completes.

Requirements

- V-Ray for 3ds Max with Scene Exporter.
- Deadline Cloud farm configured with Linux workers.
- V-Ray Standalone installed on worker nodes.
- FFmpeg or similar tool for tile assembly (can be provided using conda).

3ds Max plugins

Because 3ds Max is installed with host configuration scripts instead of conda packages, third-party 3ds Max plugins are also installed with host configuration scripts that run when each worker launches. The following architectural visualization (AEC) plugins from iToo Software are included in the example scripts alongside 3ds Max and V-Ray:

- **Forest Pack** provides scattering and distribution tools for creating forests, vegetation, and other scattered objects.
- **RailClone** is a parametric modeling plugin for creating linear and area-based structures such as fences, roads, and buildings.

For example host configuration scripts that include these plugins, see 3dsmax-2025-vray-and-aec-plugins and 3dsmax-2027-vray-and-aec-plugins on the GitHub website.

You can also use the **Pencil+ 4** line rendering plugin from PSOFT. Because Pencil+ uses network (NTR) licensing, render it with the command-line render workflow to produce watermark-free output. For more information, see Command-line rendering with 3dsmaxcmd.

Advanced configurations

Using unsupported versions

Deadline Cloud only supports and tests the workstation and worker software versions in the table above. You must ensure the version of 3ds Max used by the artist is compatible with the version of 3ds Max configured in your fleet's host configuration.

Support for older 3ds Max versions is possible via host configuration scripts. However, the integrated submitter may not function due to older Python versions. In such cases, custom job bundles can still be submitted as Deadline Cloud jobs.

3ds Max renderers

Deadline Cloud supports rendering 3ds Max jobs using the following renderers when using a host configuration script that includes them:

Renderer	Renderer Version	Host Configuration Script Provided	Usage-based Licensing Support
Autodesk Scanline	Built-in	N/A	N/A
Autodesk Raytracer (ART)	Built-in	N/A	N/A
Chaos V-Ray 6	6.x	Yes	Yes
Chaos V-Ray 7	7.x	Yes	Yes
Corona	Latest	Yes	No

Open source resources

The submitter and adaptor are open source and available on the GitHub website:

- 3ds Max Submitter and Adaptor
- Deadline Cloud Samples (for 3ds Max workflow examples)
- 3ds Max Host Config example

Autodesk Maya

Autodesk Maya is a 3D computer animation, modeling, simulation, and rendering software used for creating interactive 3D applications, including video games, animated films, TV series, and visual effects. Maya is fully supported by Deadline Cloud with comprehensive integration including submitters, conda packages, usage-based licensing, and an adaptor for increased rendering performance. This guide provides step-by-step instructions for using Deadline Cloud with Autodesk Maya to render your projects faster by distributing rendering tasks across multiple machines.

Support overview

Maya is supported by the following components:

- **Submitter**: Integrated plug-in for direct job submission from Maya.
- **Conda packages**: Automatic installation on service-managed fleets when using the submitter.
- **Adaptor**: A program on worker hosts that keeps the application loaded between tasks for faster rendering and reports render progress.
- **Cross-platform compatibility**: Submitter support for Windows, macOS, and Linux with worker support for Windows and Linux.
- **Usage-based Licensing**: Pay-as-you-go for Maya and renderer licensing.

Maya version compatibility

The following table shows current support levels for Maya versions:

Major Version	Submitter Support	Conda Support	Render Engines	Usage-Based Licensing
2024	Windows, macOS, Linux	Linux	Maya Software, Arnold (MtoA)	Usage-based licensing available

Major Version	Submitter Support	Conda Support	Render Engines	Usage-Based Licensing
2025	Windows, macOS, Linux	Linux	Maya Software, Arnold (MtoA), V-Ray, Redshift	Usage-based licensing available
2026	Windows, macOS, Linux	Linux	Maya Software, Arnold (MtoA), V-Ray, Redshift	Usage-based licensing available

Deadline Cloud Conda Channel

The following table lists all conda packages applicable to Maya available to Service-managed fleets in the deadline-cloud conda channel:

OS	Package	Version	Notes
Linux	maya	2024	Includes Maya Software renderer
Linux	maya	2025	Includes Maya Software renderer
Linux	maya	2026	Includes Maya Software renderer
Linux	maya-mtoa	2024.5.3	Arnold for Maya 2024
Linux	maya-mtoa	2025.5.4	Arnold for Maya 2025
Linux	maya-mtoa	2026.5.5	Arnold for Maya 2026
Linux	maya-openjd		Includes the Maya Adaptor
Linux	maya-redshift	2025.4	Redshift for Maya 2025

OS	Package	Version	Notes
Linux	maya-redshift	2026.2.1	Redshift for Maya 2026
Linux	maya-vray	2025.7	V-Ray for Maya 2025
Linux	maya-vray	2026.7	V-Ray for Maya 2026

Getting started

To use Maya with Deadline Cloud:

1. Create a service-managed fleet and associate it with a queue. Your queue must be set up with a queue environment that supports the deadline-cloud conda channel. For more information, see Creating a queue environment.
2. Install the Deadline Cloud monitor and Maya submitter on your artist workstation using the Deadline Cloud Submitter and monitor Installers. For more information, see Set up your workstation.
3. Submit your job directly from Maya using the integrated submitter to the queue.
4. Monitor the job and download the output using the Deadline Cloud monitor.

Installation

To install the Deadline Cloud submitter for Autodesk Maya, prepare the following environment:

- Windows, Linux, or macOS workstation.
- Autodesk Maya 2024, 2025, or 2026 installation.
- Optional: Arnold (MtoA 5.3.5 or higher), V-Ray, or Redshift for Maya installation.
- Deadline Cloud monitor installed.
- Access to an Deadline Cloud farm with either:
  - A service-managed fleet.
  - A customer-managed fleet with Autodesk Maya and licensing set up.

Installing the submitter

The Autodesk Maya submitter extension allows you to submit jobs to Deadline Cloud directly from within Maya. To install the submitter:

1. Download the Deadline Cloud submitter installer.
2. Run the installer and follow the on-screen instructions.
3. Launch Maya after installation.

Updating the submitter

To update the submitter to the latest version, download and run the latest submitter installer.

Using the Autodesk Maya submitter

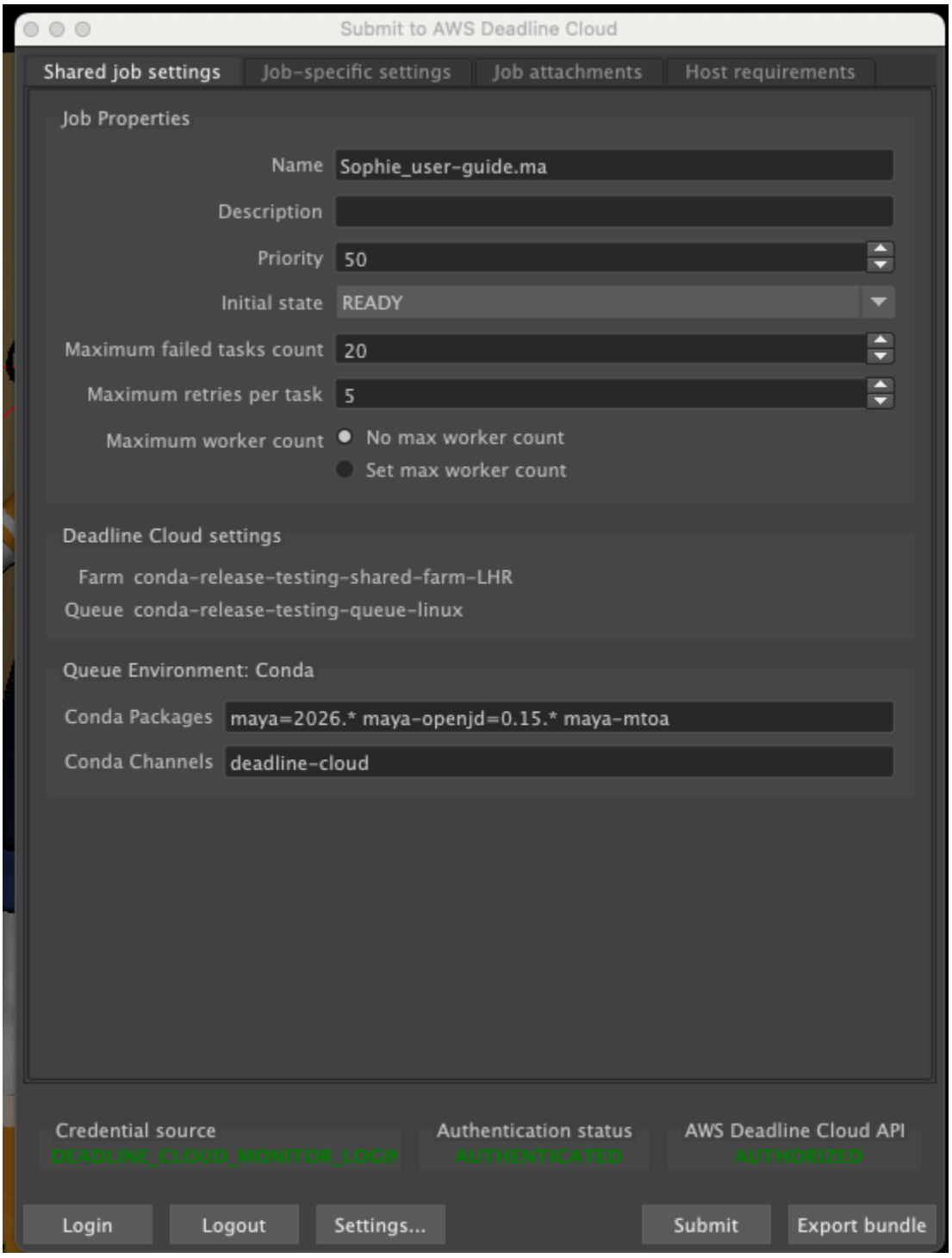
To use the Deadline Cloud submitter for Maya, ensure your farm is configured with a Maya-capable fleet, and have the submitter installed. For installation steps, see Installation. To access Deadline Cloud, log into Deadline Cloud monitor or provide AWS credentials through a configuration profile.

Submit a job

To submit a job from Maya to Deadline Cloud

1. Save your Maya file.
2. In Maya's shelf, choose the **Deadline Cloud** button.
3. Use the tabs in the dialog to customize your job.
4. (Optional) To export a job's associated files to your job history directory without submitting it, choose **Export bundle**.
5. Choose **Submit** and follow the prompts to send your job to Deadline Cloud.

Shared job settings

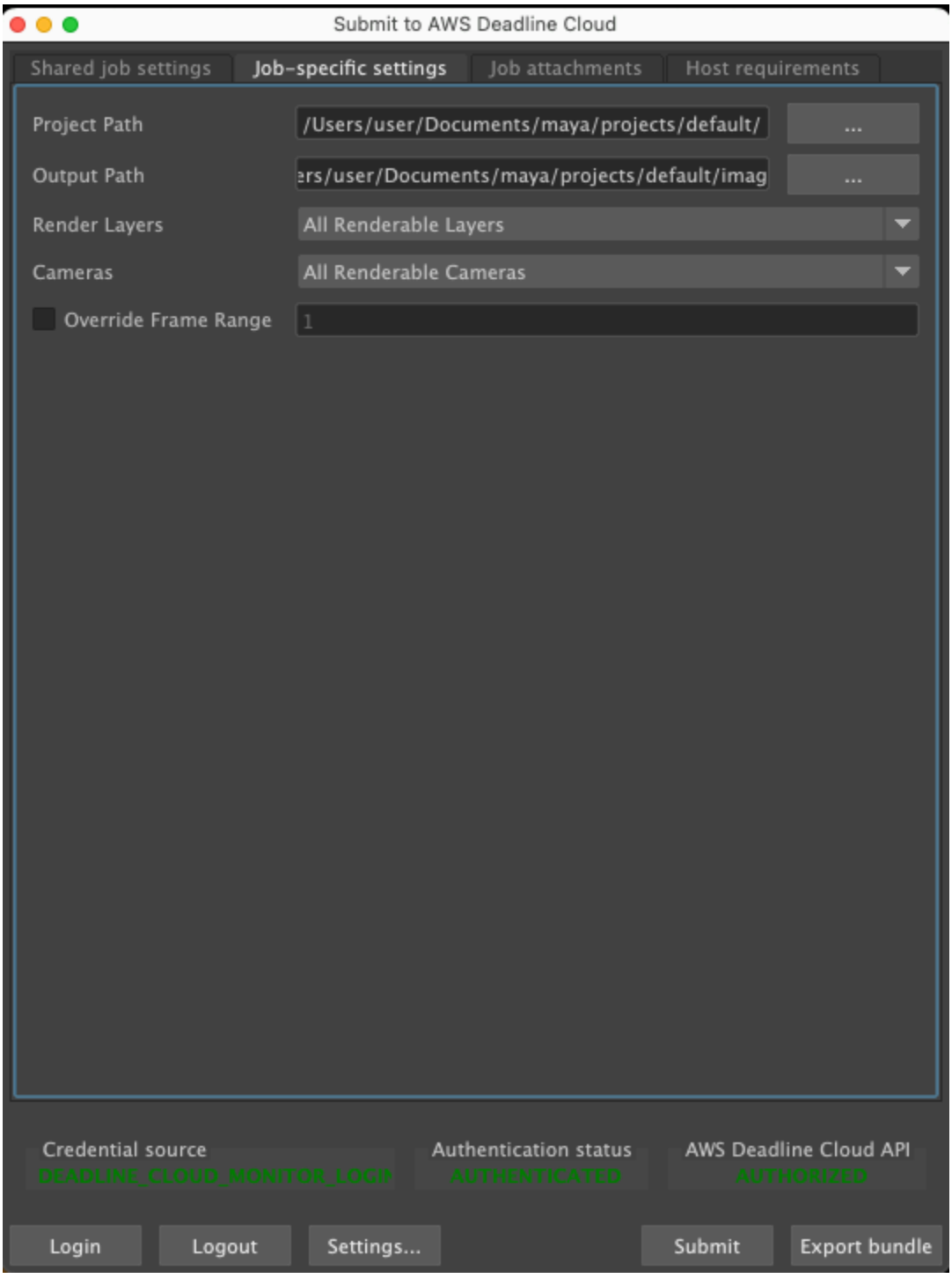


Settings that apply to the entire job:

- **Farm Selection** - Choose which farm your job will render on.
- **Queue Selection** - Select the specific queue within your chosen farm.

- **Job Name** - Give your render job a descriptive name.
- **Job Description** - Add optional details about your render job.
- **Priority** - Set job priority for queue management.
- **Initial State** - Control whether the job starts immediately or remains paused.
- **Max Failed Tasks Count** - Set the maximum number of tasks that can fail before the job is marked as failed.
- **Max Retries Per Task** - Set the number of times a failed task is retried.
- **Max Worker Count** - Set the maximum number of workers that can work on this job simultaneously.
- **Conda Packages** - Specify additional conda packages required for your render.
- **Conda Channels** - Define custom conda channels for package installation.

Maya-specific settings



Settings specific to Maya rendering:

- **Project Path** - Specify the Maya project path (automatically detected).
- **Output Path** - Specify the directory where rendered images are saved.

- **Output Filename** - Enter the base name for rendered image files.
- **Renderer** - Select the renderer to use (Arnold, V-Ray, Redshift, or Maya Software).
- **Cameras To Render** - Select specific cameras or render all renderable cameras.
- **Override Frame Range** - Optionally override the scene's frame range with custom values.
- **Render Layers** - Select which render layers to render.

Optional tabs

Options to modify the scene during submission:

- **Job Attachments** (optional) - Select which files will be uploaded and attached to the job. Files are automatically detected and attached by default.
- **Host Requirements** (optional) - Allows you to specify which types of hosts will be eligible for picking up tasks for this job.

For information about the submitter tabs, see the Deadline Cloud guide for using a submitter.

Advanced configurations

Using unsupported versions

Deadline Cloud only supports and tests the workstation and worker software versions in the table above. When using the submitter, the worker will attempt to install the same version as used on the workstation. This installation fails if the workstation version of Maya does not appear in the version table above.

If you require an unsupported version of Maya, you have the following options:

- When submitting the job from Maya, you can override the CondaPackages queue parameter to specify a supported version to use on the worker (for example, `maya=2026, maya-openjd=*`). This override might or might not work, depending on the features used by your scene and how Maya works with scenes from your workstation version.
- You can build a custom conda recipe and channel for your desired version to be installed on the worker. Use the following conda recipes for supported versions on the GitHub website as a starting point:
  - Maya conda recipe

- Maya OpenJD adaptor conda recipe

For more information about creating custom conda channels, see Creating custom conda channels.

Maya render engines

Maya supports multiple render engines that are fully compatible with Deadline Cloud:

Render Engine	Description	GPU Support	Notes	Usage-Based Licensing
Maya Software	Built-in CPU renderer	CPU-based	Legacy renderer with basic features	Included with Maya
Arnold (MtoA)	Monte Carlo ray tracer	GPU/CPU hybrid	Productio n quality rendering, MtoA 5.3.5+ required	Available for 2024-2026
V-Ray	Third-party photorealistic renderer	GPU/CPU hybrid	Requires separate licensing	Available for 2025-2026
Redshift	GPU-accelerated renderer	GPU optimized	Requires separate licensing	Available for 2025-2026

All render engines are automatically detected and configured by the Maya integrated submitter. The submitter maintains proper dependency handling and scene file management.

Maya plugins

The following table lists renderer plugins provided as conda packages. For custom plugin delivery options, see Use custom plugins with Deadline Cloud.

Plugin	Plugin Versions	Conda Recipe Provided	SMF Conda Package Provided	Usage-based Licensing Support
Arnold (MtoA)	2024.5.3, 2025.5.4, 2026.5.5	Yes	Yes	Yes
V-Ray	2025.7, 2026.7	Yes	Yes	Yes
Redshift	2025.4, 2026.2.1	Yes	Yes	Yes

Arnold for Maya (MtoA)

Arnold is supported using the maya-mtoa conda package and is automatically installed when using the Maya integrated submitter. An additional licensing cost applies when using Arnold for rendering.

For the recipe, see maya-mtoa-2026 conda recipe on the GitHub website.

V-Ray Plugin

V-Ray is supported using the maya-vray conda package and is automatically installed when using the Maya integrated submitter. An additional licensing cost applies when using V-Ray for rendering.

For the recipe, see maya-vray-2026 conda recipe on the GitHub website.

Redshift Plugin

Redshift is supported using the maya-redshift conda package and is automatically installed using the Maya integrated submitter. An additional licensing cost applies when using Redshift for rendering.

For the recipe, see maya-redshift-2026 conda recipe on the GitHub website.

Bifrost for Maya

You can use Autodesk Bifrost with Maya on Deadline Cloud. Because Bifrost requires the Autodesk installer, it isn't included in the deadline-cloud conda channel. Instead, you build your own conda

package with the maya-bifrost conda recipe and add it to a custom conda channel. The recipe provides Bifrost 2.14.1.0 for Maya 2026 on Linux workers. For more information about custom conda channels, see Creating custom conda channels.

For the recipe, see maya-bifrost conda recipe on the GitHub website.

Open source resources

The submitter and adaptor are open source and available on the GitHub website:

- Maya submitter source code
- Maya conda recipes

Autodesk VRED

Autodesk VRED is a professional 3D visualization and virtual prototyping software that brings complex 3D data to life in a realistic virtual environment. VRED is widely used by designers and engineers to create product presentations, design reviews, and virtual prototypes, particularly in the automotive industry. Use this guide to render with Deadline Cloud and Autodesk VRED by distributing tasks across multiple machines.

Support overview

VRED is partially supported by Deadline Cloud with the following components:

- **Submitters**: Integrated submitters for direct job submission from VRED Pro with automatic scene and asset detection.
- **Conda packages**: Automatic installation on service-managed fleets for Linux workers using the vredcore package.
- **Cross-platform compatibility**: Submitter support for Windows with worker support for Linux with automatic path mapping. (VRED Conda packages are available for Linux only; Windows workers require manual installation.)
- **BYOL Licensing**: VRED requires Bring Your Own License (BYOL). Unlike some other DCC applications in Deadline Cloud, usage-based licensing is not available for VRED. You must have valid VRED licenses available for your render farm fleet and configure your license server to be accessible from your workers.

VRED version compatibility

The following table shows current support levels for VRED versions:

Major Version	Submitter Support	Conda Support	Usage-Based Licensing
2026	Windows	Linux	BYOL required
2025	Windows	Linux	BYOL required

Deadline Cloud Conda Channel

The following table lists all conda packages applicable to VRED available to Service-managed fleets in the deadline-cloud conda channel:

OS	Package	Version	Notes
Linux	vredcore	2025	VRED Core for Linux
Linux	vredcore	2026	VRED Core for Linux

Requirements

To use VRED with Deadline Cloud, you need:

- VRED Pro or VRED Core 2025/2026 with valid licensing
- Python 3.11 or higher
- NVIDIA GPU driver 553.xx (recommended for optimal performance)
- Valid VRED licenses accessible from your render farm fleet
- Optionally: ImageMagick static binary for tile assembly when using region rendering with raytracing

Important

VRED integration requires **bring your own licensing (BYOL)**. You must have valid VRED licenses available for your render farm fleet and configure your license server to be accessible from worker nodes. For more information, see Connect service-managed fleets to a custom license server.

Getting started

To use VRED with Deadline Cloud:

1. Create a service-managed fleet and associate it with a queue. Ensure your fleet has access to your VRED license server.
2. Install the Deadline Cloud monitor and VRED submitter on your artist workstation using the Deadline Cloud Submitter and monitor installers. For more information, see Set up your workstation.
3. Open VRED and load your scene file.
4. Submit your job directly from VRED using the integrated submitter by selecting **Deadline Cloud** > **Submit to Deadline Cloud** from the menu.
5. Monitor the job and download the output using the Deadline Cloud monitor.

Installation

To install the Deadline Cloud submitter for Autodesk VRED, prepare the following environment:

- Windows 10+ workstation.
- VRED Pro 2025 or 2026 installation.
- Python 3.11 or higher.
- Access to an Deadline Cloud farm with either:
  - A service-managed fleet with VRED software and licensing configured.
  - A customer-managed fleet with VRED and licensing set up.

> **Important**
>
> Deadline Cloud for VRED requires bring your own licensing (BYOL). You must have valid VRED licenses available for your render farm fleet.

Installing the submitter

The Autodesk VRED submitter plugin allows you to submit jobs to Deadline Cloud directly from within VRED. To install the submitter:

1. Download the Deadline Cloud submitter installer.
2. Run the installer and follow the on-screen instructions.
3. Open VRED Pro.
4. Choose **Edit** > **Preferences**.
5. In the Preferences window, select **General Settings**, and then choose **Script**.
6. Clear **Enable Python Sandbox**.
7. In the Script section, add the following text to the end of the section:

```
from DeadlineCloudForVRED import DeadlineCloudForVRED
DeadlineCloudForVRED()
```

8. Choose **Save**.
9. Restart VRED Pro. When VRED opens, the **Deadline Cloud** menu displays in the menu bar.

For manual installation or developer workflows, see the manual installation instructions on the GitHub website.

Updating the submitter

To update the submitter to the latest version, download and run the latest submitter installer.

Using the Autodesk VRED submitter

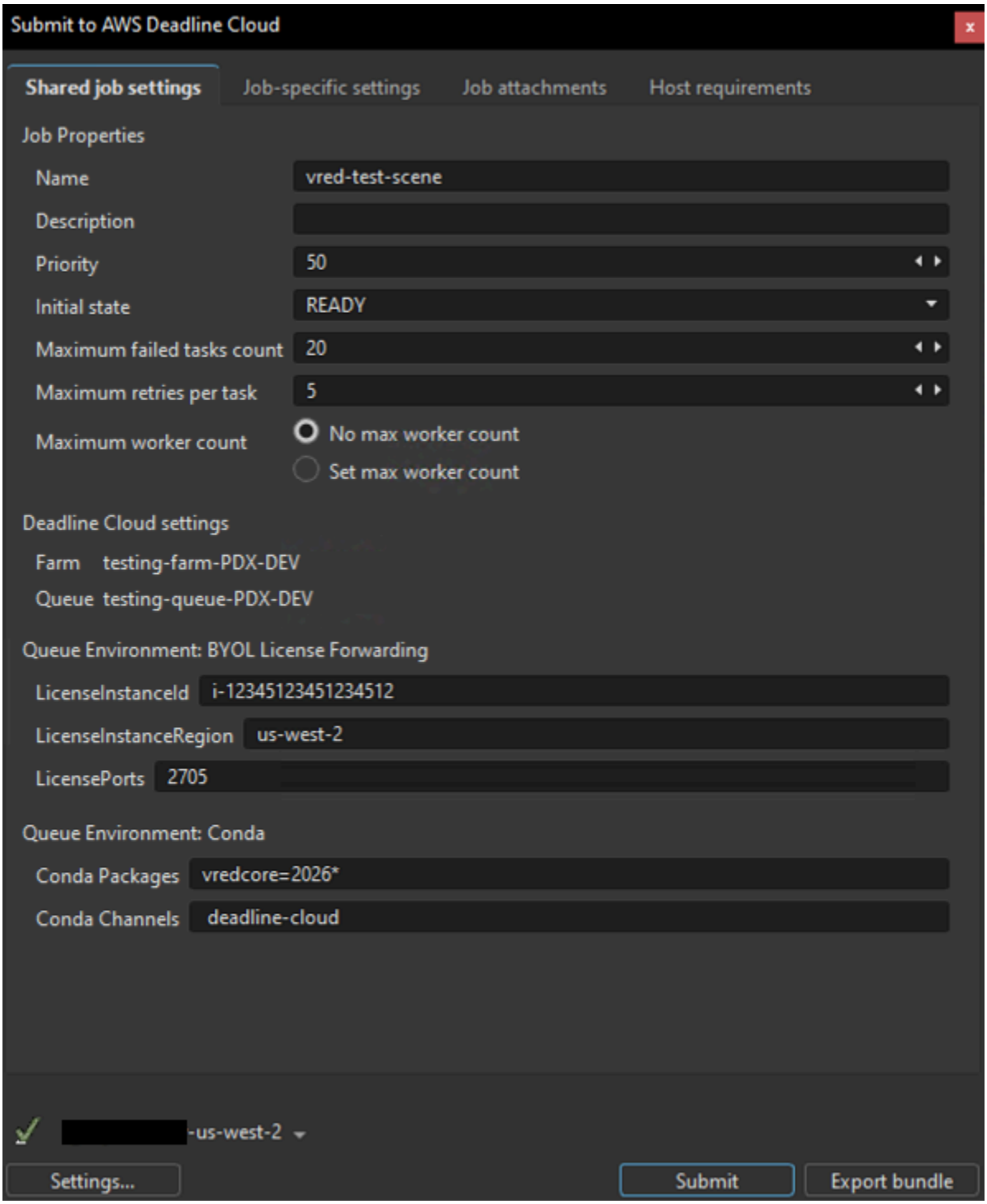
Before you use the Deadline Cloud submitter for VRED, make sure that your farm has a VRED-capable fleet configured and that the submitter is installed. You also need to authenticate with Deadline Cloud monitor or provide AWS credentials through a configuration profile.

Submitting a job

To submit a job from VRED to Deadline Cloud

1. Save your VRED scene file.
2. In VRED's menu bar, choose **Deadline Cloud** > **Submit to Deadline Cloud**.
3. If you have not already authenticated with Deadline Cloud, choose **Log in** and authenticate using your credentials in the browser window that appears.
4. Use the tabs in the dialog to customize your job.
5. (Optional) To export a job's associated files to your job history directory without submitting it, choose **Export bundle**.
6. Choose **Submit** and follow the prompts to send your job to Deadline Cloud.

Shared job settings



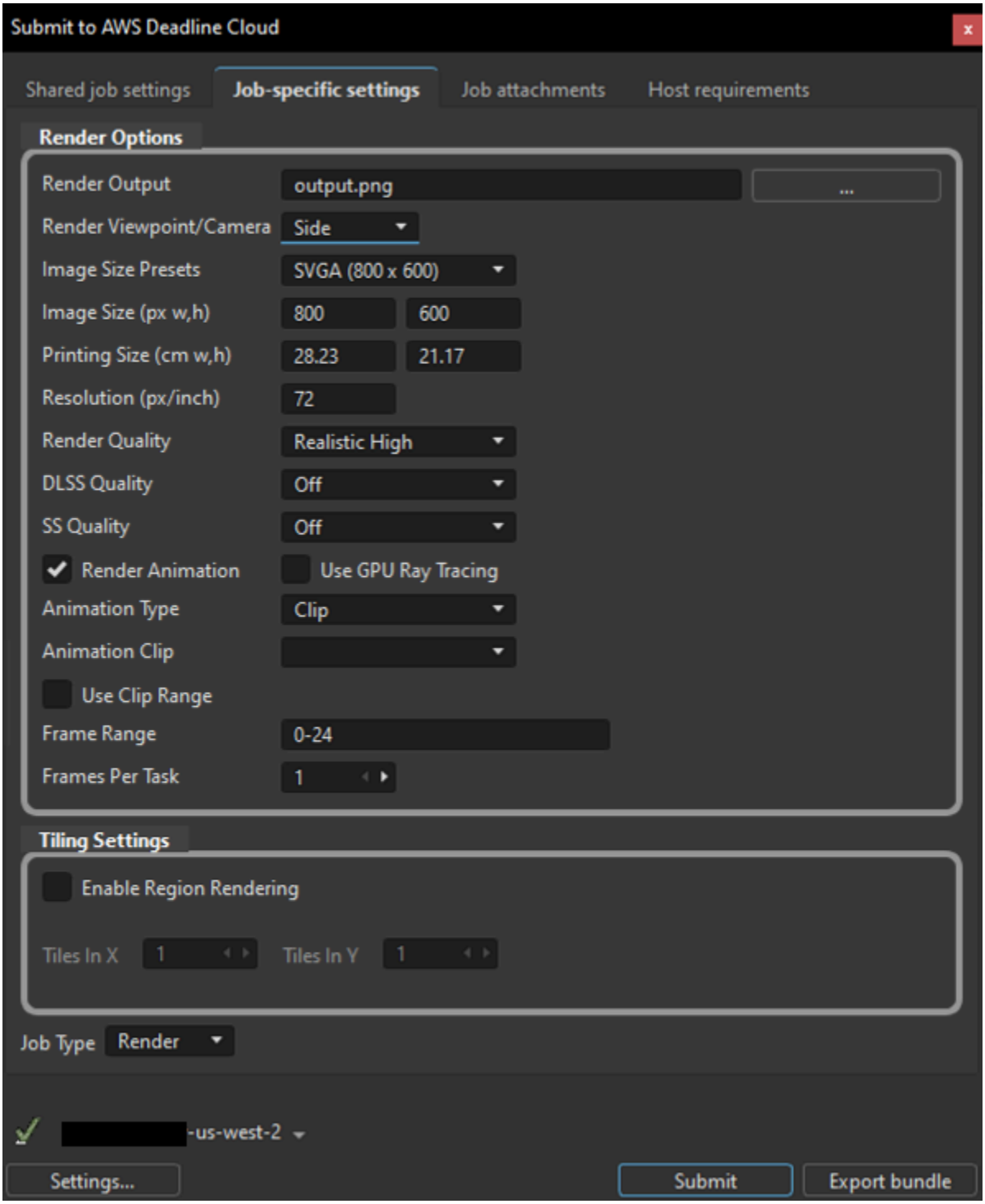
These settings apply to the entire job:

- **Name** - A descriptive name for your render job.
- **Description** - Optional details about your render job.

- **Priority** - Job priority for queue management (default: 50).
- **Initial State** - Whether the job starts immediately (READY) or remains paused.
- **Maximum failed tasks count** - Maximum tasks that can fail before the job is marked as failed (default: 20).
- **Maximum retries per task** - How many times a failed task will be retried (default: 5).
- **Maximum worker count** - Maximum workers that can process this job simultaneously.
- **Farm** - The farm where your job will render.
- **Queue** - The specific queue within your chosen farm.
- **Conda Packages** - Conda packages for the job environment (automatically configured for your VRED version).
- **Conda Channels** - Conda channels for package resolution.

VRED-specific settings

The **Job-specific settings** tab contains render options specific to VRED.



Render options

- **Render Output** - The output file path and base filename for rendered images. Use the browse button (**...**) to select the output directory and specify the filename.

- **Render Viewpoint/Camera** - Select which viewpoint or camera to render from. The dropdown lists all available viewpoints in your scene.
- **Image Size Presets** - Quick selection of common image dimensions (for example, SVGA 800x600, HD 1080, 4K).
- **Image Size (px w,h)** - Width and height in pixels for the rendered output. These values update automatically when you select a preset, or you can enter custom dimensions.
- **Printing Size (cm w,h)** - Physical print dimensions in centimeters. This value is linked to image size and resolution.
- **Resolution (px/inch)** - Dots-per-inch (DPI) setting that affects the relationship between image size and printing size (default: 72).
- **Render Quality** - Quality preset for rendering. Options include:
  - Analytic Low/High
  - Realistic Low/High
  - Raytracing
  - Non-Photorealistic (NPR)
- **DLSS Quality** - NVIDIA Deep Learning Super Sampling quality level (Off, Performance, Balanced, Quality, Ultra Performance). Requires compatible NVIDIA GPU.
- **SS Quality** - Super Sampling anti-aliasing quality (Off, Low, Medium, High, Ultra High). DLSS overrides this setting when enabled.
- **Render Animation** - When enabled, renders an animation sequence instead of a single frame. Reveals additional animation options.
- **Use GPU Ray Tracing** - Enable GPU-accelerated raytracing for higher quality renders. Requires compatible GPU hardware.
- **Animation Type** - Type of animation to render (Clip or Timeline). Visible when **Render Animation** is enabled.
- **Animation Clip** - Select which animation clip to render from the dropdown. Visible when **Animation Type** is set to Clip.
- **Use Clip Range** - When enabled, uses the frame range defined in the selected animation clip.
- **Frame Range** - Start and end frames to render (for example, `0-24`). You can specify a custom range or use the clip's range.
- **Frames Per Task** - Number of consecutive frames each worker renders per task (default: 1). Higher values improve rendering efficiency by reducing task initialization overhead, but provide

less granular progress tracking. For example, with Frames Per Task set to 5, Task 1 renders frames 1-5, Task 2 renders frames 6-10, and so on.

Tiling settings

Region rendering (tiling) divides each frame into smaller tiles that render independently as separate tasks, then assembles them into the final image. Tiling can improve performance for large, complex renders.

- **Enable Region Rendering** - Enables tile-based rendering. When enabled, **Use GPU Ray Tracing** is automatically activated.
- **Tiles In X** - Number of horizontal tile divisions (default: 1).
- **Tiles In Y** - Number of vertical tile divisions (default: 1).

> **Important**
>
> Region rendering requires **Use GPU Ray Tracing** to be enabled. Scene files must be properly configured for ray tracing (such as having sufficient lighting) to produce correct output. Using region rendering on scenes not set up for ray tracing will produce solid black tiles.

Tile assembly requirements: Region rendering requires ImageMagick to be available in the rendering environment for assembling tiles into the final image. One way to provide ImageMagick on service-managed fleets is to add the `imagemagick` conda package (for example, from the `conda-forge` channel) to your queue environment. For detailed instructions on configuring conda packages, see Configure jobs using queue environments.

Job type

- **Job Type** - Select the type of job to submit:
  - **Render** - Renders images or animation frames from the scene (default).
  - **Sequencer** - Executes VRED Sequencer workflows defined in your scene.

For information about the other submitter tabs (Job attachments, Host requirements), see the Deadline Cloud guide for using a submitter.

Advanced configuration

Using unsupported versions

Deadline Cloud only supports and tests the workstation and worker software versions in the table above. When using the submitter, the worker will attempt to install the same version as used on the workstation. This installation fails if the workstation version of VRED does not appear in the version table above.

If you require an unsupported version of VRED, you can build a custom conda recipe and channel for your desired version to be installed on the worker. Use the conda recipe for a supported version linked below as a starting point and package your desired version in a custom conda channel. For more information about creating custom conda channels, see Creating custom conda channels.

VRED render engines

VRED supports the following render engines for Deadline Cloud jobs:

Render Engine	Description	GPU Support	Notes
OpenGL	Real-time renderer	GPU optimized	Interactive visualization
Raytracing	High-quality renderer	GPU/CPU hybrid	Production quality rendering

Open source resources

The submitter and adaptor are open source and available on the GitHub website:

- VRED Submitter and Adaptor
- VRED Conda recipes are available for supported versions.

Blender

Blender is a free and open-source 3D computer graphics software toolset used for creating animated films, visual effects, art, 3D printed models, motion graphics, interactive 3D applications,

virtual reality, and computer games. Blender is supported by AWS Deadline Cloud (Deadline Cloud) with comprehensive integration including submitters, conda packages, and an adaptor for increased rendering performance. This guide provides step-by-step instructions for using Deadline Cloud with Blender to render your projects faster by distributing rendering tasks across multiple machines.

Support overview

Blender is supported by the following components:

- **Submitter**: Integrated submitter for direct job submission from Blender with automatic scene and asset detection.
- **Conda packages**: Deadline Cloud for automatic installation on service-managed fleets.
- **Adaptor**: A program on worker hosts that keeps the application loaded between tasks for faster rendering and reports render progress.
- **Cross-platform compatibility**: Submitter support for Windows, macOS, and Linux with worker support for Windows and Linux with automatic path mapping.

Blender version compatibility

The following table shows current support levels for Blender versions:

Major Version	Submitter Support	Conda Support	Render Engines
3.6	Windows, macOS, Linux	Linux	Cycles, Eevee, Workbench
4.2	Windows, macOS, Linux	Linux	Cycles, Eevee, Workbench
4.5	Windows, macOS, Linux	Linux	Cycles, Eevee, Workbench
5.0	Windows, macOS, Linux	Linux	Cycles, Eevee, Workbench
5.1	Windows, macOS, Linux	Linux	Cycles, Eevee, Workbench

Deadline Cloud Conda Channel

The following table lists all conda packages applicable to Blender available to Service-managed fleets in the deadline-cloud conda channel:

OS	Package	Version	Notes
Linux	blender	3.6	Includes all built-in render engines
Linux	blender	4.2	Includes all built-in render engines
Linux	blender	4.5	Includes all built-in render engines
Linux	blender	5.0	Includes all built-in render engines
Linux	blender	5.1	Includes all built-in render engines
Linux	blender-openjd		Includes the Blender Adaptor

Getting started

To use Blender with Deadline Cloud:

1. Create a service-managed fleet and associate it with a queue. Your queue must be set up with a queue environment that supports the deadline-cloud conda channel. For more information, see Creating a queue environment.
2. Install the Deadline Cloud monitor and Blender submitter on your artist workstation using the Deadline Cloud monitor and submitter installers. For more information, see Set up your workstation.
3. Submit your job directly from Blender using the integrated submitter to the queue.
4. Monitor the job and download the output using the Deadline Cloud monitor.

Installation

To install the Deadline Cloud for Blender submitter, you need:

- A Windows, macOS, or Linux workstation.
- Blender 3.6 or later.

There are three ways to install the Deadline Cloud for Blender submitter:

- Using the Deadline Cloud submitter installer (recommended).
- Installing the submitter from Blender.
- Manually installing the submitter from source. For more information, see Manual Installation on the GitHub website.

Using the Deadline Cloud submitter installer

You can install the Deadline Cloud for Blender submitter using the Deadline Cloud submitter installer.

To install the submitter

1. Download the Deadline Cloud submitter installer.
2. Run the installer.
   - When prompted, select each version of Blender you want to use the submitter with.
3. Launch Blender.
4. Verify the installation by checking the **Render** menu for a **Submit to Deadline Cloud** option.

If the add-on is not available from the **Render** menu, you need to manually enable it.

To manually enable the submitter add-on

1. On the **Edit** menu, choose **Preferences...**.
2. Choose **File Paths** on the left side bar.
3. Find the **Script Directories** section and choose **+**.
4. For **Name**, enter `python`.

5. For **Path**, enter the path to the `python` directory in your Blender submitter installation.
6. Restart Blender for changes to take effect.

Installing the submitter from Blender

> **Note**
>
> This feature is experimental and subject to change.

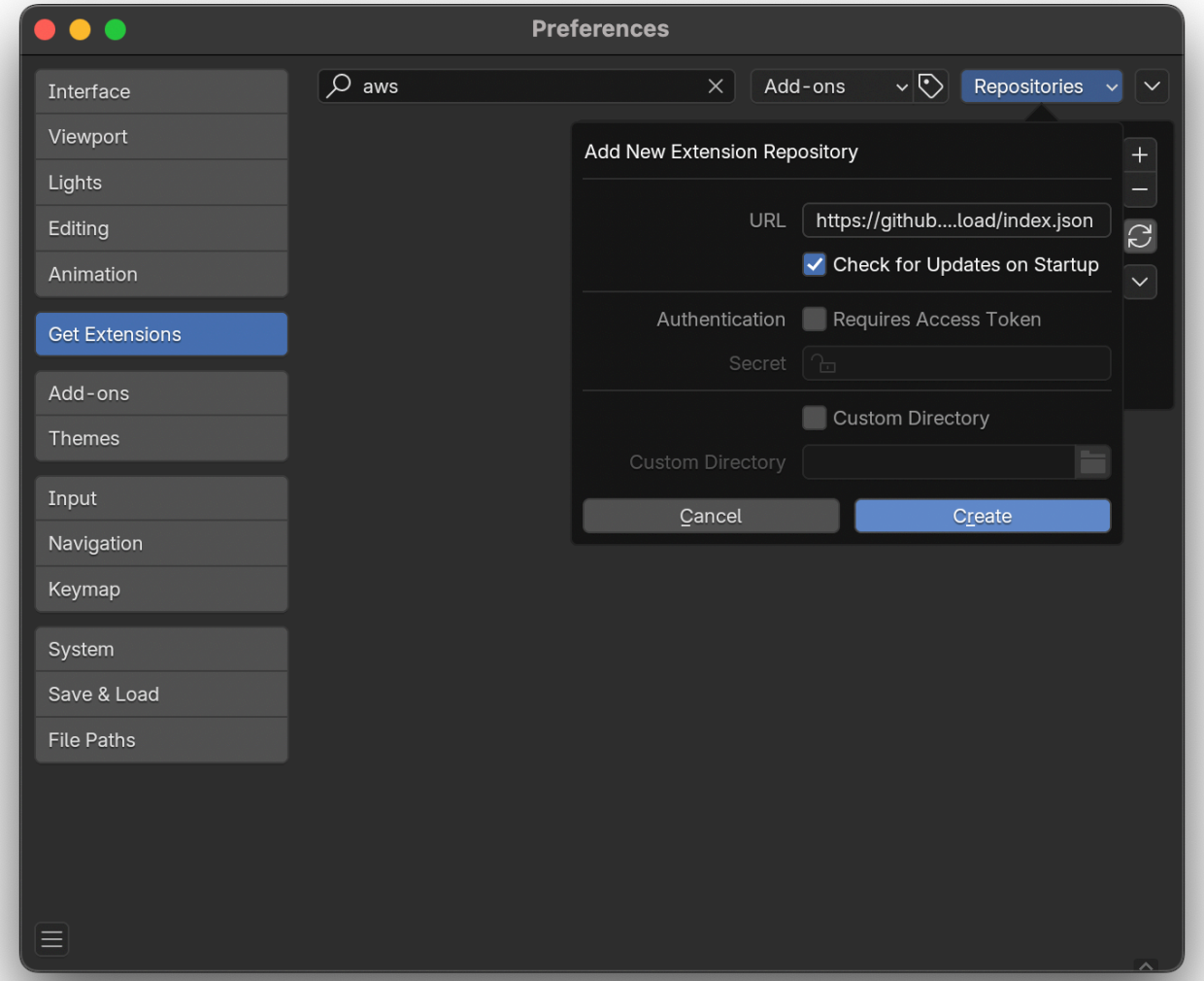
You can install and update the Blender submitter from within Blender using Blender's extension feature.

To install the Blender submitter using Blender extensions, you need:

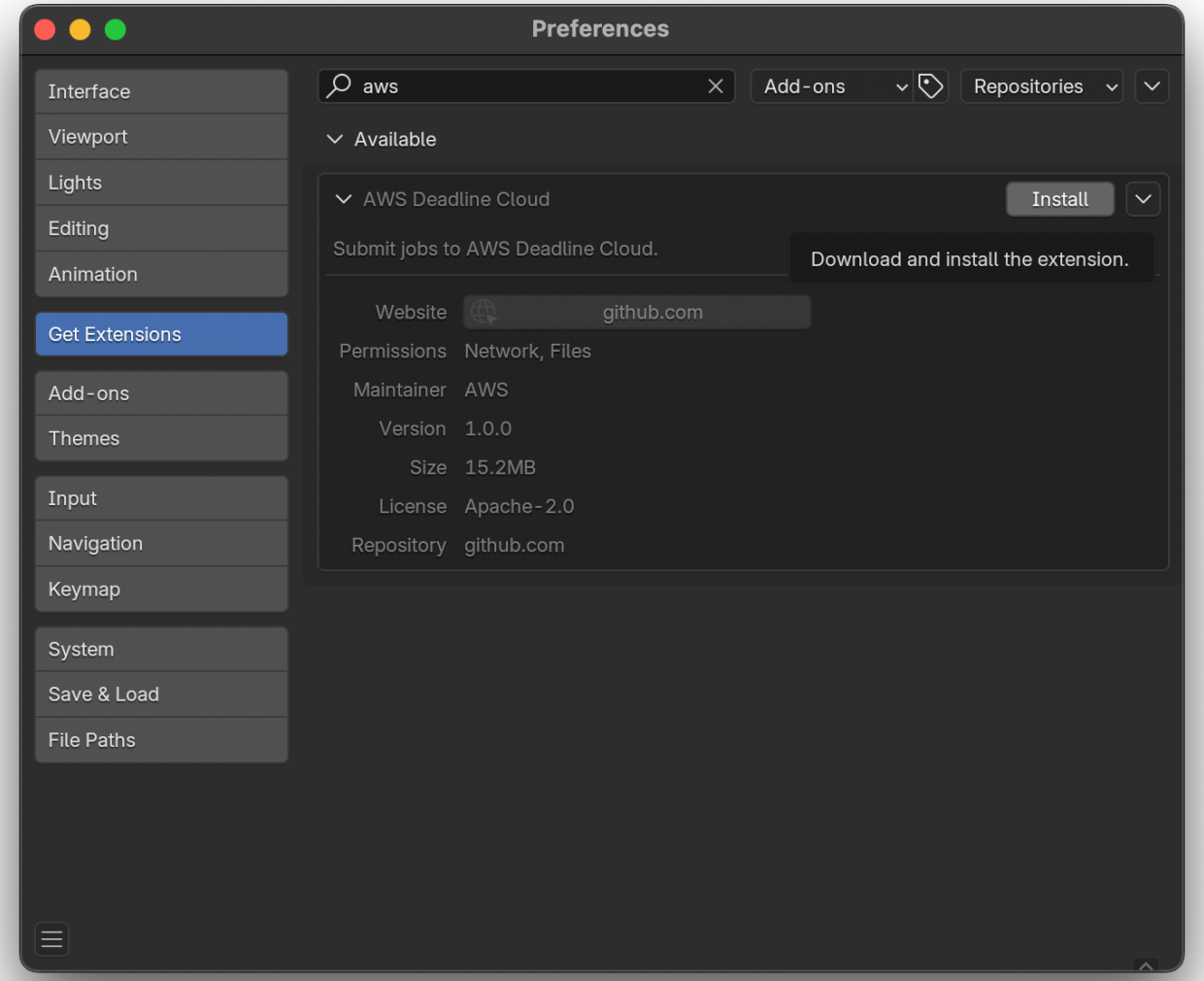
- Blender 4.2 or later.
- A workstation with consistent internet access.

To add the Blender submitter as an extension

1. Open Blender.
2. On the **Edit** menu, choose **Preferences...**.
3. Choose **Get Extensions** on the left side bar.
4. Choose **Repositories**, **+**, **Add Remote Repository**.



5. For **URL**, enter `https://github.com/aws-deadline/deadline-cloud-for-blender/releases/latest/download/index.json`.
6. Select **Check for Updates on Startup** and choose **Create**.
7. On the **Deadline Cloud** entry under **Available**, choose **Install**.



The add-on is now installed. You can use the new **Submit to Deadline Cloud** option in the **Render** menu.

When an update is available, an **Update** button appears next to the **Deadline Cloud** entry in the **Get Extensions** section.

Using the Blender submitter

To use the Deadline Cloud for Blender submitter, you need:

- A profile to submit to Deadline Cloud with.

- An Deadline Cloud farm and queue to submit to.

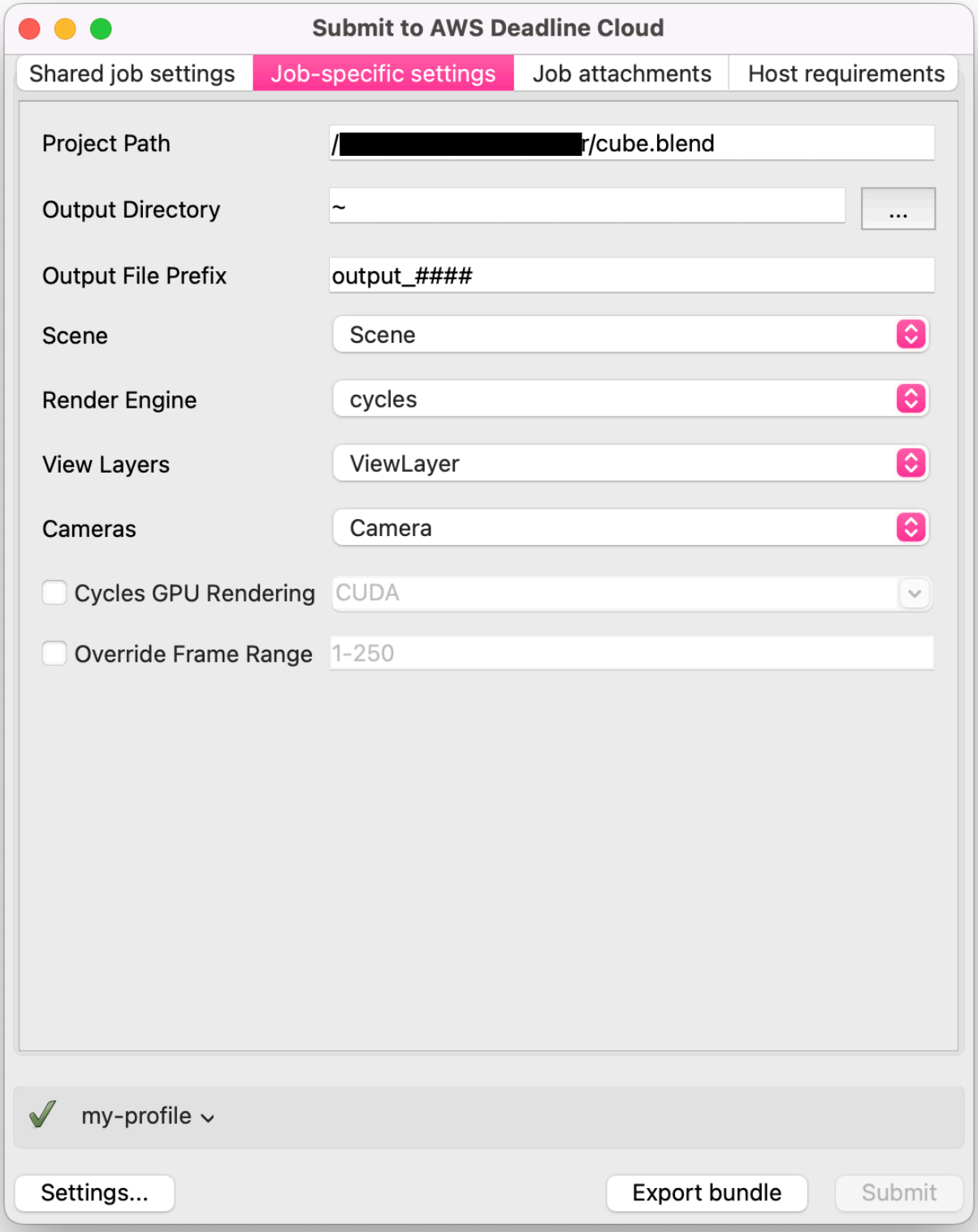
Submit a job

To submit a job from Blender to Deadline Cloud

1. Save your Blender file.
2. On the **Render** menu, choose **Submit to Deadline Cloud**.
   - You might see a pop-up to install GUI dependencies. Choose **OK** and wait for the dialog to disappear, then choose **Submit to Deadline Cloud** again.
3. Use the tabs in the dialog to customize your job.
4. (Optional) To export a job's associated files to your job history directory without submitting it, choose **Export bundle**.
5. Choose **Submit** and follow the prompts to send your job to Deadline Cloud.

Blender-specific settings

The **Job-specific settings** tab has options specific to jobs created in Blender.


Submit to AWS Deadline Cloud
Shared job settings
Job-specific settings
Job attachments
Host requirements
Project Path
/[redacted]r/cube.blend
Output Directory
~
...
Output File Prefix
output_####
Scene
Scene
Render Engine
cycles
View Layers
ViewLayer
Cameras
Camera
Cycles GPU Rendering
CUDA
Override Frame Range
1-250
my-profile
Settings...
Export bundle
Submit

- *Project Path* - The location where the current project is saved. This value can't be changed.
- *Output Directory* - The location to save file outputs from the render job.
- *Output File Prefix* - The pattern to use when naming file outputs, follows Blender's convention for file names. Output files are formatted like `[LayerName]_[CameraName]_[OutputPrefix].[EXT]`.
- *Scene* - The scene from the current project to render.
- *Render Engine* - The render engine (Cycles, EEVEE, or Workbench) to use.
- *View Layers* - The layer to render, or "All Renderable Layers" to render each applicable layer in the scene separately.
- *Cameras* - The camera to render, "All Renderable Cameras" to render each camera in the scene separately, or "Use Default Camera" to use the scene's default camera or cameras bound to timeline markers.
- *Cycles GPU Rendering* - Whether to enable GPU rendering. Choose a device type supported by Blender or specify your own. If this device type is not supported on your rendering machine, the adaptor attempts to use a compatible device type before falling back to CPU rendering.
- *Override Frame Range* - Select this option to render a different frame or frame range than is set in the scene file. Frame ranges follow the Open Job Description pattern. For more information, see the IntRangeExpr specification on the GitHub website.

For information about the other submitter tabs, see the Deadline Cloud guide for using a submitter.

Advanced configurations

Using unsupported versions

Deadline Cloud only supports and tests the workstation and worker software versions in the table above. When using the submitter, the worker will attempt to install the same version as used on the workstation. This installation fails if the workstation version of Blender does not appear in the version table above.

If you require an unsupported version of Blender, you have the following options:

- When submitting the job from Blender, you can override the CondaPackages queue parameter to specify a supported version to use on the worker (for example, `blender=4.5, blender-`

`openjd=*`). This override might or might not work, depending on the features used by your scene and how Blender works with scenes from your workstation version.

- You may build a custom conda recipe and channel for your desired version to be installed on the worker. Use the conda recipe for a supported version linked below as a starting point, and package your desired version in a custom conda channel. For more information about creating custom conda channels, see Creating custom conda channels.

Use custom Blender add-ons

You can make custom Blender add-ons available with Deadline Cloud plugin sync or a conda package. Plugin sync copies an add-on from your queue's job attachments bucket when the worker session starts. For setup options, see Use custom plugins with Deadline Cloud.

For add-ons that require installation steps or dependency resolution, build a conda package from a conda recipe and add it to a custom conda channel. When the conda environment activates on a worker, the package installs the add-on through Blender's API. When the environment deactivates, it removes the add-on. For more information, see Create a conda channel using S3 in the *Deadline Cloud Developer Guide*.

The following conda recipes are available on the GitHub website as starting points:

- FLIP Fluids add-on conda recipe packages the demo version of the FLIP Fluids liquid simulation add-on.
- Blender add-on bundle conda recipe packages multiple Blender add-ons from a single archive so that they install together when the conda environment activates.

Blender render engines

Blender includes several built-in render engines that are supported:

Render Engine	Description	GPU Support	Notes
Cycles	Physically-based path tracer	GPU/CPU hybrid	Production quality rendering with GPU acceleration

Render Engine	Description	GPU Support	Notes
Eevee	Real-time render engine	GPU optimized	Fast viewport and final rendering
Workbench	Solid shading engine	GPU optimized	For modeling and sculpting workflows

All render engines are automatically detected and configured by the Blender integrated submitter. GPU acceleration is available when using service-managed fleets with GPU-enabled instances.

Open source resources

The submitter and adaptor are open source and available on the GitHub website:

- Deadline Cloud for Blender
- Blender conda recipes are available for supported versions.

Epic Unreal Engine

Unreal Engine is a real-time 3D creation tool for photoreal visuals and immersive experiences. Unreal Engine is supported by Deadline Cloud with submitters, conda packages, and an adaptor for increased rendering performance. This guide provides step-by-step instructions for using Deadline Cloud with Unreal Engine to render your Movie Render Queue projects faster by distributing rendering tasks across multiple machines.

Support overview

Unreal Engine is supported by the following components:

- **Submitter**: Integrated submitter plugin for direct job submission from Unreal Engine with automatic scene and asset detection.
- **Conda packages**: Deadline Cloud for automatic installation on service-managed fleets.
- **Adaptor**: A program on worker hosts that keeps the application loaded between tasks for faster rendering and reports render progress.
- **Cross-platform compatibility**: Submitter and worker support for Windows only.

- **Movie Render Queue Integration**: Support for Unreal's Movie Render Queue system.

Unreal Engine version compatibility

The following table shows current support levels for Unreal Engine versions:

Major Version	Submitter Support	Conda Support
5.4	Windows	Windows
5.5	Windows	Windows
5.6	Windows	Windows
5.7	Windows	Windows
5.8	Windows	Windows

Deadline Cloud Conda Channel

The following table lists all conda packages applicable to Unreal Engine available to Service-managed fleets in the `deadline-cloud` conda channel:

OS	Package	Version
Windows	unreal-engine	5.4
Windows	unreal-engine	5.5
Windows	unreal-engine	5.6
Windows	unreal-engine	5.7
Windows	unreal-engine	5.8
Windows	unreal-engine-openjd	

Getting started

Prerequisites

Before installing the Unreal Engine submitter, ensure you have the following:

- Windows workstation (Windows 10 or later)
- Supported version of Unreal Engine installed
- Deadline Cloud monitor installed (download here)
- Access to an Deadline Cloud farm with either a GPU-enabled Windows service-managed fleet or a customer-managed fleet with Unreal Engine, the Unreal Engine adaptor, and licensing set up

Installing the submitter

The Unreal Engine submitter adds Deadline Cloud functionality as a plugin to Unreal Engine, allowing you to submit your Movie Render Queue jobs directly to Deadline Cloud for rendering.

Choosing your branch

Select the appropriate branch for your deployment:

Branch	Stability	Use case	Recommended for
release	Stable	Production	Most users
mainline	Latest features	Development and testing	Advanced users

> **Tip**
>
> Use the **release** branch for production environments to ensure stability.

Creating a new Windows Amazon EC2 instance to install Unreal on (optional)

If you're setting up on a brand new Windows Amazon Elastic Compute Cloud (Amazon EC2) instance as your submitter, a g5.2xlarge instance with 200 GB of storage is a reasonable minimum:

1. Launch an Amazon EC2 instance with a valid Instance Profile. The profile is required to download NVIDIA GRID drivers as instructed below.
2. Download the Epic Installer and install a supported version of Unreal Engine (5.4 through 5.8).
   - UE 5.5 has a known crash bug when running with the DirectX 11 plugin (see UE issue #UE-276282). If you need DirectX support on UE 5.5, use DirectX 12 or later.
3. NVIDIA GRID drivers - Follow the Windows installation instructions.

Windows long paths

Many of the following steps might create files that exceed the default Windows maximum path length. Before you build and install the Deadline Cloud for Unreal Engine submitter or adapter on a Windows machine, we recommend that you enable Windows long path support. For more information, see Maximum file path limitation on the Microsoft website, for example by running the PowerShell command.

Long path support where you build the submitter is separate from long path support at render time. Windows honors `LongPathsEnabled` only for applications that declare `longPathAware` in their manifest, so enabling it on a worker host doesn't lift the limit for every application. If jobs fail on Windows workers with missing-file errors, see Why does my job fail on Windows when my file paths are long?.

Installing build tools

The Unreal submitter plugin currently must be compiled locally.

1. Install Visual Studio using the Visual Studio Installer on the Microsoft website.
2. Verify your Visual Studio and build tools version are compatible with your version of Unreal by checking the Epic compatibility table on the Epic Games website.
3. Under **Individual Components**, ensure that the MSVC build tools version selected ("Latest" by default) matches the recommended version in the table. Even though the compatibility guidance may suggest a version "or later", build errors sometimes occur when using a newer version than the one listed as "recommended".
4. Under **Individual Components**, select a recent .NET Framework SDK (4.6.1 and 4.8.1 have been verified).
5. Under **Workloads**, select **Desktop development with C++**.

Installing Deadline Cloud monitor

Deadline Cloud monitor is used to both manage your credentials for submitting jobs to Deadline Cloud and monitor the status of your jobs.

1. Follow the Deadline Cloud monitor installation instructions.
2. Sign in.

Environment setup

1. (If not already installed) Install a recent version of Python for all users (verified with 3.12).
2. Make sure your environment variables are set correctly. In System Environment Variables, your PATH must include:
   - The path to your Python installation (for example, `C:\Program Files\Python312`).
   - The path to your Python Scripts folder (for example, `C:\Program Files\Python312\scripts`).
   - The path to your Unreal binaries (for example, `C:\Program Files\Epic Games\UE_5.5\Engine\Binaries\Win64`).

Deadline Cloud software installation

Clone or download `deadline-cloud-for-unreal-engine` from either the release branch or mainline, depending on whether you want the most recent tested release or all of the most recent commits.

```
git clone https://github.com/aws-deadline/deadline-cloud-for-unreal-engine.git
cd deadline-cloud-for-unreal-engine
git switch release
```

Optional - Build and install plugin with script

A helper script at `scripts/build_plugin.py` optionally automates the next two steps. It attempts to find the latest version of Unreal, builds your plugin and Python dependencies, and installs them in the correct locations. You can override settings such as the Unreal version. To see the full help list, run:

```
python scripts/build_plugin.py -h
```

To build and install your current copy of `deadline-cloud-for-unreal-engine` as a submitter with the latest Unreal Engine installation, run:

```
python scripts/build_plugin.py --install
```

If you've installed with this script successfully, you can skip to Creating a fleet.

Build the plugin

Adjust the first two paths below based on where your installation of Unreal lives, and where you installed `deadline-cloud-for-unreal-engine`.

From the Unreal Install Batchfiles folder (the `package` parameter can be any new directory, however you'll want it to be called `UnrealDeadlineCloudService` later):

```
cd C:\Program Files\Epic Games\UE_5.5\Engine\Build\BatchFiles
runuat.bat BuildPlugin -plugin="C:\deadline\deadline-cloud-for-unreal-
engine\src\unreal_plugin\UnrealDeadlineCloudService.uplugin" -package="C:
\UnrealDeadlineCloudService"
```

Copy the "package" folder above to your Unreal installation's Plugins folder (for example, `C:\Program Files\Epic Games\UE_5.5\Engine\Plugins\UnrealDeadlineCloudService`).

Python dependencies

There are 4 ways to install the required Python dependencies.

1. If you've built and installed the plugin from the release branch above, you can install from pip. Use the following install command, adjusting the paths to your Unreal installation:

```
"C:\Program Files\Epic Games\UE_5.5\Engine\Binaries\ThirdParty\Python3\Win64\python"
 -m pip install deadline-cloud-for-unreal-engine --target "C:\Program Files\Epic Games
\UE_5.5\Engine\Plugins\UnrealDeadlineCloudService\Content\Python\libraries"
```

2. Alternatively in your `.uplugin` file (in the above steps this would live at `C:\Program Files\Epic Games\UE_5.5\Engine\Plugins\UnrealDeadlineCloudService\UnrealDeadlineCloudService.uplugin`) you can add a `PythonRequirements` section which matches the latest release of `deadline-cloud-for-unreal-engine` in GitHub/PyPI, for example:

```
"PythonRequirements":
```

```
[
    {
        "Platform": "All",
        "Requirements":
        [
            "deadline-cloud-for-unreal-engine>=0.5.0"
        ]
    }
]
```

You may wish to deactivate the "strict hash" feature in Unreal's Python settings, or add hash settings for specific library and dependency versions you wish to consume.

3. If you're pulling from mainline you may have Python dependencies which are not yet released to PyPI - you'll need to build and install your local copy which can be done with hatch. The `.whl` file will need to be changed to reflect the version which is output by hatch build:

```
pip install hatch
hatch build
"C:\Program Files\Epic Games\UE_5.5\Engine\Binaries\ThirdParty\Python3\Win64\python"
 -m pip install dist\deadline_cloud_for_unreal_engine-0.2.2.post21-py3-none-any.whl --
target "C:\Program Files\Epic Games\UE_5.5\Engine\Plugins\UnrealDeadlineCloudService
\Content\Python\libraries"
```

4. Lastly, Python dependencies can be installed by the submitter installer. These may be out of date with your code above from the release or mainline branch, and this method should not currently be preferred.

1. Download the submitter installer from the Deadline Cloud AWS console's Downloads tab or from within Deadline Cloud monitor under Workstation Setup → Downloads.
2. Run the installer for all users. Default install location is fine.
3. Enable the Unreal Engine plugin.
4. Make sure the Unreal Engine plugin install path matches where your plugin was copied to (in particular make sure your Unreal version matches).

Update notifications

The submitter plugin automatically checks for newer releases on GitHub when Unreal Editor starts. If an update is available, a dialog will prompt you to visit the release page.

To deactivate update notifications, uncheck **Show submitter update notifications** under **General Settings** in the Deadline Cloud settings panel (**Edit** > **Project Settings** > **Plugins** > **Deadline Cloud**).

Alternatively, you can use the CLI:

```
deadline config set settings.submitter_update_notification false
```

To re-enable:

```
deadline config set settings.submitter_update_notification true
```

Creating a fleet

If you already have a Windows fleet and don't need to set up a new fleet, you can skip down to Submitting a test render.

Creating a service-managed fleet (SMF)

Follow the service-managed fleets user guide to create an SMF if you don't already have one. On service-managed fleets (SMFs), the Unreal Engine and adaptor are automatically available using the `deadline-cloud` conda channel with the default queue environment. You are ready to start rendering. Continue with Submitting a test render below to submit a test render job.

Creating a customer-managed fleet (CMF)

1. Follow Create a customer-managed fleet to create a CMF if you don't already have one.

   > **Warning**
   >
   > When associating your CMF to queues, remove the default conda queue environment if you do not use it. This removal prevents the conda environment from being used and accidentally using the default SMF-specific variables for jobs submitted to your CMF. If you use conda in your CMF, remember to update `CondaPackages` and `CondaChannels` variables in **Parameter Definition Overrides** during job submission.

2. Follow Worker host setup and configuration to set up a worker host.
3. Follow Manage access to Windows job user secrets to set up the Windows job user secrets for your CMF worker.

4. Follow Install and configure software required for jobs to install the software required to run jobs.
5. Follow Setting up a customer-managed fleet (CMF) worker to set up your worker node to run Unreal Engine jobs.

Submitting a test render

This example uses the Meerkat Demo from the Unreal Marketplace:

1. Start the Epic Games Launcher.
2. Install the Meerkat Demo from the **Samples** tab.
3. Create a project from the Meerkat Demo, then open it.
4. From the **Edit** menu, select **Plugins**, search for and enable **UnrealDeadlineCloudService**.
5. Restart Unreal if you've enabled the plugin for the first time.
6. Under **Edit** > **Project Settings**, search for the **Movie Render Pipeline** section.
   - For **Default Remote Executor**, select **MoviePipelineDeadlineCloudRemoteExecutor**.
   - For **Default Executor Job**, select **MoviePipelineDeadlineCloudExecutorJob**.
   - Under **Default Job Settings Classes**, choose the add icon, and add **DeadlineCloudRenderStepSetting**.
7. Search for **Deadline Cloud** settings and verify authentication:
   - Ensure your Status shows "AUTHENTICATED" and Deadline Cloud API shows "AUTHORIZED".
   - If it does not appear, first try using the **Login** button. If that doesn't work, open Deadline Cloud monitor and ensure you're logged in.
   - In the **Deadline Cloud Workstation Configuration** section:
     - Under **Global Settings**, ensure your AWS Profile is set correctly to your Deadline Cloud monitor profile.
     - Under **Profile**, ensure your Default Farm is set to your farm.
     - Under **Farm**, ensure your Default Queue is set to a queue that is associated with the fleet you set up above.
8. Exit the Project Settings window.
9. Choose **Windows**, **Cinematics**, **Movie Render Queue**.
   - Choose **+ Render**, and select **Main_SEQ**.

- Choose **UnsavedConfig** in the settings column.
  - In the popup window, you should see Deadline Cloud settings on the left. This window can then be closed.
- On the right side of the dialog, configure the job settings:
  - Under **Preset Overrides** (you may need to widen this dialog):
    - Expand **Job Shared Settings**:
      - Set **Name** to `Unreal Test Job`.
      - Set **Maximum retries** to 2.
    - Expand **Job Attachments**:
      - Under **Input Files**, select **Show Auto-Detected**.
      - Verify that the list of auto-detected files populates correctly.
  - Under **Job Template Overrides**:
    - Update the Unreal Engine version in **CondaPackages** if you are using a different version than 5.6.
      - Unreal Engine version autodetection is coming in a future release.
- Choose **Render (Remote)**.

10. You can go to Deadline Cloud monitor and watch the progress of your job.

Setting up a customer-managed fleet (CMF) worker

This section walks you through setting up an Amazon EC2 instance as a CMF worker for Deadline Cloud with Unreal Engine.

Overview

The following are key differences between CMF and SMF:

- **CMF**: Manual installation of Unreal Engine and adaptor on worker hosts.
- **SMF**: Automatic availability through the `deadline-cloud` conda channel.

Choosing your branch

Select the appropriate branch for your deployment:

Branch	Stability	Use case
release	Stable	Production deployments
mainline	Latest	Development and testing

> **Important**
>
> Ensure your worker version matches your submitter version to avoid compatibility issues.

Amazon EC2 instance setup

The following is the recommended instance configuration:

- **Instance Type**: g5.2xlarge or higher.
- **Storage**: 200 GB minimum.
- **OS**: Windows Server 2019/2022.

Software installation

1. Install Unreal Engine:

1. Download the Epic Games Launcher.
2. Install Unreal Engine 5.4 or higher.

> **Note**
>
> Unreal Engine 5.4+ is required for Deadline Cloud compatibility.

2. Install NVIDIA GRID drivers:

- Follow the AWS NVIDIA GRID driver installation guide.
- Required for GPU-accelerated rendering on Amazon EC2 instances.

Installing build tools

The Unreal plugin currently must be compiled locally.

1. Install Visual Studio using the Visual Studio Installer on the Microsoft website.
2. Verify your Visual Studio and build tools version are compatible with your version of Unreal by checking the Epic compatibility table on the Epic Games website.
3. Under **Individual Components**, ensure that the MSVC build tools version selected ("Latest" by default) matches the recommended version in the table.
4. Under **Individual Components**, select a recent .NET Framework SDK (4.6.1 and 4.8.1 have been verified).
5. Under **Workloads**, select **Desktop development with C++**.

Deadline Cloud software installation

Clone or download `deadline-cloud-for-unreal-engine` from either the release branch or mainline. Ensure your worker version of the libraries is compatible with the version being used from your submitters.

```
git clone https://github.com/aws-deadline/deadline-cloud-for-unreal-engine.git
cd deadline-cloud-for-unreal-engine
git switch release
```

Optional - Build and install plugin and dependencies with script:

A helper script exists at `scripts/build_plugin.py` that can automate the remaining installation steps. It attempts to find the latest version of Unreal, build your plugin and python dependencies, and install them in the correct locations. Settings such as the Unreal version to use can be overridden. To see the full help list, run:

```
python scripts/build_plugin.py -h
```

To build and install your current copy of `deadline-cloud-for-unreal-engine` as a worker with the latest Unreal Engine installation, run:

```
python scripts/build_plugin.py --install --worker
```

Configure the Deadline Cloud worker agent by running:

```
install-deadline-worker ^
  --farm-id FARM_ID ^
  --fleet-id FLEET_ID ^
  --region REGION ^
  --allow-shutdown
```

If you've installed with this script and configured the worker agent successfully, you can skip to Starting the Deadline Cloud worker agent service.

```
python -m pip install deadline-cloud-worker-agent
```

The correct version of the adaptor must be installed depending on the version of the submitter being used. If you are using the version of the submitter from the release branch in GitHub, you can simply install with pip:

```
python -m pip install deadline-cloud-for-unreal-engine
```

If you're using mainline or a custom in-development version of the submitter, to avoid compatibility issues we recommend that you build and install from the same version of code or transfer over the `.whl` file from your submitter build:

```
pip install hatch
hatch build
python -m pip install dist\my-built-wheel.whl
```

Building the plugin

Adjust the first two paths below based on where your installation of Unreal lives, and where you installed `deadline-cloud-for-unreal-engine`.

From the Unreal Install Batchfiles folder (the `package` parameter can be any new directory, however you'll want it to be called `UnrealDeadlineCloudService` later):

```
cd C:\Program Files\Epic Games\UE_5.5\Engine\Build\BatchFiles
runuat.bat BuildPlugin -plugin="C:\deadline\deadline-cloud-for-unreal-
engine\src\unreal_plugin\UnrealDeadlineCloudService.uplugin" -package="C:
\UnrealDeadlineCloudService"
```

Copy the "package" folder above to your Unreal installation's Plugins folder (for example, `C:\Program Files\Epic Games\UE_5.5\Engine\Plugins\UnrealDeadlineCloudService`).

pywin32

Unreal's version of Python will need pywin32. Pip install using a copy of Unreal's third-party Python installation:

```
"C:\Program Files\Epic Games\UE_5.5\Engine\Binaries\ThirdParty\Python3\Win64\python" -m
 pip install pywin32
```

Starting the Deadline Cloud worker agent service

On your CMF worker instance:

1. Open **Task Manager**.
2. Choose the **Services** tab on the right.
3. Find **DeadlineWorker**.
   - If you don't see it listed you've likely missed steps (`install-deadline-worker` in particular) from the CMF host setup steps.
4. If the status of the service isn't currently "Running", right-click it and select **Start**.
5. If your DeadlineWorker service isn't starting, check the worker agent launch logs in these locations:
   - `C:\ProgramData\Amazon\Deadline\Logs\worker-agent.log`
   - `C:\ProgramData\Amazon\Deadline\Logs\queue-<queueid>\session-<sessionid>.log`

Perforce credentials management

This section covers secure credential management for integrating Perforce with Deadline Cloud workers.

Overview

There are several approaches to configure Perforce credentials for Deadline Cloud workers. Each method has different security implications and use cases:

Method	Security level	Deployment support	Recommended use
AWS Secrets Manager (Secrets Manager)	High	SMF + CMF	**Production (recommended)**
Job environment variables	Low	SMF + CMF	Development and testing only
Queue environment variables	Low	SMF + CMF	Development and testing only
Windows registry	Medium	CMF only	Legacy CMF setups
Pre-configured admin user	Medium	CMF only	Simplified CMF setups

Important

Use Secrets Manager for production environments. It provides centralized, encrypted credential storage with audit trails and works with both SMF and CMF deployments.

P4 credentials basics

To retrieve connection settings, including Perforce server URL and port, username, and password, Perforce follows the following priority:

1. Connection parameters in any framework for Perforce (P4) (`p4python.P4` for example).
2. User/system environment variables: `P4PORT`, `P4USER`, `P4CLIENT`, `P4PASSWD`.
3. Windows registry: `HKEY_LOCAL_MACHINE\SOFTWARE\Perforce\Environment` (system-wide settings) or `HKEY_CURRENT_USER\SOFTWARE\Perforce\Environment` (user-specific settings).

With these priorities, if you have the following setup:

- Connection parameter `password` is `password.from.connection`
- Environment variable `%P4PORT%` is `ssl:perforce.from.env:1666`
- Windows registry `HKEY_CURRENT_USER\SOFTWARE\Perforce\Environment:P4PORT` is `ssl:perforce.from.registry:1666`
- Windows registry `HKEY_CURRENT_USER\SOFTWARE\Perforce\Environment:P4USER` is `user.from.registry`
- Windows registry `HKEY_CURRENT_USER\SOFTWARE\Perforce\Environment:P4PASSWD` is `password.from.registry`

The resulting Perforce connection will be:

```
Port = ssl:perforce.from.env:1666
User = user.from.registry
Password = password.from.connection
```

Secrets Manager (recommended)

Secrets Manager provides the most secure and scalable solution for both CMF and SMF deployments.

This approach provides the following benefits:

- **Centralized security**: Store all P4 credentials in one encrypted location.
- **No credential exposure**: Credentials never appear in job configurations or logs.
- **Automatic rotation**: Support credential rotation without job reconfiguration.
- **Audit trails**: Track credential access and usage.
- **Universal support**: Works with both CMF and SMF deployments.
- **Log redaction**: Connection credentials automatically redacted in job logs.

Step 1: Create a secret in Secrets Manager

Create a secret containing your Perforce connection parameters.

Required key-value pairs:

- P4PORT - Perforce server URL and port.
- P4USER - Perforce username.
- P4PASSWD - Perforce password.

> **Important**
>
> Key names must exactly match P4 connection parameters to work with `deadline-cloud-for-unreal-engine`.

Example secret:

```
{
   "P4PORT": "ssl:your-perforce-server.com:1666",
   "P4USER": "your-perforce-username",
   "P4PASSWD": "your-perforce-password"
}
```

To create the secret:

1. Open the Secrets Manager console.
2. Choose "Store a new secret".
3. Select "Other type of secret".
4. Enter the key-value pairs above.
5. Name your secret (for example, `deadline-cloud-p4-credentials`).
6. Complete the creation process.

Step 2: Grant worker access to the secret

Workers need `secretsmanager:GetSecretValue` permission to access the secret. This follows the same pattern as managing Windows job user secrets.

To grant access:

1. Open the Secrets Manager console and navigate to your secret.
2. In the "Resource permissions" section, add this policy:

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "QUEUE_ROLE_ARN"
      },
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": "*"
    }
  ]
}
```

3. Replace QUEUE_ROLE_ARN with your actual queue role ARN.
4. Save the policy.

> **Note**
>
> Save the secret name - you'll need it when configuring P4 render jobs. See Creating a Perforce render job for next steps.

Job environment variables

> **Warning**
>
> This approach is not recommended for production as it exposes credentials in job configurations and logs. Use Secrets Manager for production environments.

You can pass connection credentials within the job environment where workspace creation happens. Examples on the GitHub website include p4_sync_smf_environment and ugs_sync_smf_environment, or similar environments for CMF. Alternatively, create a new environment template and prepend it to your job.

```
name: P4Credentials
```

```
variables:
  P4PORT: ssl:my-perforce.com:1666
  P4USER: j.doe
  P4PASSWD: MyVeRyS3cretP4ssW0rd
```

This approach has the following security risks:

- Credentials are visible in job configurations.
- Passwords may appear in logs (not automatically redacted).
- No centralized credential management.
- Difficult to rotate credentials.

Queue environment variables

> **Warning**
>
> This approach is not recommended for production as it stores credentials in queue configurations. Use Secrets Manager for secure credential management.

Per the Deadline Cloud user guide, you can use queue environments to provide software applications, environment variables, and other resources to jobs in the queue. For queue environment samples, see the queue_environments folder in deadline-cloud-samples on the GitHub website.

Add queue environment using Deadline Cloud monitor or console

1. Open Deadline Cloud monitor or the AWS console.
2. Navigate to the farm and queue you are working with.
3. Select **Queue environments** tab.
4. Choose **Action** and select **Create new with YAML**.
5. Add the following and save:

```
specificationVersion: environment-2023-09
environment:
   name: P4Credentials
```

```
   variables:
      P4PORT: ssl:my-perforce.com:1666
      P4USER: j.doe
      P4PASSWD: MyVeRyS3cretP4ssW0rd
```

Add queue environment using CLI

1. Create a `p4_credentials.yaml` file with the sample above.
2. Run the following CLI command:

```
aws deadline create-queue-environment \
 --farm-id FARM_ID \
 --queue-id QUEUE_ID \
 --priority 1 \
 --template-type YAML \
 --template file://p4_credentials.yaml
```

Windows registry (CMF only)

> **Important**
>
> This solution is only suitable for CMF where you can configure worker hosts directly. For SMF deployments, use Secrets Manager.

Windows registry provides local credential storage on worker machines. This method uses the standard Perforce priority system where credentials are stored in:

- `HKEY_LOCAL_MACHINE\SOFTWARE\Perforce\Environment` (system-wide settings).
- `HKEY_CURRENT_USER\SOFTWARE\Perforce\Environment` (user-specific settings).

This approach is only suitable for CMF deployments where you have direct access to configure worker machines.

Pre-configured admin user (CMF only)

> **Important**
>
> This solution is only suitable for CMF where you can configure worker hosts directly. For SMF deployments, use Secrets Manager.

Create a dedicated P4 user for render nodes with a non-expiring connection to eliminate the need to pass secret data such as usernames and passwords. In this case, a single dedicated P4 user can be used to connect to all P4 servers, including the Commit (Master) and Edge servers where the project depot is accessible.

Therefore, you only need to pass the port to connect to, if the default is not configured on workers. You can pass the port by adding the following in job environment or queue environment similar to the steps documented above:

```
name: P4Sync
variables:
  P4PORT: ssl:my-perforce.com:1666
```

Creating a Perforce render job

This section walks you through configuring Perforce-integrated render job data assets so you can submit Perforce-integrated render jobs to Deadline Cloud from Unreal Engine.

Prerequisites

Before submitting a Movie Render Queue (MRQ) job from Unreal Engine in a Perforce repository:

Unreal project setup:

- Project must be within the Perforce workspace.
- Verify Perforce connection is established and you're logged in.

Deadline Cloud setup:

- Complete Installing the submitter.
- Configure Perforce credentials management for workers.

Perforce requirements:

- Valid Perforce workspace with project files.
- Appropriate Perforce permissions for sync operations.

Perforce render job architecture

P4 render jobs extend the standard render job with Perforce synchronization capabilities:

```
Deadline Cloud Perforce render job
### Environments
#   ### Apply Perforce Credentials from Secrets Manager
#   ### Sync Perforce Environment (CMF/SMF)
#   #   ### Initializes Perforce workspace and syncs repository on workers
#   ### Launch UE Environment
#       ### Starts Unreal Engine with Perforce workspace paths
### Steps
    ### Render Step
        ### Executes Movie Render Queue process
```

Key differences from standard jobs:

- **Credential management**: Secure Perforce credential application from Secrets Manager.
- **Repository sync**: Additional Perforce sync environment for workspace management.
- **Path resolution**: Environment variables reference Perforce workspace paths.
- **Dependency collection**: Automatic collection and sync of Perforce-tracked assets.

How Perforce sync scales for production projects

Production Unreal Engine projects can reach hundreds of gigabytes across tens of thousands of files. The Perforce integration keeps sync time proportional to what changed rather than to the size of the project:

Jobs are pinned to a changelist

> At submission, the submitter sets the `PerforceChangelistNumber` job parameter to the changelist your workspace has synced. Every task in the job renders against that same revision, so your job stays reproducible even when artists submit new changes while it runs. Workers can start a task only after the pinned changelist is available on the Perforce server they sync from.

Workers reuse a stable client workspace

Each worker creates its Perforce client workspace under a persistent root directory. It records the pairing in a `workspace_info.json` registry file stored with the workspace. A replacement worker that receives the same storage reads the registry and reuses the existing client. It doesn't create a new workspace. Because Perforce tracks the have-list on the server under the client name, a normal sync then downloads only the file revisions that changed since the last job.

To benefit from client workspace reuse on a service-managed fleet, enable persistent storage so the volume holding the synced workspace is reattached to future workers.

If the network path between your fleet's Region and your Perforce server has low throughput or high latency, a Perforce edge server in the fleet's Region can act as a regional cache for depot data. For more information, see Perforce source control in the *Deadline Cloud Developer Guide*.

Setting up Perforce render job components

Follow these steps to create the required data assets for Perforce-integrated render jobs.

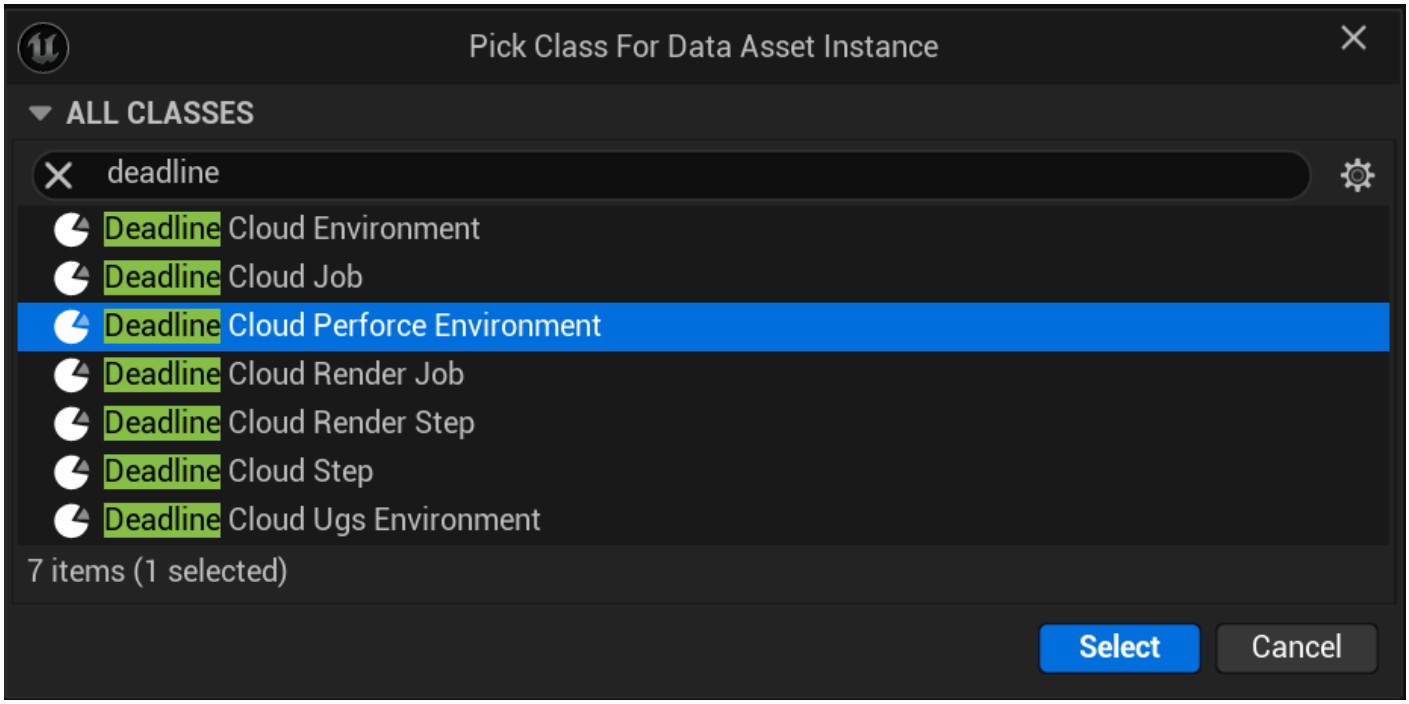
1. Create Apply Perforce Credentials data asset

Set up an OpenJD environment for retrieving Perforce credentials from Secrets Manager and applying them to the Perforce connection.

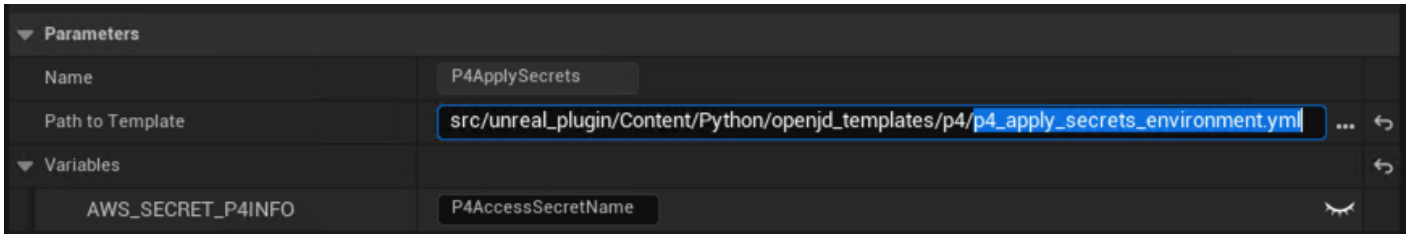
> **Note**
>
> Environment variables appear in logs with the `openjd_env` prefix, but sensitive data (port, user, password) is automatically redacted for security.

Configuration steps:

1. Create a new **Deadline Cloud Perforce Environment** data asset.



2. Name the data asset descriptively (for example, "ApplyP4SecretEnv").
3. Select the `p4_apply_secrets_environment.yml` template from `Content/Python/openjd_templates/p4/`.
4. Configure the secret reference:
   - Enter your Perforce credential secret name in **AWS_SECRET_P4INFO**.
   - The value should match the secret name created in Perforce credentials management.



2. Create Perforce sync environment data asset

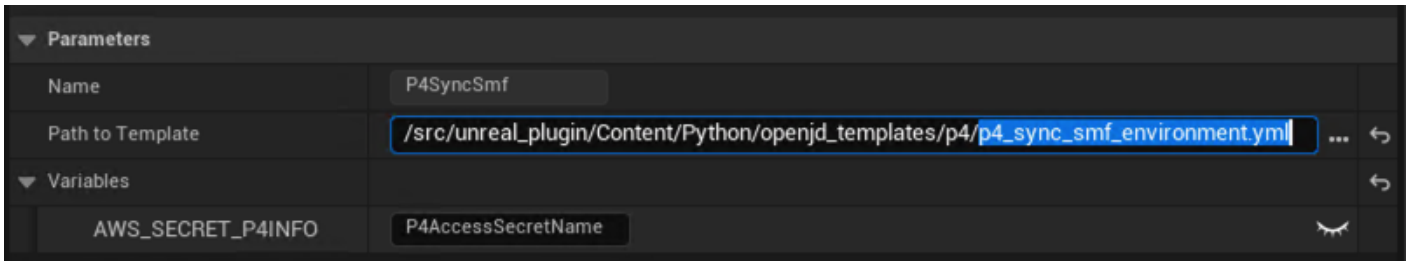
Set up an OpenJD environment for creating a Perforce workspace, syncing files from the Perforce server, and cleaning up the workspace after rendering.

Configuration steps:

1. Create a new **Deadline Cloud Perforce Environment** data asset.
2. Name the data asset based on your fleet type:

- **P4SyncSMFEnv** for service-managed fleet (SMF).
- **P4SyncCMFEnv** for customer-managed fleet (CMF).

3. Select the appropriate template from `Content/Python/openjd_templates/p4/`:
   - **SMF**: `p4_sync_smf_environment.yml`.
   - **CMF**: `p4_sync_cmf_environment.yml`.
4. Configure the secret reference:
   - Enter your Perforce credential secret name in **AWS_SECRET_P4INFO**.
   - Must match the secret from step 1.

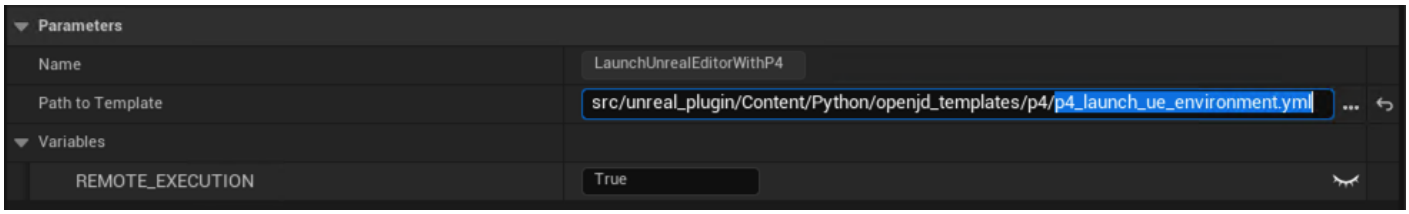


3. Create Perforce Launch UE environment data asset

Set up an OpenJD environment for launching Unreal Engine with Perforce integration. This automatically references the Perforce workspace created by the sync environment.

Configuration steps:

1. Create a new **Deadline Cloud Environment** data asset.
2. Name the data asset descriptively (for example, "P4LaunchUEEnv").
3. Select the `p4_launch_ue_environment.yml` template from `Content/Python/openjd_templates/p4/`.
4. Configure environment settings:
   - Set **REMOTE_EXECUTION** to `True`.
   - This setting enables remote rendering capabilities.

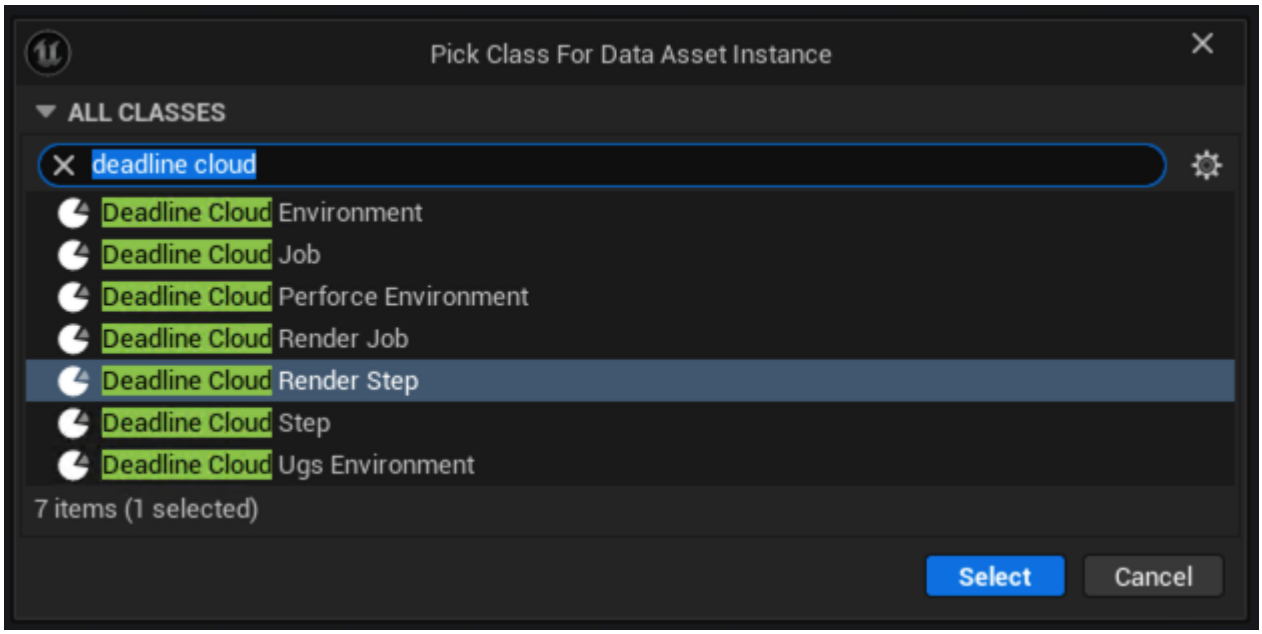


4. Create Perforce render step data asset

Set up an OpenJD render step for executing the rendering process.

Configuration steps:

1. Create a new **Deadline Cloud Render Step** data asset.



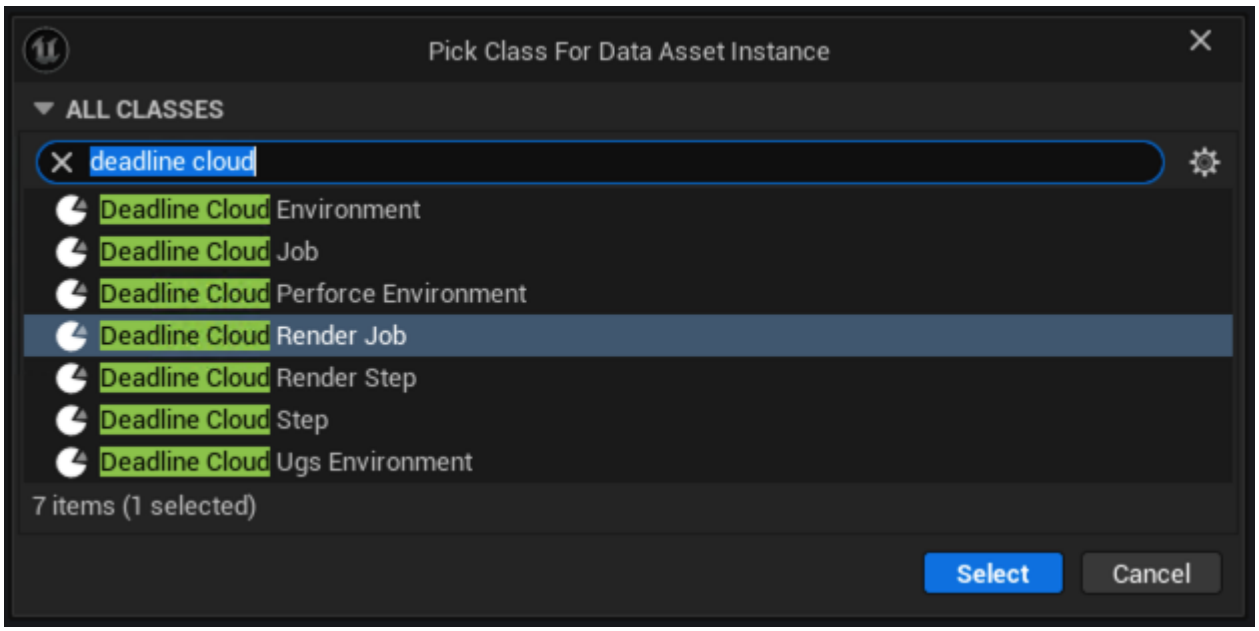
2. Name the data asset descriptively (for example, "P4RenderStep").
3. Select the `p4_render_step.yml` template from `Content/Python/openjd_templates/p4/`.

5. Create Perforce render job data asset

Set up an OpenJD render job that orchestrates the entire rendering workflow.

Configuration steps:

1. Create a new **Deadline Cloud Render Job** data asset.



2. Name the data asset descriptively (for example, "P4RenderJob").
3. Select the `p4_render_job.yml` template from `Content/Python/openjd_templates/p4/`.
4. **Review parameter definitions**: The template includes these parameters with their default behaviors:

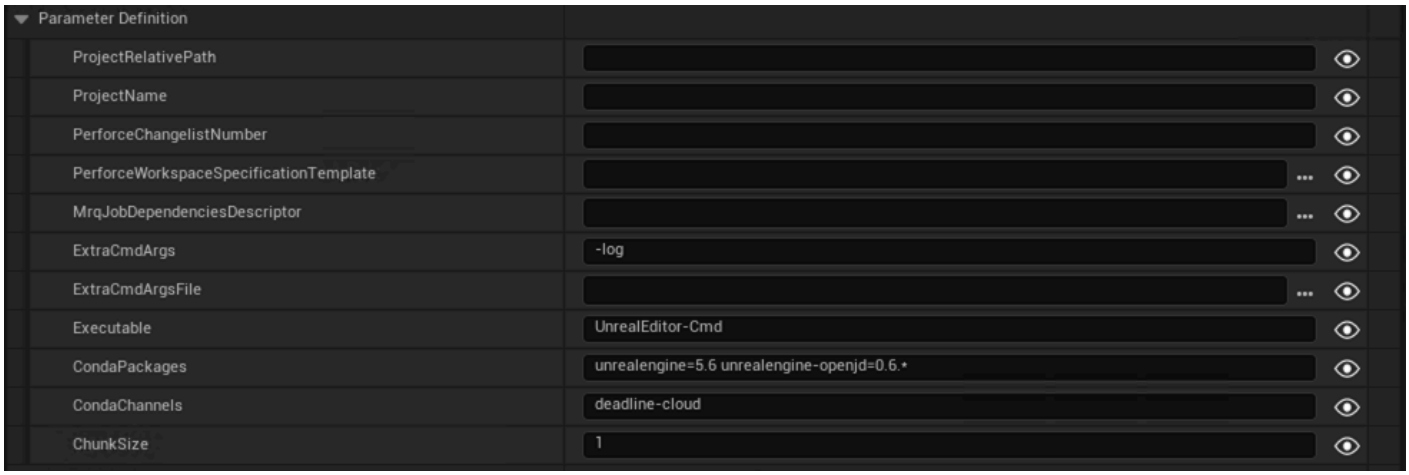
Parameter	Description	Auto-filled	Action required
`ProjectRelativePath`	Project path relative to Perforce workspace root	Yes	Leave empty - auto-populated
`ProjectName`	Project name for Perforce workspace creation	Yes	Leave empty - auto-populated
`PerforceChangelistNumber`	Perforce changelist to sync workspace to	Yes	Leave empty - auto-populated
`PerforceWorkspaceSpecificationTemplate`	Perforce client spec with `{workspace_name}` token	Yes	Leave empty - auto-populated

Parameter	Description	Auto-filled	Action required
`MrqJobDependenciesDescriptor`	JSON file with MRQ dependencies for sync	Yes	Leave empty - auto-populated
`ExtraCmdArgsFile`	File for extra args (avoids 1024 char limit)	No	Optional - use default for standard setups
`FramesPerTask`	Number of frames to render per task	No	Optional - use default (0) to divide tasks by shots
`ExtraCmdArgs`	Additional Unreal launch arguments	No	Optional - use default for standard setups
`Executable`	Unreal executable name for render node	No	Configure - use default for standard setups
`CondaPackages`	Conda packages needed to render the job	No	Configure - use default for standard setups
`CondaChannels`	Conda channels where packages are stored	No	Configure - use default for standard setups
`ShotsPerTask`	Number of shots grouped in a single render session	No	Configure - default: 1 (tune for performance)

Parameter	Description	Auto-filled	Action required
`SubmitMode`	Push render outputs into Perforce (`''`, `submit`, or `shelve`)	No	Optional - default `''` deactivates Perforce output submission
`MarketplacePluginsDir`	Path to engine Marketplace plugins	Yes	Leave empty - auto-populated

Parameter configuration guidelines:

- **Auto-populated parameters**: Leave these empty - they're filled automatically during job submission.
- **Manual parameters**: Review defaults and adjust based on your specific requirements.
- **ShotsPerTask**: Start with 1, increase for better performance with simple shots.



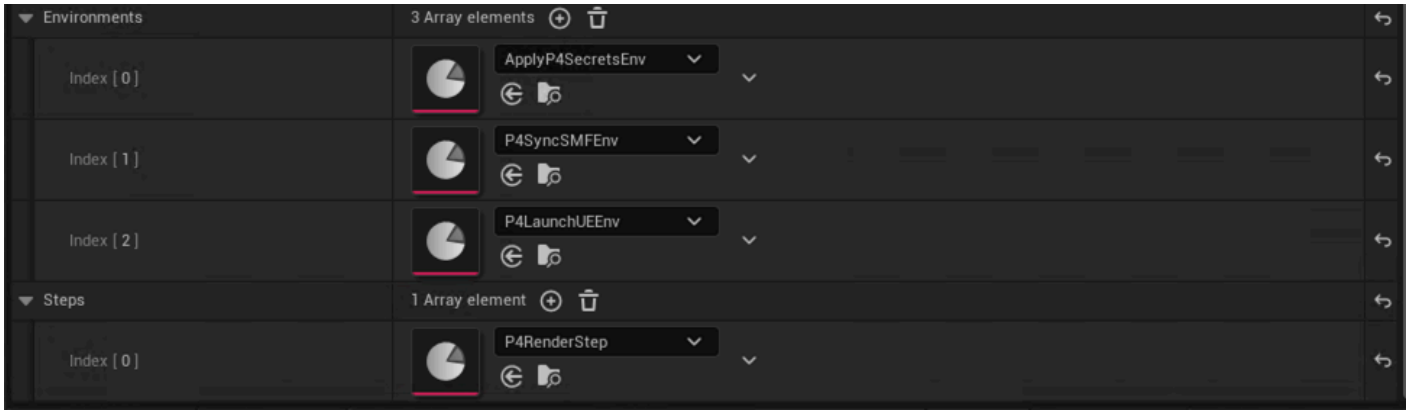
5. **Configure environments** (in this exact order):

Order	Environment	Purpose
1st	"ApplyP4SecretEnv"	Apply Perforce credentials from Secrets Manager

Order	Environment	Purpose
2nd	"P4SyncSMFEnv" or "P4SyncCMFEnv"	Sync Perforce workspace and files
3rd	"P4LaunchUEEnv"	Launch Unreal Engine with Perforce paths

> **Important**
>
> Environment order is essential for proper dependency resolution and credential flow.

6. **Add render step**: Add "P4RenderStep" to the Steps section.



Submitting render outputs back to Perforce

By default, a Perforce render job uploads outputs through job attachments only. The `SubmitMode` job parameter opts in to pushing rendered outputs into Perforce instead, so finished frames land in your depot alongside the project.

`SubmitMode` value	Behavior
`''` (default)	Job attachments only. No Perforce write occurs. Existing jobs behave unchanged.
`submit`	Each render task shelves the files it produced. When every task has finished, an `AssembleShelves` step combines all task shelves into a

SubmitMode value	Behavior
	single aggregate changelist and submits it. The job log records the final changelist number.
`shelve`	Same aggregation as `submit`, but the final aggregate changelist is shelved instead of submitted, so a person can review the aggregate before committing it.

Only files that changed since the last sync end up in the changelist. Each task reconciles the exact files it produced, and the integration reverts files whose content is identical to the depot revision, so re-rendering an unchanged frame produces no changelist entry.

> **Note**
>
> When `SubmitMode` is `submit` or `shelve`, the job doesn't upload render outputs to Amazon S3 through job attachments. Perforce becomes the sole delivery path for the frames. The download outputs feature in Deadline Cloud monitor doesn't find the frames, and downstream jobs that consume outputs through job attachments don't see them. Input attachments are still uploaded through job attachments as normal.

Configure your workers to authenticate to Perforce as the same Perforce user for all tasks in a job, because the aggregation step finds task shelves by owner. The recommended credential setup in Perforce credentials management, a single shared secret in Secrets Manager, already satisfies that requirement.

Best practices

Pre-submission checklist

- **Credentials**: Perforce credentials properly configured (see Perforce credentials management).
- **Workspace**: Specification includes all necessary view mappings.
- **Testing**: Test with a small render task before large job submissions.
- **Dependencies**: Verify all project dependencies are in Perforce and accessible.
- **Permissions**: Confirm worker roles have access to Secrets Manager.

Performance optimization

The following table shows how the `ShotsPerTask` setting affects performance:

Shots per task	Use case	Performance impact
1-2 shots	Complex shots, detailed review needed	Lower throughput, higher quality control
4-8 shots	Balanced workload, typical projects	Optimal balance of speed and manageability
10+ shots	Simple shots, batch processing	Maximum throughput, minimal overhead

Consider the following additional optimizations:

- **Dependencies**: Minimize unnecessary asset dependencies to reduce sync time.
- **Workspace views**: Optimize Perforce workspace views to sync only required files.

Monitoring and debugging

- **Job logs**: Monitor Perforce sync process in job execution logs.
- **Workspace status**: Check workspace creation and sync completion.
- **Dependency collection**: Verify all required assets were captured.
- **Performance metrics**: Track sync times and render performance.

Troubleshooting

Common issues and solutions

Issue	Symptoms	Solution
P4 connection failures	Authentication errors, timeout	Verify Perforce credentials; check network connectivity; validate Perforce server accessibility

Issue	Symptoms	Solution
Workspace sync errors	Sync failures, permission denied	Check Perforce user permissions; verify workspace specification; ensure changelist exists
Missing dependencies	Render failures, missing assets	Review `MrqJobDependencies Descriptor` ; check soft references collection; verify all assets in Perforce
Path resolution issues	File not found errors	Verify `P4_CLIENT_DIRECTORY` variable; check environment order; validate workspace root paths

Debug steps

1. **Check job logs**: Review Perforce sync environment logs for detailed error messages.
2. **Validate workspace**: Ensure Perforce workspace specification is correct.
3. **Test locally**: Verify Perforce operations work from submitter machine.
4. **Verify permissions**: Confirm worker roles have secret access (if using Secrets Manager).

Custom host requirements

This section describes how to create and use custom host requirements for Deadline Cloud rendering in Unreal Engine.

Overview

Host requirements define which fleet are eligible to run a particular render step. They are stored inside a `UDeadlineCloudHostRequirements` asset, which is then referenced by a `DeadlineCloudRenderStep` asset.

Requirement type	Configuration location	Use case
Base system requirements	CPU / RAM / GPU fields	Restrict rendering to machines with specific hardware
Custom amount requirements	Name + min/max values	Require numeric resource levels (licenses, tokens, quotas)
Custom attribute requirements	Attribute + value list	Required custom attributes on fleets to run the step

Note

Hiding a requirement using the eye icon only hides it in the MRQ Submit UI. The requirement still applies when submitting the job.

Important

The `Name` and `Attribute` values used for custom amount requirements and custom attribute requirements must strictly match the valid identifiers defined in the official Open Job Description documentation. For more information, see Open Job Specifications on the GitHub website.

Step 1: Create the host requirements asset

1. Open Content Browser in Unreal Engine.
2. Create a new asset: **Add** → **Miscellaneous** → **Data Asset** → **DeadlineCloudHostRequirements**.
3. Name the asset, for example, `MyHostRequirements`.

Step 2: Load the default YAML template

1. In the asset details panel, locate the field **Path to Template**.

2. Select the default template: `Plugins/UnrealDeadlineCloudService/Content/Python/openjd_templates/host_requirements.yml`.

This loads base requirements such as CPU, Memory, GPU, OS, and architecture.

> **Note**
>
> Base requirements cannot be removed; only their values may be changed.

Step 3: Configure base (system) requirements

Setting	Example
Operating System	`windows`
CPU Architecture	`x86_64`
vCPUs (Min)	16
GPUs (Min)	1

If Max = 0, then there is no upper limit for that requirement.

Step 4: Add custom amount requirements

Used when the requirement is numeric (for example, number of licenses, concurrency limits).

1. Find the section **Custom Amount Requirements**.
2. Add a new entry and set a name, for example: `amount.custom.license`.
3. Set Min and Max values:
   - Min: 1.
   - Max: 0 (0 means unlimited).

Step 5: Add custom attribute requirements

Used to filter workers based on tags and labels.

1. Find the section **Custom Attributes Requirements**.
2. Add a new attribute name, for example: `attr.custom.gpu.vendor`.
3. Choose the matching rule:
   - **AllOf** - all values must match.
   - **AnyOf** - at least one must match.
4. Enter attribute values separated by spaces, for example: `nvidia amd`.

Step 6: Visibility in MRQ (optional)

Each requirement entry has an eye icon.

Eye	Behavior
Visible	The requirement is shown in MRQ Submit UI
Hidden	The requirement is hidden but still included when submitting the job

This setting is useful for simplifying UI for artists while preserving technical constraints.

Step 7: Attach the requirements to a render step

1. Open the relevant `DeadlineCloudRenderStep` asset.
2. Find the **Host Requirements** field.
3. Assign your created asset (for example, `MyHostRequirements`).

All MRQ submissions using this step now enforce your host selection logic.

Summary

Using the host requirements asset, you can:

- Control which worker machines execute your render jobs.
- Specify minimum resource levels.
- Target specific worker pools using attribute matching.
- Simplify UI while keeping technical rules intact.

- Reuse settings across multiple MRQ steps.

This setup improves job routing consistency and ensures rendering happens on the correct hardware.

Advanced configurations

Service-managed fleets vs customer-managed fleets

Service-managed fleets (SMF)

On service-managed fleets, the Unreal Engine and adaptor are automatically available using the `deadline-cloud` conda channel with the default queue environment. This setup provides the easiest experience.

Customer-managed fleets (CMF)

For customer-managed fleets, Unreal Engine and the adaptor must be manually installed on worker hosts. This setup provides more control and supports additional features like Perforce integration. For detailed instructions, see Setting up a customer-managed fleet (CMF) worker.

Unreal Engine rendering features

Unreal Engine's rendering system provides comprehensive support for:

Feature	Description	Notes
Movie Render Queue	High-quality offline rendering	Integration with job submission
Sequencer	Timeline-based animation system	Automatic shot detection and processing
Project Plugins	Custom plugin support	Automatic detection and inclusion
Asset Dependencies	Content file management	Comprehensive asset tracking
Sticky Rendering	Application persistence between shots	Improved performance for multi-shot sequences

All rendering features are automatically detected and configured by the Unreal Engine integrated submitter. The adaptor maintains proper dependency handling and supports efficient multi-shot rendering without restarting Unreal Engine.

Open source resources

The submitter and adaptor are open source. For more information, see Deadline Cloud for Unreal Engine on the GitHub website.

Foundry Nuke

Foundry Nuke is a node-based digital compositing and visual effects application used for television and film post-production. Nuke is supported by AWS Deadline Cloud (Deadline Cloud) with submitters, conda packages, and an adaptor for increased rendering performance. This guide provides step-by-step instructions for using Deadline Cloud with Nuke to render your projects faster by distributing rendering tasks across multiple machines.

Support overview

Nuke is supported by the following components:

- **Submitter**: Integrated submitter plugin for direct job submission from Nuke with automatic scene and asset detection.
- **Conda packages**: Packages to install nuke versions 15, 16, and 17 are available on the Deadline Cloud conda channel for service-managed fleets.
- **Adaptor**: A program on worker hosts that keeps the application loaded between tasks for faster rendering and reports render progress.
- **Cross-platform compatibility**: Submitter support for Windows, macOS, and Linux with worker support for Linux only with automatic path mapping.

Nuke version compatibility

The following table shows current support levels for Nuke versions:

Major Version	Submitter Support	Conda Support
15	Windows, macOS, Linux	Linux

Major Version	Submitter Support	Conda Support
16	Windows, macOS, Linux	Linux
17	Windows, macOS, Linux	Linux

Deadline Cloud Conda Channel

The following table lists conda packages applicable to Nuke available to Service-managed fleets in the deadline-cloud conda channel:

OS	Package	Version	Notes
Linux	nuke	15	Includes built-in compositing engine
Linux	nuke	16	Includes built-in compositing engine
Linux	nuke	17	Includes built-in compositing engine
Linux	nuke-openjd		Includes the Nuke Adaptor

Getting started

To use Nuke with Deadline Cloud:

1. Create a service-managed fleet and associate it with a queue. Your queue must be set up with a queue environment that supports the deadline-cloud conda channel. For more information, see Creating a queue environment.
2. Install the Deadline Cloud monitor and Nuke submitter on your artist workstation using the Deadline Cloud Submitter and monitor Installers. For more information, see Set up your workstation.
3. Submit your job directly from Nuke using the integrated submitter to the queue.

4. Monitor the job and download the output using the Deadline Cloud monitor.

Launch the submitter

To launch the Deadline Cloud submitter in Nuke

> **Note**
>
> Support for Nuke is provided using the conda environment for service-managed fleets. For more information, see Default conda queue environment.

1. Install the Deadline Cloud monitor and Nuke submitter on your artist workstation using the Deadline Cloud Submitter and monitor Installers. For more information, see Set up your workstation.
2. Open **Nuke**.
3. Open a Nuke script with dependencies that exist within the asset root directory.
4. Choose **AWS Deadline**, and then choose **Submit to Deadline Cloud** to launch the submitter.
5. If you are not already authenticated, choose **Login** and log in with your user credentials in the browser window.
6. Choose **Submit**.

Installation

To install the Deadline Cloud for Nuke submitter, you need:

- A Windows, macOS, or Linux workstation.
- Nuke 14, 15, 16, or 17. We recommend Nuke 15 or later over Nuke 14, because these versions are supported by the default conda queue environment on service-managed fleets. To use Nuke 14 with a service-managed fleet, you need to make Nuke 14 available to the worker. The recommended way is to create your own conda package by following Create a conda package for an application or plugin.

There are two ways to install the Deadline Cloud for Nuke submitter:

- Using the Deadline Cloud submitter installer (recommended).

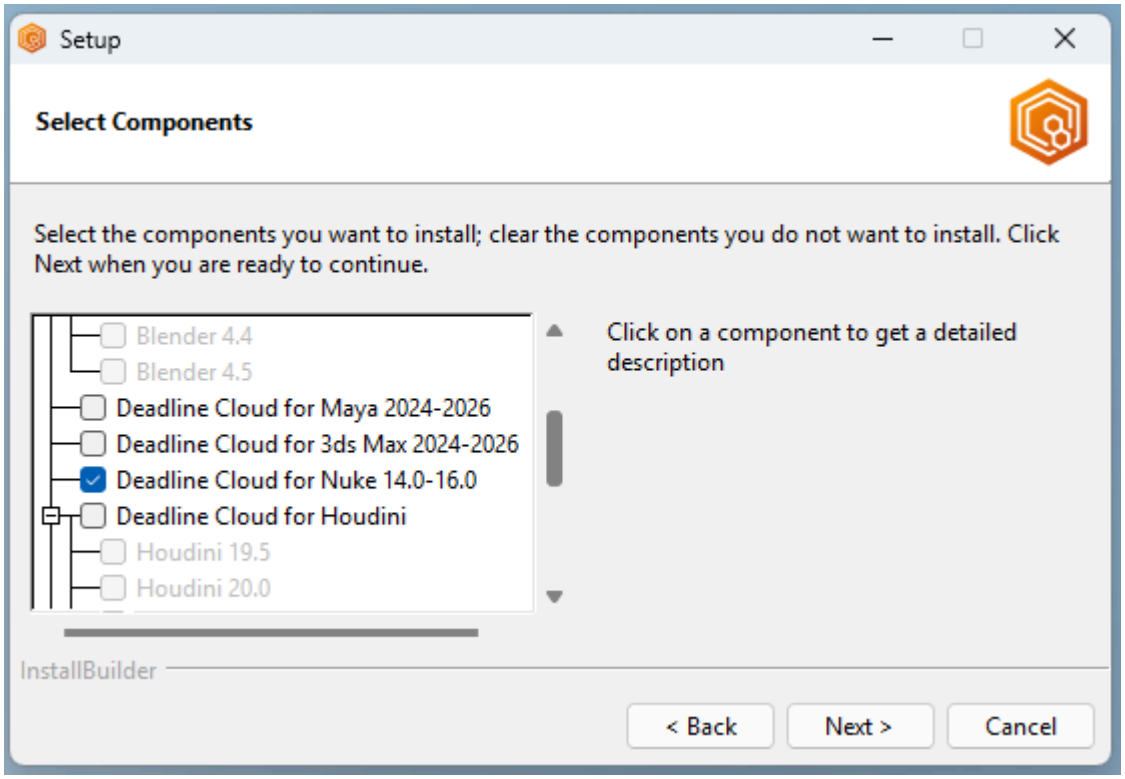
- Manually installing the submitter from source. For instructions, see Manual installation on the GitHub website.

Using the Deadline Cloud submitter installer

You can install the Deadline Cloud for Nuke submitter using the Deadline Cloud submitter installer.

To install the submitter:

1. Download the Deadline Cloud submitter installer.
2. Run the installer.
3. When prompted to select components, find and mark the checkbox for Nuke.



4. Finish running the installer.
5. Launch Nuke.
6. Verify the installation by checking if **Deadline Cloud** has been added to the top navigation bar.

Using the Nuke submitter

The Deadline Cloud for Nuke submitter supports two types of jobs:

- Render jobs - Render the output files created by one or more of the Write nodes in your Nuke script. For more information, see Write nodes on the Foundry website.
- CopyCat training jobs - Perform training for a CopyCat node in your Nuke script. For more information, see CopyCat on the Foundry website.

Render jobs

To use the Deadline Cloud for Nuke submitter, you need:

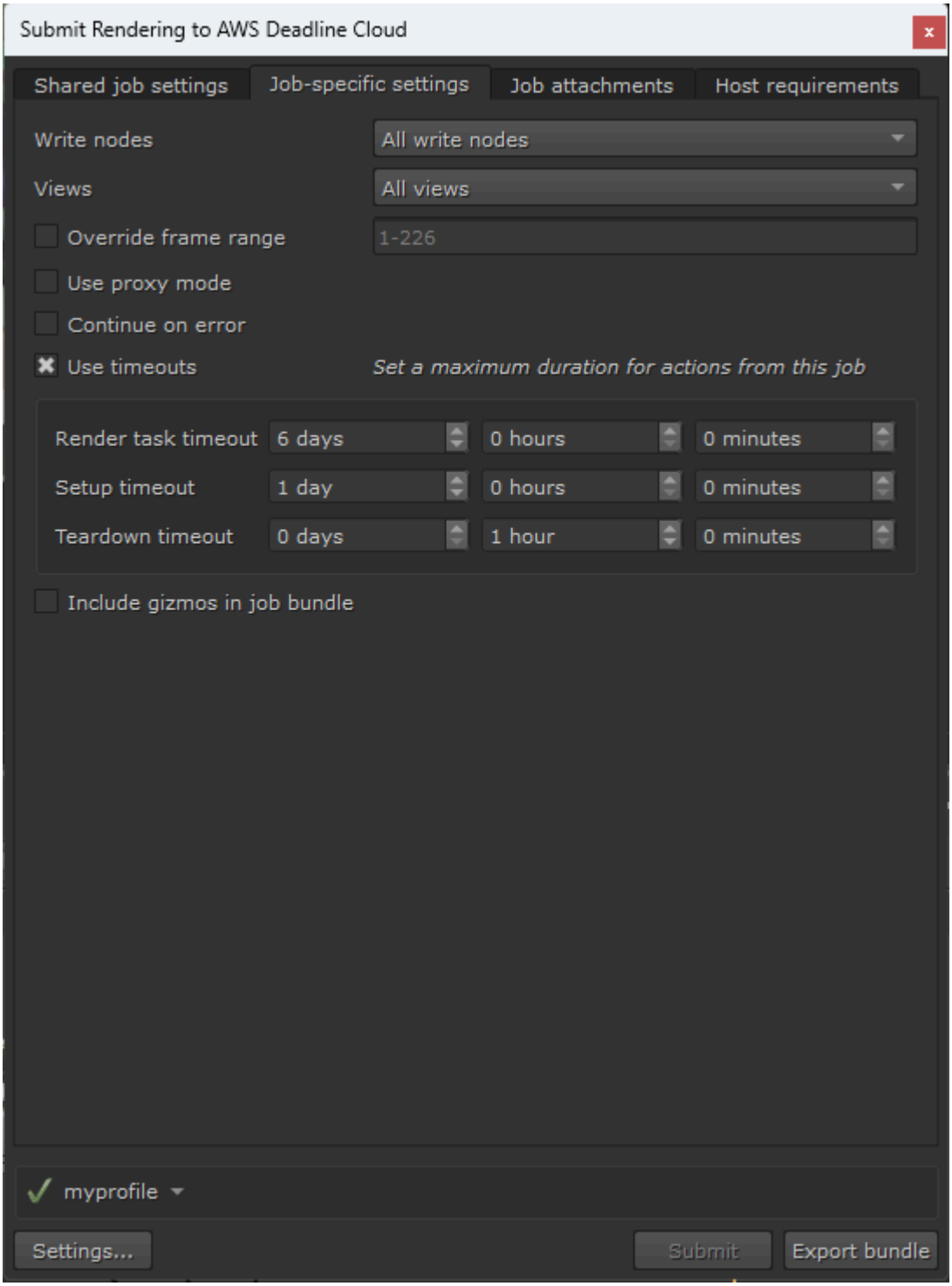
- A profile to submit to Deadline Cloud with.
- An Deadline Cloud farm and queue to submit to.

To submit a render job from Nuke to Deadline Cloud:

1. Save your Nuke file.
2. From the top navigation bar, choose **Deadline Cloud**. From the drop-down menu, choose **Submit to Deadline Cloud**.
3. Use the tabs in the dialog to customize your job.
4. (Optional) To export a job's associated files to your job history directory without submitting it, choose **Export bundle**.
5. Choose **Submit** and follow the prompts to send your job to Deadline Cloud.

Nuke render-specific settings

The **Job-specific settings** tab has options specific to jobs created in Nuke:



- *Write nodes* - Which write nodes to render outputs for. You can either select to render all write nodes, or select a specific node. For more information, see Write nodes on the Foundry website.
- *Views* - Which views should be rendered. For more information, see Setting up stereo views on the Foundry website.

- *Override frame range* - Select this option to render a different frame or frame range than is set in Nuke. Frame ranges follow the Open Job Description specification on the GitHub website.
- *Use proxy mode* - Manages whether to use proxy mode in the submitted job. For more information, see Proxy mode on the Foundry website.
- *Continue on error* - If selected, Nuke tries to continue rendering when it encounters an error. If cleared, Nuke fails the task when it encounters an error.
- *Chunk size* - Number of frames to group into each chunk (1-150). Use 1 for one frame per task (default). Higher values group frames into contiguous chunks to reduce per-task overhead. For more information, see Task chunking for job templates.
- *Target chunk duration (seconds)* - When you specify a value, the scheduler dynamically adjusts chunk sizes based on observed runtimes of completed chunks, aiming for this duration for each chunk. Leave at 0 to use a fixed chunk size for all chunks.
- *Use timeouts* - Whether to use user-configured timeouts.
- *Render task timeout* - Maximum duration of each action which performs a render. Default is 6 days.
- *Setup timeout* - Maximum duration of each action which sets up the job for rendering, such as scene load. Default is 1 day.
- *Teardown timeout* - Maximum duration of action which tears down the setup required for rendering. Default is 1 hour.
- *Include gizmos in job bundle* - Whether to include gizmos in the job bundle. For more information, see Creating and sourcing gizmos on the Foundry website.

For information about the other submitter tabs, see the Deadline Cloud guide for using a submitter.

CopyCat training jobs

To use the Deadline Cloud for Nuke submitter to train CopyCat nodes, you need:

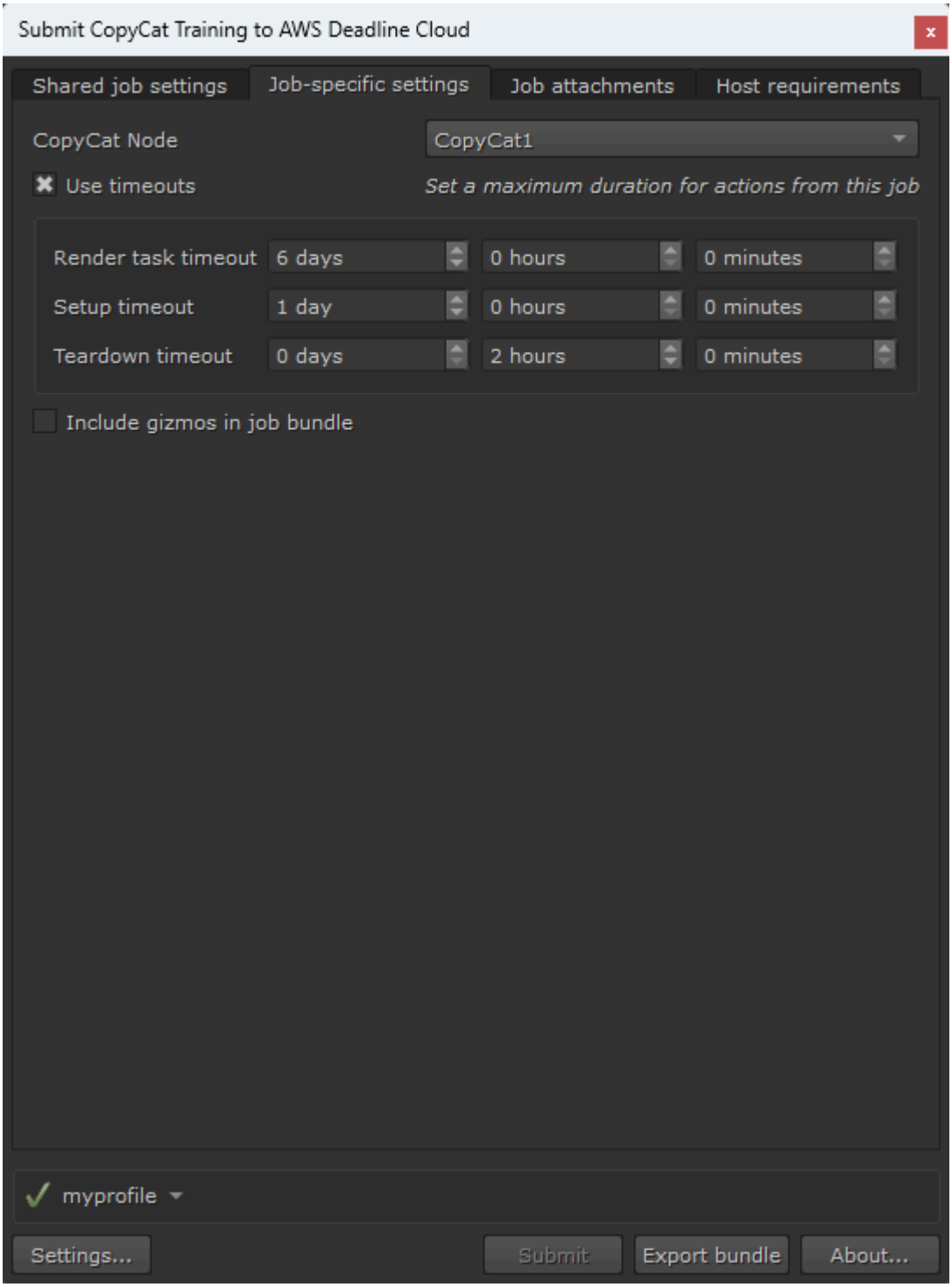
- A profile to submit to Deadline Cloud with.
- An Deadline Cloud farm and queue to submit to.
- An Deadline Cloud fleet with GPU-enabled workers associated with the queue you will be submitting to. For instructions on creating a service-managed fleet with GPU access, see Managing service-managed fleets.

To submit a CopyCat training job from Nuke to Deadline Cloud:

1. Create or open a Nuke script containing a CopyCat node.
2. Attach ground-truth and input nodes to the CopyCat node, and configure knobs on the node to desired values. For details on using CopyCat, see CopyCat on the Foundry website.
3. Save your Nuke file.
4. From the top navigation bar, choose **Deadline Cloud**. From the drop-down menu, choose **Submit CopyCat Training to Deadline Cloud**.
5. Use the tabs in the dialog to customize your job.
6. (Optional) To export a job's associated files to your job history directory without submitting it, choose **Export bundle**.
7. Choose **Submit** and follow the prompts to send your job to Deadline Cloud.

Nuke CopyCat training-specific settings

The **Job-specific settings** tab has options specific to CopyCat training jobs created in Nuke.



- *CopyCat Node* - Select which CopyCat node to train by node name.
- *Use timeouts* - Whether to use user-configured timeouts.
- *Render task timeout* - Maximum duration of each action. In the case of CopyCat, the training is a single action. Default is 6 days.

- *Setup timeout* - Maximum duration of each action which sets up the job, such as scene load. Default is 1 day.
- *Teardown timeout* - Maximum duration of action which tears down the setup. Default is 1 hour.
- *Include gizmos in job bundle* - Whether to include gizmos in the job bundle. For more information, see Creating and sourcing gizmos on the Foundry website.

For information about the other submitter tabs, see the Deadline Cloud guide for using a submitter.

Advanced configurations

Using unsupported versions

Deadline Cloud only supports and tests the workstation and worker software versions in the table above. When using the submitter, the worker attempts to install the same version as used on the workstation. This fails if the workstation version of Nuke does not appear in the version table above.

If you require an unsupported version of Nuke, you have the following options:

- When submitting the job from Nuke, you can override the CondaPackages queue parameter to specify a supported version to use on the worker (for example, `nuke=17, nuke-openjd=*`). This override might or might not work, depending on the features used by your composition and how Nuke works with compositions from your workstation version.
- You can build a custom conda recipe and channel for your desired version to be installed on the worker. Use the conda recipe for a supported version linked below as a starting point, and package your desired version in a custom conda channel. For more information about creating custom conda channels, see Creating custom conda channels.

Custom Nuke executable

You can set the `NUKE_EXECUTABLE` environment variable to point to a specific Nuke executable if it's not available on the PATH.

OpenColorIO support

The Nuke integration includes full support for OpenColorIO (OCIO) color management workflows. Color configurations are automatically detected and included with job submissions to ensure consistent color handling across the render farm.

Nuke plugins

You can add third-party Nuke plugins to a service-managed fleet by building a conda package from a conda recipe and adding it to a custom conda channel. For more information, see Create a conda package for an application or plugin. To include Nuke gizmos with a job instead, use the **Include gizmos in job bundle** option in the submitter.

RevisionFX DENoise

RevisionFX DENoise is a noise-reduction plugin for Nuke compositing jobs. You can build a conda package for DENoise 3.6.9 on Linux workers with the nuke-denoise conda recipe, and then add it to a custom conda channel. A commercial DENoise license is required.

For the recipe, see nuke-denoise conda recipe on the GitHub website.

Nuke compositing features

Nuke's compositing engine provides comprehensive support for:

Feature	Description	Notes
Write Nodes	Multiple output formats and codecs	Automatically detected by submitter
Frame Ranges	Custom frame range specification	Supports override and default ranges
Multiple Views	Stereo and multi-view rendering	Proper handling of view-specific outputs
Color Management	OpenColorIO integration	Automatic OCIO configuration detection

Feature	Description	Notes
Path Mapping	Cross-platform path translation	Windows/Linux compatibility
CopyCat	ML-based paint and rotoscoping	Requires Nuke 14.0 or later

Compositing features are automatically detected and configured by the Nuke integrated submitter. The submitter maintains proper dependency handling and asset management for complex compositions.

Open source resources

The submitter and adaptor are open source and available on the GitHub website:

- Deadline Cloud for Nuke
- Nuke conda recipes for supported versions.

KeyShot Studio

KeyShot Studio is a real-time ray tracing and global illumination program developed by Luxion for rendering 3D models and animations. This guide provides step-by-step instructions for using AWS Deadline Cloud (Deadline Cloud) with KeyShot Studio to render your projects faster by distributing rendering tasks across multiple machines.

Support overview

KeyShot Studio is supported by the following components:

- **Submitter**: Integrated submitter extension for direct job submission from KeyShot with automatic scene and asset detection.
- **Cross-platform compatibility**: Submitter support for Windows and macOS with worker support for Windows.
- **Licensing (BYOL)**: Bring Your Own License for KeyShot rendering on your farm.

KeyShot version compatibility

The following table shows current support levels for Keyshot versions:

Major Version	Submitter Support	Render Engines	Licensing
2024	Windows, macOS	Built-in ray tracer	BYOL required
2025	Windows, macOS	Built-in ray tracer	BYOL required

Prerequisites

KeyShot requires **Bring Your Own License (BYOL)**. You must have valid KeyShot licenses available for your render farm fleet. Configure your license server to be accessible from worker nodes. For more information, see Connect service-managed fleets to a custom license server.

To use KeyShot on service-managed fleets, you must create a conda package and host it in a custom conda channel. A sample conda recipe for KeyShot is available on the GitHub website. For more information about creating custom conda channels, see Creating custom conda channels.

Before you begin, make sure that you have:

- A Windows or macOS workstation.
- KeyShot Studio 2023 - 2025.
- Deadline Cloud monitor installed.
- Access to an Deadline Cloud farm with a fleet that has KeyShot Studio installed and licensed.

Getting started

To use KeyShot with Deadline Cloud:

1. Create a service-managed fleet and associate it with a queue. Your queue must be set up with a queue environment that includes your custom conda channel that contains the KeyShot package. For more information, see Creating a queue environment.
2. Install the Deadline Cloud monitor and KeyShot submitter on your artist workstation using the Deadline Cloud Submitter and monitor installers. For more information, see Set up your workstation.

Installation

The KeyShot submitter extension allows you to submit jobs to Deadline Cloud directly from within KeyShot.

Installing the submitter

To install the submitter:

1. Download the Deadline Cloud submitter installer.
2. Run the installer and follow the on-screen instructions.
3. Launch KeyShot after installation.

Updating the submitter

To update the submitter to the latest version, download and run the latest submitter installer.

Using the KeyShot submitter

Preparing your scene

Before submitting a job:

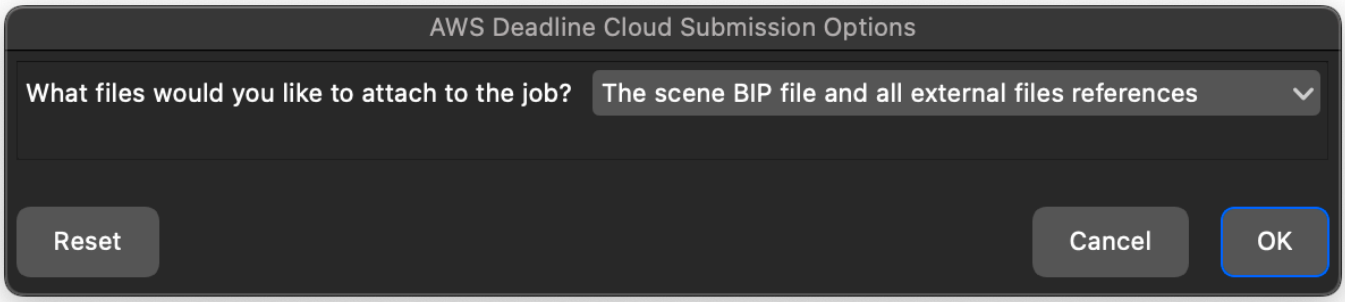
1. Make sure that your scene is saved.
2. Set up your camera angles, materials, and lighting as desired.
3. Configure animation frames if rendering an animation.

Submitting a job

1. In KeyShot, on the top toolbar, choose **Scripting Console**.
2. In the Scripting Console, navigate to **Scripts** > **Submit to Deadline Cloud**.
3. Choose **Run**.

Submission options

When you run the submitter, a dialog asks how you want to handle file attachments.



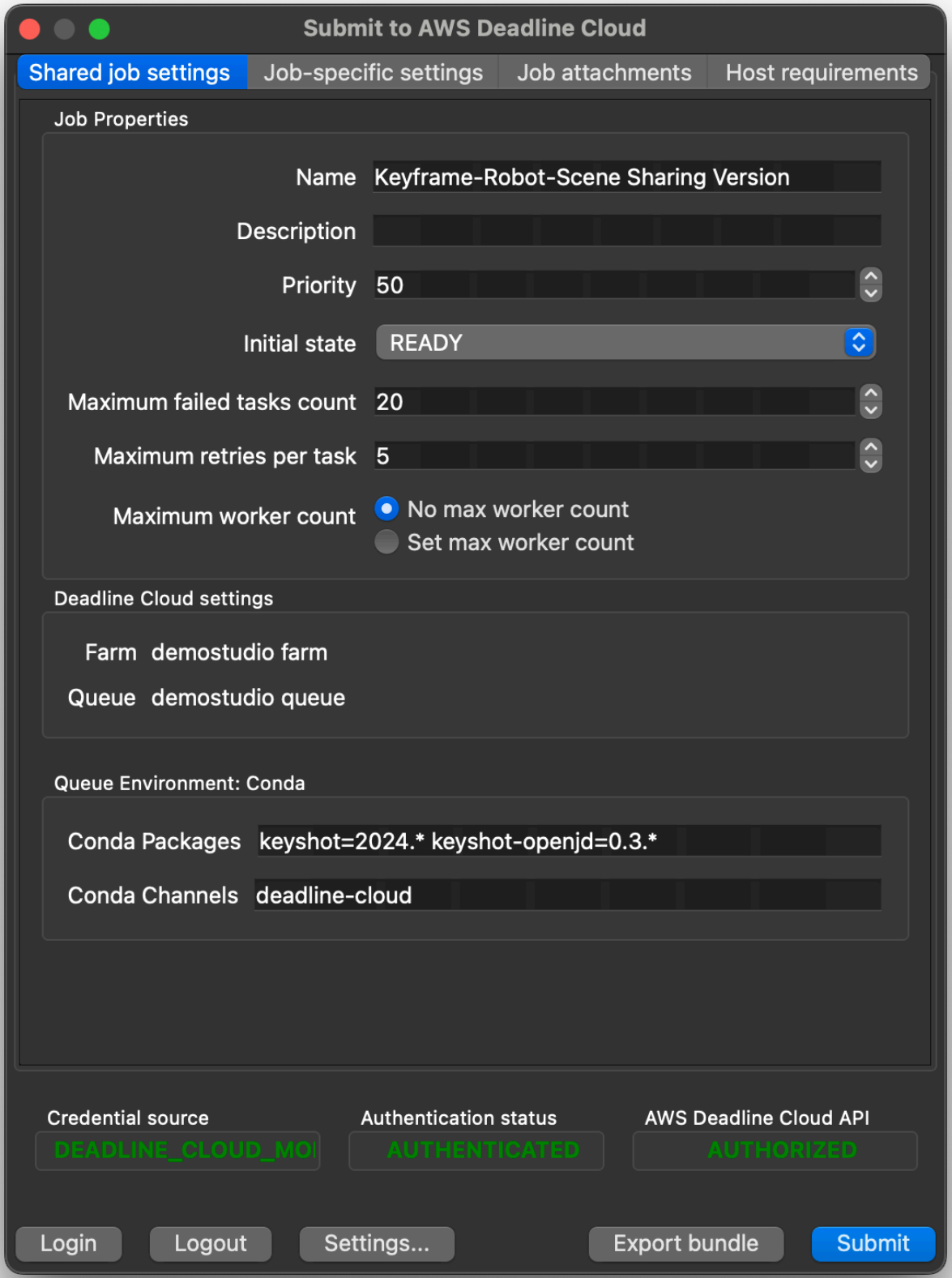
Choose one of the following options:

- **The scene BIP file and all external file references** (Recommended)
  - Automatically packages your scene file and all referenced files. Internally, the submitter creates a KeyShot Package (KSP) which bundles all linked files and uses relative paths.
  - Best for scenes with textures, models, and other external assets.
  - Ensures workers have all necessary files to render your scene.
- **Only the scene BIP file**
  - Only submits the KeyShot scene file.
  - Use this option if your workers already have access to all referenced files.
  - Requires shared network storage or another method to access external files.

Render settings

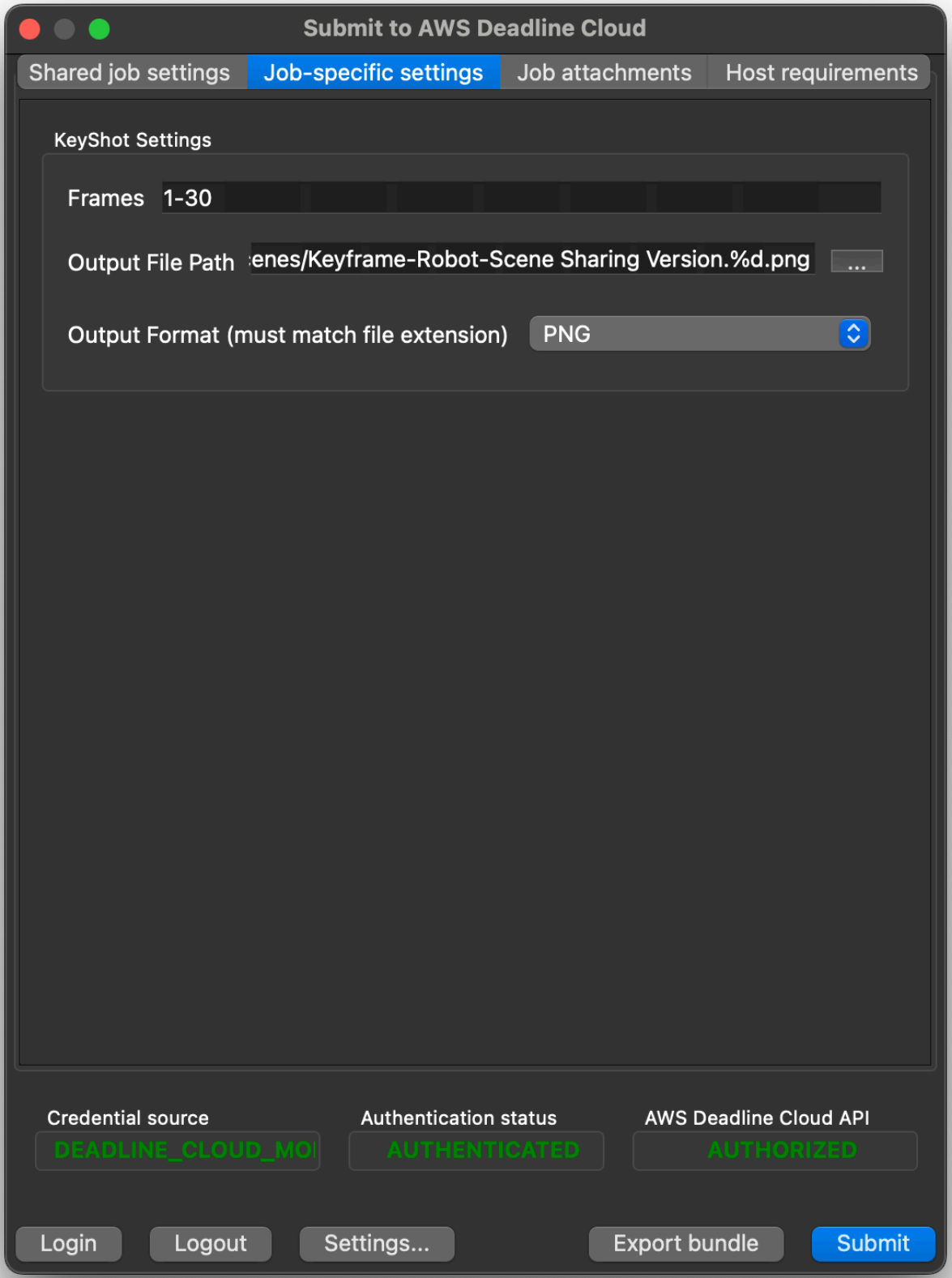
After selecting your submission option, the Deadline Cloud submitter interface appears. Configure your render settings:

1. **Shared job settings**.


Submit to AWS Deadline Cloud
Shared job settings
Job-specific settings
Job attachments
Host requirements
Job Properties
Name
Keyframe-Robot-Scene Sharing Version
Description
Priority
50
Initial state
READY
Maximum failed tasks count
20
Maximum retries per task
5
Maximum worker count
No max worker count
Set max worker count
Deadline Cloud settings
Farm demostudio farm
Queue demostudio queue
Queue Environment: Conda
Conda Packages
keyshot=2024.* keyshot-openjd=0.3.*
Conda Channels
deadline-cloud
Credential source
DEADLINE_CLOUD_MO
Authentication status
AUTHENTICATED
AWS Deadline Cloud API
AUTHORIZED
Login
Logout
Settings...
Export bundle
Submit

- **Job Name**: Give your job a descriptive name.
- If you are using the submitter for the first time, you might need to set your farm and queue. To set them, choose the **Settings** button.

2. **Job-specific settings**.


Submit to AWS Deadline Cloud
Shared job settings
Job-specific settings
Job attachments
Host requirements
KeyShot Settings
Frames 1-30
Output File Path enes/Keyframe-Robot-Scene Sharing Version.%d.png
...
Output Format (must match file extension)
PNG
Credential source
DEADLINE_CLOUD_MO
Authentication status
AUTHENTICATED
AWS Deadline Cloud API
AUTHORIZED
Login
Logout
Settings...
Export bundle
Submit

- **Frames**: Specify which frames to render (for example, `1-30` for frames 1 through 30).
- **Output File Path**: Set the location and naming pattern for rendered images. The path must include the file's extension, and the extension must match the output format. Use `%d` as a placeholder for the frame number.
- **Output Format**: Choose the image format (PNG, JPEG, EXR, TIFF, PSD).

3. **Job attachments** (optional). Select which files are uploaded and attached to the job. The submitter automatically detects and attaches files by default.
4. **Host requirements** (optional). Specify which types of hosts are eligible to pick up tasks for this job.
5. Choose **Submit** to send your job to Deadline Cloud.

Advanced configurations

Using unsupported versions

Deadline Cloud only supports and tests the workstation and worker software versions in the table above. When using the submitter, the worker uses the KeyShot version from your custom conda package. Ensure that your custom conda channel contains packages for all KeyShot versions that you intend to use.

If you require an unsupported version of KeyShot, you can build a custom conda recipe and channel for your desired version to be installed on the worker. Use the sample conda recipe for KeyShot on the GitHub website as a starting point. For more information about creating custom conda channels, see Creating custom conda channels.

Open source resources

The submitter is open source and available on the GitHub website:

- Deadline Cloud for KeyShot.
- KeyShot conda recipe.
- Standalone KeyShot job bundle.

Maxon Cinema 4D

Cinema 4D is a professional 3D animation, modeling, simulation, and rendering software solution from Maxon. Cinema 4D is supported by AWS Deadline Cloud (Deadline Cloud) including a submitter, conda packages, usage-based licensing, and an adaptor for improved performance. This guide walks you through using Deadline Cloud with Cinema 4D - from installation to your first successful render.

The following are reasons to use Deadline Cloud for Cinema 4D:

- **Scale your renders** - Render complex scenes faster by distributing frames across multiple instances, reducing render times from hours to minutes.
- **Free up your workstation** - Submit renders to Deadline Cloud and continue working on your next project while your scenes render in the background.
- **Pay only for what you use** - No upfront costs or long-term commitments. Pay only for the compute time you actually use.
- **Redshift-ready** - Full support for Maxon Cinema 4D and Maxon Redshift.

Support overview

Cinema 4D is supported by the following components:

- **Submitter**: Integrated submitter for direct job submission from Cinema 4D with automatic scene and asset detection.
- **Conda packages**: Automatic installation on service-managed fleets when using the submitter.
- **Adaptor**: A program on worker hosts that keeps the application loaded between tasks for faster rendering and reports render progress.
- **Cross-platform compatibility**: Submitter support for Windows and macOS with worker support for Windows and Linux with automatic path mapping.
- **Usage-based Licensing**: Pay-as-you-go licensing for Cinema 4D, Redshift, and Red Giant licensing.

Cinema 4D version compatibility

The following table shows current support levels for Cinema 4D versions:

Major Version	Submitter Support	Conda Support	Usage-Based Licensing
2024	Windows, macOS	Windows	Usage-based licensing available
2025	Windows, macOS	Windows, Linux	Usage-based licensing available
2026	Windows, macOS	Windows, Linux	Usage-based licensing available

Deadline Cloud Conda Channel

The following table lists all conda packages applicable to Cinema 4D available to Service-managed fleets in the deadline-cloud conda channel:

OS	Package	Version	Notes
Windows	cinema4d	2024	Includes Standard, Physical and Redshift renderers
Windows, Linux	cinema4d	2025	Includes Standard, Physical and Redshift renderers
Windows, Linux	cinema4d	2026	Includes Standard, Physical and Redshift renderers
Windows, Linux	cinema4d-c4dtoa	2025	Cinema4D to Arnold
Windows	cinema4d-c4dtoa	2026	Cinema4D to Arnold
Windows, Linux	cinema4d-openjd		Includes the Cinema 4D Adaptor

Note

For **Cinema 4D**, the Linux conda package does not support substance 3D materials. Jobs with this material fail with one of the following errors:

```
Commandline: ./modules/io_substance/source/substance_framework/src/details/
detailsengine.cpp:794:
 SubstanceAir::Details::Engine::Context::Context(SubstanceAir::Details::Engine&,
 SubstanceAir::RenderCallbacks*): Assertion `res==0' failed.
```

```
/home/job-user/.conda/envs/<hash>/Lib/deadline/cinema4d_adaptor/Cinema4DAdaptor/
adaptor.sh: line 44: 10832 Segmentation fault      (core dumped) $C4DEXE
 ${ARGS[*]}
```

We recommend that you submit jobs with substance materials to Windows instead.
In Cinema 4D 2025.3.3 on Linux, globalized asset paths can cause segmentation faults. Therefore, the Linux conda package contains Cinema 4D 2025.3.1 with Redshift 2025.6.0 instead. If you need features or bug fixes from Cinema 4D 2025.3.3, we recommend two options: upgrade to Cinema 4D 2026 or submit those jobs to Windows instead.
For **Cinema 4D OpenJD,** to prevent any timeout issues, we recommend you set task run timeouts to double their expected render time, instead of using the default 2 day timeout.

Getting started

To use Cinema 4D fully-managed on Deadline Cloud:

1. Create a service-managed fleet and associate it with a queue. Configure the fleet with GPU support if you intend to use Redshift or Red Giant features that require a GPU. Your queue should be set up with a queue environment that supports the deadline-cloud conda channel. For more information, see Creating a queue environment.
2. Install the Deadline Cloud monitor and Cinema 4D submitter on your artist workstation using the Deadline Cloud Submitter and monitor Installers. For more information, see Set up your workstation.
3. Submit your job directly from Cinema 4D using the integrated submitter to the queue.
4. Monitor the job and download the output using the Deadline Cloud monitor.

Quick start

Set up Cinema 4D and Deadline Cloud in just a few steps.

What you need

- **Cinema 4D 2024 - 2026** installed on your workstation.
  - Redshift, Arnold, and Cargo are supported natively.
- **Windows or macOS** workstation for job submission.
- Deadline Cloud monitor installed.
- **Access to an Deadline Cloud farm** with either:
  - A Windows service-managed fleet, or
  - A customer-managed fleet with Cinema 4D, the Cinema 4D adaptor, and licensing set up.

Step 1: Install the submitter

The submitter adds Deadline Cloud functionality to Cinema 4D's Extensions menu, allowing you to submit your scene directly to Deadline Cloud to manage the rendering.

Download the official installer (recommended).

1. Run the installer and follow the on-screen instructions.
2. Launch Cinema 4D after installation.
3. Verify the submitter appears in **Extensions** > **Deadline Cloud Submitter**.

Updating the submitter

To update the submitter to the latest version, download and run the latest submitter installer.

System-wide installation for multiple users (Windows)

For shared workstations or enterprise environments where multiple users need access to the Cinema 4D submitter, you can perform a system-wide installation.

Make sure that you have the following prerequisites:

- Administrator account access.

- Cinema 4D installed on the system.

Installation steps:

1. Install the submitter as Administrator:
   - Run the Deadline Cloud submitter installer as Administrator.
   - Select "system installation" option during installation.
2. Initial dependency setup:
   - Open Cinema 4D as Administrator (right-click → "Run as administrator").
   - Choose **Extensions**, **Deadline Cloud Submitter**.
   - Choose "Yes" when prompted to install GUI dependencies.
   - This step configures permissions so all users can access the installed packages.
3. Regular usage:
   - After initial setup, any user can open Cinema 4D normally (without Administrator privileges).
   - The Deadline Cloud submitter is available to all users.

For troubleshooting permission issues, see Troubleshooting.

Step 2: Submit your first render

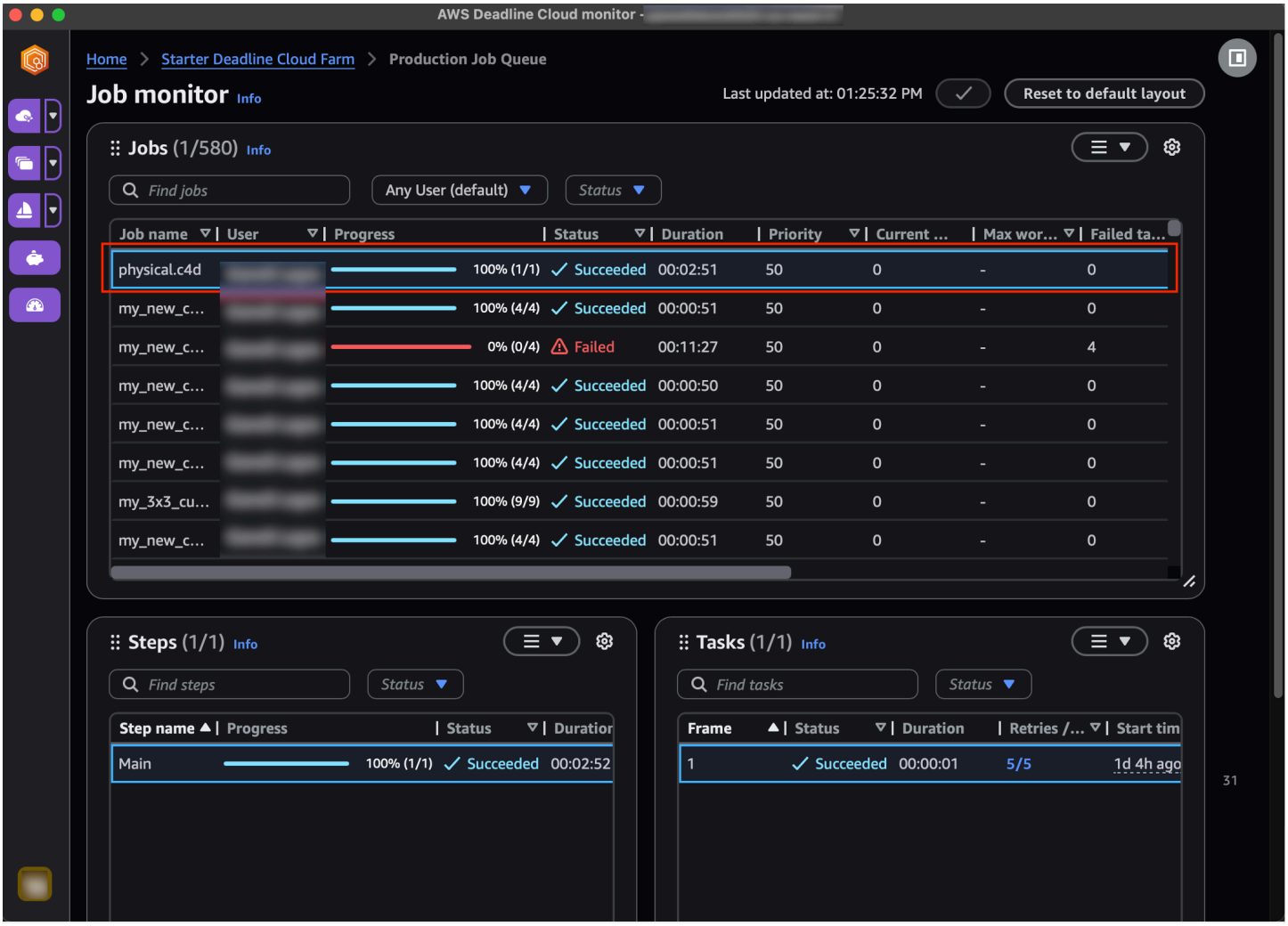
1. Open Cinema 4D and load a scene.
2. Make sure your scene is saved.
3. Set up your camera angles, materials, and lighting as desired.
4. Choose **Extensions**, **Deadline Cloud Submitter**.

5. Review your render settings.
6. Choose **Submit**.

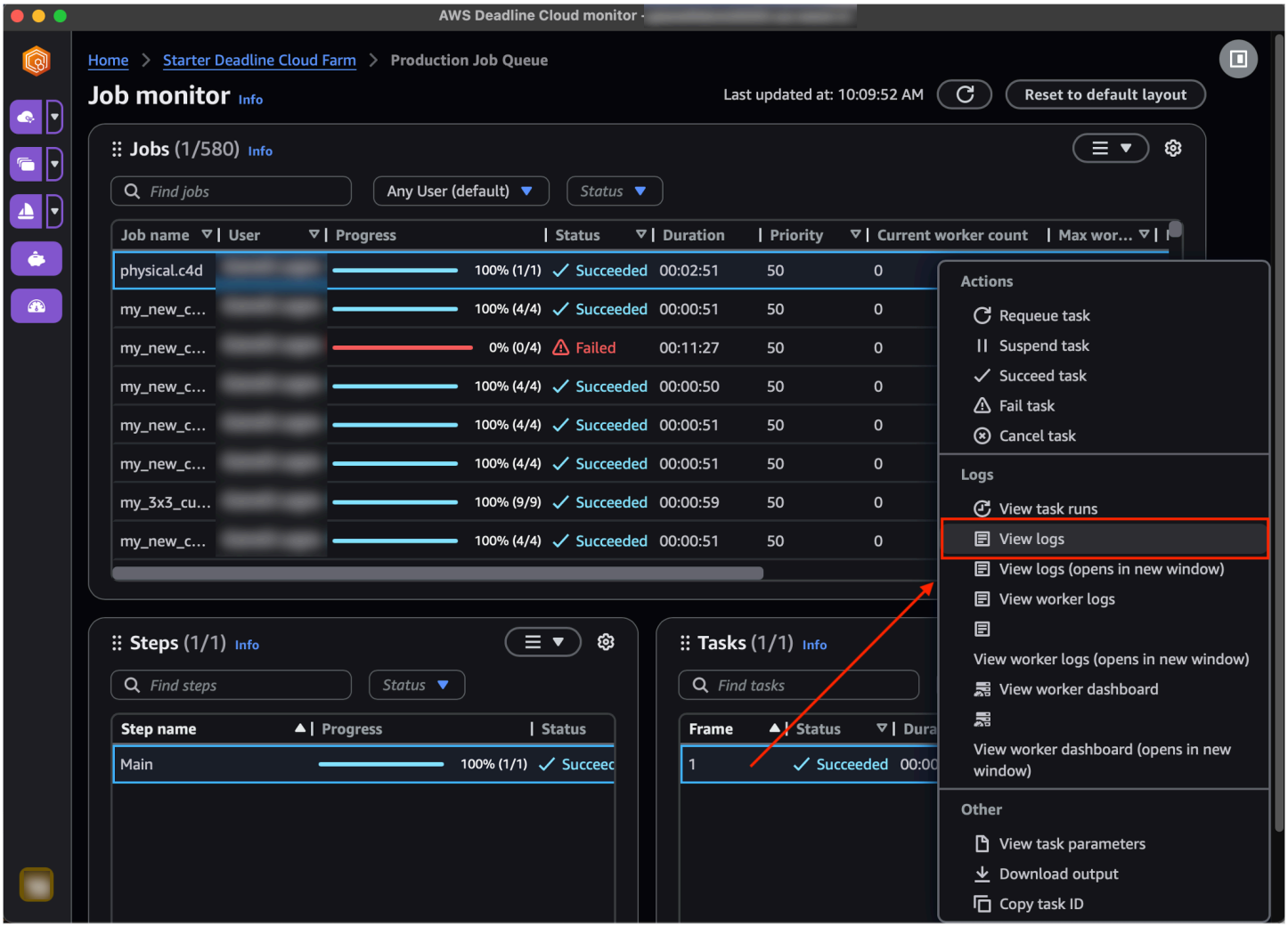
Step 3: Monitor your renders

If you haven't already, install Deadline Cloud monitor from the requirements above.

After submitting a job, open Deadline Cloud monitor to view the job's progress. The submitter creates a job with a single step and one task per frame.



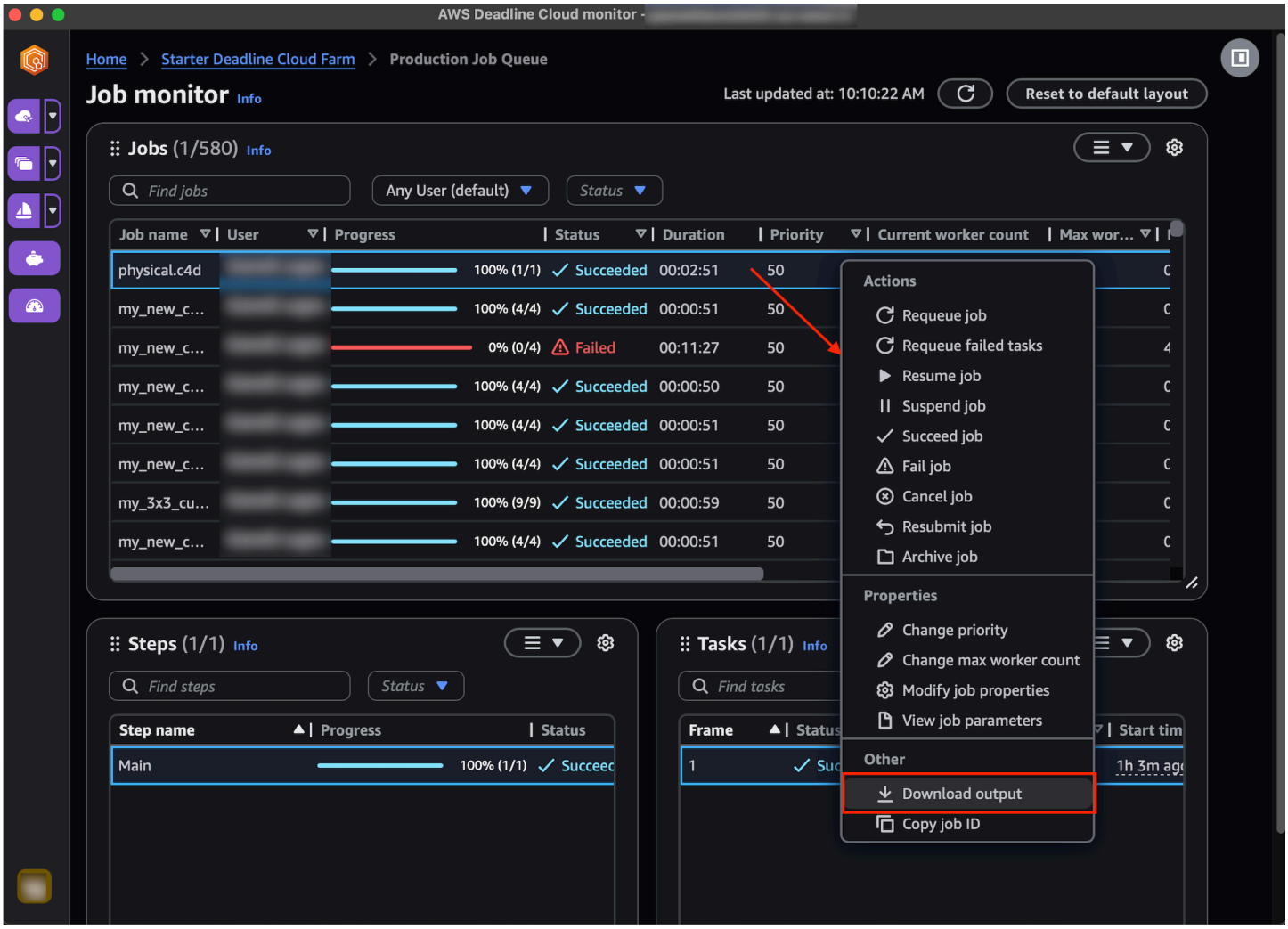
To view rendering logs, open the context (right-click) menu for a task, and then choose **View logs**. Viewing logs is especially useful for troubleshooting failed jobs.



Step 4: Download your results

Once your render job completes successfully, you can download the rendered frames.

1. In Deadline Cloud monitor, locate your completed job.
2. Open the context (right-click) menu for the job.
3. Choose **Download output**.
4. Choose where to save your rendered files.
5. The download begins automatically.



Your rendered frames are organized in the same structure as specified in your output settings.

Submitter features

The Cinema 4D submitter provides automation and configuration options for your rendering workflow.

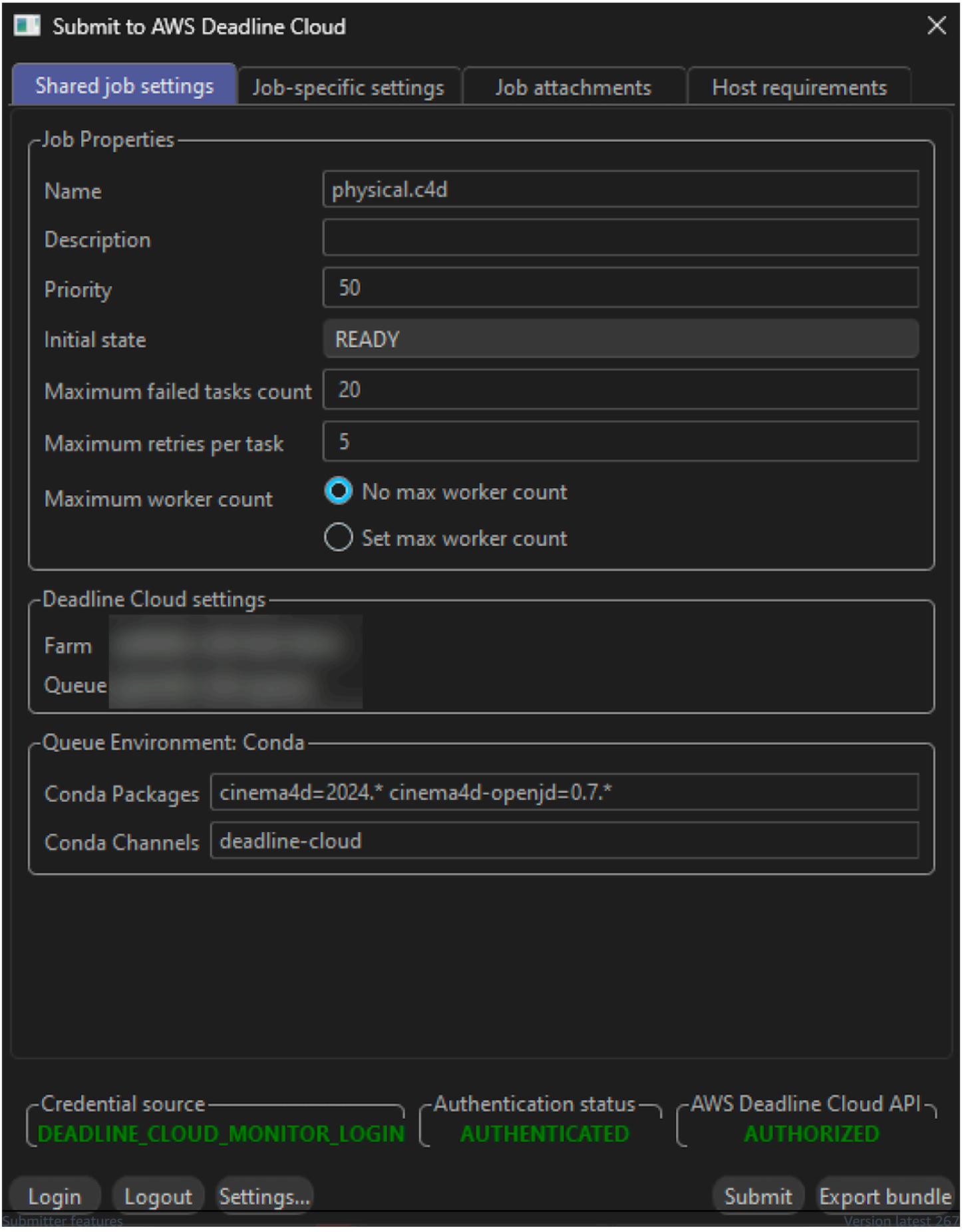
Key benefits

- **Smart asset detection** - Automatically finds and includes all textures, models, and other files your scene needs. No more missing asset errors or manual file hunting.
- **Advanced settings** - Provides additional configuration options beyond basic settings, allowing you to customize output paths, frame ranges, takes, and error checking to fit your workflow.

Shared job settings

Settings that apply to the entire job:

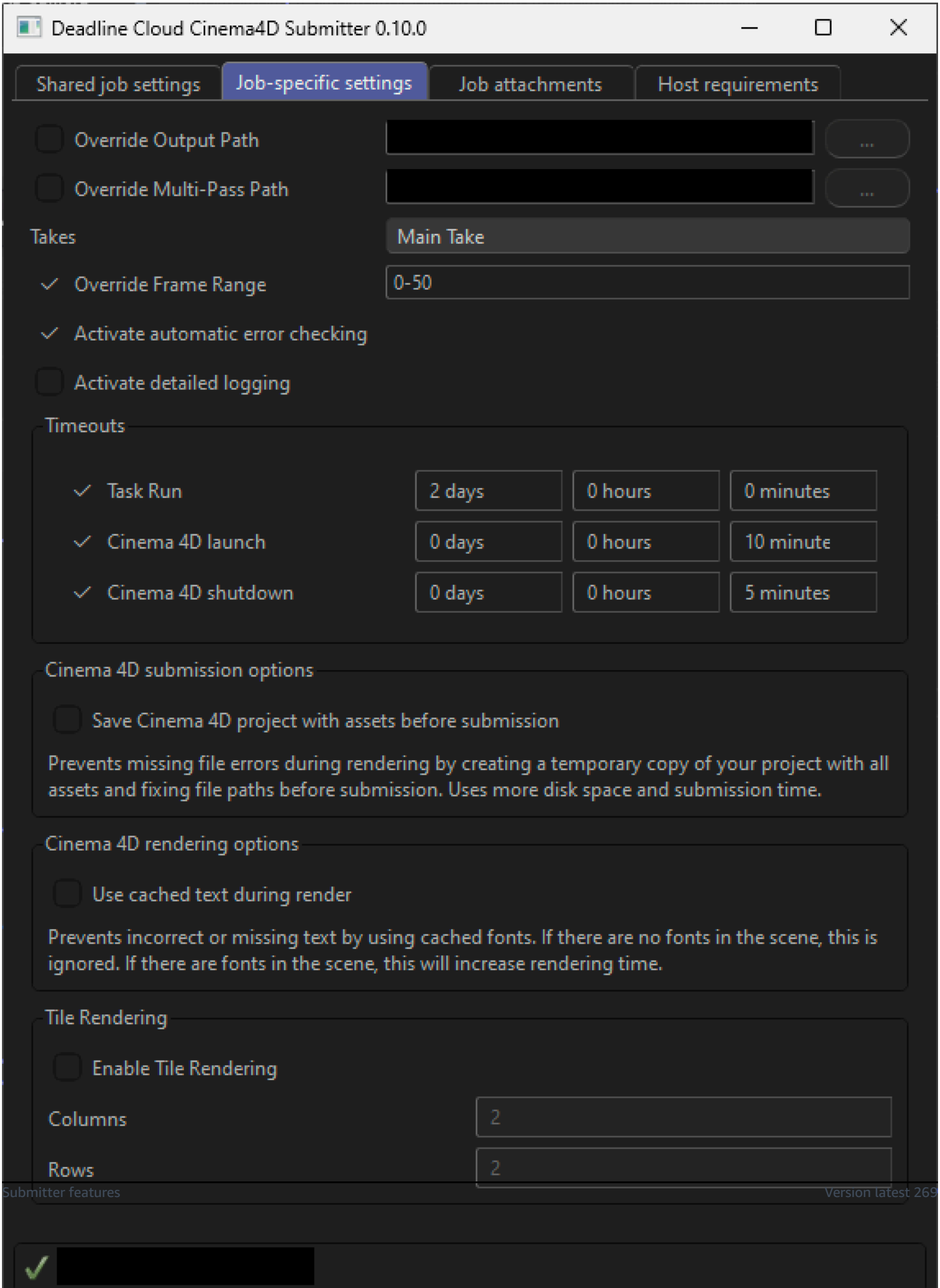
- **Farm Selection** - Choose which farm your job will render on.
- **Queue Selection** - Select the specific queue within your chosen farm.
- **Job Name** - Give your render job a descriptive name.
- **Job Description** - Add optional details about your render job.
- **Priority** - Set job priority for queue management.
- **Initial State** - Control whether the job starts immediately or remains paused.
- **Max Failed Tasks Count** - Maximum number of tasks that can fail before the job is marked as failed.
- **Max Retries Per Task** - Number of times a failed task will be retried.
- **Max Worker Count** - Maximum number of workers that can work on this job simultaneously.
- **Conda Packages** - Specify additional conda packages required for your render.
- **Conda Channels** - Define custom conda channels for package installation.


Submit to AWS Deadline Cloud
Shared job settings
Job-specific settings
Job attachments
Host requirements
Job Properties
Name
physical.c4d
Description
Priority
50
Initial state
READY
Maximum failed tasks count
20
Maximum retries per task
5
Maximum worker count
No max worker count
Set max worker count
Deadline Cloud settings
Farm
Queue
Queue Environment: Conda
Conda Packages
cinema4d=2024.* cinema4d-openjd=0.7.*
Conda Channels
deadline-cloud
Credential source
DEADLINE_CLOUD_MONITOR_LOGIN
Authentication status
AUTHENTICATED
AWS Deadline Cloud API
AUTHORIZED
Login
Logout
Settings...
Submit
Export bundle
Submitter features
Version latest 267

Job-specific settings

Settings specific to your Cinema 4D render:

- **Override Output Path** - Override the main render output path from your scene settings.
- **Override Multipass Path** - Override the multipass output path for additional render passes.
- **Takes** – Select which Cinema 4D takes to render: **Main Take**, **All Takes**, **Marked Takes** (takes with the check mark set in the Marker column of the Take Manager), or **Current Take** (the take activated with the take assignment icon at the far left of the take's name in the Take Manager). For more information, see Take system questions.
- **Override Frame Range** - Override the frame range from your scene settings.
- **Automatic Error Checking** - Optional checkbox to activate or deactivate error checking during rendering.
- **Activate detailed logging** - Enable detailed logging to capture detailed logs for debugging rendering issues. When enabled, debug logs are automatically captured and printed to the job output for easy review.
- **Task Run Timeout** - Maximum time allowed for each task to complete.
- **Cinema 4D Launch Timeout** - Maximum time allowed for Cinema 4D to start up.
- **Cinema 4D Shutdown Timeout** - Maximum time allowed for Cinema 4D to shut down cleanly.
- **Save Cinema 4D Project with Assets** - Prevents missing file errors during rendering by creating a temporary copy of your project with all assets and fixing file paths before submission. Uses more disk space and submission time.
- **Use cached text during render** - Prevents incorrect or missing text by re-animating the text on the worker using the cached fonts for each frame. Will increase rendering time.
- **Tile Rendering** - Split each frame into a grid of tiles that render in parallel across multiple workers, then automatically assemble into the final image. Configure the number of columns and rows (1-99 each, default 2x2). See Tile rendering below for details.
- **Frames per chunk** – Number of frames to group into each chunk (1-150). Use 1 for one frame per task (default). Higher values reduce per-task overhead. When you set Target chunk duration, this value serves only as the initial chunk size. For more information, see Task chunking for job templates.
- **Target chunk duration** – Target render time per chunk in seconds. Deadline Cloud automatically adjusts how many frames to group together to reach this target. To always use the fixed frames-per-chunk value, set this to 0.


Deadline Cloud Cinema4D Submitter 0.10.0
Shared job settings
Job-specific settings
Job attachments
Host requirements
Override Output Path
Override Multi-Pass Path
Takes
Main Take
Override Frame Range
0-50
Activate automatic error checking
Activate detailed logging
Timeouts
Task Run
2 days
0 hours
0 minutes
Cinema 4D launch
0 days
0 hours
10 minute
Cinema 4D shutdown
0 days
0 hours
5 minutes
Cinema 4D submission options
Save Cinema 4D project with assets before submission
Prevents missing file errors during rendering by creating a temporary copy of your project with all assets and fixing file paths before submission. Uses more disk space and submission time.
Cinema 4D rendering options
Use cached text during render
Prevents incorrect or missing text by using cached fonts. If there are no fonts in the scene, this is ignored. If there are fonts in the scene, this will increase rendering time.
Tile Rendering
Enable Tile Rendering
Columns
2
Rows
2

Optional tabs

- **Job Attachments** (optional) - Select which files will be uploaded and attached to the job. Files are automatically detected and attached by default.
- **Host Requirements** (optional) - Allows you to specify which types of hosts will be eligible for picking up tasks for this job.

The submitter handles the technical details so you can focus on your creative work.

Tile rendering

Tile rendering splits each frame into a grid of smaller tiles that render independently across multiple workers, then automatically assembles them into the final full-resolution image. Tile rendering is useful for large or complex single-frame scenes where a single frame takes a long time to render.

How to enable

1. In the **Job-Specific Settings** tab, find the **Tile Rendering** group.
2. Select **Enable Tile Rendering**.
3. Set the number of **Columns** and **Rows** for the tile grid (default: 2x2).

How it works

When enabled, the submitter creates a two-step job for each take:

1. **Render step** - Each tile is a separate task. A 3x3 grid produces 9 tile tasks per frame, each rendering only its assigned tile.
2. **Assembly step** - After all tiles for a frame finish, an assembly task automatically stitches them into the final full-resolution image. Both beauty and multi-pass outputs are assembled.

No manual stitching or external tools required.

Downloading outputs

The Render step produces intermediary tile image files (for example, `image_0_tile_0_0.`png, `image_0_tile_1_0.`png) that are used as input for assembly. If you only need the final

composited images, download the output from the **Assemble Tiles** step rather than the Render step. The Assemble Tiles step contains only the final full-resolution images with all tiles stitched together.

Detailed logging for debugging

The **Activate detailed logging** feature helps you troubleshoot rendering issues by capturing detailed logs.

When to use detailed logging

Enable detailed logging when you need to:

- Debug Redshift rendering issues or unexpected behavior.
- Investigate rendering artifacts or errors.
- Analyze renderer performance and behavior.
- Capture detailed information for technical support.

How it works

When you enable the "Activate detailed logging" checkbox in the Job-Specific Settings:

1. **Log capture** - The system automatically enables Redshift debug logging by setting the `REDSHIFT_DEBUGCAPTURE` environment variable.
2. **Log output** - After rendering completes, all Redshift logs are automatically printed to the job output for easy review.

Viewing the logs

To view the detailed logs:

1. Open Deadline Cloud monitor.
2. Navigate to your completed job.
3. Open the context (right-click) menu for a task, and then choose **View logs**.
4. Enable the **View logs for all tasks** button.
5. Scroll through the task run logs to find the "Shut down DetailedLogging" section for detailed logs.

The Redshift logs are output in HTML format and include timestamps at the start of each line. If you want to save and view the logs in a web browser without timestamps, you can use the provided cleanup utility:

1. Download the detailed log file from Deadline Cloud monitor. Open the context (right-click) menu for the task, and then choose **Download logs**.
2. Run the cleanup script with the log file path:

   ```
   cd deadline-cloud-for-cinema-4d/scripts
   python clean_redshift_detailed_logs.py /path/to/detailed_logs.log
   ```

   Or run it without arguments and enter the path when prompted:

   ```
   python clean_redshift_detailed_logs.py
   ```

3. Open the generated `redshift_log_cleaned.html` file in your web browser.

The cleanup script automatically extracts the Redshift HTML logs from the detailed log file and removes timestamp prefixes (for example, `2024/11/17 14:23:45-08:00`) from each line, making the logs easier to read and compare.

Important notes

- Detailed logging is **disabled by default** to minimize overhead.
- Only enable it when you need to debug specific issues.
- Logs are captured on a per-job basis - each job has its own log output.
- The feature works across Windows and Linux worker nodes.

Troubleshooting

The following sections describe common issues you might encounter when using Cinema 4D with Deadline Cloud and how to resolve them.

Rendering questions

Q: How many tiles should I use?

A: It depends on your scene. A 3x3 or 4x4 grid is a good starting point. More tiles means more parallelism but also more tasks and overhead. The total tasks per frame is (columns x rows) + 1 for assembly. Deadline Cloud has a maximum of 10,000 tasks per step. Exceeding this limit will cause the job to fail with a `CREATE_FAILED` status. For example, a 99x99 grid on a single frame would produce 9,801 render tasks, which is close to the limit.

Q: Why don't fonts work while submitting with macOS?

A: Font functionality is currently only supported on Windows due to technical limitations. This limitation is a known behavior in mixed macOS/Windows environments, as confirmed by Maxon's official documentation. For more information, see How to resolve missing fonts in Team Render for Cinema 4D on the Maxon website.

Take system questions

The Cinema 4D Take system stores multiple states of a scene (different cameras, render settings, object variations) in a single file. The **Takes** setting in the Job-Specific Settings tab of the submitter controls which of these takes the job renders. For more information about the Take system, see the Take Manager documentation on the Maxon website.

Q: I submitted with "Current Take" selected, but the wrong take rendered (usually the Main take). Why?

A: The cause is usually a mix-up between two separate controls in the Cinema 4D Take Manager:

- The **take assignment icon** at the far left of the take's name sets the **current take**. It turns white when active, the take is displayed in the viewport, and the take name appears in the Cinema 4D title bar next to the file name. The submitter's **Current Take** option renders this take.
- The **check mark in the Marker column** only *marks* a take. Marking a take does **not** make it the current take. The submitter only renders marked takes when you select the **Marked Takes** option.

If you check mark a take but don't activate it with the take assignment icon, submitting with **Current Take** renders whichever take is actually current, which is typically the Main take. To fix this, either select the take assignment icon on the take you want before submitting with **Current Take**, or select **Marked Takes** in the submitter to render the takes you check-marked.

Before you submit, verify the correct take is current by checking that its name appears in the Cinema 4D title bar and that the viewport shows the expected scene state.

Q: I rendered multiple takes but only got one set of output files. Why?

A: When you submit multiple takes (**All Takes** or **Marked Takes**) and the output path doesn't distinguish between takes, each take writes to the same file names and the outputs overwrite each other. Include the `$take` token in your output path (in your render settings, or in the submitter's **Override Output Path** field) so each take renders to its own location, for example `renders/$take/frame`. The submitter shows a warning at submission time if you select multiple takes without the `$take` token in the output paths.

Common issues

Q: My submitter button doesn't appear in Cinema 4D.

A: Make sure you've installed the extension correctly and restarted Cinema 4D. Check the Console (**Extensions** > **Console**) for any error messages. Consider re-installing the submitter if the issue persists.

Q: My job completed successfully but I get "no outputs found" when trying to download outputs. Why?

A: The cause is most likely a version mismatch between the Cinema 4D submitter and Deadline Cloud monitor. Jobs submitted with `deadline-cloud-for-cinema-4d` 0.11.1 (or newer) are incompatible with Deadline Cloud monitor 1.1.7 and earlier, which causes output downloads to fail even though the render itself succeeded.

How to verify you're affected:

1. Check your submitter version. The version is displayed in the submitter window title. If the version is `0.11.1` or later, you have the new submitter.
2. Check your Deadline Cloud monitor version. Open Deadline Cloud monitor and view the **About** dialog (on macOS, **Deadline Cloud monitor** menu > **About Deadline Cloud monitor**; on Windows, **Help** menu > **About Deadline Cloud monitor**). If the version is `1.1.7` or earlier, you have the old Deadline Cloud monitor.

If both conditions are true, you are hitting this issue.

Fix: Update Deadline Cloud monitor to **1.1.8 or later**. You can download the latest version from the Deadline Cloud monitor downloads page. After updating, re-open the job in Deadline Cloud monitor and the outputs will download as expected. No re-submission is required.

Q: Why are some of my textures or assets missing in the rendered output?

A: The cause is typically a path mapping issue. Cinema 4D sometimes stores deep links (absolute paths) to assets in the scene file that cannot be edited. When the scene is submitted and rendered on the farm, it might still reference the original workstation paths even though the assets were uploaded.

Workaround: Enable "Save Cinema 4D Project with Assets" in the Job-Specific Settings tab of the submitter. This setting consolidates all assets into the project folder and fixes the paths before submission, ensuring they render correctly on the farm.

Q: Are nested Redshift proxy files detected when submitting Cinema 4D jobs to Deadline Cloud?

A: No, Redshift proxy files are not detected when submitting Cinema 4D jobs to Deadline Cloud. This behavior is a limitation of the Redshift `*.rs` file format. When you export a Cinema 4D scene containing RS Proxy objects to `*.rs`, all referenced proxy data is flattened or inlined into a single file - no external references are preserved. The Cinema 4D SDK cannot read `*.rs` files to discover nested dependencies, and the Redshift Core doesn't expose this functionality either. For more information, see Nested Redshift proxy files on the Maxon website.

Q: I'm getting permission errors with a system-wide installation on Windows. How do I fix this?

A: If users encounter permission errors when accessing the submitter after a system-wide installation:

1. Verify initial setup was completed:
   - Ensure Cinema 4D was opened as Administrator at least once.
   - Ensure the dependency installation prompt was accepted.
   - Check that the installation completed without errors.
2. Check file permissions:
   - Navigate to the installation directory (for example, `C:\Program Files \DeadlineCloudSubmitter\`).
   - Right-click → Properties → Security tab.
   - Verify that "Users" group has "Read & execute" permissions.
   - Permissions should be inherited by all subdirectories and files.
3. Manual permission fix (if needed):

- Open Command Prompt as Administrator.
- Run: `icacls "C:\Program Files\DeadlineCloudSubmitter" /grant *S-1-5-32-545:(OI)(CI)(RX) /T`.
- This grants read and execute permissions to all users.

4. Verify Cinema 4D Python environment:
   - Open Cinema 4D as the affected user.
   - Choose **Extensions**, **Console**.
   - Try importing: `import deadline`.
   - If this fails, the permissions may not be correctly applied.

Getting support

This section guides you through troubleshooting steps and how to get support when you need it.

Before you contact support

Before reaching out for help, try these troubleshooting steps. They often resolve common issues and will help you provide better information if you do need to contact support.

Troubleshooting checklist:

- **Render one frame locally** - Before submitting to the cloud, render at least one frame locally in Cinema 4D to verify your scene renders correctly. This step helps identify scene-specific issues vs. cloud rendering issues.
- **Update to the latest submitter** - We release updates frequently with bug fixes and improvements. Your issue may already be fixed in a newer version. To check if you're running the latest version:
  - Find your current version: The version is displayed in the submitter window title.
  - Compare with the latest release: Visit the releases page on the GitHub website to see the most recent version.
  - If your version is older, update the submitter and test again before reporting the issue.
- **Check your Job-Specific Settings** - Review the Job-Specific Settings tab in the submitter. In particular, enable the **Save Cinema 4D Project with Assets** checkbox. This setting creates a temporary copy of your project with all assets and fixes file paths, helping identify missing files and organizing assets for render farms. See Job-specific settings for more information.

- **Try different Cinema 4D versions** - If you're experiencing issues, test with Cinema 4D 2024, 2025, or 2026 to see if the problem is version-specific.
- **Check existing GitHub issues** - Search the issues page on the GitHub website to see if someone else has already reported your problem and found a solution.
- **Try a different fleet operating system** - Submit your job to both Windows and Linux fleets if available. Windows generally has better support and compatibility for Cinema 4D features.
- **Review session logs** - Download and review the session logs from Deadline Cloud monitor. These logs often contain error messages that generally pinpoint the issue.
- **Create a scene project file** - Use Cinema 4D's **File** > **Save Project with Assets** to create a self-contained project that includes all dependencies. Zip this file for easy sharing with support.

When to contact support

Different types of issues should be directed to different support channels.

AWS general support

Contact AWS Support for:

- AWS account issues.
- Billing questions.
- General AWS service questions.
- Deadline Cloud internal server errors in Deadline Cloud monitor or while using the CLI.

You can also report Cinema 4D submitter or adaptor issues through AWS Support, but note that these requests might take longer because they need to be routed to the maintainers of the integration repository. For faster response on Cinema 4D-specific issues, we recommend using GitHub issues.

Contact AWS Support.

Cinema 4D submitter or adaptor support

Use GitHub issues as your primary channel for Cinema 4D-specific problems:

- Submitter bugs or crashes.
- Adaptor issues.

- Rendering failures specific to Cinema 4D.
- Feature requests.
- Integration problems.

To report an issue, see Cinema 4D submitter and adaptor issues on the GitHub website.

How to report issues on GitHub

Bug reports

Before creating a new bug report:

1. Search existing bugs on the GitHub website to see if your problem has already been reported or fixed in latest versions.
2. If you find an existing issue that matches your problem:
   - Add a thumbs-up reaction to help us prioritize.
   - Comment with any additional details or reproduction steps you can provide.
   - This information helps us understand how many users are affected.

If no existing issue matches, create a new bug report on the GitHub website using the bug report template. The template will guide you through providing all the necessary information.

Feature requests

Before creating a new feature request:

1. Search existing feature requests on the GitHub website to see if someone has already suggested your idea.
2. If you find an existing request that matches:
   - Add a thumbs-up reaction to show your support.
   - Comment with your specific use case - this strengthens the request and helps us understand different needs.
   - The more users who express interest, the higher priority it receives.

If no existing request matches, create a new feature request on the GitHub website using the feature request template. The template will guide you through providing all the necessary information.

> **Note**
>
> Feature requests help us prioritize development, but there's no guarantee of implementation timeline.

What to include in your support request

Required information checklist:

When contacting support or creating a GitHub issue, always include:

- **Cinema 4D version** - for example, Cinema 4D 2025.1.0.
- **Operating system** - for example, Windows 11, macOS 14.2.
- **Submitter version** - Copy the entire contents of the About panel (**Extensions** > **Deadline Cloud Submitter** > **About**).
- **Renderer** - Standard, Redshift, Arnold, etc.
- **Fleet configuration** - Worker OS (Windows or Linux), memory requirements, disk space, GPU type if using Redshift.
- **Error messages** - Complete error text, not paraphrased.

How to gather log files

Logs are essential for diagnosing issues. To enable detailed logging:

1. In the Cinema 4D submitter, select **Activate detailed logging** in Job-Specific Settings.
2. Submit your job.
3. After the job completes (or fails), retrieve the logs:
   - Open Deadline Cloud monitor.
   - Navigate to your job.
   - Choose **Download logs** and select **Entire session** to download the full session logs for sharing with support.

How to create scene project files

A scene project file packages your Cinema 4D scene with all its assets:

1. First, remove any confidential assets or data from your scene before saving.
2. In Cinema 4D, go to **File** > **Save Project with Assets**.
3. Choose a destination folder.
4. Cinema 4D will copy your scene and all referenced assets to this folder.
5. Zip the entire project folder.
6. Share the zip file with support.

> ⚠ **Important**
>
> Remove confidential assets *before* using "Save Project with Assets" - not after. Removing assets after saving can cause missing file errors that make it harder to diagnose your original issue.

Tip: The best way to share a reproducible test case is with a publicly available scene or a simplified scene that demonstrates the issue without confidential material. If you can recreate the problem with non-proprietary assets, that makes it much easier for support to investigate.

Advanced configurations

Using unsupported versions

Deadline Cloud only supports and tests the workstation and worker software versions in the table above. When using the submitter, the worker will attempt to install the same version as used on the workstation. This installation fails if the workstation version of Cinema 4D does not appear in the version table above.

If you require an unsupported version of Cinema 4D, you can build a custom conda recipe and channel for your desired version to be installed on the worker. Use the conda recipe for a supported version linked in the Open Source Resources section below as a starting point, and package your desired version in a custom conda channel. For more information about creating custom conda channels, see Creating custom conda channels.

If you create a conda package for a different version of Cinema 4D, you should ensure it will acquire a license correctly. If the version is compatible with licensing for a supported version in the table above, then usage-based licensing will work automatically. You may also bring your own license to a service-managed fleet by following Connect service-managed fleets to a custom license server.

Cinema 4D plugins

Plugin	Plugin Version	Conda Recipe Provided	SMF Conda Package Provided	Usage-based Licensing Support
Redshift	2026.3.0	Bundled*	Yes	Yes
Redshift	2025.6.0	Bundled*	Yes	Yes
Red Giant	2025.x	No	No	Yes
V-Ray	7.x	Yes	No	Yes
Insydium X-Particles	2024.x	Yes	No	N/A
C4DtoArnold	4.8.4.1	Yes	Yes	Yes

*Included in the base Cinema 4D package recipe

Maxon Redshift

The Redshift renderer is included with all Cinema 4D conda packages and is automatically used when appropriate when using the Cinema 4D integrated submitter. An additional licensing cost applies when using Redshift for rendering. For more information about Deadline Cloud pricing, see Deadline Cloud pricing.

Maxon Red Giant

Red Giant is a comprehensive toolkit designed for video post-production, motion graphics, and visual effects. It offers rich color grading, smooth transitions, realistic visual effects, motion design templates and tools to create and edit your visuals. For more information, see Red Giant on the Maxon website.

Red Giant requires custom setup on service-managed fleets. A host configuration script is provided which you can use in your Deadline Cloud fleet. Once configured, Red Giant is supported by Deadline Cloud Usage-based Licensing and requires no further configuration to operate.

V-Ray Plugin

V-Ray is a 3D photorealistic ray-traced rendering plug-in. V-Ray for Cinema 4D is not currently fully supported in Service-managed fleets. We provide a conda recipe that you can use to create your own conda channel for use in your Deadline Cloud farm. For more information about creating custom conda channels, see Creating custom conda channels. After installation, V-Ray is supported by Deadline Cloud Usage-based Licensing and requires no further configuration to operate.

C4DToArnold

Autodesk Arnold software is an advanced Monte Carlo ray tracing renderer. For more information, see Arnold on the Autodesk website. C4DToArnold is fully supported in Service-managed fleets.

Insydium X-Particles

X-Particles is a fully-featured advanced particle and VFX system for Maxon's Cinema 4D. For more information, see X-Particles on the Insydium website. Insydium X-Particles is not currently fully supported in Service-managed fleets. We provide a conda recipe that you can use to create your own conda channel for use in your Deadline Cloud farm. For more information about creating custom conda channels, see Creating custom conda channels. When you create the conda package from your X-Particles package, it will include your purchased license. No additional configuration is necessary to operate on service-managed fleets.

Open source resources

The submitter and adaptor are open source and available on the GitHub website:

- Deadline Cloud for Cinema 4D
- Cinema 4D Conda recipes are available for C4D 2024, C4D 2025, the INSYDIUM X-PARTICLES plugin, the C4DtoA plugin, and the V-Ray Plugin.
- Host Configuration script is included to support Red Giant plugins.

SideFX Houdini

SideFX Houdini is a 3D procedural software for modeling, rigging, animation, VFX, look development, lighting, and rendering in film, TV, advertising, and video game pipelines. Houdini is fully supported by Deadline Cloud with comprehensive integration including submitters, conda packages, and an adaptor for increased rendering performance. This guide provides step-by-step instructions for using AWS Deadline Cloud with Houdini to render your projects faster by distributing rendering tasks across multiple machines.

Support overview

Houdini is supported by the following components:

- **Submitter**: Integrated render output node (ROP) for direct job submission from Houdini with automatic scene and asset detection.
- **Conda packages**: Deadline Cloud for automatic installation on service-managed fleets.
- **Adaptor**: A program on worker hosts that keeps the application loaded between tasks for faster rendering and reports render progress.
- **Cross-platform compatibility**: Submitter support for Windows, macOS, and Linux with worker support for Windows and Linux with automatic path mapping.

Houdini version compatibility

The following table shows current support levels for Houdini versions:

Major Version	Submitter Support	Conda Support	Render Engines	Usage-Based Licensing
19.5	Windows, macOS, Linux	Linux	Mantra, Karma CPU, Karma XPU	Usage-based licensing available
20.0	Windows, macOS, Linux	Linux	Mantra, Karma CPU, Karma XPU	Usage-based licensing available

Major Version	Submitter Support	Conda Support	Render Engines	Usage-Based Licensing
20.5	Windows, macOS, Linux	Linux	Mantra, Karma CPU, Karma XPU	Usage-based licensing available
21.0	Windows, macOS, Linux	Linux	Mantra, Karma CPU, Karma XPU	Usage-based licensing available
22.0	Windows, macOS, Linux	Linux	Mantra, Karma CPU, Karma XPU	Usage-based licensing available

Deadline Cloud Conda Channel

The following table lists all conda packages applicable to Houdini available to Service-managed fleets in the deadline-cloud conda channel:

OS	Package	Version	Notes
Linux	houdini	19.5	Includes Mantra and Karma renderers
Linux	houdini	20.0	Includes Mantra and Karma renderers
Linux	houdini	20.5	Includes Mantra and Karma renderers
Linux	houdini	21.0	Includes Mantra and Karma renderers
Linux	houdini	22.0	Includes Mantra and Karma renderers

OS	Package	Version	Notes
Linux	houdini-openjd		Includes the Houdini Adaptor

Getting started

To use Houdini with Deadline Cloud:

1. Create a service-managed fleet and associate it with a queue. Your queue must be set up with a queue environment that supports the deadline-cloud conda channel. For more information, see Creating a queue environment.
2. Install the Deadline Cloud monitor and Houdini submitter on your artist workstation using the Deadline Cloud Submitter and monitor installers. For more information, see Set up your workstation.
3. Submit your job directly from Houdini using the integrated submitter to the queue.
4. Monitor the job and download the output using the Deadline Cloud monitor.

Installation

To install the Deadline Cloud for Houdini submitter, you need:

- A Windows, macOS (arm64), or Linux workstation.
- A supported version of Houdini.

Installing the submitter

To install the submitter

1. Download the Deadline Cloud submitter installer.
2. Run the installer.
   - When prompted, select each version of Houdini you want to use the submitter with.
3. Launch Houdini.

The Deadline Cloud submitter is automatically available as a render output (ROP) node.

> **Note**
>
> The submitter installer is available for Windows, macOS, and Linux. For manual installation, see the manual installation instructions on the GitHub website.

Verifying the submitter is installed correctly

1. Open Houdini.
2. In the **Network Editor**, choose the `/out` network.
3. Open the context menu (right-click or press **Tab**) and search for `deadline`.
4. Choose **Deadline Cloud** to create a new node.

Using the Houdini submitter

The Deadline Cloud for Houdini submitter is a node that accepts a render output (ROP) node as input. You can configure and submit your job through this node. When you submit a job, it includes steps for each ROP in the graph.

Submitting a job from Houdini

To use the Deadline Cloud for Houdini submitter, you need:

- A profile to submit to Deadline Cloud with.
- An Deadline Cloud farm and queue to submit to.

To submit a job from Houdini to Deadline Cloud

1. In the Network Editor, choose the **/out** network.
2. Open the context menu (right-click or press **Tab**) and search for `deadline` to create an Deadline Cloud node.
3. Connect the output of a ROP to the input of the Deadline Cloud node.
   - When you connect a node to the Deadline Cloud node, the submitted job renders the input ROP and all ROPs in its graph.
4. Select the Deadline Cloud node.

5. Use the options in the node editor to configure your job. See Houdini-specific settings for information about what each option does.
6. (Optional) To export a job's associated files to your job history directory without submitting it, choose **Export Bundle**.
7. Choose **Submit** to send your job to Deadline Cloud.

Houdini-specific settings

The **Job-specific settings** tab of the Deadline Cloud node provides options specific to Houdini jobs.

- *Submit Dependencies as Separate Steps* - Split the ROP graph into separate rendering steps for easier monitoring and debugging. When enabled, each connected render node becomes its own step in the job.
- *Include Adaptor Wheels* - Enable custom builds of the adaptor (called *wheels*) that change rendering behavior. When enabled, you can specify a directory containing adaptor wheels. You can build adaptor wheels by running the build_wheels.sh script on the GitHub website.
- *Adaptor Wheels* - Specify the directory path containing custom adaptor wheels (only available when **Include Adaptor Wheels** is enabled).
- *Automatically unlock ROPs* - Automatically unlock dependency ROPs during submission. Locked ROPs use existing outputs and won't re-render, which can block dependencies from re-rendering.
- *Automatically parse scene (.hip) references* - Automatically discover and attach the job's input and output file names and directories based on the ROP graph during job submission.
- *Automatically save scene (.hip) file* - Automatically save the scene (`.hip`) file to `$HIP` when submitting a job.

For information about the other submitter options, see the Deadline Cloud guide for using a submitter.

Overriding the render strategy for Deadline Cloud jobs

For many types of nodes, frames can be rendered independently and in any order. For others like simulations, each frame depends on the result of the previous frame and must be rendered sequentially. The submitter chooses a rendering strategy for each node based on its type, but also allows you to override the default.

Parallel vs. sequential rendering

For parallel rendering, each frame has its own task, and the tasks are distributed across available workers. For sequential rendering, all frames for a node are rendered in a single task running on a single worker.

By default, if a node is a geometry node with **Initialize Simulation OPs** enabled, it renders sequentially. Otherwise the node renders in parallel.

Adding a render strategy parameter

You can override the render strategy by creating a `deadline_cloud_render_strategy` parameter on your render node (for example, Mantra or Karma) with a value of either `SEQUENTIAL` or `PARALLEL`.

To override render strategy by adding a parameter

1. Open the context menu for a node in the **/out** network (right-click).
2. Choose **Parameters and Channels**, **Edit Parameter Interface**.
3. Under **Create Parameters**, **By Type**, choose **Ordered Menu**.
4. Add an ordered menu to **Existing parameters** by selecting the right arrow next to the **Create Parameters** column.
5. Select the new parameter under **Existing Parameters**, then edit its configuration under **Parameter Description**:
   - In the **Parameter** tab:
     - For **Name**, enter `deadline_cloud_render_strategy`.
     - For **Label**, enter `Deadline Cloud Render Strategy`.
   - In the **Menu** tab, add menu items for:

Token	Label
SEQUENTIAL	Sequential
PARALLEL	Parallel

6. Choose **Accept**.

Now in the parameter editor for your node, you can use the **Deadline Cloud Render Strategy** menu to specify submitter behavior.

Husk rendering and USD workflows

The following sections describe current limitations of USD export workflows in the Houdini submitter and an alternative example job bundle for rendering exported USD scenes with Husk.

USD export workflow support

The Deadline Cloud submitter for Houdini does not currently have built-in support for USD export workflows.

You cannot use the submitter node to create a single job that will export a USD scene from Houdini and then call Husk standalone to render without consuming a Houdini Engine license.

Alternative: example Husk job bundle

Deadline Cloud provides an example Husk job bundle on the GitHub website that enables USD export rendering workflows outside of the Houdini submitter. You will need to export the USD scene yourself separately from Houdini before using the example job bundle.

The Husk example job bundle:

- Allows direct submission of USD scenes for rendering using Husk and a chosen Hydra render delegate without launching Houdini and consuming a Houdini engine license during the render.
- Automatically introspects USD files to find any file dependencies within to attach using job attachments.
- Provides a simple GUI for configuration of common Husk settings and submission.

Prerequisites

Before using the Husk example job bundle, you need:

- A scene exported to USD format.
  - See the USD output documentation on the SideFX website for information on writing out USD files in Houdini.
- The Deadline Cloud CLI installed and configured.

  - The CLI can be installed from either the submitter installer or directly following the getting started guide on the GitHub website.
- A git clone of the deadline-cloud-samples repository on the GitHub website.
- The Hydra render delegate available on the worker nodes.
  - Karma is included with Houdini. If you want to use other Hydra render delegates, you must provide them on the worker. See the conda recipes for V-Ray and Redshift on the GitHub website as one option to make them available on the worker nodes.

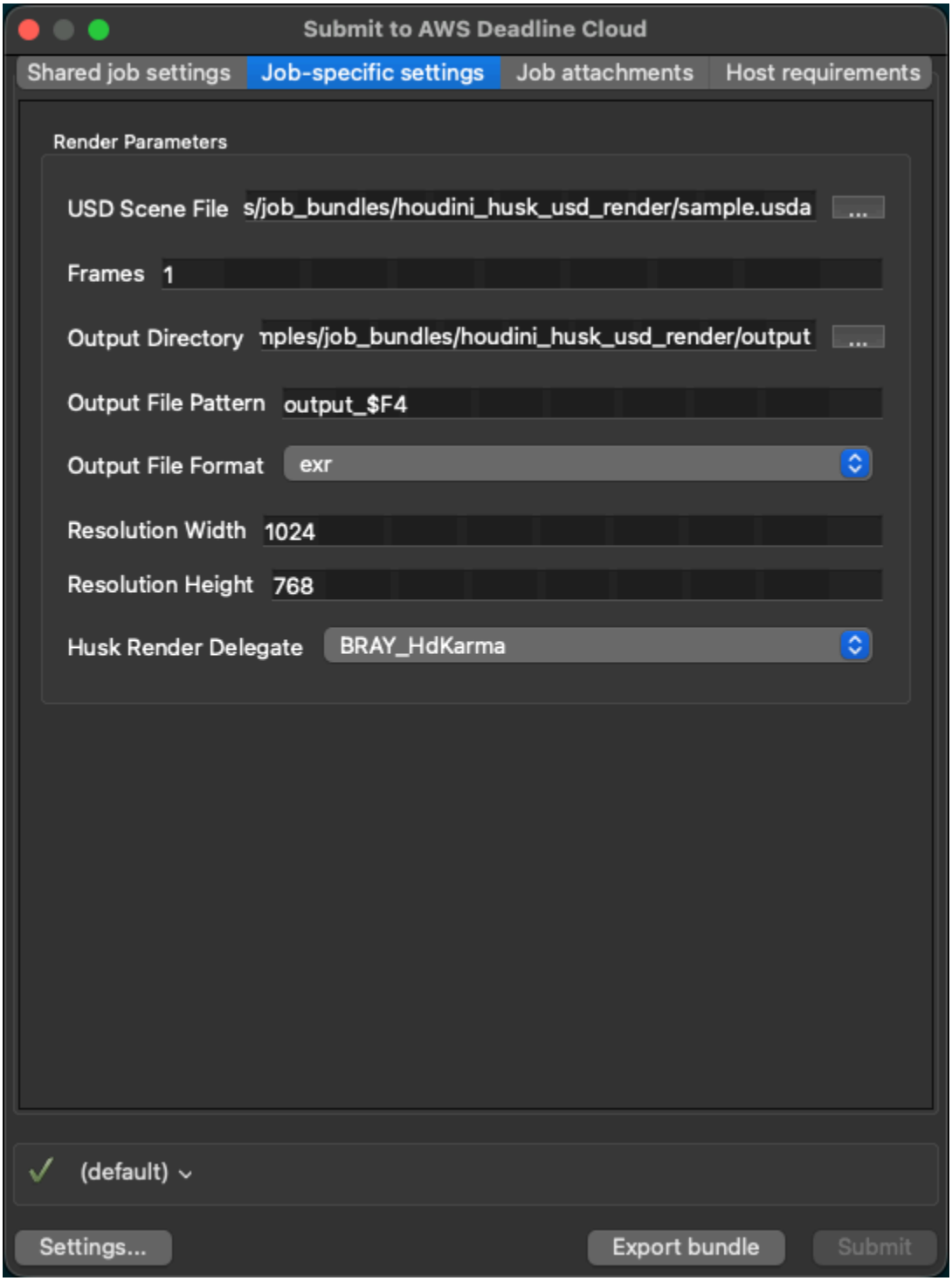
Using the Husk example job bundle

To use the Husk example job bundle

1. Submit the bundle using the Deadline Cloud CLI:

```
deadline bundle gui-submit ./deadline-cloud-samples/job_bundles/
houdini_husk_usd_render
```

2. Configure your USD file, output settings, frame range, and any other applicable settings to submit.



Additional resources

The following resources are available on the GitHub website and the SideFX website:

- deadline-cloud-samples repository
- SideFX Husk documentation

Troubleshooting

The following sections describe common errors and questions you might encounter when using the Deadline Cloud submitter for Houdini, and how to resolve them.

Why do I get "incomplete asset definitions" errors while rendering?

Jobs from this submitter that run in your farm may produce errors in the logs that look like:

```
The following node types are using incomplete asset definitions:
  Driver/deadline_cloud
```

These errors are safe to ignore. The Deadline Cloud submitter exists as a node in your Houdini scene. When a worker in your farm loads the scene, the scene still contains the Deadline Cloud node, but the worker may not have the submitter installed. Because the worker does not have the files needed to run the Deadline Cloud node, it logs "incomplete asset definition" errors. The Deadline Cloud node itself is not rendered as part of the job, so these errors can be ignored.

Does the Deadline Cloud submitter support USD export render workflows using Husk?

The Houdini submitter does not directly support export workflows using Husk at this time. Jobs created through the submitter always run the adaptor which uses `hython` and therefore a Houdini engine license for the duration of the render. If you want to render an exported USD scene using just Husk and a Hydra render delegate, you can use an example job bundle on the GitHub website. This approach is useful to render USD scenes with only a render license (for example, Karma) without needing a Houdini engine license for the entire render. For more information on rendering USD scenes with Husk on Deadline Cloud, see Husk rendering and USD workflows.

Advanced configurations

Using unsupported versions

Deadline Cloud only supports and tests the workstation and worker software versions in the table above. When using the submitter, the worker will attempt to install the same version as used on the workstation. This installation might fail if the workstation version of Houdini does not appear in the version table above.

If you require an unsupported version of Houdini, you have the following options:

- When submitting the job from Houdini, you can override the CondaPackages queue parameter to specify a supported version to use on the worker (for example, `houdini=21.0, houdini-openjd=*`). This override might or might not work, depending on the features used by your scene and how Houdini works with scenes from your workstation version.
- you can build a custom conda recipe and channel for your desired version to be installed on the worker. Use the conda recipe for a supported version linked below as a starting point, and package your desired version in a custom conda channel. For more information about creating custom conda channels, see Creating custom conda channels.

Houdini render engines

Houdini supports multiple render engines that are compatible with Deadline Cloud:

Render Engine	Description	GPU Support
Karma CPU	Modern USD-based renderer (CPU variant)	CPU-based
Karma XPU	Modern USD-based renderer (GPU variant)	GPU accelerated
Mantra	Traditional Houdini renderer	CPU-based
Arnold	Third-party Monte Carlo ray tracer	GPU/CPU hybrid
V-Ray	Third-party photorealistic renderer	GPU/CPU hybrid
Redshift	GPU-accelerated renderer	GPU optimized

These render engines are automatically detected and configured by the Houdini integrated submitter and usage is automatically licensed. The submitter maintains dependency trees between connected render output nodes (ROPs).

Open source resources

The submitter, adaptor, sample scenes and workflows, and conda recipes for supported versions are open source and available on the GitHub website:

- Houdini submitter source code
- Sample scenes and workflows
- Conda recipes for supported versions

Use custom plugins with Deadline Cloud

Jobs can load custom plugins and add-ons on both fleet types. The developer guide provides setup instructions and helps administrators choose a delivery method. For more information, see Use custom plugins with Deadline Cloud in the *Deadline Cloud Developer Guide*.

File storage for Deadline Cloud

Workers must have access to the storage locations that contain the input files necessary to process a job, and to the locations that store the output. AWS Deadline Cloud provides the following options for storage:

- With *persistent storage*, service-managed fleet workers use dedicated Amazon Elastic Block Store (Amazon EBS) volumes that preserve data across worker lifecycle events. Application caches, conda package installations, and workspaces persist when workers are recycled, eliminating cold-start delays. For more information, see Persistent storage for service-managed fleets.
- With *job attachments*, Deadline Cloud transfers the input and output files for your jobs back and forth between a workstation and Deadline Cloud workers. To enable the file transfers, Deadline Cloud uses an Amazon Simple Storage Service (Amazon S3) bucket in your AWS account.

  When you use job attachments with a Linux based service-managed fleet, you can enable a virtual file system (VFS) to mount job attachments files and access them as needed instead of syncing them to the worker at the start of the job.
- With *shared storage*, you use file sharing with your operating system to provide access to files.

  When you use cross-platform shared storage, you can create a *storage profile* so that workers can map the path to files between two different operating systems.

  You can also integrate third-party cloud storage solutions, such as LucidLink, with service-managed fleets using host configuration scripts. For more information, see Set up LucidLink with service managed fleet scripts for Deadline Cloud on the AWS for M&E Blog.

Topics

Storage profiles in Deadline Cloud

When you use workstations and fleet worker hosts from multiple operating systems or with different file system mounts, you can create storage profiles in your farm to indicate where the same file systems are mounted on different systems. When Deadline Cloud runs a job on a different

storage profile than on the workstation it was submitted from, it will transform file system paths that are in directories configured in the storage profiles.

Using storage profiles in your Deadline Cloud farm enables the following behaviors:

- When submitting a job to a queue, the files that the job references will be categorized by the workstation storage profile:
  - Files that are under a shared file system location will be left alone.
  - Files that are under a local file system location will be attached to the job by uploading them to the job attachments S3 bucket. Files that were previously uploaded are not uploaded again.
  - Files that are not under any file system location will be attached to the job as well. The job submitter will warn about these file paths unless they are under a known path in the local Deadline Cloud settings.
- When jobs are running on a fleet worker host with a different operating system or storage profile than the submitting workstation, file paths used by the job will be mapped from the submitting storage profile to the fleet storage profile.
- When downloading job output, jobs that were submitted for a different operating system or storage profile will have their paths mapped from the submitting storage profile to the local workstation storage profile.

For more information, see Storage profiles and path mapping in the *AWS Deadline Cloud Developer Guide*.

To create a storage profile

1. Open the Deadline Cloud console.
2. From **Get started**, choose **Go to Deadline Cloud dashboard**.
3. Choose a farm, and then choose the **Storage profiles** tab.
4. Choose **Create storage profile**.
5. From the dropdown, select an **Operating system**.
6. Enter a **Storage profile name**. The name is how you choose a storage profile for a workstation. For example, names like *Windows-Workstation* or *Windows-OnPremFleet* can make it easy to identify it later.
7. Create a file system location of **Shared** type for each shared file system that is mounted on both workstations and fleet worker hosts.

    1. Enter a **name** that identifies the mount, such as *Projects* for a shared file system that contains project data, or *Tools* for a shared file system with tools to use.
    2. Enter the mount location for the chosen shared file system on the operating system of the storage profile.
8. Create a file system location of **Local** type for each shared file system that is only for workstations. For example when your fleets are on AWS and you want job attachments to handle the data transfer. You can also create this kind of file system location for directories that are local to each workstation, to specify equivalent paths on different operating systems even if they are not mounted storage.

    1. Enter a **name** that identifies the mount, such as *Projects* for a shared file system that contains project data, or *Tools* for a shared file system with tools to use.
    2. Enter the chosen file system location on the operating system of the storage profile.
9. (Optional) To add another file system location, choose **Add new required file system location** and enter the required data.
10. After you have added all of the required file system locations, choose **Create**.

To set up your storage profile for use

1. Navigate to the queue that you want to use this storage profile with, and select the **Allowed storage profiles** tab.
2. Choose **Configure storage profiles**.
3. From the dropdown list to associate storage profiles, select the storage profile you created.
4. In the Required file system location list, select the **file system location names** that you want to ensure are available on any storage profile for the associated fleets.
5. (Optional) If you created the storage profile for a fleet, navigate to the fleet and select the **Configurations** tab.

    a. From the Storage profiles section, choose **Configure storage profile**.

    b. Select the storage profile, and then choose **Save changes**.

To configure a storage profile on a workstation

On each workstation that will submit jobs to a queue, select the workstation's default storage profile.

1. Open a Deadline Cloud submitter, such as the submitter in your digital content creation (DCC) application or the submitter that the CLI command `deadline bundle gui-submit` opens.
2. On the **Shared job settings** tab, find the **Job submission settings** section and choose the default farm and queue.
3. Choose the workstation's storage profile from the **Default storage profile** list. The list contains the storage profiles that the queue can use, and only appears after you choose a queue that has storage profiles.

Storage profiles for shared file systems

You can configure your Deadline Cloud fleets to mount shared file systems by using VPC resource endpoints on service-managed fleets, or by configuring the hosts of a customer-managed fleet on AWS or on-premises. When workstations have the same shared file systems mounted as your fleets, you can create file system locations of the shared type in your storage profiles to configure where each shared file system appears as a local path.

For example, suppose you have one shared file system for projects and another one for tools. Your workstations and fleets include the three operating systems Windows, macOS, and Linux. You can create one storage profile for each operating system with the following values:

- **Storage profile name**: Linux-Host, **operating system family**: Linux.
  - **File system location name**: Projects, **path**: /mnt/projects, **type**: Shared.
  - **File system location name**: Tools, **path**: /mnt/projects, **type**: Shared.
- **Storage profile name**: Windows-Host, **operating system family**: Windows.
  - **File system location name**: Projects, **path**: X:\projects, **type**: Shared.
  - **File system location name**: Tools, **path**: Z:, **type**: Shared.
- **Storage profile name**: MacOS-Host, **operating system family**: MacOS.
  - **File system location name**: Projects, **path**: /Volumes/Projects, **type**: Shared.
  - **File system location name**: Tools, **path**: /Volumes/Tools, **type**: Shared.

When you submit a job from Windows that uses a path X:\Projects\ProjectA\Textures\texture.jpg, Deadline Cloud will add a field containing the Windows-Host storage profile id to the job.

If the job runs on a Linux fleet worker host, Deadline Cloud will create two path mapping rules for the job based on corresponding file system location names: X:\Projects -> /mnt/projects, Z: -> /mnt/tools. The job will apply these rules to resolve the original paths to where the Linux host sees them.

If job attachments are also configured for your queue, any paths that are not under a file system location of type shared will be attached to the job and uploaded to the job attachments S3 bucket. This behavior lets you attach data files to the job instead of requiring that they always be copied to a shared file system. For example, providing auxiliary files defined by the job bundle you submit.

Storage profiles for job attachments

You can configure your Deadline Cloud queue to use job attachments for transferring asset data referenced by your jobs to and from AWS. When workstations mount the same shared file systems, but your fleets do not, you can create file system locations of the local type in your storage profiles. This configuration lets you configure where you will upload and download files from, and how to map paths between operating systems.

For example, suppose you have one shared file system for projects and another one for tools. Your workstations and fleets include the three operating systems Windows, macOS, and Linux. Everything is the same as in the **Storage profiles for shared file systems** topic except the file systems are not shared with the farm. They are for the local area network containing your workstations. You can create one storage profile for each operating system with the following values:

- **Storage profile name**: Linux-Host, **operating system family**: Linux.
  - **File system location name**: Projects, **path**: /mnt/projects, **type**: Local.
  - **File system location name**: Tools, **path**: /mnt/projects, **type**: Local.
- **Storage profile name**: Windows-Host, **operating system family**: Windows.
  - **File system location name**: Projects, **path**: X:\projects, **type**: Local.
  - **File system location name**: Tools, **path**: Z:, **type**: Local.
- **Storage profile name**: MacOS-Host, **operating system family**: MacOS.
  - **File system location name**: Projects, **path**: /Volumes/Projects, **type**: Local.
  - **File system location name**: Tools, **path**: /Volumes/Tools, **type**: Local.

When you submit a job from Windows that uses a path X:\Projects\ProjectA\Textures\texture.jpg, Deadline Cloud will add a field containing the Windows-Host storage profile id to the job and upload the file to the job attachments S3 bucket if it wasn't uploaded already.

If the job runs on a Linux fleet worker host, Deadline Cloud will make the texture file available in a local temporary directory, then create a path mapping rule from one of the directories containing the texture to the temporary directory. For example X:\Projects\ProjectA -> /sessions/session-123/projects, so that X:\Projects\ProjectA\Textures\texture.jpg maps to /sessions/session-123/projects/Textures/texture.jpg. When a task of the job is complete, it collects the output from directories specified by the job. Suppose /sessions/session-123/projects/Output/frame0032.png is an output file. This output is recorded on the job as X:\Projects\ProjectA\Output\frame0032.jpg, matching on the storage profile for the workstation submitting the job.

When you download the job output on a macOS workstation, Deadline Cloud will create path mapping rules from the Windows workstation: X:\Projects -> /Volumes/Projects, Z: -> /Volumes/Tools. It applies the rule to all output paths, downloading the example output file to /Volumes/Projects/ProjectA/Output/frame0032.jpg.

If an output file path of a job is not contained under any of the storage profile file system locations, Deadline Cloud will not be able to determine its path for download when the storage profile is different from the submitting workstation. Depending on the command you use for download, that file will either be skipped or you will have to manually select a download directory.

Job attachments in Deadline Cloud

With *job attachments* you can transfer files back and forth between your workstation and AWS Deadline Cloud. With job attachments, you don't need to manually set up an Amazon S3 bucket for your files. Instead, when you create a queue with the Deadline Cloud console, you choose the bucket for your job attachments.

The first time that you submit a job to Deadline Cloud, all of the files for the job are transferred to Deadline Cloud. For subsequent submissions, only the files that have changed are transferred, saving both time and bandwidth.

After processing is complete, you can download the result from the job detail page, or by using the Deadline Cloud CLI `deadline job download-output` command.

You can use the same S3 bucket for multiple queues. Set a different root prefix for each queue to organize the attachments in the bucket.

When you create a queue with the console, you can either choose an existing AWS Identity and Access Management (IAM) role or you can have the console create a new role. If the console creates the role, it sets permissions to access the bucket that's specified for the queue. If you choose an existing role, you must grant the role permissions to access the S3 bucket.

Encryption for job attachment S3 buckets

Job attachment files are encrypted in your S3 bucket by default. This encryption helps secure your information from unauthorized access. You don't need to do anything to have your files encrypted with keys provided by Deadline Cloud. For more information, see Amazon S3 now automatically encrypts all new objects in the *Amazon S3 User Guide*.

You can use your own customer managed AWS Key Management Service key to encrypt the S3 bucket that contains your job attachments. To do so, you must modify the IAM role for the queue associated with the bucket to allow access to the AWS KMS key.

To open the IAM policy editor for the queue role

1. Sign in to the AWS Management Console and open the Deadline Cloud console. From the main page, in the **Get started** section, choose **View farms**.
2. From the list of farms, choose the farm that contains the queue to modify.
3. From the list of queues, choose the queue to modify.
4. In the **Queue details** section, choose the **Service role** to open the IAM console for the service role.

Next, complete the following procedure.

To update the role policy with permission for AWS KMS

1. From the list of **Permissions policies**, choose the policy for the role.
2. In the **Permissions defined in this policy** section, choose **Edit**.
3. Choose **Add new statement**.
4. Copy and paste the following policy into the editor. Change the *Region*, *accountID*, and *keyID* to your own values.

```
{
   "Effect": "Allow",
   "Action": [
```

```
            "kms:Decrypt",
            "kms:DescribeKey",
            "kms:GenerateDataKey"
        ],
        "Resource": [
            "arn:aws:kms:us-east-1:111122223333:key/keyID"
        ]
}
```

5. Choose **Next**.
6. Review the changes to the policy, and then when you're satisfied, choose **Save changes**.

Replace job attachments bucket

You can replace your current job attachments bucket with a different job attachments bucket. You will find a button under the **Job Attachments** tab in the queue details. You can use it to either change the job attachments bucket or replace the root folder inside the same bucket to upload the job attachments.

To access job attachments settings

1. Go to **Queue details**, then locate the **Job Attachments** tab.
2. From the job attachments tab, there are 2 options:
    a. Change the job attachments bucket by doing the following:
        i. Select a new S3 bucket.
        ii. Update the queue's service role policy to grant access to the new bucket.

        OR
    b. Change the root folder within an existing bucket by doing the following:
        i. Modify the root folder name.
        ii. Update the resource ARN in the queue service role.

To update the service role

1. Navigate to your farm > queue > queue service role.

2. Choose **Edit in JSON**.
3. Locate the resource ARN (default root folder is **DeadlineCloud**):

```
  "arn:aws:s3:::<your-job-attachments-bucket-name>/DeadlineCloud/*"
]
```

4. Update the ARN with new bucket or folder:

```
 "arn:aws:s3:::<your-job-attachments-NEW-bucket-name>/NEW-ROOT-FOLDER-NAME/*"
]
```

5. Verify permissions after making these changes to ensure proper access.

Managing job attachments in S3 buckets

Deadline Cloud stores the job attachment files required for your job in an S3 bucket. These files accumulate over time, leading to increased Amazon S3 costs. To reduce costs, you can apply an S3 Lifecycle configuration to your S3 bucket. This configuration can automatically delete files in the bucket. Because the S3 bucket is in your account, you can choose to modify or remove the S3 Lifecycle configuration at any time. For more information, see Examples of S3 Lifecycle configuration in the *Amazon S3 User Guide*.

For a more granular S3 bucket management solution, you can set up your AWS account to expire objects in an S3 bucket based on the last time that they were accessed. For more information, see Expiring Amazon S3 objects based on last accessed date to decrease costs on the AWS Architecture Blog.

Deadline Cloud virtual file system

Virtual file system support for job attachments in AWS Deadline Cloud enables client software on workers to communicate directly with Amazon Simple Storage Service. Workers can load files only when needed instead of downloading all files before processing. Files are stored locally. This approach avoids downloading assets used more than once multiple times. All files are removed after the job completes.

- The virtual file system provides a significant performance boost for specific job profiles. In general, smaller subsets of total files with larger fleets of workers show the most benefit. Small numbers of files with fewer workers have roughly equivalent processing times.

- Virtual file system support is only available for Linux workers in service-managed fleets.
- The Deadline Cloud virtual file system supports the following operations, but is not POSIX compliant:
  - File `create`, `delete`, open, `close`, `read`, `write`, append, `truncate`, `rename`, `move`, `copy`, `stat`, `fsync`, and `falloc`
  - Directory `create`, `delete`, `rename`, `move`, `copy`, and `stat`
- The virtual file system is designed to reduce data transfer and improve performance when your tasks access only part of a large data set, and is not optimized for all workloads. You should test your workload before running production jobs.

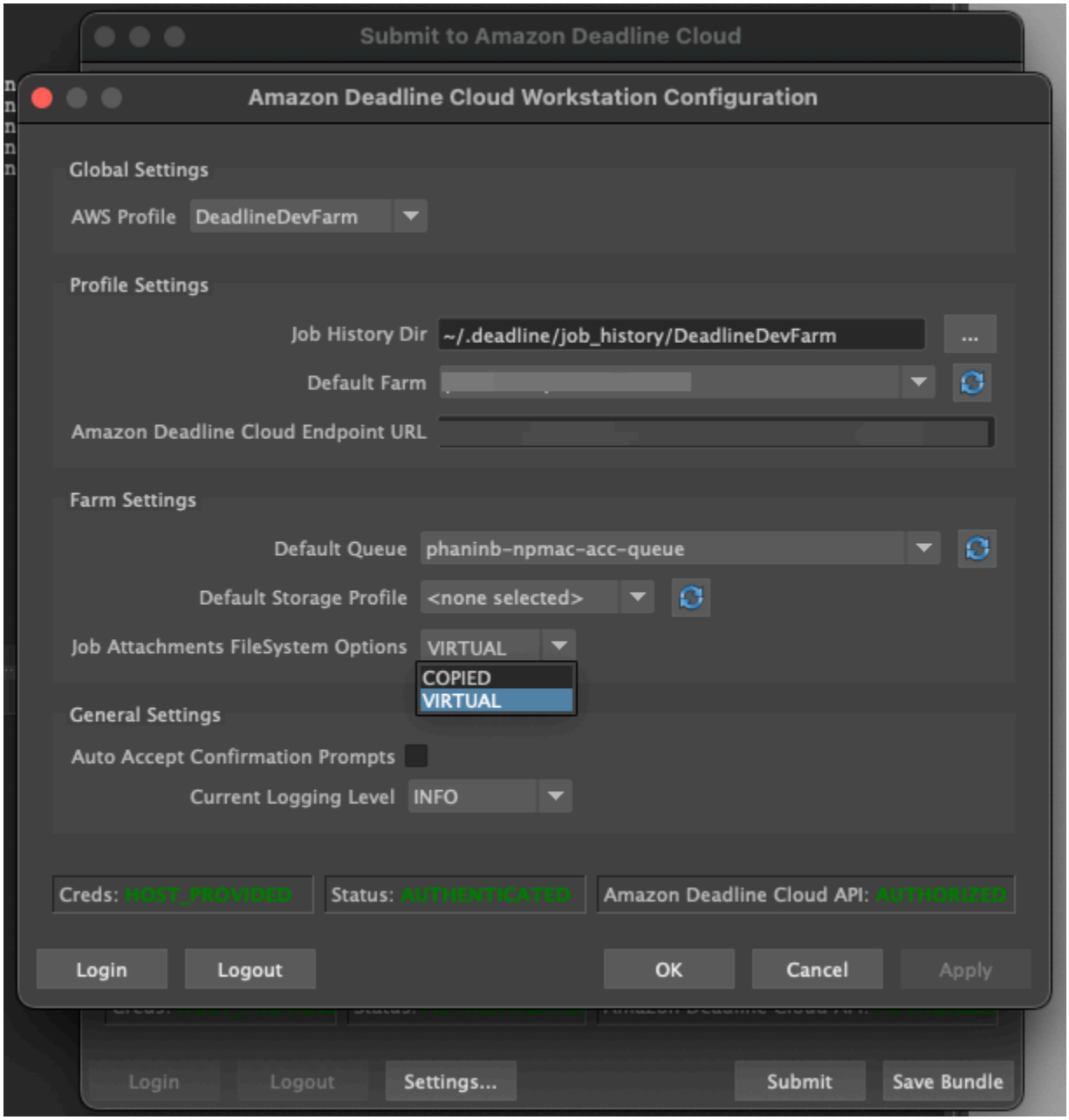
Enable VFS support

Virtual file system support (VFS) is enabled for each job. A job falls back to the default job attachments framework in these cases:

- A worker instance profile does not support a virtual file system.
- Problems prevent launching the virtual file system process.
- The virtual file system can't be mounted.

To enable virtual file system support using the submitter

1. When submitting a job, choose the **Settings** button to open the **AWS Deadline Cloud workstation configuration panel**.
2. From the **Job attachments filesystem options** dropdown, choose **VIRTUAL**.



3. To save your changes, choose **OK**.

To enable virtual file system support using the AWS CLI

- Use the following command when you submit a saved job:

```
deadline bundle submit-job --job-attachments-file-system VIRTUAL
```

To verify that the virtual file system launched successfully for a particular job, review your logs in Amazon CloudWatch Logs. Look for the following messages:

```
Using mount_point mount_point
Launching vfs with command command
Launched vfs as pid PID number
```

If the log contains the following message, virtual file system support is disabled:

```
Virtual File System not found, falling back to COPIED for JobAttachmentsFileSystem.
```

Troubleshooting virtual file system support

You can view logs for your virtual file system using the Deadline Cloud monitor. For instructions, see View session and worker logs in Deadline Cloud.

Virtual file system logs are also sent to the CloudWatch Logs group that's associated with the queue shared with the worker agent output.

Automatic downloads

The Deadline CLI provides a command to download the output of all tasks in a queue that completed since the last time the same command ran. You can configure this as a cron job or scheduled task to run repeatedly. This configuration sets up automatic downloading of output on a continuous basis.

Before setting up automatic downloads, follow the steps in Storage profiles for job attachments to configure all paths of asset data for upload and download. If a job uses an output path that is not in its storage profile, then the automatic download skips downloading that output and prints warning messages to summarize the files it did not download. Similarly, if a job is submitted without a storage profile, the automatic download skips that job and prints a warning message. By default, Deadline Cloud submitters display warning messages for paths that are outside of storage profiles to help ensure correct configuration.

Configuring AWS credentials

Automatic downloads use the Deadline CLI to continuously download job outputs. To authenticate these downloads, you need long-term IAM credentials. Deadline Cloud monitor credentials expire, so you can't use them for this purpose.

Follow the steps below to set up long-term credentials.

> **Important**
>
> Heed the following warnings:
>
> - **Do NOT** use your account's root credentials to access AWS resources. These credentials provide unrestricted account access and are difficult to revoke.
> - **Do NOT** put literal access keys or credential information in your application files. If you do, you create a risk of accidentally exposing your credentials if, for example, you upload the project to a public repository.
> - **Do NOT** include files that contain credentials in your project area.
> - Secure your access keys. Do not provide your access keys to unauthorized parties, even to help find your account identifiers. By doing this, you might give someone permanent access to your account.
> - Be aware that any credentials stored in the shared AWS credentials file are stored in plain text.
>
> For more details, see Best practices for managing AWS access keys in the *AWS General Reference*.

Create an IAM user

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, select **Users** and then select **Create user**.
3. Name the user **`deadline-output-downloader`**. Clear the checkbox for **Provide user access to the AWS Management Console**, then choose **Next**.
4. Choose **Attach policies directly**.
5. Choose **Create policy** to create a custom policy with minimum required permissions.
6. In the JSON editor, specify the following permissions:

   JSON

```
{
                    "Version":"2012-10-17",
```

```
                "Statement": [
                    {
                        "Sid": "DeadlineCloudOutputDownload",
                        "Effect": "Allow",
                        "Action": [
                            "deadline:AssumeQueueRoleForUser",
                            "deadline:ListQueueEnvironments",
                            "deadline:ListSessions",
                            "deadline:ListSessionActions",
                            "deadline:SearchJobs",
                            "deadline:GetJob",
                            "deadline:GetQueue",
                            "deadline:GetStorageProfileForQueue"
                        ],
                        "Resource": "*"
                    }
                ]
            }
```

7. Name the policy **`DeadlineCloudOutputDownloadPolicy`** and choose **Create policy**.
8. Return to the user creation page, refresh the policy list, and select the **DeadlineCloudOutputDownloadPolicy** you just created, then choose **Next**.
9. Review the user details and then choose **Create user**.

Create an access key

1. From the user details page, select the **Security credentials** tab. In the **Access keys** section, choose **Create access key**.
2. Indicate that you want to use the key for Other, then choose **Next**, then choose **Create access key**.
3. On the **Retrieve access keys** page, choose **Show** to reveal the value of your user's secret access key. You can copy the credentials or download a .csv file.

Store the user access keys

- Store the user access keys in the AWS credentials file on your system:
  - On Linux, the file is located at `~/.aws/credentials`
  - On Windows, the file is located at `%USERPROFILE%\.aws\credentials`

Replace the following keys:

```
[deadline-downloader]
aws_access_key_id=ACCESS_KEY_ID
aws_secret_access_key=SECRET_ACCESS_KEY
region=YOUR_AWS_REGION
```

Important

When you no longer need this IAM user, we recommend that you remove it to align with the AWS security best practice. We recommend that you require your human users to use temporary credentials through AWS IAM Identity Center when accessing AWS.

Prerequisites

Complete the following steps before creating a cron job or scheduled task for automatic download.

1. If you haven't already, install Python from the Python website.
2. Install the Deadline CLI by running:

```
python -m pip install deadline
```

3. Confirm the version of the Deadline CLI is 0.52.1 or newer with the following command.

```
$ deadline --version
deadline, version 0.52.1
```

To see download status in the Deadline Cloud monitor, use version 0.60.4 or newer. That version started recording the download status that the monitor reads. For more information, see View output download status in Deadline Cloud.

Test the output download command

To verify the command works in your environment

1. Get the path to Deadline

   Linux and macOS

   ```
   $ which deadline
   ```

   Windows

   ```
   C:\> where deadline
   ```

   PowerShell

   ```
   PS C:\> Get-Command deadline
   ```

2. Run the sync-output command to bootstrap.

   ```
   /path/to/deadline queue sync-output \
   --profile deadline-downloader \
   --farm-id YOUR_FARM_ID \
   --queue-id YOUR_QUEUE_ID \
   --storage-profile-id YOUR_PROFILE_ID \
   --checkpoint-dir /path/to/checkpoint/directory \
   ```

3. You only need to do this step if your downloading machine is the same as submitting machine. Replace `--storage-profile-id YOUR_PROFILE_ID \` above with `--ignore-storage-profiles`.

4. Submit a test job.

   a. Download the .zip file from GitHub.

      i. Open the deadline-cloud-samples repository on the GitHub website.

      ii. Choose **Code** and then, from the dropdown menu, select **Download ZIP**.

      iii. Unzip the downloaded archive to a local directory.

   b. Run

```
 cd /path/to/unzipped/deadline-cloud-samples-mainline/job_bundles/
job_attachments_devguide_output
```

c. Run

```
deadline bundle submit .
```

- If you don't have the default deadline config setup, you might need to supply the following in the command line.

```
--farm-id YOUR-FARM-ID --queue-id YOUR-QUEUE-ID
```

d. Wait for the job to complete before going to the next step.

5. Run the sync-output command again.

```
/path/to/deadline queue sync-output \
 --profile deadline-downloader \
 --farm-id YOUR_FARM_ID \
 --queue-id YOUR_QUEUE_ID \
 --storage-profile-id YOUR_PROFILE_ID \
 --checkpoint-dir /path/to/checkpoint/directory
```

6. Verify the following:

- Your test job's outputs appear in the destination directory.
- A checkpoint file is created in your specified checkpoint directory.

Set up scheduled downloads

Select the tab for your operating system to learn how to configure automatic downloads for every 5 minutes.

Linux

1. **Verify Deadline CLI Installation**

   Get the exact path to your deadline executable:

```
$ which deadline
```

Note this path (e.g., `/opt/homebrew/bin/deadline`) for use in the plist file.

2. **Create Checkpoint Directory**

   Create the directory where checkpoint files will be stored. Ensure proper permissions for your user to run the command.

   ```
   $ mkdir -p /path/to/checkpoint/directory
   ```

3. **Create Log Directory**

   Create a directory for cron job logs:

   ```
   $ mkdir -p /path/to/logs
   ```

   Consider setting up log rotate on the log file using https://www.redhat.com/en/blog/setting-logrotate

4. **Check Current Crontab**

   View your current crontab to see existing jobs:

   ```
   $ crontab -l
   ```

5. **Edit Crontab**

   Open your crontab file for editing:

   ```
   $ crontab -e
   ```

   The first time you run the command, you might be prompted to choose an editor (nano, vim, and so on).

6. **Add Cron Job Entry**

   Add the following line to run the job every 5 minutes (replace paths with actual values from steps 1 and 2):

   ```
   */5 * * * * /path/to/deadline queue sync-output --profile deadline-downloader
    --farm-id YOUR_FARM_ID --queue-id YOUR_QUEUE_ID --storage-profile-id
   ```

```
 YOUR_PROFILE_ID --checkpoint-dir /path/to/checkpoint/directory >> /path/to/
logs/deadline_sync.log 2>&1
```

7. **Verify Cron Job Installation**

   After saving and exiting the editor, verify the cron job was added:

   ```
   $ crontab -l
   ```

   You should see your new job listed.

8. **Check Cron Service Status**

   Ensure the cron service is running:

   ```
   # For systemd systems (most modern Linux distributions)
   $ sudo systemctl status cron
   # or
   $ sudo systemctl status crond

   # For older systems
   $ sudo service cron status
   ```

   If not running, start it:

   ```
   $ sudo systemctl start cron
   $ sudo systemctl enable cron  # Enable auto-start on boot
   ```

macOS

1. **Verify Deadline CLI Installation**

   Get the exact path to your deadline executable:

   ```
   $ which deadline
   ```

   Note this path (e.g., `/opt/homebrew/bin/deadline`) for use in the plist file.

2. **Create Checkpoint Directory and Log Directory**

   Create the directory where checkpoint files will be stored:

```
$ mkdir -p /path/to/checkpoint/directory
$ mkdir -p /path/to/logs
```

Consider setting up log rotate on the log file using https://formulae.brew.sh/formula/logrotate

3. **Create a Plist file**

   Create a configuration file at `~/Library/LaunchAgents/com.user.deadlinesync.plist` with the following content (replace `/path/to/deadline` with the actual path from step 1):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.user.deadlinesync</string>
    <key>ProgramArguments</key>
    <array>
        <string>/path/to/deadline</string>
        <string>queue</string>
        <string>sync-output</string>
        <string>--profile</string>
        <string>deadline-downloader</string>
        <string>--farm-id</string>
        <string>YOUR_FARM_ID</string>
        <string>--queue-id</string>
        <string>YOUR_QUEUE_ID</string>
        <string>--storage-profile-id</string>
        <string>YOUR STORAGE PROFILE ID</string>
        <string>--checkpoint-dir</string>
        <string>/path/to/checkpoint/dir</string>
    </array>
    <key>RunAtLoad</key>
    <true/>
    <key>UserName</key>
    <string>YOUR_USER_NAME</string>
    <key>StandardOutPath</key>
    <string>/path/to/logs/deadline_sync.log</string>
    <key>StartInterval</key>
```

```
    <integer>300</integer>
</dict>
</plist>
```

Replace `--storage-profile-id` *`YOUR_PROFILE_ID`* above with `--ignore-storage-profiles` if your downloading machine is the same as submitting machine.

4. **Validate Plist File**

   Validate the XML syntax of your plist file:

   ```
   $ plutil -lint ~/Library/LaunchAgents/com.user.deadlinesync.plist
   ```

   The command returns "OK" if the file is valid.

5. **Check for Existing Launch Agents or Launch Daemons**

   Check if a launch agent is already loaded:

   ```
   $ launchctl list | grep deadlinesync
   OR
   $ sudo launchctl list | grep deadlinesync
   ```

   If one exists, unload it first:

   ```
   $ launchctl bootout gui/$(id -u)/com.user.deadlinesync
   OR
   $ sudo launchctl bootout system/com.user.deadlinesync
   ```

6. **Create and bootstrap**

   To run this task while the user is logged in, run it as **LaunchAgent**. To run this task without a user being logged in every time the machine is running, run it as a **LaunchDaemon**.

   a. To run as **LaunchAgent:**

      i. Use the configuration created under `~/Library/LaunchAgents/com.user.deadlinesync.plist`

      ii. Then load the configuration using the bootstrap command:

```
$ launchctl bootstrap gui/$(id -u) ~/Library/LaunchAgents/
com.user.deadlinesync.plist
```

b. To run as **LaunchDaemon**:

i. Move the Pilst file and change permissions by running the following:

```
$ sudo mv ~/Library/LaunchAgents/com.user.deadlinesync.plist /Library/
LaunchDaemons/
$ sudo chown root:wheel /Library/LaunchDaemons/
com.user.deadlinesync.plist
$ sudo chmod 644 /Library/LaunchDaemons/com.user.deadlinesync.plist
```

ii. Load the launch agent using the modern bootstrap command:

```
$ sudo launchctl bootstrap system /Library/LaunchDaemons/
com.user.deadlinesync.plist
```

7. **Verify Status**

If you bootstrapped a LaunchAgent run the following to confirm it's loaded:

```
$ launchctl list | grep deadlinesync
```

If you bootstrapped a LaunchDaemon, confirm it is loaded by running:

```
$ sudo launchctl list | grep deadlinesync
```

The output should look like

```
SOME_PID_NUMBER 0 com.user.deadlinesync
```

For detailed status information:

```
$ launchctl print gui/$(id -u)/com.user.deadlinesync
```

This shows the current state, program arguments, environment variables, run interval, and execution history.

Windows

> **Note**
>
> The scheduled task created using these instructions only work when the user is logged in.
> To set it up at system startup without requiring user login, see the official Windows documentation on the Microsoft website.

For all steps below use Command Prompt - run as Administrator:

1. **Verify Deadline CLI Installation**

   Find the deadline executable:

   ```
   C:\> where deadline
   ```

   Note the full path (e.g., `C:\Program Files\Amazon\DeadlineCloud\deadline.exe`) for use in the task.

2. **Create Checkpoint Directory**

   Create the directory where checkpoint files will be stored:

   ```
   C:\> mkdir "path\to\checkpoint\directory"
   ```

3. **Create Log Directory**

   Create a directory for task logs:

   ```
   C:\> mkdir "path\to\logs"
   ```

4. **Create Batch File Wrapper**

   Create the batch file with the following content:

   ```
   C:\> notepad C:\path\to\deadline_sync.bat
   ```

   ```
   YOUR_PATH_TO_DEADLINE.EXE queue sync-output --profile deadline-downloader
    --farm-id YOUR_FARM_ID --queue-id YOUR_QUEUE_ID --storage-profile-
   ```

```
id YOUR_PROFILE_ID --checkpoint-dir path\to\checkpoint\checkpoints > path\to
\logs\deadline.log 2>&1
```

5. **Test Batch File**

   Test the batch file manually:

   ```
   C:\> .\path\to\deadline_sync.bat
   ```

   Check the log file was created:

   ```
   C:\> notepad path\to\logs\deadline_sync.log
   ```

6. **Check Task Scheduler Service**

   Ensure Task Scheduler service is running:

   ```
   C:\> sc query "Schedule"
   ```

   If the service doesn't exist, try alternative names:

   ```
   C:\> sc query "TaskScheduler"
   C:\> sc query "Task Scheduler"
   ```

   If not running, start it:

   ```
   C:\> sc start "Schedule"
   ```

7. **Create Scheduled Task**

   Create the task to run every 5 minutes.

   ```
   C:\> schtasks /create /tn "DeadlineOutputSync" /tr "C:\path\to
   \deadline_sync.bat" /sc minute /mo 5
   ```

   Command breakdown:

   - `/tn` - Task name
   - `/tr` - Task to run (your batch file)

- `/sc minute /mo 5` - Schedule: every 5 minutes

8. **Verify Task Creation**

   Check that the task was created successfully:

   ```
   schtasks /query /tn "DeadlineOutputSync" /v /fo LIST
   ```

   Look for:

   - **Task To Run**: Should show your batch file path
   - **Next Run Time**: Should show a time within 5 minutes

9. **Test Task Execution**

   Run the task manually to test:

   ```
   schtasks /run /tn "DeadlineOutputSync"
   ```

   Check task status:

   ```
   schtasks /query /tn "DeadlineOutputSync"
   ```

Verify the setup

To verify the automatic downloads setup was successful, complete the following steps.

1. Submit a new test job.
2. Wait for one scheduler interval to complete, which in this case is 5 minutes.
3. Verify that new outputs are downloaded automatically.

If the outputs do not download, check the Troubleshooting section for the process logs.

Troubleshooting automatic downloads

If you encounter issues with the automatic downloads, check the following:

Download error codes

When a download fails, the **Download status** column in the Deadline Cloud monitor names the reason, and the `deadline queue sync-output` command records one of the following error codes. Most of these errors are resolved on the machine that runs the download command, which is often a different machine from the one you view the monitor on.

`PERMISSION_DENIED` (Permission denied)

The downloader isn't allowed to write to the output location, or its AWS credentials were denied access. Grant the user that runs the download command write access to the output location, confirm the command's AWS profile has access to the queue, and run the command again.

`DISK_FULL` (Disk full)

The machine running the download ran out of disk space. Free up space on the drive that holds the output location, then run the command again.

`PATH_NOT_FOUND` (Path not found)

The output destination doesn't exist or isn't mounted on the machine running the download. Create the directory or mount the shared drive, then run the command again.

`NETWORK_ERROR` (Network error)

A network interruption stopped the transfer. Network errors are usually transient. Run the command again, and check the machine's connectivity if the error repeats.

`UNKNOWN` (Failed)

The downloader couldn't identify a specific cause. The monitor shows **Failed** for this code and for any code it doesn't recognize. Check the log output of the `deadline queue sync-output` command for the underlying error.

A job whose download fails is retried automatically on each later run. After five failed attempts, the downloader stops retrying the job and prints a warning. After you fix the cause, recover the job by running the download command with a lookback window that covers when the job finished:

```
deadline queue sync-output --farm-id FARM_ID --queue-id QUEUE_ID \
    --storage-profile-id STORAGE_PROFILE_ID \
    --force-bootstrap --bootstrap-lookback-minutes 1440
```

Pass both flags together. The `--bootstrap-lookback-minutes` option defaults to 0, so `--force-bootstrap` on its own recovers nothing.

Why was my job skipped?

When the downloader skips a job, the **Download status** column names the reason directly in the cell:

No attachments (`no_attachments`)

The job wasn't submitted with job attachments, so it has no recorded output files to download. Some jobs never produce downloadable output, so a skipped job with no attachments usually needs no action.

Missing storage profile (`missing_storage_profile`)

The job was submitted without a storage profile while the download command uses one, so the downloader doesn't know where the job's files belong on each machine. Nothing downloads for the job until a storage profile is configured for it. A job's storage profile is set when the job is submitted, so submit the job with a storage profile and pass the same profile to the download command. In the tasks table, the **Missing storage profile** status is a link that opens an explanation, a documentation link, and a **Troubleshoot with AI** button. For more information, see Storage profiles for job attachments.

A skipped job with no reason shown was canceled or stopped before producing output. It needs no action.

Is it a download problem or a render problem?

A problem in a task's **Download status** column can mean two different things, and they have different fixes. The status text tells you which one you have:

- **No outputs** on a task whose run status is `FAILED` means the task failed to render on the farm. No file was ever produced, so there was nothing to download. Your drive and your network are fine. Check the task's logs to find the render error, fix it, and requeue the task. A task can also show **No outputs** after finishing successfully without writing any files, which is normal and needs no action.
- An error name, such as **Permission denied**, means the task rendered and its output exists, and copying the files to your file system failed. Resolve the error using Download error codes.

In the jobs table, both problems appear as a red segment in the job's download progress bar. Open the job's tasks table to tell them apart. A job where every task failed shows a dash instead of a download error, because the job never produced output. The job's own status column reports that failure. Diagnosing the download in these render-failure cases wastes time, so always check the task's run status first.

Troubleshoot with AI

The download failure messages in the Deadline Cloud monitor, including the red **Output sync failed** indicator, include a **Troubleshoot with AI** button. The button opens the Deadline Cloud assistant, which reads the download status record for your queue and walks you through your specific failure, including the commands to run to fix it.

Reach for it when a download keeps failing after you tried the fix for its error code, when you see an error you don't recognize, or when you aren't sure which machine the problem is on. The button appears when the Deadline Cloud assistant is enabled for your monitor. For more information, see Deadline Cloud assistant.

Storage profile issues

- An error like `[Errno 2] No such file or directory` or `[Errno 13] Permission denied` in the log file could be related to missing or misconfigured storage profiles.
- See Storage profiles for information about how to set up your storage profiles when the downloading machine is different from the submitting machine.
- For same-machine downloads, try the `--ignore-storage-profiles` flag.

Directory permissions

- Ensure the scheduler service user has:
  - Read/write access to the checkpoint directory
  - Write access to the output destination directory
- For Linux and macOS, use `ls -la` to check permissions.
- For Windows, review Security settings in the Properties folder.

Checking scheduler logs

Linux

1. Check if cron service is running:

```
# For systemd systems
$ sudo systemctl status cron
# or
$ sudo systemctl status crond

# Check if your user has cron job correctly configured
$ crontab -l
```

2. View cron execution logs:

```
# Check system logs for cron activity (most common locations)
$ sudo tail -f /var/log/syslog | grep CRON
$ sudo tail -f /var/log/cron.log | grep deadline

# View recent cron logs
$ sudo journalctl -u cron -f
$ sudo journalctl -u crond -f  # On some systems
```

3. Check your specific cron job logs:

```
# View the log file specified in your cron job
$ tail -100f /path/to/logs/deadline_sync.log
```

4. Search for cron job execution in system logs:

```
# Look for your specific cron job executions
$ sudo grep "deadline.*sync-output" /var/log/syslog

# Check for cron job starts and completions
$ sudo grep "$(whoami).*CMD.*deadline" /var/log/syslog
```

5. Check checkpoint file updates:

```
# List checkpoint files with timestamps
$ ls -la /path/to/checkpoint/directory/

# Check when checkpoint was last modified
$ stat /path/to/checkpoint/directory/queue-*_download_checkpoint.json
```

6. Check the log file:

```
$ ls -la /path/to/log/deadline_sync.log
```

macOS

Viewing Launch Agent Execution Logs:

1. Check if the launch agent is running:

```
$ sudo launchctl list | grep deadlinesync
```

Output shows: `PID Status Label` (PID will be - when not currently running, which is normal for interval jobs)

2. View detailed launch agent status:

```
$ sudo launchctl print system/com.user.deadlinesync
```

This shows execution history, last exit code, number of runs, and current state.

3. View launch agent execution logs:

```
# View recent logs (last hour)
log show --predicate 'subsystem contains "com.user.deadlinesync"' --last 1h

# View logs from a specific time period
log show --predicate 'subsystem contains "com.user.deadlinesync"' --start
 '2024-08-27 09:00:00'
```

4. Force run the launch agent for immediate testing:

```
$ sudo launchctl kickstart gui/$(id -u)/com.user.deadlinesync
```

   This immediately triggers the job regardless of the schedule, useful for testing.

5. Check checkpoint file updates:

```
# List checkpoint files with timestamps
$ ls -la /path/to/checkpoint/directory/
```

6. Check the log file:

```
$ ls -la /path/to/log/deadline_sync.log
```

Windows

1. Check if Task Scheduler service is running:

```
C:\> sc query "Schedule"
```

   If the service doesn't exist, try alternative names:

```
C:\> sc query "TaskScheduler"
```

```
C:\> sc query "Task Scheduler"
```

2. View your scheduled tasks:

```
C:> schtasks /query /tn "DeadlineOutputSync"
```

3. Check your task's log file:

```
# View the log file created by your batch script
C:> notepad C:\path\to\logs\deadline_sync.log
```

4. Check checkpoint file updates:

```
# List checkpoint files with timestamps
C:> dir "C:\path\to\checkpoint\directory" /od
```

Track spending and usage for Deadline Cloud farms

The AWS Deadline Cloud budget manager and usage explorer are cost management tools that provide the approximate cost of using Deadline Cloud based on available information about cost variables. The cost management tools don't guarantee the amount owed for your actual use of Deadline Cloud and other AWS services.

To help you manage costs for Deadline Cloud, you can use the following features:

- **Budget manager** – With the Deadline Cloud budget manager, you can create and edit budgets to help manage project costs.
- **Usage explorer** – With the Deadline Cloud usage explorer, you can view how many AWS resources are used and the estimated costs for those resources.
- **Cost scale factor** – With the cost scale factor, you can adjust how costs are displayed in the usage explorer and budget manager to reflect discounts or premiums that apply to your organization.
- **AWS cost allocation tags** – With cost allocation tags, you can track detailed costs for all of your AWS services. For more information, see Organizing and tracking costs using AWS cost allocation tags.

Cost assumptions

The basic calculation used by the Deadline Cloud cost management tools is:

```
Cost per job =
    (CMF run time x CMF compute rate) +
    (SMF run time x SMF compute rate) +
    (License run time x license rate)
```

- *Run time* is the sum of all tasks in a job, from start time to end time.
- *Compute rate* is determined by the AWS Deadline Cloud pricing for service-managed fleets. For customer-managed fleets, the compute rate is estimated to be $1 per worker hour.
- *License rate* is determined by the Deadline Cloud base license price and is only available for service-managed fleets. Additional tiers are not included. For more information about license pricing, see AWS Deadline Cloud pricing.

The cost estimate from the Deadline Cloud cost management tools may vary from your actual costs for a number of reasons. Common reasons include:

- *Customer owned resources and their pricing*. You can choose to bring your own resources, either from AWS or externally from on-premises or other cloud providers. Actual costs of these resources are not calculated.
- *Idle worker costs*. Idle worker costs are not included when the worker status is IDLE. This situation can happen for fleets with a minimum instance count greater than zero, or when workers transition between jobs. Idle worker cost are not included in calculations.
- *Worker stop and start time*. After workers complete a job, the cost for moving from IDLE to STOPPING and from STOPPING to STOPPED is not included in Deadline Cloud cost estimates.
- *Promotional credits, discounts, and custom pricing agreements*. The cost management tools don't account for promotional credits, private pricing agreements, or other discounts. You may be eligible for other discounts that are not part of the estimate. To adjust displayed costs to reflect these factors, use the Cost scale factor.
- *Asset storage*. Asset storage is not included in the cost and usage estimates.
- *Changes in price*. AWS offers pay-as-you-go pricing for most services. Prices may change over time. The cost management tools use the most up-to-date prices publicly available, but there may be delays after changes.
- *Taxes*. The cost management tools don't include taxes applied to our purchase of the service.
- *Rounding*. The cost management tool perform mathematical rounding of pricing data.
- *Currency*. Cost estimates are made in U.S. dollars. Global exchange rates vary over time. If you translate estimates to a different currency base on the current exchange, changes in the exchange rate affect the estimate.
- *Outside licensing*. If you choose to use pre-purchased licences (Software licensing for service-managed fleets), Deadline Cloud cost management tools can't account for this cost.

Cost scale factor

The cost scale factor is a farm-level setting that applies a multiplier to the calculated costs displayed in the usage explorer and budget manager. Use the cost scale factor to align cost estimates with your organization's actual pricing, such as private pricing agreements, promotional credits, or internal cost allocation markups.

Cost scale factor values

The cost scale factor accepts values from 0 to 100:

- **Values less than 1** represent discounts. For example, a value of 0.75 applies a 25% discount to displayed costs.
- **Values greater than 1** represent premiums or markups. For example, a value of 1.5 applies a 50% markup to displayed costs.
- **A value of 1** (the default) leaves costs unchanged.

Configure the cost scale factor

You can configure the cost scale factor when you create a farm or by editing an existing farm's settings.

To configure the cost scale factor for an existing farm

1. Open the AWS Deadline Cloud (Deadline Cloud) console. In the navigation pane, choose **Farms and other resources**.
2. Select the farm you want to modify.
3. Choose **Actions**, then choose **Edit**.
4. For **Cost scale factor**, enter a value between 0 and 100.
5. Choose **Save changes**.

Effects of the cost scale factor on cost tools

After you configure a cost scale factor, the value affects the usage explorer and budget manager in the following ways:

- **Usage explorer** – All new queries display cost data modified by the cost scale factor.
- **New budgets** – Budgets created after you configure the cost scale factor use the new value for all cost calculations.
- **Existing budgets** – Existing budgets use the cost scale factor for new cost calculations, but their accumulated cost history is not recalculated. To recalculate accumulated costs with the new factor, delete and recreate the budget.

Understand the cost model for service-managed fleets

AWS Deadline Cloud service-managed fleets (SMFs) with job attachments have a fundamentally different cost structure than traditional on-premises render farms or customer-managed fleets that use network file systems. Understanding this difference helps you plan budgets, optimize spending, and take advantage of elastic scaling without unexpected costs.

Traditional render farm costs

On a traditional render farm, costs come from two main areas:

- **File storage** – A high-performance network file system (NFS or SAN) is required to serve assets to every worker simultaneously. The cost of this storage scales with the throughput it must support, which indirectly limits how many workers the farm can scale to.
- **Compute** – Workers (render nodes) are provisioned and maintained regardless of how much work is in the queue. Idle workers still incur hardware, power, and cooling costs.
- **Farm management** – A traditional farm also requires infrastructure and staff time for the render farm scheduler, its job database, software configuration, monitoring and reporting, and ongoing maintenance and upgrades.

Because the file system must be provisioned to handle peak throughput, scaling the farm up and down is expensive and slow. Adding more workers requires additional file system capacity, and that capacity cannot be released when workers are idle.

How service-managed fleet costs differ

With service-managed fleets, costs are structured differently:

Compute (worker time)

You pay for EC2 instances only while they are processing jobs. There is no cost to provision or decommission workers. When the fleet scales to zero workers, your compute cost drops to zero. For more information about pricing, see AWS Deadline Cloud pricing.

Storage (Amazon EBS)

Each worker uses a local Amazon Elastic Block Store (Amazon EBS) volume. Deadline Cloud charges for Amazon EBS storage only while the worker instance exists. The storage cost is included in the Deadline Cloud service-managed fleet pricing.

Farm management

The scheduler, job database, fleet scaling, monitoring, and maintenance are part of the Deadline Cloud service. There is no separate infrastructure to run or maintain for farm management.

File transfer (job attachments)

Job attachments use Amazon Simple Storage Service (Amazon S3) to transfer files between your workstation and workers. There is no charge for throughput or bytes transferred between Amazon S3 and workers. The cost is based primarily on the amount of data stored in Amazon S3. Amazon S3 API request fees apply for file uploads and downloads but are typically not a significant cost driver. For more information about Amazon Simple Storage Service pricing, see Amazon Simple Storage Service pricing.

Note

There is no high-performance file system to provision or maintain. Job attachments throughput is virtually unlimited, scales automatically with the number of workers, and requires no capacity planning.

Why auto scaling is cost efficient with service-managed fleets

Because there is no fixed storage infrastructure to provision, SMFs can scale up and down freely without incurring additional overhead. This changes how you think about fleet sizing:

- **Scale up aggressively** – When a large job arrives, the fleet can grow to hundreds of workers. Each additional worker accesses job attachments in Amazon S3 without placing additional load on a shared file system.
- **Scale down to zero** – When work completes, the fleet can shrink to zero workers with no ongoing compute or storage cost (only Amazon S3 storage for cached assets).
- **Burst for deadlines** – Temporarily spinning up a large fleet to meet a deadline adds compute cost proportional to the duration, with no penalty for the scale.

For information about configuring auto scaling, see Auto scaling configuration.

Cost components at a glance

The following table summarizes where costs come from when using SMFs with job attachments.

Cost component	What drives the cost	How to optimize
Amazon EC2 compute	Worker running time × instance size	Use Spot Instances, right-size instance types, reduce task duration
Amazon EBS storage	Volume size × worker running time, plus IOPS/throughput above baseline	Use the default volume size unless workloads need more local space
Amazon S3 storage (job attachments)	Total bytes stored in the job attachments bucket	Apply an Amazon S3 Lifecycle policy to delete old assets automatically
Amazon S3 requests (job attachments)	Number of PUT and GET requests during upload and download	Typically not a significant cost driver; no action needed for most workloads
Usage-based licensing (optional)	License type × instance size × job duration	Use only for jobs that require the licensed software
Amazon CloudWatch Logs (optional)	Volume of worker and task logs collected	Reduce log verbosity, set retention policies

Example: Comparing service-managed fleet costs to a traditional farm

Consider a project that renders 1,000 frames with an average render time of 20 minutes per frame, using workers with at least 16 vCPUs. The total input asset size is 50 GB.

Traditional farm approach

You maintain a network file system provisioned for the throughput required by your peak worker count. The file system costs the same whether the farm is idle or running at full capacity. Adding workers beyond the file system's throughput limit requires an expensive upgrade.

SMF with job attachments approach

Workers scale up to process all frames and then scale down to zero. Each worker downloads only the assets it needs from Amazon S3. Your costs are:

- **Compute**: ~333 worker-hours (1,000 frames × 20 minutes) at the applicable instance rate.
- **Amazon S3 storage**: 50 GB of assets stored at standard Amazon S3 rates (a few dollars per month).
- **Amazon S3 requests**: A small number of GET requests as workers download assets (fractions of a cent per 1,000 requests).

When the job finishes, costs return to the Amazon S3 storage charge only. There is no ongoing file system cost.

For additional pricing examples, see AWS Deadline Cloud pricing.

Tips for managing service-managed fleet costs

- **Use Spot Instances** – Spot Instances provide significant savings over On-Demand pricing. Because render tasks are typically short and can be retried, Spot interruptions have minimal impact.
- **Set fleet maximum size** – Limit the maximum number of workers in your fleet to control the peak compute cost per job. For more information, see Auto scaling configuration.
- **Use budgets** – Create an Deadline Cloud budget to set spending limits and receive notifications. For more information, see Control costs with a budget.
- **Manage job attachment storage** – Apply an Amazon S3 Lifecycle configuration to automatically delete old job attachment files. Because job attachments uses content-addressable storage, unchanged files are not re-uploaded, which keeps storage costs low for iterative workflows.
- **Right-size your instances** – Choose the smallest instance type that meets your workload's CPU and memory requirements. Larger instances cost more per hour but may complete tasks faster, so compare total cost (rate × duration) across instance sizes.
- **Consider Wait and Save** – For non-urgent workloads, Wait and Save offers lower compute prices in exchange for flexible job start times.

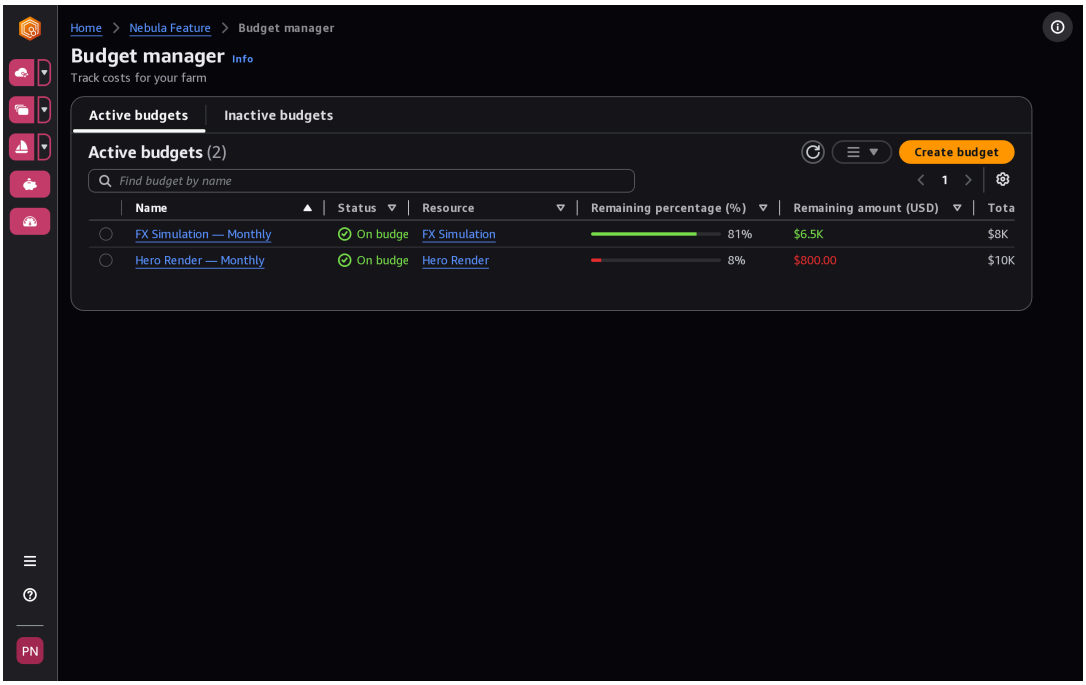
Related resources

- AWS Deadline Cloud pricing

- Cost management
- Control costs with a budget
- Amazon Simple Storage Service pricing

Control costs with a budget

The Deadline Cloud budget manager helps you control spending on a queue. You can create budget amounts and limits, and set automated actions to help reduce or stop additional spending against the budget. Each budget tracks the estimated cost of one queue, so you can give each project, department, or vendor its own queue and its own spending cap.



The following sections provide you with the steps for using the Deadline Cloud budget manager.

Topics

- Prerequisite
- Open the Deadline Cloud budget manager
- How budget actions affect running and new work
- Create a budget for a Deadline Cloud queue
- View a Deadline Cloud queue budget
- Edit a budget for a Deadline Cloud queue
- Deactivate a budget for a Deadline Cloud queue

- Monitor a budget with EventBridge events

Prerequisite

To use the Deadline Cloud budget manager, you must have `OWNER` access level. To grant `OWNER` permission, follow the steps in Managing users in Deadline Cloud.

Open the Deadline Cloud budget manager

To open the Deadline Cloud budget manager, use the following procedure.

1. Sign in to the AWS Management Console and open the Deadline Cloud console.
2. Choose **View farms**.
3. Locate the farm that you want to get information about, then choose **Manage jobs**.
4. In the Deadline Cloud monitor, in the left navigation pane, choose **Budgets**.

The budget manager summary page displays a list of both active and inactive budgets:

- **Active** budgets track against the selected resource (a queue).
- **Inactive** budgets have either expired or been canceled by a user, and are no longer tracking costs against this budget's limits.

After you choose a budget, the budget summary page contains basic information about the budget. Information provided includes the budget name, status, resources, remaining percentage, remaining amount, total budget, start date, and end date.

How budget actions affect running and new work

When spending against a budget reaches a limit you set, Deadline Cloud applies the action that you chose for that limit. Both actions stop the queue from scheduling new tasks. They differ in what happens to tasks that are already running on workers:

Stop after finishing current work (`STOP_SCHEDULING_AND_COMPLETE_TASKS`)

The queue stops assigning new tasks to workers. Tasks that are already running finish normally and continue to incur cost until they complete. Choose this action to stop new spending without losing partially completed frames.

Immediately stop work (`STOP_SCHEDULING_AND_CANCEL_TASKS`)

The queue stops assigning new tasks to workers and cancels tasks that are running, including partially completed frames. Choose this action when a hard spending cap matters more than finishing frames that are in flight.

You can add several limit actions to one budget at different remaining amounts. For example, stop scheduling new work when $500 remains so that current frames finish, and cancel all work when $0 remains as a hard stop.

Each budget tracks the estimated cost of a single queue. To give each project, department, or vendor its own spending cap, route each one's jobs through its own queue and create a budget for that queue. For an overview of how budgets combine with worker counts, resource limits, and job priority, see Control costs and concurrency.

To receive notifications before a budget reaches its limit, use the EventBridge events that Deadline Cloud sends as spending crosses each threshold percentage. For more information, see Monitor a budget with EventBridge events.

You can also create and adjust budgets programmatically with the CreateBudget and UpdateBudget API operations. When a budget action triggers, the queue reports a blocked reason of `BUDGET_THRESHOLD_REACHED` in the GetQueue response.

Create a budget for a Deadline Cloud queue

To create a budget, use the following procedure.

1. If you haven't already, sign in to the AWS Management Console, open the Deadline Cloud console, choose a farm, and then choose **Manage jobs**.
2. From the **Budget manager** page, choose **Create budget**.
3. In the details section, enter a **Budget name** for the budget.
4. (Optional) In the description field, enter a brief description of the budget.
5. From **Resource**, use the **Queue** dropdown to select the queue that you want to create a budget for.
6. For **Period**, set the start and end date for the budget by completing the following steps:

    a. For **Start date**, enter the first date of the budget tracking in YYYY/MM/DD format, or choose the **calendar** icon and select a **date**.

The default start date is the date that the budget is created.

b. For **End date**, enter the last date of the budget tracking in YYYY/MM/DD format or choose the **calendar** icon and select a **date**.

The default end date is 120 days from the start date.

7. For **Budget amount**, enter the dollar amount of the budget.
8. (Optional) We recommend that you create limit alerts. In the **Limit actions** section, you can implement automated actions that occur when specific amounts remain in the budget. To do this, complete the following steps:

a. Choose **Add new action**.

b. For **Remaining amount**, enter the dollar amount that you want to start the action.

c. In the **Action** dropdown, choose the action that you want. Actions include:

- **Stop after finishing current work** – All work currently running when the threshold amount is met continue to run (and incur costs) until finished.
- **Immediately stop work** – All work is canceled immediately when the threshold amount is met.

For details about how each action affects running and new work, see How budget actions affect running and new work.

d. To create additional limit alerts, choose **Add new action** and repeat the previous steps.

9. Choose **Create budget**.

View a Deadline Cloud queue budget

After you create a budget, you can view the budget on the **Budget manager** page. From there, you can view the budget's total amount and the overall cost allocated to the specific budget.

To view a budget, use the following procedure.

1. If you haven't already, sign in to the AWS Management Console, open the Deadline Cloud console, choose a farm, and then choose **Manage jobs**.
2. Choose **Budgets** from the left side navigation pane. The **Budget Manager** page appears.

3. To view an active budget, choose the **Active budgets** tab, and choose the name of the budget that you want to view. The budget details page appears.
4. To view the budget details for an expired budget, choose the **Inactive budgets** tab. Then, choose the name of the budget that you want to view. The budget details page appears.

Edit a budget for a Deadline Cloud queue

You can edit any active budget. To edit an active budget, use the following procedure.

1. If you haven't already, sign in to the AWS Management Console, open the Deadline Cloud console, choose a farm, and then choose **Manage jobs**.
2. From the **Budget Manager** page, in the **Active budgets** tab, choose the button next to the budget you want to edit.
3. From the **Actions** dropdown menu, select **Edit budget**.
4. Make the changes that you want, and then choose **Update budget**.

Deactivate a budget for a Deadline Cloud queue

You can deactivate any active budget. Deactivating a budget changes its status from **Active** to **Inactive**. When a budget is deactivated, it no longer tracks a resource to that budget's amount.

To deactivate a budget, use the following procedure.

1. If you haven't already, sign in to the AWS Management Console, open the Deadline Cloud console, choose a farm, and then choose **Manage jobs**.
2. From the **Budget manager** page, in the **Active Budgets** tab, choose the button next to the budget that you want to deactivate.
3. From the **Actions** dropdown menu, select **Deactivate budget**. In a few moments, the selected budget will change from **Active** to **Inactive** and will move from the **Active Budgets** tab to the **Inactive Budgets** tab.

Monitor a budget with EventBridge events

Deadline Cloud sends budget-related events, using Amazon EventBridge, to your default EventBridge event bus. You can create custom functions that receive the events and act on them to

send notifications to automatically notify users via email, Slack, or other channels when a budget reaches predefined levels. For example, you can send SMS messages when a budget reaches a certain threshold. These notifications help you stay on top of your spending and make informed decisions before your budget is exhausted.

Deadline Cloud periodically aggregates usage and cost data for each render farm. Then it checks to see if any of the budget thresholds has been crossed. If a threshold is crossed, Deadline Cloud triggers an event to alert you so that you can take the appropriate action. An event is triggered whenever a budget crosses one of these thresholds, specified in percent of the budget used:

- 10, 20, 30, 40, 50, 60, 70, 75, 80, 85, 90, 95, 96, 97, 98, 99, 100

The budget usage thresholds get closer together as a budget approaches 100 percent usage. This frequency helps you closely monitor usage as the budget reaches its limit. You can also set your own budget thresholds. Deadline Cloud sends an event when usage passes your custom thresholds. After your budget reaches 100 percent, Deadline Cloud stops sending events. If you adjust your budget, Deadline Cloud sends events for your thresholds based on the new budget amount.

You can use the EventBridge console (https://console.aws.amazon.com/events/) to create rules to send the Deadline Cloud events to the appropriate target for the event. For example, you can send the event to an Amazon Simple Queue Service queue and from there to multiple targets, such as AWS End User Messaging SMS or a Amazon Relational Database Service database for logging.

For examples of an EventBridge rule, see the following topics:

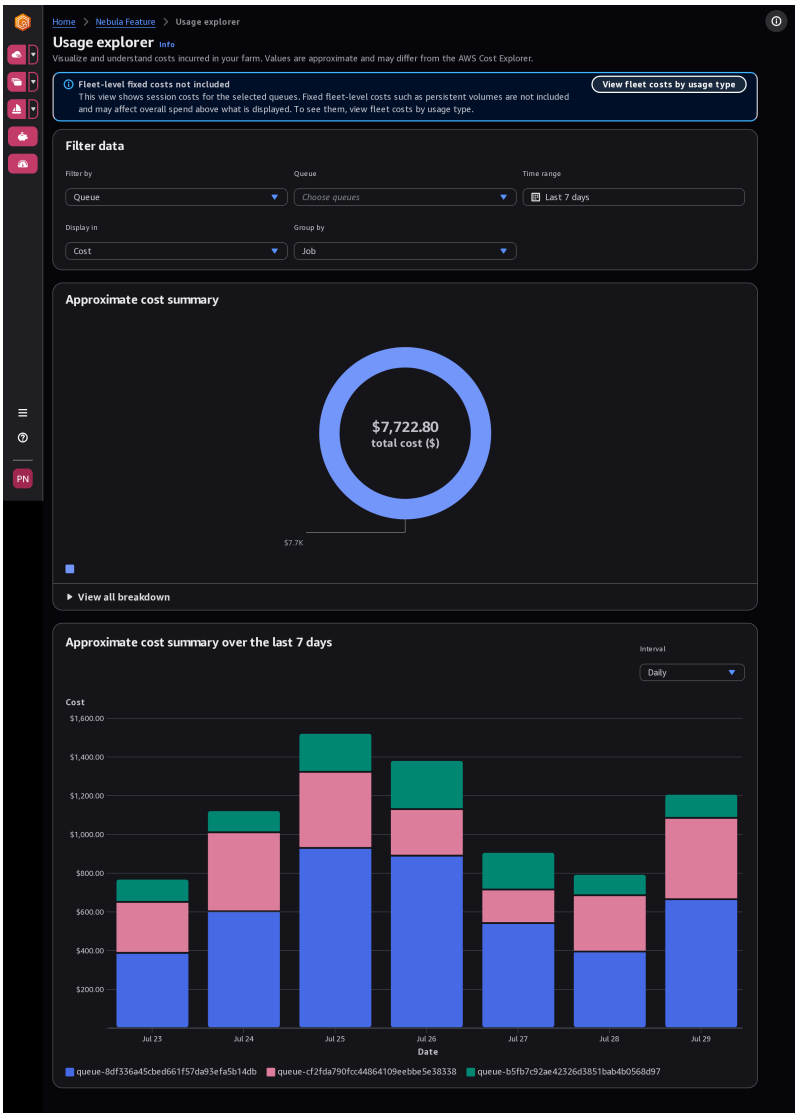
- Send an email when events happen using Amazon EventBridge.
- Creating an Amazon EventBridge rule that sends notifications to Amazon Q Developer in chat applications.
- Getting started with Amazon EventBridge.

For more information about budget events, see the Budget Threshold Reached event in the *Deadline Cloud Developer Guide*.

Track usage and costs with the Deadline Cloud usage explorer

With the Deadline Cloud usage explorer, you can see real-time metrics on the activity happening on each farm. You can look at the farm's costs by different variables, such as queue, fleet, job,

license product, or instance types. Select various time frames to see usage during a specific period of time, and look at usage trends over the course of time. You can also see a detailed breakdown of selected data points, allowing for a closer look into metrics. Usage can be shown by time (minutes and hours) or by cost ($USD).



The following sections show you the steps for accessing and using the Deadline Cloud usage explorer.

Topics

- Prerequisite
- Open the usage explorer
- Use the usage explorer

Prerequisite

To use the Deadline Cloud usage explorer, you must have either `MANAGER` or `OWNER` farm permissions. For more information, see How permissions work in Deadline Cloud.

> **Note**
>
> If your time zone doesn't align to a full hour, such as India Standard Time (UTC+5:30), the usage explorer doesn't show usage metrics. To see metrics, set your time zone to a zone that aligns to a full hour.

Open the usage explorer

To open the Deadline Cloud usage explorer, use the following procedure.

1. Sign in to the AWS Management Console and open the Deadline Cloud console.
2. To see all available farms, choose **View farms**.
3. Locate the farm that you want to get information about, then choose **Manage jobs**. The Deadline Cloud monitor opens in a new tab.
4. In the Deadline Cloud monitor, from the left menu, select **Usage explorer**.

Use the usage explorer

From the usage explorer page, you can select specific parameters in which the data can be displayed. By default, you see total usage in time (hours and minutes) within the last 7 days. You can change these parameters, and the information displayed changes dynamically in accordance to the parameter settings.

You can group the results based on the queue, fleet, job, user, usage type, instance type, or license product. If you choose license product, costs are calculated for specific licenses. If you filter by fleet and group by usage type, you can see persistent volume cost associated with your fleets. For all other groups the time is calculated by adding up the time taken for each task to run.

You can filter results by queues or by fleets, but you cannot filter by both at the same time.

The usage explorer returns only 100 results based on the filter criteria that you set. The results are listed in descending order by the date created timestamp. If there are more than 100 results, you get an error message. You can refine your query to reduce the number of results:

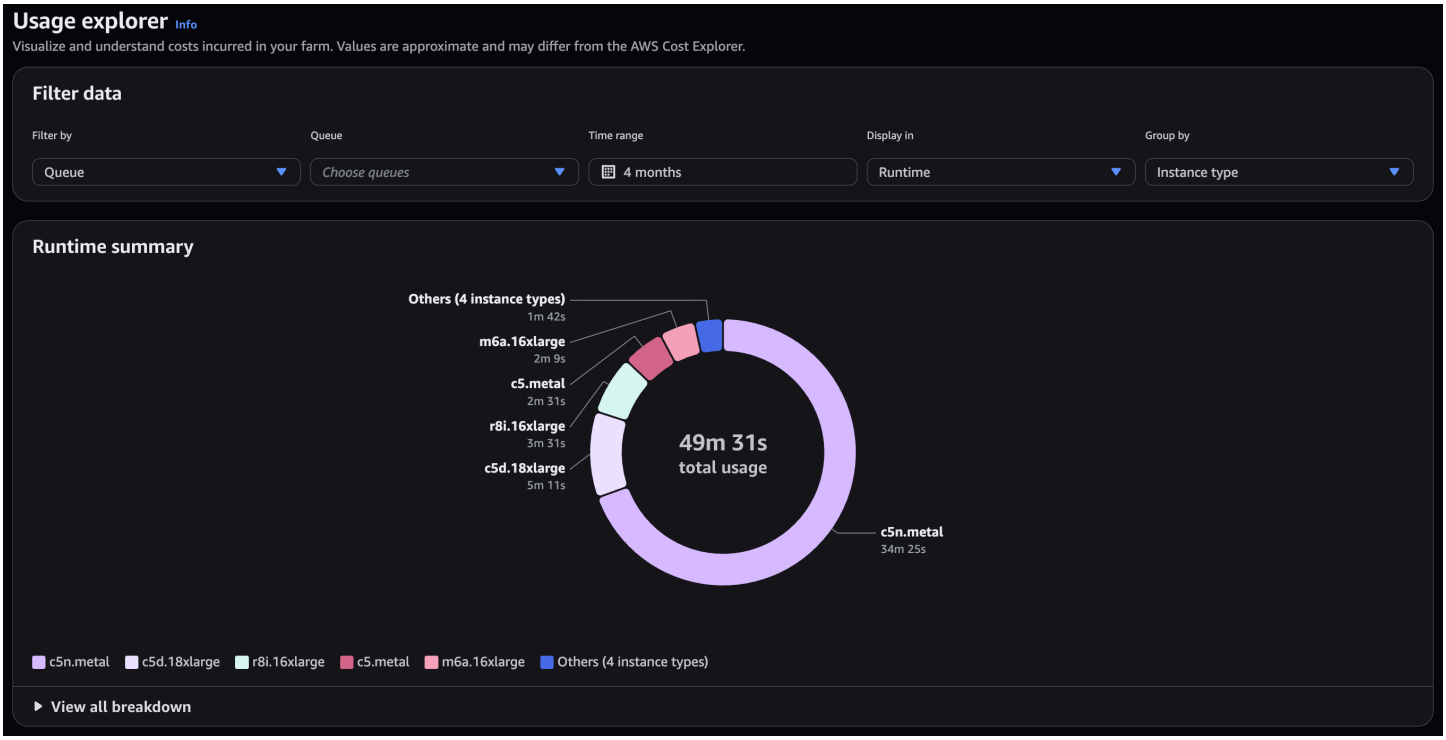
- Select a smaller time range
- Select fewer queues or fleets
- Select a different grouping, such as grouping by queue or fleet instead of job

Topics

Use visual graphs to review data

You can review data in a visual format to identify trends and potential areas that might need more analysis or attention. Usage explorer offers a pie chart that displays overall usage and cost with the option to group the totals into smaller subtotals.

> **Note**
>
> The chart *only* displays the top five results with other results combined in an "others" section. You can view all results in the breakdown section below the chart.



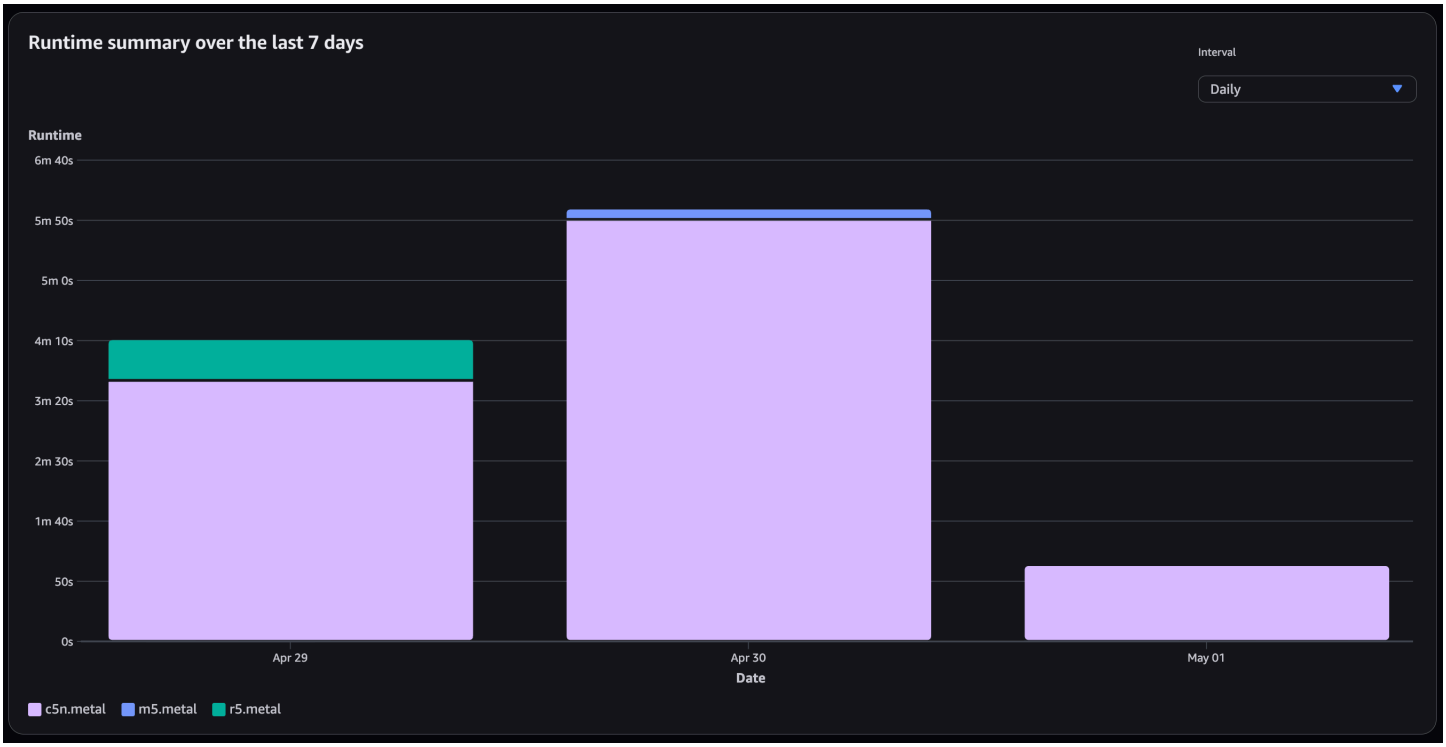
View a breakdown of metrics

Beneath the pie chart, usage explorer offers a more detailed breakdown of specific metrics, which will change as parameters change. By default, five results display in the usage explorer. You can scroll through results using the pagination arrows in the breakdown section.

Breakdown is minimized by default. To expand and display the results, select the **View all breakdown** arrow. To download the breakdown, choose **Download data**.

View approximate runtime of queues and fleets

You can also view the approximate runtime of your queues or fleets based on different intervals that you specify. The interval options are hourly, daily, weekly, and monthly. After you select an interval, the graph displays the approximate runtime of your queues or fleets.



Cost management

AWS Deadline Cloud provides budgets and the usage explorer to help you control and visualize costs for your jobs. However, Deadline Cloud uses other AWS services, such as Amazon S3. Costs for those services are not reflected in Deadline Cloud budgets or the usage explorer and are charged separately based on usage. Depending on how you configure Deadline Cloud, you may use the following AWS services, as well as others:

Service	Pricing page
Amazon CloudWatch Logs	Amazon CloudWatch Logs pricing
Amazon Elastic Compute Cloud	Amazon Elastic Compute Cloud pricing
AWS Key Management Service	AWS Key Management Service pricing
AWS PrivateLink	AWS PrivateLink pricing
Amazon Simple Storage Service	Amazon Simple Storage Service pricing
Amazon Virtual Private Cloud	Amazon Virtual Private Cloud pricing

Cost management best practices

Using the following best practices can help you understand and control your costs when using Deadline Cloud and the tradeoffs you can make between cost and efficiency.

> **Note**
>
> The final cost of using Deadline Cloud depends on the interaction between a number of AWS services, the amount of work that you process, and the AWS Region where you run your jobs. The following best practices are guidelines and may not significantly reduce costs.

Best practices for CloudWatch Logs

Deadline Cloud sends worker and task logs to CloudWatch Logs. You are charged to collect, store, and analyze these logs. You can reduce costs by logging only the minimum amount of data required to monitor your tasks.

When you create a queue or fleet, Deadline Cloud creates a CloudWatch Logs log group with the following names:

- `/aws/deadline/<FARM_ID>/<FLEET_ID>`
- `/aws/deadline/<FARM_ID>/<QUEUE_ID>`

By default, these logs never expire. You can adjust the retention policy of log groups to remove old logs and help reduce storage costs. You can also export logs to Amazon S3. Amazon S3 storage costs are lower than those for CloudWatch. For more information, see Exporting log data to Amazon S3.

Best practices for Amazon EC2

You can use Amazon EC2 instances for both service-managed and customer-managed fleets. There are three considerations:

- For service-managed fleets, you can choose to have one or more instances available at all times by setting the minimum worker count for the fleet. When you set the minimum worker count above 0, the fleet always has this many workers running. This setting can reduce the amount of

time that it takes for Deadline Cloud to start processing jobs, however you are charged for the instance's idle time.

- For service-managed fleets, set a maximum size for the fleet. This setting limits the number of instances that a fleet can auto scale to. Fleets won't grow past this size even if there are more jobs waiting to be processed.
- For both service-managed and customer-managed fleets, you can specify the Amazon EC2 instance types in your fleets. Using smaller instances costs less per minute, but may take longer to complete a job. Conversely, a larger instance costs more per minute, but can reduce the time to complete a job. Understanding the demands that your jobs place on an instance can help reduce your costs.
- When possible, choose Amazon EC2 Spot instances for your fleet. Spot instances are available for a reduced price, but may be interrupted by on-demand requests. On-demand instances are charged by the second and are not interrupted.

Best practices for AWS KMS

By default, Deadline Cloud encrypts you data with an AWS owned key. You are not charged for this key.

You may choose to use a customer managed key to encrypt your data. When you use your own key, you are charged based on how your key is used. If you use an existing key, the additional use is an incremental cost.

Best practices for AWS PrivateLink

You can use AWS PrivateLink to create a connection between your VPC and Deadline Cloud using an interface endpoint. When you create a connection, you can call all of the Deadline Cloud API actions. You are charged per hour for each endpoint that you create. If you use PrivateLink, you must create at least three endpoints, and depending on your configuration, you may need as many as five.

Best practices for Amazon S3

Deadline Cloud uses Amazon S3 to store assets for processing, job attachments, output, and logs. To reduce the costs associated with Amazon S3, reduce the amount of data that you store. Some suggestions:

- Only store assets that are currently in use or that will be used shortly.

- Use an S3 Lifecycle configuration to automatically delete unused files from an S3 bucket.

Best practices for Amazon VPC

When you use usage-based licensing for your customer-managed fleet, you create a Deadline Cloud license endpoint, which is a Amazon VPC endpoint created in your account. This endpoint is charged at an hourly rate. To reduce costs, remove the endpoints when you are not using usage-based licenses.

Control costs and concurrency

On a traditional render farm, fixed infrastructure creates indirect limits on concurrency and cost. A shared file system's throughput caps how fast workers can access data. A fixed number of render nodes limits how many jobs run at the same time. These bottlenecks slow down rendering, but they also prevent unexpected spending.

With AWS Deadline Cloud, you have precise, direct controls over both cost and concurrency. You can scale rapidly to finish jobs quickly, with budgets that enforce hard dollar caps and fleet settings that limit peak concurrency. Unlike a traditional farm, you don't need to trade render speed for cost control—you can have both.

With Deadline Cloud, you can also limit resource usage for other reasons, such as matching a fixed number of software licenses. The following mechanisms each operate at a different level. You can combine them to match your organization's requirements.

Comparison of cost and concurrency controls

The following table summarizes the controls available in Deadline Cloud. Choose one or more based on what you want to limit.

Control	Level	What it does	Best for
Fleet maximum worker count	Fleet	Caps the total number of workers that can run simultaneously in a fleet, regardless of how many jobs are queued.	Hard cap on peak compute across all jobs in a fleet. Replaces the natural limit of a fixed number of render nodes.

Control	Level	What it does	Best for
Job maximum worker count	Job	Caps the number of workers assigned to a single job. Other workers in the fleet remain available for other jobs.	Preventing a single large job from consuming the entire fleet while smaller jobs wait.
Resource limits	Farm or queue	Caps the number of tasks that can simultaneously use a shared resource (such as software licenses or a file server).	Matching concurrency to a constrained resource shared across jobs or queues. Replaces the natural limit of file system throughput or license count.
Budgets with limit actions	Queue	Tracks cumulative spending on a queue. When spending reaches a threshold, an action stops scheduling new work or cancels running work.	Enforcing a dollar cap on total spending for a project or time period.
Job priority	Job	Determines which jobs are processed first when multiple jobs compete for the same workers.	Ensuring urgent work processes before less-critical jobs without changing fleet capacity.

Fleet maximum worker count

The **maximum worker count** setting on a fleet limits how many workers can run simultaneously. When the fleet reaches this maximum, it stops starting new workers even if more jobs are waiting. This setting is the most direct replacement for the natural concurrency limit of a fixed-size render farm.

Use this setting when you want to cap your peak compute cost at the fleet level. For example, setting the maximum to 50 workers means you never pay for more than 50 concurrent instances, regardless of queue depth.

For more information, see Auto scaling configuration.

Job maximum worker count

The **max-worker-count** option on a job limits how many workers can process that specific job. When the maximum is reached, no more workers are assigned to the job even if workers are available in the fleet. Other jobs in the queue can still use the remaining workers.

Use this setting when you want to prevent a single large job from monopolizing the fleet. For example, if your fleet has 100 workers and you submit a 10,000-frame job with `--max-worker-count 50`, the remaining 50 workers stay available for other jobs. You can also change this value after submission.

```
deadline bundle submit my_job --max-worker-count 50
```

Resource limits

Resource limits cap the number of tasks that can simultaneously use a constrained resource, such as floating software licenses or a file server with limited throughput. Limits are defined at the farm level and associated with one or more queues. Steps in a job that declare a limit requirement only run tasks up to the available count.

Use limits when you have a shared resource with a fixed capacity. For example, if you have 25 floating licenses for a renderer, create a limit of 25. With this limit, no more than 25 tasks use that license concurrently across all queues that share it.

For more information, see Create resource limits for jobs in the *Deadline Cloud Developer Guide*.

Budgets with limit actions

An Deadline Cloud **budget** tracks cumulative estimated spending on a queue over a time period. You configure *limit actions* that trigger when spending reaches a threshold. Available actions include:

- **Stop scheduling new work** – Running tasks complete, but the fleet stops accepting new tasks.
- **Stop all work immediately** – All running tasks are canceled and no new tasks are assigned.

Use budgets when you want to enforce a dollar cap for a project or billing period. You can create multiple thresholds with different actions. For example, stop scheduling at $5,000 remaining to allow current work to finish gracefully, and cancel all work if spending reaches $0 remaining.

For more information, see Control costs with a budget.

Job priority

Job priority influences the order in which jobs are processed when multiple jobs compete for workers in the same queue. Priority ranges from 0 to 100, with higher numbers generally processed first. Jobs with the same priority are processed in the order received.

Priority does not limit concurrency or cost directly. Instead, it helps ensure that the most important work gets workers first when the fleet is at capacity. Combine priority with a fleet maximum worker count to control both which jobs run first and how much compute runs in total.

Limit your spend rate

A fleet's cost comes from its running workers, so you can cap a fleet's spend rate by setting the maximum number of workers. Two controls work together:

- The **fleet maximum worker count** sets your peak spend rate by capping how many workers can run at once. For more information about what makes up the hourly cost of a worker, see Understand the cost model for service-managed fleets.
- A **budget** caps cumulative spending. A budget tracks total estimated spending over a time period rather than a rate, so it adds a hard dollar limit alongside the fleet's capacity cap.

Mix Spot and On-Demand capacity

Each service-managed fleet uses a single instance market option: Spot, On-Demand, or Wait and Save. To combine market options, create a separate fleet for each option and associate the fleets with the same queue. For more information, see Service-managed fleets and Associate a queue and fleet.

When a queue has more than one fleet, it distributes jobs evenly across those fleets. To instead treat one fleet as primary capacity and another as overflow, adjust the fleets' maximum worker counts. The capacity manager sample in the *Deadline Cloud Developer Guide* automates that adjustment for a hybrid Wait and Save plus Spot setup.

Temporarily raise limits during busy periods

All of these controls are adjustable at any time. You can temporarily increase limits during periods of increased job activity. For example, before a delivery deadline you might raise a fleet's maximum

worker count, increase a budget threshold, and add standby workers to reduce job start latency. After the deadline, lower the settings again.

You can update fleet auto scaling settings and budgets from the Deadline Cloud console. To change capacity on a schedule, such as raising the standby worker count during working hours, use the sample CloudFormation template at fleet_standby_scheduling on the GitHub website. If you need more capacity than your account's service quotas allow, see Service quotas and throttling for Deadline Cloud.

Combine controls

Each control on its own addresses one need. The following scenarios show how to combine controls:

"I don't want one huge job to starve smaller jobs"

Set `--max-worker-count` on large jobs to reserve fleet capacity for other work. Optionally, set higher priority on the smaller jobs.

"I want a combination of cost control and fair sharing"

Set a fleet maximum worker count, use per-job max worker counts for large jobs, and add a budget as a safety net. This combination gives you a peak capacity limit, per-job fairness, and a dollar backstop.

Security in Deadline Cloud

Security in Deadline Cloud comes down to three questions: who can see and manage your farm, what the service and your jobs can do on your behalf, and how workloads stay separate from each other. For a map of the controls that answer each question and where to configure them, start with Security controls in Deadline Cloud. For hands-on guidance for each part of your farm, see Security best practices for Deadline Cloud.

Topics

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to AWS Deadline Cloud, see AWS services in Scope by Compliance Program. AWS Deadline Cloud is in scope for SOC 1, 2, and 3 compliance. For more information, see the section called “Compliance validation”.

- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

Security controls in Deadline Cloud

A render farm runs jobs from many people on shared infrastructure. Artists submit scenes, worker hosts run them, and the results land in shared storage. AWS Deadline Cloud gives you a set of controls to decide who can see and change farm resources, what jobs can do while they run, and how workloads stay separate from each other.

Three questions organize the controls:

- **Who can see and manage farm resources?** Monitor users with access levels, and IAM policies for the console and API.
- **What can the service, workers, and jobs do on your behalf?** The IAM service roles for fleets, worker hosts, queues, and monitors.
- **How are workloads kept separate?** Farm and queue structure, operating system users for jobs, and per-queue job attachment buckets.

The following table maps each control to what it governs and where to learn more.

Security controls in Deadline Cloud

Control	What it governs	More information
Monitor users and access levels	Which people can see each farm, queue, and fleet in the monitor, and whether they can view, contribute, manage, or own those resources. Users come from AWS IAM Identity Center (IAM Identity Center).	Managing users in the *Deadline Cloud User Guide*
IAM identity-based policies	What people and tools with AWS credentials can do with the Deadline Cloud console and API, such as creating farms or submitting jobs from a pipeline.	Identity and Access Management in Deadline Cloud

Control	What it governs	More information
Service roles	What Deadline Cloud can do on your behalf. The fleet, worker host, queue, and monitor roles each grant a different part of the service a scoped set of permissions.	Service roles
Queue operating system user	The operating system permissions that job processes have on worker hosts. On a customer-managed fleet, give each queue its own OS user. On a service-managed fleet, all jobs run as one service-managed user; separate queues by giving them separate fleets.	Run jobs as dedicated OS users
Farm and queue structure	The isolation boundaries between workloads . Farms don't share Deadline Cloud resources with each other, and queues within a farm can be separated by fleet, OS user, and bucket.	Isolate workloads with farms, fleets, and queues
Job attachment buckets	Which Amazon S3 data each queue can read and write through its queue role, bucket, and root prefix.	Secure job attachment and software buckets
Encryption keys	How farm data is encrypted at rest, with an AWS owned key or a customer managed AWS KMS key.	Key management
Network controls	How traffic reaches Deadline Cloud: private connectivity with AWS PrivateLink, and the endpoints to allowlist in restricted networks.	Access AWS Deadline Cloud using an interface endpoint (AWS PrivateLink), Restricted network environments

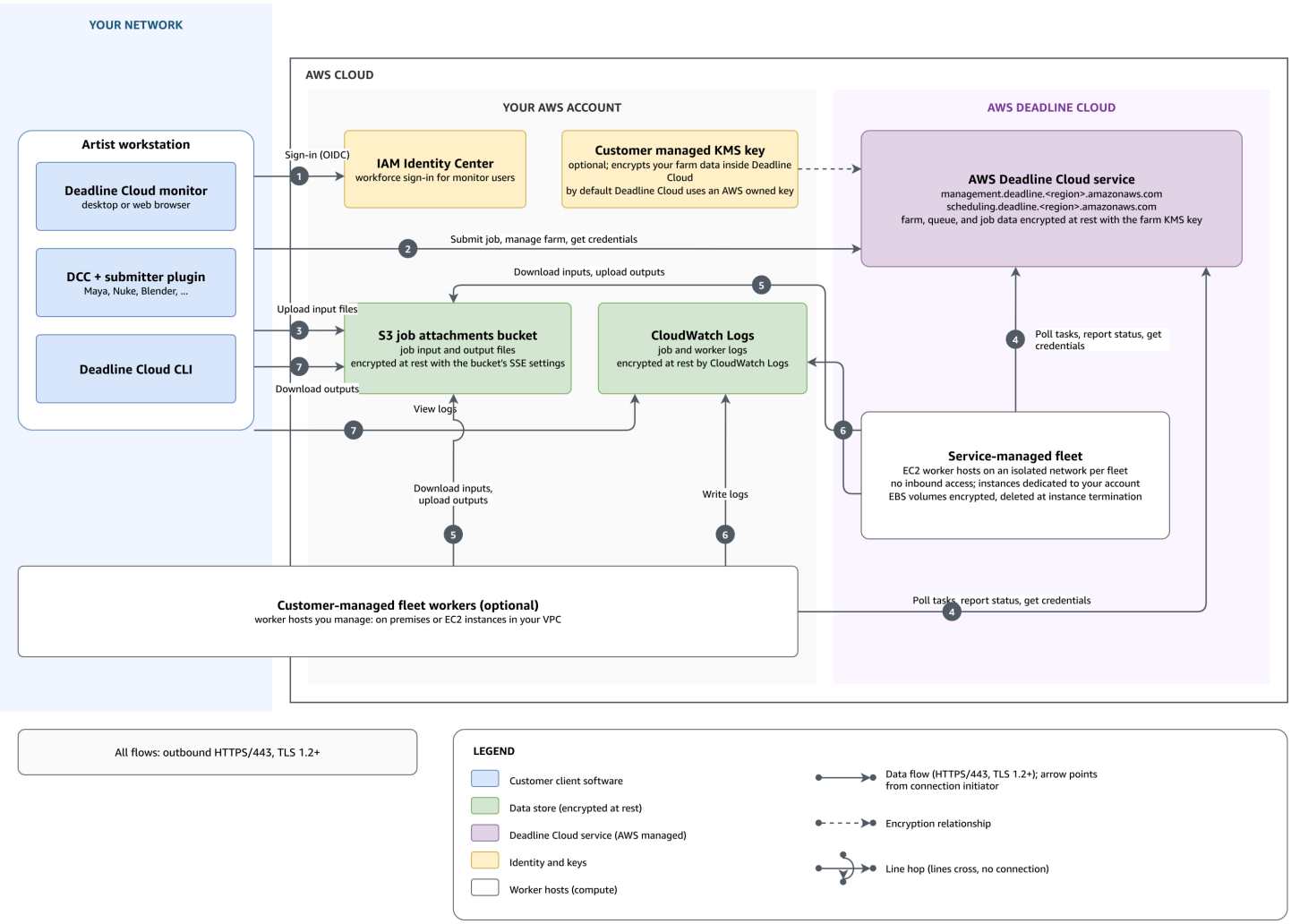
Where to start

Pick the entry point that matches your role:

- If you administer a studio and want to control which artists see which projects, start with Managing users and the Isolate workloads with farms, fleets, and queues guidance.
- If you build pipelines that submit jobs or read results, start with Identity-based policy examples for Deadline Cloud and the section called "Job attachment buckets".
- If you want to control what jobs can access while they run, start with the section called "Job OS users" for OS permissions and Service roles for the queue role that grants AWS permissions.
- If you run customer-managed fleets, start with Service roles and Secure worker hosts.

Data flow, ports, and encryption in Deadline Cloud

Content security reviews, such as studio security audits and vendor onboarding questionnaires, ask for a data flow diagram. The diagram covers the components of the render farm, where each component runs, how data moves between them, and how connections and stored data are encrypted. The diagram and tables on this page answer those questions for a typical AWS Deadline Cloud deployment. For more information about the security controls that you can configure for each part of the farm, see Security controls in Deadline Cloud.



Components and where they run

A Deadline Cloud render farm spans three locations: your network, your AWS account, and Deadline Cloud itself, which AWS manages. Customer-managed fleet workers are the exception: you choose where they run.

The following table describes each component and where it runs.

Component	Where it runs	Purpose
Deadline Cloud monitor, integrated submitters, and the Deadline Cloud CLI	Your network	Client tools on artist workstations that submit jobs, upload and download files, and track job progress.

Component	Where it runs	Purpose
AWS IAM Identity Center	Your AWS account	Sign-in and access levels for monitor users. For more information, see Managing users.
Amazon Simple Storage Service job attachments bucket	Your AWS account	Input and output files for jobs. Each queue has its own bucket and root prefix, so queue permission boundaries also apply to files.
Amazon CloudWatch Logs	Your AWS account	Job logs and worker logs.
AWS Key Management Service key	Your AWS account, or AWS managed	The key that encrypts your farm data inside Deadline Cloud. By default, Deadline Cloud uses an AWS owned key that you don't manage. You can supply a customer managed key from your account when you create the farm. For more information, see Key management.
Customer-managed fleet workers	Your choice: machines on your network, or Amazon Elastic Compute Cloud instances in your VPC	Worker hosts that you manage, running your software and the Deadline Cloud worker agent. A customer-managed fleet only needs outbound HTTPS access. If workers run in your VPC, optional AWS PrivateLink endpoints keep their traffic inside the VPC.
Deadline Cloud service	AWS managed	The service endpoints `management.deadline.` *`region`*`.amazonaws.com` and `scheduling.deadline.` *`region`*`.amazonaws.com`, and the farm, queue, and job scheduling data behind them.

Component	Where it runs	Purpose
Service-managed fleet workers	AWS managed	Amazon EC2 instances on an isolated network for each fleet. The instances are dedicated to your account and accept no inbound connections.

Data flows, ports, and protocols

Every client and worker in the diagram initiates its connections outbound to an AWS endpoint over HTTPS on TCP port 443. We require TLS 1.2 to encrypt connections in transit, and we recommend TLS 1.3. Deadline Cloud doesn't make inbound connections to your network. The flow numbers in the following table match the numbers in the diagram.

Flow	Connection	Description
1	Monitor to IAM Identity Center	Monitor users sign in through AWS IAM Identity Center with OpenID Connect (OIDC).
2	Workstation to the management endpoint	Submitters, the CLI, and the monitor submit jobs, manage farm resources, and request temporary role credentials through `management.deadline.` *`region`*`.amazonaws.com .`
3	Workstation to Amazon S3	Job attachments uploads input files to the queue's Amazon S3 bucket.
4	Workers to the scheduling endpoint	Workers in both fleet types poll for tasks, report progress and status, and request temporary role credentials through `scheduling.deadline.` *`region`*`.amazonaws.com .`
5	Workers to Amazon S3	Workers download input files from the queue's bucket and upload output files to it.
6	Workers to CloudWatch Logs	The worker agent writes job and worker logs.

Flow	Connection	Description
7	Workstation to CloudWatch Logs and Amazon S3	The monitor reads job logs, and the monitor and CLI download output files.

Neither the client tools nor the workers hold long-lived credentials for the queue's resources. Both request temporary credentials from Deadline Cloud and receive the permissions that you attach to the queue and fleet roles:

- The monitor, the CLI, and the submitters call `AssumeQueueRoleForUser` to upload and download job attachments, and `AssumeQueueRoleForRead` for read-only access such as reading job logs.
- A worker calls `AssumeFleetRoleForWorker` for the fleet role, then `AssumeQueueRoleForWorker` for the queue role of the job it runs.

For more information about the roles and what each one grants, see Service roles.

If your studio filters outbound web traffic, see Restricted network environments for the full list of endpoints to allowlist. To keep traffic between your VPC and Deadline Cloud off the public internet, use AWS PrivateLink interface endpoints. For more information, see Access AWS Deadline Cloud using an interface endpoint (AWS PrivateLink).

Access to the monitor web application

Your monitor URL, `https://`*`monitor-alias`*`.`*`region`*`.deadlinecloud.amazonaws.com`, is reachable from any location on the internet, and you can't restrict the URL to a set of IP addresses. The URL serves the monitor web application, which is the same application for every Deadline Cloud monitor. The application holds none of your farm, queue, or job data, and anyone who opens the URL sees a sign-in page.

Your farm data reaches the monitor through the Deadline Cloud APIs, so the API calls are where access is granted or denied. After you sign in through AWS IAM Identity Center (IAM Identity Center), Deadline Cloud assumes the monitor role for you. The monitor then calls the APIs from the browser with the role's temporary credentials. IAM authorizes every call. The managed policies on the monitor role allow each action only at the access levels that you granted. For more information, see Monitor role and Managing users.

To make farm data reachable only from your own network, apply the restriction to the API calls instead of the URL. You can allow only your IP ranges, or only requests that arrive through your VPC endpoints. For more information, see Restrict farm access to your network.

Encryption summary

Each data store is encrypted at rest independently. The farm AWS KMS key applies only to the data that Deadline Cloud holds; your Amazon S3 bucket and your log groups use their own encryption settings.

- **In transit** – All connections use HTTPS with TLS 1.2 required and TLS 1.3 recommended. For more information, see Encryption in transit.
- **Farm data** – The Deadline Cloud service encrypts your farm, queue, and job data at rest with the farm's AWS KMS key. You can use the default AWS owned key or supply a customer managed key when you create the farm. For more information, see Key management.
- **Job attachments** – Amazon Simple Storage Service encrypts your input and output files at rest with the server-side encryption configured on the bucket. By default, Amazon S3 uses Amazon S3 managed keys (SSE-S3). To use your own AWS KMS key instead, configure the bucket for SSE-KMS and grant the queue role access to the key. For more information, see Job attachments in Deadline Cloud.
- **Worker volumes** – In service-managed fleets, Deadline Cloud encrypts the Amazon Elastic Block Store volumes attached to worker instances and deletes the volumes when instances terminate. For more information, see Encryption at rest.
- **Logs** – Amazon CloudWatch Logs encrypts job logs and worker logs at rest with its own server-side encryption. To use your own AWS KMS key instead, associate the key with the log group. For more information, see Encrypt log data in CloudWatch Logs using AWS KMS in the *Amazon CloudWatch Logs User Guide*.

Security best practices for Deadline Cloud

Each of the following topics covers securing one part of your AWS Deadline Cloud (Deadline Cloud) farm. For a map of all the security controls in Deadline Cloud and where to configure them, see Security controls in Deadline Cloud.

Topics

- Isolate workloads with farms, fleets, and queues

- Run jobs as dedicated OS users
- Secure job attachment and software buckets
- Secure worker hosts
- Secure artist workstations
- Restrict farm access to your network

Isolate workloads with farms, fleets, and queues

You can arrange Deadline Cloud farms, fleets, and queues many ways. The arrangement you choose sets the isolation boundaries between your workloads, so decide how much separation your teams, shows, or clients need before you build.

Two facts about fleets drive the decision:

- When a fleet is associated with more than one queue, jobs from all of those queues run on the same worker hosts. After a job runs, data can remain on the host, such as files in a temporary directory or the queue user's home directory, and a job can leave processes running that later jobs can observe.

  Some of that data remains by design. A job can cache data under the job user's home directory so that later sessions reuse it. For example, the conda queue environment caches downloaded packages there. On a service-managed fleet, all jobs run as one user, so every queue on the fleet shares those caches. Cached data remains until the worker shuts down, or across worker replacement when the fleet uses persistent storage. For more information, see Sessions in the *Deadline Cloud Developer Guide* and Persistent storage in the *Deadline Cloud User Guide*.
- Worker hosts run jobs as an operating system user. On a customer-managed fleet, you can give each queue its own user, so queues can share a fleet while their jobs' files and processes stay separated. On a service-managed fleet, all jobs run as a single user, so give queues that need separation their own fleets. In both cases, the hosts run only jobs from queues in your farm.

With those facts in mind, choose the arrangement that fits your isolation needs:

- **Separate farms** – A farm doesn't share Deadline Cloud resources such as fleets, queues, and storage profiles with other farms, so separate farms give workloads the strongest separation. The tradeoff is more resources to set up and manage, and worker capacity that one farm can't

lend to another. Sharing external AWS resources, such as an Amazon S3 bucket, between farms weakens the boundary, so give each farm its own.

- **One farm, a fleet for each queue** – Queues that need separation from each other each get their own fleet, so their jobs never share worker hosts. You keep a single farm to manage, and monitor users can be scoped to each queue. For more information about scoping users, see Managing users in the *Deadline Cloud User Guide*. The tradeoff is partitioned capacity: idle workers in one fleet can't pick up another queue's jobs.
- **One farm, queues sharing fleets** – Sharing fleets gives you the simplest setup and the best worker utilization. Because the queues share worker hosts, treat all queues that share a fleet as one security boundary. On a customer-managed fleet you can also give each queue a distinct operating system user to separate job processes and files from each other.

Whichever arrangement you choose, keep resource sharing within a security boundary:

- Share an Amazon S3 bucket and root prefix for job attachments only between queues in the same security boundary. For more information, see the section called “Job attachment buckets”.
- Share an operating system user only between queues in the same security boundary. For more information, see Run jobs as dedicated OS users.
- Apply the same boundary to any other AWS resources that you integrate into the farm, such as shared file systems.

Run jobs as dedicated OS users

Each queue can specify an operating system (OS) user and its primary group, called the job run as user. The worker agent runs every job process submitted to the queue as that OS user, so the jobs inherit that user's OS permissions, including access to files and other resources on the worker host.

Give the queue user the least-privilege OS permissions that the queue's workloads need. A dedicated user for each queue keeps a job's files and processes separated from jobs in other queues, from other installed software, and from other users with access to the worker host.

The two fleet types handle the queue user differently:

- On a customer-managed fleet, you create the OS users on your worker hosts. Use a separate user for each queue unless the queues are in the same security boundary.
- On a service-managed fleet, the service manages the worker hosts, and all jobs run as a single OS user. The hosts are dedicated to your account and run only jobs from the queues that you

associate with the fleet, so you separate queues by giving them separate fleets instead of separate users.

When a queue doesn't specify a user, its jobs run as the worker agent user. The worker agent user can impersonate (`sudo`) any queue user, so a queue without a user can escalate privileges to any other queue's user and data.

On a customer-managed fleet, the worker agent enforces the boundary between queue users with operating system permissions. The agent creates each session's working directory so that only the agent user and the job user's primary group can access it. The agent also stores each queue's role credentials in a queue-specific directory with the same access restriction. When each queue has its own user and primary group, a job can't read another queue's session files or AWS credentials.

Standard operating system user permissions on a shared kernel separate queue users on a shared worker host. Give workloads that require stronger separation, such as virtual machine isolation, their own fleets.

Secure job attachment and software buckets

Each queue stores its job attachments in an Amazon S3 bucket that you own. Three pieces configure the connection between a queue and its bucket:

- Job attachments write to and read from a root prefix in the Amazon S3 bucket. You specify this root prefix in the `CreateQueue` API call.
- The bucket has a corresponding `Queue Role`, which specifies the role that grants queue users access to the bucket and root prefix. When creating a queue, you specify the `Queue Role` Amazon Resource Name (ARN) alongside the job attachments bucket and root prefix.
- Authorized calls to the `AssumeQueueRoleForRead`, `AssumeQueueRoleForUser`, and `AssumeQueueRoleForWorker` API operations return a set of temporary security credentials for the `Queue Role`.

Queues that share an Amazon S3 bucket and root prefix also share access to the job attachments stored there. Give each queue its own bucket and root prefix so that queue permission boundaries also apply to job attachments. For example, when QueueA and QueueB have separate buckets, an artist with access to only QueueA can't read QueueB's job attachments. If several queues belong to the same security boundary, such as the queues for one show, sharing a bucket between them is a

reasonable simplification. The console sets up each queue with its own bucket and root prefix by default.

Separate buckets keep job attachments separated in Amazon S3. On the worker host, a job runs code as the queue's operating system user. The job can write files to any location that user can write to, including locations that other queues' users can read. Separation on the host comes from the queue operating system users. For more information, see Run jobs as dedicated OS users.

To isolate your queues, you must configure the `Queue Role` to only allow queue access to the bucket and root prefix. In the following example, replace each *`placeholder`* with your resource-specific information.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::JOB_ATTACHMENTS_BUCKET_NAME",

 "arn:aws:s3:::JOB_ATTACHMENTS_BUCKET_NAME/JOB_ATTACHMENTS_ROOT_PREFIX/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "111122223333"
                }
            }
        },
        {
            "Action": [
                "logs:GetLogEvents"
            ],
            "Effect": "Allow",
```

```
            "Resource": "arn:aws:logs:us-east-1:111122223333:log-group:/aws/
deadline/FARM_ID/*"
        }
    ]
}
```

You must also set a trust policy on the role. In the following example, replace the *placeholder* text with your resource-specific information.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "deadline.amazonaws.com"
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:deadline:us-
east-1:111122223333:farm/FARM_ID"
                }
            }
        },
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "credentials.deadline.amazonaws.com"
            },
            "Condition": {
```

```
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:deadline:us-
east-1:111122223333:farm/FARM_ID"
                }
            }
        }
    ]
}
```

Custom software Amazon S3 buckets

You can add the following statement to your `Queue Role` to access custom software in your Amazon S3 bucket. In the following example, replace *SOFTWARE_BUCKET_NAME* with the name of your S3 bucket and *BUCKET_ACCOUNT_OWNER* with the AWS account ID that owns the bucket.

```
"Statement": [
    {
        "Action": [
            "s3:GetObject",
            "s3:ListBucket"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::SOFTWARE_BUCKET_NAME",
            "arn:aws:s3:::SOFTWARE_BUCKET_NAME/*"
        ],
        "Condition": {
         "StringEquals": {
            "aws:ResourceAccount": "BUCKET_ACCOUNT_OWNER"
         }
      }
    }
]
```

For more information about Amazon S3 security best practices, see Security best practices for Amazon S3 in the *Amazon Simple Storage Service User Guide*.

Secure worker hosts

On customer-managed fleets, you configure the worker host operating system. The following practices keep the worker agent, queue users, and their data separated from each other on the host:

- Review host configuration scripts before you use them, and test configuration changes before rolling them out to your fleet. An incorrect configuration can make workers unstable or stop them from working.
- Don't use the same `jobRunAsUser` value with multiple queues unless jobs submitted to those queues are within the same security boundary.
- Don't set the queue `jobRunAsUser` to the name of the OS user that the worker agent runs as. For the reason, see Run jobs as dedicated OS users.
- Grant queue users the least-privileged OS permissions required for the intended queue workloads. Ensure that they don't have filesystem write permissions to worker agent program files or other shared software.
- Ensure that only the root user on Linux, or the `Administrator` account on Windows, owns and can modify the worker agent program files.
- On Linux worker hosts, consider configuring a `umask` override in `/etc/sudoers` that allows the worker agent user to launch processes as queue users. This configuration helps ensure other users can't access files written to the queue.
- Restrict permissions to local DNS override configuration files (`/etc/hosts` on Linux and `C:\Windows\system32\etc\hosts` on Windows), and to route tables on worker host operating systems, so that worker traffic to Deadline Cloud can't be redirected.
- Regularly patch the operating system and all installed software. This approach includes software specifically used with Deadline Cloud such as submitters, adaptors, worker agents, OpenJD packages, and others.
- Use strong passwords for the Windows queue `jobRunAsUser`, and rotate the passwords regularly.
- Grant least-privileged access to the Windows password secrets, and delete unused secrets.
- Don't give the queue `jobRunAsUser` permission to schedule commands to run in the future:
  - On Linux, deny these accounts access to `cron` and `at`.
  - On Windows, deny these accounts access to the Windows task scheduler.

Secure artist workstations

Workstations with Deadline Cloud submitters installed hold credentials that can submit jobs to your farm, and jobs submitted from them run on your fleets at your expense. The following practices protect those credentials and the submission path:

- Prefer temporary credentials for workstation access to Deadline Cloud. Users who sign in through the Deadline Cloud monitor receive temporary credentials automatically. If a workstation uses an AWS profile with long-lived IAM access keys, such as for the Deadline Cloud CLI, secure those keys. For more information, see Managing access keys for IAM users in the *IAM User Guide*.
- When you install the Deadline Cloud submitters and adaptors, download them through the Deadline Cloud monitor and verify the installers before running them. For more information, see Verify the authenticity of downloaded software. If you build your own submitters, see Build a custom submitter for Deadline Cloud in the *Deadline Cloud Developer Guide*.
- Use secure permissions on Deadline Cloud submitter program files to prevent tampering.
- Restrict permissions to local DNS override configuration files (`/etc/hosts` and `/etc/resolv.conf` on Linux and macOS, and `C:\Windows\system32\etc\hosts` on Windows), and to route tables, so that workstation traffic to Deadline Cloud can't be redirected.
- Regularly patch the operating system and the software used with Deadline Cloud, including submitters, adaptors, and OpenJD packages.

Restrict farm access to your network

Every request to your farm goes through the Deadline Cloud APIs. The monitor, the AWS CLI, the submitters, and the AWS Management Console all call the APIs with temporary credentials, so the network boundary belongs on the API calls. The monitor URL is reachable from any location on the internet and holds no farm data. For more information, see Access to the monitor web application.

Add a network condition to the IAM policies of the credentials that your people use:

- The monitor gets its credentials from the monitor role. So do the AWS CLI and the submitters when they use the profile that the Deadline Cloud monitor creates. Add a custom policy to the monitor role. For more information, see Adding permissions for advanced workflows.
- The AWS Management Console and pipeline scripts use their own IAM roles or users. Add the condition to those policies, or to the permission sets that you assign for AWS Management Console access in IAM Identity Center.

Allow only your IP ranges

An `aws:SourceIp` condition limits requests to the public addresses that you list. List the addresses that your studio sends traffic from. Include your office ranges, your VPN exit addresses, and the NAT gateway addresses of any VPC where your workstations run.

The following policy denies Deadline Cloud requests from any other address. Attach it to the monitor role, alongside the managed policies that the role already has.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "deadline:*",
      "Resource": "*",
      "Condition": {
        "NotIpAddress": {
          "aws:SourceIp": [
            "203.0.113.0/24",
            "198.51.100.10/32"
          ]
        },
        "BoolIfExists": {
          "aws:ViaAWSService": "false"
        }
      }
    }
  ]
}
```

The policy covers your files as well as your farm. Job attachments and job logs need queue role credentials. The monitor, the AWS CLI, and the submitters get those credentials by calling `AssumeQueueRoleForUser`. The deny blocks the call from any address that you didn't list.

If you sign in from an address that you didn't list, you still reach the sign-in page. The monitor then reports errors instead of showing farm data. An allow can't override a deny, so verify your ranges before you attach the policy.

Important

The preceding policy also denies requests that arrive through an Deadline Cloud interface endpoint. Endpoint traffic carries `aws:VpcSourceIp` instead of `aws:SourceIp`, and a `NotIpAddress` condition on an absent key matches, so the deny applies. If your people reach Deadline Cloud through an interface endpoint, use the VPC form in the next section. To allow both paths, put both keys in the same condition block. The deny then applies only when neither key matches. Test the policy from both networks before you roll it out.

For more information about the keys, see AWS global condition context keys in the *IAM User Guide*.

Allow access only from your VPC

To require that people work from hosts in a VPC, such as virtual workstations, combine three pieces:

1. Run the monitor, the AWS CLI, and the submitters on hosts in the VPC.
2. Create an interface endpoint in the VPC for the Deadline Cloud management endpoint. Add Amazon S3 and CloudWatch Logs endpoints if your people upload job attachments and read job logs. Enable private DNS so that the client tools use the endpoints without extra configuration. For more information, see Access AWS Deadline Cloud using an interface endpoint (AWS PrivateLink).
3. Attach a policy to the monitor role that denies requests from outside the VPC. The VPC keys hold strings, so the condition operator is `StringNotEquals` rather than `NotIpAddress`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "deadline:*",
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:SourceVpc": "vpc-0123456789abcdef0"
        },
        "BoolIfExists": {
          "aws:ViaAWSService": "false"
```

```
        }
      }
    }
  ]
}
```

To name individual endpoints instead of the whole VPC, use `aws:SourceVpce` with your endpoint IDs. To name address ranges inside the VPC, use `aws:VpcSourceIp` with `NotIpAddress`.

The monitor web application, the IAM Identity Center portal, and AWS Sign-In have no interface endpoints. The browser on the host therefore needs outbound access to the sign-in domains listed in Restricted network environments. You can route that traffic through a NAT gateway or a web proxy that allows only those domains. Farm data doesn't travel over the sign-in path.

Keep worker access working

Your workers share the queue role with your people. A worker uses queue role credentials to read inputs and write outputs while it runs a job. The monitor and the AWS CLI use the same role when someone uploads inputs or downloads outputs. Restrict the monitor role, and leave the queue role and the fleet role without a network condition.

Service-managed fleet workers run on the AWS network, so their addresses aren't in your ranges and their requests don't come through your endpoints. A network condition on the queue role, the fleet role, or the bucket policy of the job attachments bucket denies those requests. Jobs then fail. A bucket policy condition works only if all of your fleets are customer-managed and run in your VPC. In that case, list both your workstation endpoint and your fleet endpoint in an `aws:SourceVpce` condition.

Queue role credentials carry no network condition of their own. Someone who obtains them from an address that you allow can use them from anywhere until they expire. The condition on the monitor role controls who gets credentials, rather than every request that uses them.

Verify the authenticity of downloaded software

Verify your software's authenticity after downloading the installer to protect against file tampering. This procedure works for both Windows and Linux systems.

Windows

To verify the authenticity of your downloaded files, complete the following steps.

1. In the following command, replace *file* with the file that you want to verify. For example, ***C:\PATH\TO\MY*DeadlineCloudSubmitter-windows-x64-installer.exe** . Also, replace *signtool-sdk-version* with the version of the SignTool SDK installed. For example, **10.0.22000.0**.

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin\signtool-sdk-version\x86\signtool.exe" verify /v file
   ```

2. For example, you can verify the Deadline Cloud submitter installer file by running the following command:

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin
   \10.0.22000.0\x86\signtool.exe" verify /v DeadlineCloudSubmitter-windows-x64-installer.exe
   ```

3. Confirm that the output contains `Successfully verified: DeadlineCloudSubmitter-windows-x64-installer.exe`. The successful verification means that you can run the installer.

Linux

To verify the authenticity of your downloaded files, use the gpg command line tool.

1. Import the OpenPGP key by running the following command:

```
 gpg --import --armor <<EOF
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGlANDUBEACg6zffjN43gqe5ryPhk+wQM10rEdvmItw4WPWaVsN+/at/OIJw
MGCagSYXcgR+jKbsHQOQoEQdo5SrxxHjpKTEs3KQhGvf+ehrU1Ac7koXKIBWtes+
BI9F0slRECz0nXTOy/cd/90RXjpF07mreTLIKNIbybULfad82nYykpITjFr5XRGj
/shYkucxRQZdwkgkIYyV25pPICPd2RsX+Zua85jV8mCqVffDfRXvgcPe3+ofClj/
2CE8UfUIqO8Csua4YEkSqr3aeoTOEFT4kuQR5nFXVzorOEkQtO3gB35KNWKMlIOU
2vA+wyoL7nWSii4yfYtW3EZ+3gq6HxvnT9Zs8MC53uTOiOdamASXecYREwGmY/io
6n5XTEA/35LNbl4A756vSTZ7h4VFJAN5BpuqxstI1D7ou94skoSmcPoC/iniTvY9
kZylU5OCH/nifMAHM2a5jrQel80cW4oko9eyc8ENQpSy15JElFOKFf7D/4tcZJLF
F0VBTXbhfvq3dPfoq94IWt7p54Ovwj0S//CEu3jZYbNl2QC/3YiHE2H2XyGCQbq6
2MjcuxLnEapoRIqfbi8GPtCWVPzm28WGyKIDofWICczzeJFFJnvzrY3wRG64ibKJ
bR/uedwua1UuiC482V1FD5ffmzSSs8ktTp9hgj7RGDXlc9NTcF1jHxG9hwARAQAB
tCxBV1MgRGVhZGxpbmUgQ2xvdWQgPGF3cy1kZWFkbGluZUBhbWF6b24uY29tPokC
VwQTAQgAQRYhBJmXd7So2csyehiIYsg71N18bhtjBQJpQDQ1AhsvBQkDwmcABQsJ
CAcCAiICBhUKCQgLAgQWAgMBAh4HAheAAAoJEMg71N18bhtjk2UP/3h4KlEzZO/7
```

```
BxRmkbixuo1QuqOGvA6tXbSWaM8QH5jglcvL12PZLALklLT4v82uCsLR1lF8/Tch
cCl0SZEOFIS+XxAaw1Xfai6jlyLhabOwKF2ylq5eJlLcw1lh2nAArDRb4fLD0m1g
Dfqetq/XEpyXpOSkWxGRV4RlUdjQfytxrmcUnsT5/fk5f9VDdblu6K/lEmwfyYjB
lXv0uUCkqPot0SmbvOh3PY3Hi3n54ncy8NfTeV+TUvSe3C1s1zNl8aqHoTxJB/eU
kp+LFZ9m+igpSYnKeglKnytylH3KGCjTHglT/QXnI1wNTqmj1kFBVwtt/y1mtnA+
CPIUHP1CtbKsHaLtpp4llBm5TVtPN/Wqqicn5QLl4khg7R4K+V2aaA4ubY6p1tG9
0fFhN5tTnHDSKWMfmb83wfh5Zkcg85c3egjoit+wgGQRAQVqbznx7NqAHs9VoDIu
SPcAr+C329AOBzod4gyNGH7Ah5DkMITo4O4+axnAU9yhFOHcMJmTIask/fNg1Aum
OqYPMUwcgv1GZjLaTJyfGGC1xALsYR0KHnwIehD06MHR/Z98bGkcV8+Y0q8UPsd1
VN1fc1rjCJh/AT3w6owvG4DaEwspseSjzHv16mW4e2N6Uu23SPzgQsJ5qYN2g8D+
P7N9LGDfP8DaYc5JM9mlyFmYI2Q94ufL
=rY5l
-----END PGP PUBLIC KEY BLOCK-----
EOF
```

2. Determine whether to trust the OpenPGP key. When deciding, consider that AWS has taken measures to secure the hosting of the OpenPGP public key on this page, and that you obtained the key from the official Deadline Cloud documentation.
3. If you decide to trust the OpenPGP key, edit the key to trust with gpg similar to the following example:

```
$ gpg --edit-key 0xC83BD4DD7C6E1B63

    gpg (GnuPG) 2.3.7; Copyright (C) 2021 Free Software Foundation, Inc.
    This is free software: you are free to change and redistribute it.
    There is NO WARRANTY, to the extent permitted by law.


    pub  rsa4096/C83BD4DD7C6E1B63  created: 2025-12-15  expires: 2027-12-15
 usage: SCEA
                         trust: unknown       validity: unknown
    [ unknown] (1). AWS Deadline Cloud <aws-deadline@amazon.com>

    gpg> trust
    pub  rsa4096/C83BD4DD7C6E1B63  created: 2025-12-15  expires: 2027-12-15
 usage: SCEA
                         trust: unknown       validity: unknown
    [ unknown] (1). AWS Deadline Cloud <aws-deadline@amazon.com>

    Please decide how far you trust this user to correctly verify other users'
 keys
    (by looking at passports, checking fingerprints from different sources,
 etc.)
```

```
      1 = I don't know or won't say
      2 = I do NOT trust
      3 = I trust marginally
      4 = I trust fully
      5 = I trust ultimately
      m = back to the main menu

   Your decision? 5
   Do you really want to set this key to ultimate trust? (y/N) y

   pub  rsa4096/C83BD4DD7C6E1B63  created: 2025-12-15  expires: 2027-12-15
usage: SCEA
                        trust: ultimate      validity: unknown
   [ unknown] (1). AWS Deadline Cloud <aws-deadline@amazon.com>
   Please note that the shown key validity is not necessarily correct
   unless you restart the program.

   gpg> quit
```

4. **Verify the Deadline Cloud submitter installer**

   To verify the Deadline Cloud submitter installer, complete the following steps:

   a. Download the signature file for the Deadline Cloud submitter installer.

      Download signature file (.sig)

   b. Verify the signature of the Deadline Cloud submitter installer by running:

      ```
      gpg --verify ./DeadlineCloudSubmitter-linux-x64-installer.run.sig ./
      DeadlineCloudSubmitter-linux-x64-installer.run
      ```

5. **Verify the Deadline Cloud monitor**

   > **Note**
   >
   > You can verify the Deadline Cloud monitor download using signature files or platform specific methods. For platform specific methods, see the Linux (Debian) tab, the Linux (RPM) tab, or the Linux (AppImage) tab based on your downloaded file type.

To verify the Deadline Cloud monitor desktop application with signature files, complete the following steps:

a. Download the corresponding signature file for your Deadline Cloud monitor installer:

- Download .deb signature file
- Download .rpm signature file
- Download .AppImage signature file

b. Verify the signature:

For .deb:

```
gpg --verify ./deadline-cloud-monitor_amd64.deb.sig ./deadline-cloud-
monitor_amd64.deb
```

For .rpm:

```
gpg --verify ./deadline-cloud-monitor.x86_64.rpm.sig ./deadline-cloud-
monitor.x86_64.rpm
```

For .AppImage:

```
gpg --verify ./deadline-cloud-monitor_amd64.AppImage.sig ./deadline-cloud-
monitor_amd64.AppImage
```

c. Confirm that the output looks similar to the following:

```
gpg: Signature made Mon Jan 5 21:10:14 2026 UTC

gpg: using RSA key C83BD4DD7C6E1B63
```

If the output contains the phrase `Good signature from "AWS Deadline Cloud"`, it means that the signature has successfully been verified and you can run the Deadline Cloud monitor installation script.

Historical keys

If you downloaded an installer that was signed before the current key was issued, import the following historical key and repeat the verification steps.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGX6GQsBEADduUtJgqSXI+q76O6fsFwEYKmbnlyL0xKvlq32EZuyv0otZo5L
le4m5Gg52AzrvPvDiUTLooAlvYeozaYyirIGsK08Ydz0Ftdjroiuh/mw9JSJDJRI
rnRn5yKet1JFezkjopA3pjsTBP6lW/mb1bDBDEwwwtH0x9lV7A03FJ9T7Uzu/qSh
qO/UYdkafro3cPASvkqgDt2tCvURfBcUCAjZVFcLZcVD5iwXacxvKsxxS/e7kuVV
I1+VGT8Hj8XzWYhjCZxOLZk/fvpYPMyEEujN0fYUp6RtMIXve0C9awwMCy5nBG2J
eE2Ol5DsCpTaBd4Fdr3LWcSs8JFA/YfP9auL3NczOozPoVJt+fw8CBlVIXO0J7l5
hvHDjcC+5v0wxqAlMG6+f/SX7CT8FXK+L3iOJ5gBYUNXqHSxUdv8kt76/KVmQa1B
Akl+MPKpMq+lhw++S3G/lXqwWaDNQbRRw7dSZHymQVXvPp1nsqc3hV7KlOM+6s6g
1g4mvFY4lf6DhptwZLWyQXU8rBQpojvQfiSmDFrFPWFi5BexesuVnkGIolQoklKx
AVUSdJPVEJCteyy7td4FPhBaSqT5vW3+ANbr9b/uoRYWJvn17dN0cc9HuRh/Ai+I
nkfECo2WUDLZ0fEKGjGyFX+todWvJXjvc5kmE9Ty5vJp+M9Vvb8jd6t+mwARAQAB
tCxBV1MgRGVhZGxpbmUgQ2xvdWQgPGF3cy1kZWFkbGluZUBhbWF6b24uY29tPokC
VwQTAQgAQRYhBLhAwIwpqQeWoHH6pfbNPOa3bzzvBQJl+hkLAxsvBAUJA8JnAAUL
CQgHAgIiAgYVCgkICwIDFgIBAh4HAheAAAoJEPbNPOa3bzzvKswQAJXzKSAY8sY8
F6Eas2oYwIDDdDurs8FiEnFghjUEO6MTt9AykF/jw+CQg2UzFtEyObHBymhgmhXE
3buVeom96tgM3ZDfZu+sxi5pGX6oAQnZ6riztN+VpkpQmLgwtMGpSML13KLwnv2k
WK8mrR/fPMkfdaewB7A6RIUYiW33GAL4KfMIs8/vIwIJw99NxHpZQVoU6dFpuDtE
1OuxGcCqGJ7mAmo6H/YawSNp2Ns80gyqIKYo7o3LJ+WRroIRlQyctq8gnR9JvYXX
42ASqLq5+OXKo4qh81blXKYqtc176BbbSNFjWnzIQgKDgNiHFZCdcOVgqDhwO15r
NICbqqwwNLj/Fr2kecYx180Ktpl0jOOw5IOyh3bf3MVGWnYRdjvA1v+/CO+55N4g
z0kf50Lcdu5RtqV10XBCifn28pecqPaSdYcssYSRl5DLiFktGbNzTGcZZwITTKQc
af8PPdTGtnnb6P+cdbW3bt9MVtN5/dgSHLThnS8MPEuNCtkTnpXshuVuBGgwBMdb
qUC+HjqvhZzbwns8dr5WI+6HWNBFgGANn6ageYl58vVp0UkuNP8wcWjRARciHXZx
ku6W2jPTHDWGNrBQO2Fx7fd2QYJheIPPAShHcfJO+xgWCof45D0vAxAJ8gGg9Eq+
gFWhsx4NSHn2gh1gDZ41Ou/4exJ1lwPM
=uVaX
-----END PGP PUBLIC KEY BLOCK-----
```

Linux (AppImage)

To verify packages that use a Linux .AppImage binary, first complete steps 1-3 in the Linux tab, then complete the following steps.

1. From the AppImageUpdate releases page on the GitHub website, download the **validate-x86_64.AppImage** file.
2. After downloading the file, to add execute permissions, run the following command.

```
chmod a+x ./validate-x86_64.AppImage
```

3. To add execute permissions, run the following command.

```
chmod a+x ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
```

4. To verify the Deadline Cloud monitor signature, run the following command.

```
./validate-x86_64.AppImage ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
```

   If the output contains the phrase `Validation successful`, it means that the signature has successfully been verified and you can safely run the Deadline Cloud monitor installation script.

Linux (Debian)

To verify packages that use a Linux .deb binary, first complete steps 1-3 in the Linux tab.

dpkg is the core package management tool in most debian based Linux distributions. You can verify the .deb file with the tool.

1. Download the Deadline Cloud monitor .deb file:

   Download Deadline Cloud monitor (.deb)

2. Verify the .deb file:

```
dpkg-sig --verify deadline-cloud-monitor_amd64.deb
```

3. The output will be similar to:

```
Processing deadline-cloud-monitor_amd64.deb...
GOODSIG _gpgbuilder C83BD4DD7C6E1B63 1765815349
```

4. To verify the .deb file, confirm that `GOODSIG` is present in the output.

Linux (RPM)

To verify packages that use a Linux .rpm binary, first complete steps 1-3 in the Linux tab.

1. Download the Deadline Cloud monitor .rpm file:

   Download Deadline Cloud monitor (.rpm)

2. Verify the .rpm file:

```
gpg --export --armor "Deadline Cloud" > key.pub
sudo rpm --import key.pub
rpm -K deadline-cloud-monitor.x86_64.rpm
```

3. The output will be similar to:

```
deadline-cloud-monitor.x86_64.rpm: digests signatures OK
```

4. To verify the .rpm file, confirm that `digests signatures` OK is in the output.

Identity and Access Management in Deadline Cloud

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Deadline Cloud resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting AWS Deadline Cloud identity and access)
- **Service administrator** - determine user access and submit permission requests (see How Deadline Cloud works with IAM)
- **IAM administrator** - write policies to manage access (see Identity-based policy examples for Deadline Cloud)

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. You must specify a principal in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How Deadline Cloud works with IAM

Before you use IAM to manage access to Deadline Cloud, learn what IAM features are available to use with Deadline Cloud.

IAM features you can use with AWS Deadline Cloud

IAM feature	Deadline Cloud support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Yes
Temporary credentials	Yes
Forward access sessions (FAS)	Yes
Service roles	Yes
Service-linked roles	No

To get a high-level view of how Deadline Cloud and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for Deadline Cloud

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can

perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for Deadline Cloud

To view examples of Deadline Cloud identity-based policies, see Identity-based policy examples for Deadline Cloud.

Resource-based policies within Deadline Cloud

Supports resource-based policies: No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for Deadline Cloud

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Deadline Cloud actions, see Actions defined by AWS Deadline Cloud in the *Service Authorization Reference*.

Policy actions in Deadline Cloud use the following prefix before the action:

```
deadline
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "deadline:action1",
      "deadline:action2"
         ]
```

To view examples of Deadline Cloud identity-based policies, see Identity-based policy examples for Deadline Cloud.

Policy resources for Deadline Cloud

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. As a best practice, specify a resource using its Amazon Resource Name (ARN). For actions that don't support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of Deadline Cloud resource types and their ARNs, see Resources defined by AWS Deadline Cloud in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by AWS Deadline Cloud.

To view examples of Deadline Cloud identity-based policies, see Identity-based policy examples for Deadline Cloud.

Policy condition keys for Deadline Cloud

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element specifies when statements execute based on defined criteria. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of Deadline Cloud condition keys, see Condition keys for AWS Deadline Cloud in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by AWS Deadline Cloud.

To view examples of Deadline Cloud identity-based policies, see Identity-based policy examples for Deadline Cloud.

ACLs in Deadline Cloud

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with Deadline Cloud

Supports ABAC (tags in policies): Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes called tags. You can attach tags to IAM entities and AWS resources, then design ABAC policies to allow operations when the principal's tag matches the tag on the resource.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with Deadline Cloud

Supports temporary credentials: Yes

Temporary credentials provide short-term access to AWS resources and are automatically created when you use federation or switch roles. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM and AWS services that work with IAM in the *IAM User Guide*.

Forward access sessions for Deadline Cloud

Supports forward access sessions (FAS): Yes

Forward access sessions (FAS) use the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. For policy details when making FAS requests, see Forward access sessions.

Service roles for Deadline Cloud

Supports service roles: Yes

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break Deadline Cloud functionality. Edit service roles only when Deadline Cloud provides guidance to do so.

Service-linked roles for Deadline Cloud

Supports service-linked roles: No

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a Yes in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for Deadline Cloud

By default, users and roles don't have permission to create or modify Deadline Cloud resources. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by Deadline Cloud, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for AWS Deadline Cloud in the *Service Authorization Reference*.

Topics

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Deadline Cloud resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies

that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.

- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the Deadline Cloud console

To access the AWS Deadline Cloud console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Deadline Cloud resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To grant full access to the Deadline Cloud console, attach the policy in the section called "Policy to access the console". To grant users access to farm, fleet, queue, and job data based on their farm memberships and access levels, attach the AWS managed policies described in AWS managed policies for Deadline Cloud. For more information about attaching policies, see Adding permissions to a user in the *IAM User Guide*.

Policy to access the console

To grant access to all functionality in the Deadline Cloud console, attach this identity policy to a user or role you want to have full access.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "EC2InstanceTypeSelection",
        "Effect": "Allow",
        "Action": [
            "ec2:DescribeInstanceTypeOfferings",
            "ec2:DescribeInstanceTypes",
            "ec2:GetInstanceTypesFromInstanceRequirements",
            "pricing:GetProducts"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "VPCResourceSelection",
        "Effect": "Allow",
        "Action": [
            "ec2:DescribeVpcs",
            "ec2:DescribeSubnets",
            "ec2:DescribeSecurityGroups"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "ViewVpcLatticeResources",
        "Effect": "Allow",
        "Action": [
            "vpc-lattice:ListResourceConfigurations",
            "vpc-lattice:GetResourceConfiguration",
```

```
            "vpc-lattice:GetResourceGateway"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "ManageVpcEndpointsViaDeadline",
        "Effect": "Allow",
        "Action": [
            "ec2:CreateVpcEndpoint",
            "ec2:DescribeVpcEndpoints",
            "ec2:DeleteVpcEndpoints",
        "ec2:CreateTags"
        ],
        "Resource": ["*"],
        "Condition": {
        "StringEquals": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "ChooseJobAttachmentsBucket",
        "Effect": "Allow",
        "Action": ["s3:GetBucketLocation", "s3:ListAllMyBuckets"],
        "Resource": "*"
    },
    {
        "Sid": "CreateDeadlineCloudLogGroups",
        "Effect": "Allow",
        "Action": ["logs:CreateLogGroup"],
        "Resource": "arn:aws:logs:*:*:log-group:/aws/deadline/*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "ValidateDependencies",
        "Effect": "Allow",
        "Action": ["s3:ListBucket"],
        "Resource": "*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "RoleSelection",
```

```
        "Effect": "Allow",
        "Action": ["iam:GetRole", "iam:ListRoles",
"iam:ListAttachedRolePolicies"],
        "Resource": "*"
    },
    {
        "Sid": "PassRoleToDeadlineCloud",
        "Effect": "Allow",
        "Action": ["iam:PassRole"],
        "Condition": {
        "StringLike": { "iam:PassedToService": "deadline.amazonaws.com" }
    },
        "Resource": "*"
    },
    {
        "Sid": "KMSKeySelection",
        "Effect": "Allow",
        "Action": ["kms:ListKeys", "kms:ListAliases"],
        "Resource": "*"
    },
    {
        "Sid": "IdentityStoreReadOnly",
        "Effect": "Allow",
        "Action": [
            "identitystore:DescribeUser",
            "identitystore:DescribeGroup",
            "identitystore:ListGroups",
            "identitystore:ListUsers",
            "identitystore:IsMemberInGroups",
            "identitystore:ListGroupMemberships",
            "identitystore:ListGroupMembershipsForMember",
            "identitystore:GetGroupMembershipId"
    ],
        "Resource": "*"
    },
    {
        "Sid": "OrganizationAndIdentityCenterIdentification",
        "Effect": "Allow",
        "Action": [
            "sso:ListDirectoryAssociations",
            "organizations:DescribeAccount",
            "organizations:DescribeOrganization",
            "sso:DescribeRegisteredRegions",
            "sso:GetManagedApplicationInstance",
```

```
            "sso:GetSharedSsoConfiguration",
            "sso:ListInstances",
            "sso:GetApplicationAssignmentConfiguration",
            "sso:GetSSOStatus",
            "sso:ListRegions",
            "sso:DescribeRegion"
    ],
        "Resource": "*"
    },
    {
        "Sid": "ManagedDeadlineCloudIDCApplication",
        "Effect": "Allow",
        "Action": [
            "sso:CreateApplication",
            "sso:PutApplicationAssignmentConfiguration",
            "sso:PutApplicationAuthenticationMethod",
            "sso:PutApplicationGrant",
            "sso:DeleteApplication",
            "sso:UpdateApplication"
    ],
        "Resource": "*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "ChooseSecret",
        "Effect": "Allow",
        "Action": ["secretsmanager:ListSecrets"],
        "Resource": "*"
    },
    {
        "Sid": "DeadlineMembershipActions",
        "Effect": "Allow",
        "Action": [
            "deadline:AssociateMemberToFarm",
            "deadline:AssociateMemberToFleet",
            "deadline:AssociateMemberToQueue",
            "deadline:AssociateMemberToJob",
            "deadline:DisassociateMemberFromFarm",
            "deadline:DisassociateMemberFromFleet",
            "deadline:DisassociateMemberFromQueue",
            "deadline:DisassociateMemberFromJob",
            "deadline:ListFarmMembers",
```

```
            "deadline:ListFleetMembers",
            "deadline:ListQueueMembers",
            "deadline:ListJobMembers"
    ],
        "Resource": ["*"]
    },
    {
        "Sid": "DeadlineControlPlaneActions",
        "Effect": "Allow",
        "Action": [
            "deadline:CreateMonitor",
            "deadline:GetMonitor",
            "deadline:UpdateMonitor",
            "deadline:DeleteMonitor",
            "deadline:ListMonitors",
            "deadline:CreateFarm",
            "deadline:GetFarm",
            "deadline:UpdateFarm",
            "deadline:DeleteFarm",
            "deadline:ListFarms",
            "deadline:CreateQueue",
            "deadline:GetQueue",
            "deadline:UpdateQueue",
            "deadline:DeleteQueue",
            "deadline:ListQueues",
            "deadline:CreateFleet",
            "deadline:GetFleet",
            "deadline:UpdateFleet",
            "deadline:DeleteFleet",
            "deadline:ListFleets",
            "deadline:ListWorkers",
            "deadline:CreateQueueFleetAssociation",
            "deadline:GetQueueFleetAssociation",
            "deadline:UpdateQueueFleetAssociation",
            "deadline:DeleteQueueFleetAssociation",
            "deadline:ListQueueFleetAssociations",
            "deadline:CreateQueueEnvironment",
            "deadline:GetQueueEnvironment",
            "deadline:UpdateQueueEnvironment",
            "deadline:DeleteQueueEnvironment",
            "deadline:ListQueueEnvironments",
            "deadline:CreateLimit",
            "deadline:GetLimit",
            "deadline:UpdateLimit",
```

```
            "deadline:DeleteLimit",
            "deadline:ListLimits",
            "deadline:CreateQueueLimitAssociation",
            "deadline:GetQueueLimitAssociation",
            "deadline:DeleteQueueLimitAssociation",
            "deadline:UpdateQueueLimitAssociation",
            "deadline:ListQueueLimitAssociations",
            "deadline:CreateStorageProfile",
            "deadline:GetStorageProfile",
            "deadline:UpdateStorageProfile",
            "deadline:DeleteStorageProfile",
            "deadline:ListStorageProfiles",
            "deadline:ListStorageProfilesForQueue",
            "deadline:ListBudgets",
            "deadline:TagResource",
            "deadline:UntagResource",
            "deadline:ListTagsForResource",
            "deadline:CreateLicenseEndpoint",
            "deadline:GetLicenseEndpoint",
            "deadline:DeleteLicenseEndpoint",
            "deadline:ListLicenseEndpoints",
            "deadline:ListAvailableMeteredProducts",
            "deadline:ListMeteredProducts",
            "deadline:PutMeteredProduct",
            "deadline:DeleteMeteredProduct",
            "deadline:GetMonitorSettings",
            "deadline:UpdateMonitorSettings",
            "deadline:GetVolume",
            "deadline:ListVolumes",
            "deadline:DeleteVolume"
        ],
        "Resource": ["*"]
      }]
}
```

Policy to submit jobs to a queue

In this example, you create a scoped-down policy that grants permission to submit jobs to a specific queue in a specific farm.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SubmitJobsFarmAndQueue",
            "Effect": "Allow",
            "Action": "deadline:CreateJob",
            "Resource": "arn:aws:deadline:us-east-1:111122223333:farm/FARM_A/
queue/QUEUE_B/job/*"
        }
    ]
}
```

Policy to allow creating a license endpoint

In this example, you create a scoped-down policy that grants the required permissions to create and manage license endpoints. Use this policy to create the license endpoint for the VPC associated with your farm.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "CreateLicenseEndpoint",
        "Effect": "Allow",
        "Action": [
            "deadline:CreateLicenseEndpoint",
            "deadline:DeleteLicenseEndpoint",
            "deadline:GetLicenseEndpoint",
            "deadline:ListLicenseEndpoints",
            "deadline:PutMeteredProduct",
            "deadline:DeleteMeteredProduct",
            "deadline:ListMeteredProducts",
            "deadline:ListAvailableMeteredProducts",
            "ec2:CreateVpcEndpoint",
            "ec2:DescribeVpcEndpoints",
            "ec2:DeleteVpcEndpoints"
```

```
        ],
        "Resource": [
            "arn:aws:deadline:*:111122223333:*",
            "arn:aws:ec2:*:111122223333:vpc-endpoint/*"
        ]
    }]
}
```

Policy to allow monitoring a specific farm queue

In this example, you create a scoped-down policy that grants permission to monitor jobs in a specific queue for a specific farm.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "MonitorJobsFarmAndQueue",
        "Effect": "Allow",
        "Action": [
            "deadline:SearchJobs",
            "deadline:ListJobs",
            "deadline:GetJob",
            "deadline:SearchSteps",
            "deadline:ListSteps",
            "deadline:ListStepConsumers",
            "deadline:ListStepDependencies",
            "deadline:GetStep",
            "deadline:SearchTasks",
            "deadline:ListTasks",
            "deadline:GetTask",
            "deadline:ListSessions",
            "deadline:GetSession",
            "deadline:ListSessionActions",
            "deadline:GetSessionAction"
        ],
        "Resource": [
            "arn:aws:deadline:us-east-1:123456789012:farm/FARM_A/queue/QUEUE_B",
            "arn:aws:deadline:us-east-1:123456789012:farm/FARM_A/queue/QUEUE_B/*"
        ]
```

```
    }]
}
```

Policy to manage queue–fleet associations for a specific fleet

The queue–fleet associations on a fleet determine which jobs can be scheduled to the fleet's workers. If you reserve a fleet for sensitive content, control the associations so that no one can attach the fleet to an unapproved queue.

In this example, you create a scoped-down policy that grants permission to manage queue–fleet associations only between a specific fleet and a specific queue. The `CreateQueueFleetAssociation` operation authorizes against the farm, queue, and fleet resources, so a principal with this policy cannot associate the fleet with any other queue. In the following example, replace each *`placeholder`* with your resource-specific information.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "deadline:CreateQueueFleetAssociation",
                "deadline:UpdateQueueFleetAssociation",
                "deadline:DeleteQueueFleetAssociation",
                "deadline:GetQueueFleetAssociation",
                "deadline:ListQueueFleetAssociations"
            ],
            "Resource": [
                "arn:aws:deadline:REGION:ACCOUNT_ID:farm/FARM_ID",
                "arn:aws:deadline:REGION:ACCOUNT_ID:farm/FARM_ID/queue/QUEUE_ID",
                "arn:aws:deadline:REGION:ACCOUNT_ID:farm/FARM_ID/fleet/FLEET_ID"
            ]
        }
    ]
}
```

The scoping is effective only if no other policy grants the association operations on the restricted fleet. Audit the policies attached to your farm administrators, and for a stronger guarantee add an explicit Deny statement for the restricted fleet's Amazon Resource Name (ARN) to the principals

that must not change its associations. For more information about the security boundaries this protects, see the section called “Isolate workloads”.

AWS managed policies for Deadline Cloud

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AWSDeadlineCloud-FleetWorker

You can attach the `AWSDeadlineCloud-FleetWorker` policy to your AWS Identity and Access Management (IAM) identities.

This policy grants workers in this fleet the permissions that are needed to connect to and receive tasks from the service.

Permissions details

This policy includes the following permissions:

- `deadline` – Allows principals to manage workers in a fleet.

For a JSON listing of the policy details, see AWSDeadlineCloud-FleetWorker in the *AWS Managed Policy reference guide*.

AWS managed policy: AWSDeadlineCloud-WorkerHost

You can attach the `AWSDeadlineCloud-WorkerHost` policy to your IAM identities.

This policy grants the permissions that are needed to initially connect to the service. It can be used as an Amazon Elastic Compute Cloud (Amazon EC2) instance profile.

Permissions details

This policy includes the following permissions:

- `deadline` – Allows the user to create workers, assume the fleet role for workers, and apply tags to workers

For a JSON listing of the policy details, see AWSDeadlineCloud-WorkerHost in the *AWS Managed Policy reference guide*.

AWS managed policy: AWSDeadlineCloud-UserAccessFarms

You can attach the `AWSDeadlineCloud-UserAccessFarms` policy to your IAM identities.

This policy allows users to access farm data based on the farms that they are members of and their membership level.

Permissions details

This policy includes the following permissions:

- `deadline` – Allows the user to access farm data.
- `ec2` – Allows users to see details about Amazon EC2 instance types.
- `identitystore` – Allows users to see user and group names.
- `kms` – Allows users to configure AWS Key Management Service (AWS KMS) customer-managed keys for their AWS IAM Identity Center (IAM Identity Center) instance.

For a JSON listing of the policy details, see AWSDeadlineCloud-UserAccessFarms in the *AWS Managed Policy reference guide*.

AWS managed policy: AWSDeadlineCloud-UserAccessFleets

You can attach the `AWSDeadlineCloud-UserAccessFleets` policy to your IAM identities.

This policy allows users to access fleet data based on the farms that they are members of and their membership level.

Permissions details

This policy includes the following permissions:

- `deadline` – Allows the user to access farm data.
- `ec2` – Allows users to see details about Amazon EC2 instance types.
- `identitystore` – Allows users to see user and group names.

For a JSON listing of the policy details, see AWSDeadlineCloud-UserAccessFleets in the *AWS Managed Policy reference guide*.

AWS managed policy: AWSDeadlineCloud-UserAccessJobs

You can attach the `AWSDeadlineCloud-UserAccessJobs` policy to your IAM identities.

This policy allows users to access job data based on the farms that they are members of and their membership level.

Permissions details

This policy includes the following permissions:

- `deadline` – Allows the user to access farm data.
- `ec2` – Allows users to see details about Amazon EC2 instance types.
- `identitystore` – Allows users to see user and group names.

For a JSON listing of the policy details, see AWSDeadlineCloud-UserAccessJobs in the *AWS Managed Policy reference guide*.

AWS managed policy: AWSDeadlineCloud-UserAccessQueues

You can attach the `AWSDeadlineCloud-UserAccessQueues` policy to your IAM identities.

This policy allows users to access queue data based on the farms that they are members of and their membership level.

Permissions details

This policy includes the following permissions:

- `deadline` – Allows the user to access farm data.
- `ec2` – Allows users to see details about Amazon EC2 instance types.
- `identitystore` – Allows users to see user and group names.

For a JSON listing of the policy details, see AWSDeadlineCloud-UserAccessQueues in the *AWS Managed Policy reference guide*.

Deadline Cloud updates to AWS managed policies

View details about updates to AWS managed policies for Deadline Cloud since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Deadline Cloud Document history page.

Change	Description	Date
AWSDeadlineCloud-UserAccessFarms – Change	Deadline Cloud added new action `kms:Decrypt` so you can use an AWS KMS customer-managed key with your IAM Identity Center instance.	December 22, 2025
AWSDeadlineCloud-WorkerHost – Change	Deadline Cloud added new actions `deadline:TagResource` and `deadline:ListTagsForResource` to allow you to add and view tags	May 30, 2025

Change	Description	Date
	associated with workers in your fleet.	
AWSDeadlineCloud-UserAccessFarms – Change AWSDeadlineCloud-UserAccessJobs – Change AWSDeadlineCloud-UserAccessQueues – Change	Deadline Cloud added new actions `deadline:GetJobTemplate` and `deadline:ListJobParameterDefinitions` to allow you to resubmit jobs.	October 7, 2024
Deadline Cloud started tracking changes	Deadline Cloud started tracking changes to its AWS managed policies.	April 2, 2024

Service roles

How Deadline Cloud uses IAM service roles

Deadline Cloud automatically assumes IAM roles and provides temporary credentials to workers, jobs, and the Deadline Cloud monitor. This approach eliminates manual credential management while maintaining security through role-based access control.

Four roles cover the lifecycle of work in a farm. On a customer-managed fleet, a new host starts with the *worker host role*, uses it to register as a worker, and trades it for *fleet role* credentials. On a service-managed fleet, the service performs that bootstrap for you, so only the other three roles apply. The worker uses the fleet role to receive work and report progress, and receives *queue role* credentials while it runs each job. People and tools get their credentials from the *monitor role* when they sign in to the Deadline Cloud monitor.

Deadline Cloud service roles at a glance

Role	Who uses its credentials	What it grants	Associated resource
Worker host role	Customer-managed fleet hosts during startup	Register a new worker and assume the fleet role	The worker host, for example through an Amazon EC2 instance profile
Fleet role	Workers on the fleet	Receive work, report progress, and write worker logs to CloudWatch Logs	The fleet
Queue role	Jobs while they run; monitor and CLI users working with job attachments and logs	Read and write the queue's job attachments bucket, read job logs, download third-party software, plus any permissions you add for your jobs	The queue
Monitor role	People signed in to the Deadline Cloud monitor, and the CLI and submitter s through the profile it creates	Access to farm, fleet, queue, and job data based on each user's memberships and access level	The monitor

When you create monitors, fleets, and queues in the console, Deadline Cloud can create the fleet, queue, and monitor roles for you with the necessary permissions. You create the worker host role yourself when you set up a customer-managed fleet. Use the following sections to understand each role for troubleshooting, to create the roles yourself, or to extend them.

Fleet role

Configure a fleet role to give Deadline Cloud workers the permissions they need to receive work and report progress on that work.

You usually do not have to configure this role yourself. This role can be created for you in the Deadline Cloud console to include the necessary permissions. Use the following guide to understand the specifics of this role for troubleshooting.

When creating or updating fleets programmatically, specify the fleet role ARN using the `CreateFleet` or `UpdateFleet` API operations.

What the fleet role does

The fleet role provides workers with permissions to:

- Receive new work and report progress on ongoing work to the Deadline Cloud service
- Manage worker lifecycle and status
- Record log events to Amazon CloudWatch Logs for the worker logs

Set up the fleet role trust policy

Your fleet role must trust the Deadline Cloud service and be scoped to your specific farm.

As a best practice, the trust policy should include security conditions for Confused Deputy protection. To learn more about Confused Deputy protection, see Confused Deputy in the *Deadline Cloud User Guide*.

- `aws:SourceAccount` ensures only resources from the same AWS account can assume this role.
- `aws:SourceArn` restricts role assumption to a specific Deadline Cloud farm.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDeadlineCredentialsService",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
```

```
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:deadline:REGION:YOUR_ACCOUNT_ID:farm/YOUR_FARM_ID"
        }
      }
    }
  ]
}
```

Attach the Fleet role permissions

Attach the following AWS managed policy to your fleet role:

AWSDeadlineCloud-FleetWorker

This managed policy provides permissions for:

- `deadline:AssumeFleetRoleForWorker` - Allows workers to refresh their credentials.
- `deadline:UpdateWorker` - Allows workers to update their status (for example, to STOPPED when exiting).
- `deadline:UpdateWorkerSchedule` - For obtaining work and reporting progress.
- `deadline:BatchGetJobEntity` - For fetching job information.
- `deadline:AssumeQueueRoleForWorker` - For accessing queue role credentials during job execution.

Add KMS permissions for encrypted farms

If your farm was created using a KMS key, add these permissions to your fleet role to ensure the worker can access encrypted data in the farm.

The KMS permissions are only necessary if your farm has an associated KMS key. The `kms:ViaService` condition must use the format `deadline.{region}.amazonaws.com`.

When creating a fleet, a CloudWatch Logs log group is created for that fleet. The worker's permissions are used by the Deadline Cloud service to create a log stream specifically for that

particular worker. After the worker is set up and running, the worker will use these permissions to send log events directly to CloudWatch Logs.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateLogStream",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "deadline.REGION.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "ManageLogEvents",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*"
    },
    {
      "Sid": "ManageKmsKey",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:GenerateDataKey"
      ],
      "Resource": "YOUR_FARM_KMS_KEY_ARN",
      "Condition": {
        "StringEquals": {
```

```
          "kms:ViaService": "deadline.REGION.amazonaws.com"
        }
      }
    }
  ]
}
```

Modifying the fleet role

Permissions for the fleet role are not customizable. The described permissions are always required and adding additional permissions has no effect.

Worker host role

Set up a worker host role if you use customer-managed fleets on Amazon EC2 instances or on-premises hosts.

What the worker host role does

The worker host role (`WorkerHost`) bootstraps workers on customer-managed fleet hosts. It provides the minimal permissions needed for a host to:

- Create a worker in Deadline Cloud
- Assume the fleet role to fetch operational credentials
- Tag workers with fleet tags (if tag propagation is enabled)

Set up worker host role permissions

Attach the following AWS managed policy to your worker host role:

AWSDeadlineCloud-WorkerHost

This managed policy provides permissions for:

- `deadline:CreateWorker` - Allows the host to register a new worker.
- `deadline:AssumeFleetRoleForWorker` - Allows the host to assume the fleet role.
- `deadline:TagResource` - Allows tagging workers during creation (if enabled).
- `deadline:ListTagsForResource` - Allows reading fleet tags for propagation.

Understand the bootstrap process

The worker host role is only used during initial worker startup:

1. The worker agent starts on the host using worker host role credentials.
2. It invokes `deadline:CreateWorker` to register with Deadline Cloud.
3. It then invokes `deadline:AssumeFleetRoleForWorker` to fetch fleet role credentials.
4. From this point forward, the worker uses only fleet role credentials for all operations.

After the worker starts running, it no longer uses the worker host role. Service-managed fleets don't need this role because the service performs the bootstrap automatically.

Queue role

The queue role is assumed by the worker when processing a task. This role provides the permissions needed to complete the task.

When creating or updating queues programmatically, specify the queue role ARN using the `CreateQueue` or `UpdateQueue` API operations.

Set up the queue role trust policy

Your queue role must trust the Deadline Cloud service.

As a best practice, the trust policy should include security conditions for Confused Deputy protection. To learn more about Confused Deputy protection, see Confused Deputy in the *Deadline Cloud User Guide*.

- `aws:SourceAccount` ensures only resources from the same AWS account can assume this role.
- `aws:SourceArn` restricts role assumption to a specific Deadline Cloud farm.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
```

```
          "credentials.deadline.amazonaws.com",
          "deadline.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:deadline:us-west-2:123456789012:farm/{farm-id}"
        }
      }
    }
  ]
}
```

Understand queue role permissions

The queue role doesn't use a single managed policy. Instead, when you configure your queue in the console, Deadline Cloud creates a custom policy for your queue based on your configuration.

This automatically created policy provides access to:

Job attachments

Read and write access to your specified Amazon S3 bucket for job input and output files:

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
    "s3:PutObject",
    "s3:ListBucket",
    "s3:GetBucketLocation"
  ],
  "Resource": [
    "arn:aws:s3:::YOUR_JOB_ATTACHMENTS_BUCKET",
    "arn:aws:s3:::YOUR_JOB_ATTACHMENTS_BUCKET/YOUR_PREFIX/*"
  ],
  "Condition": {
    "StringEquals": {
      "aws:ResourceAccount": "YOUR_ACCOUNT_ID"
```

```
    }
  }
}
```

Job logs

Read access to CloudWatch Logs for jobs in this queue. Each queue has its own log group and each session has its own log stream:

```
{
  "Effect": "Allow",
  "Action": [
    "logs:GetLogEvents"
  ],
  "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*"
}
```

Third-party software

Access to download third-party software supported by Deadline Cloud (such as Maya, Blender, and others):

```
{
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
    "s3:GetObject"
  ],
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "s3:DataAccessPointArn": "arn:aws:s3:*:*:accesspoint/deadline-software-*"
    },
    "StringEquals": {
      "s3:AccessPointNetworkOrigin": "VPC"
    }
  }
}
```

Add permissions for your jobs

Add permissions to your queue role for AWS services that your jobs need to access. When writing OpenJobDescription step scripts, the AWS CLI and SDK will automatically use credentials from your queue role. Use this to access additional services needed to complete your job.

Example use cases include:

- for fetching custom data
- SSM permissions to tunnel to a custom license server
- CloudWatch for emitting custom metrics
- Deadline Cloud permission to create new jobs for dynamic workflows

How queue role credentials are used

Deadline Cloud provides queue role credentials to:

- Workers during job execution
- Users via Deadline Cloud CLI and monitor when interacting with job attachments and logs

Deadline Cloud creates separate CloudWatch Logs log groups for each queue. The Deadline Cloud CLI and monitor use the queue role (through `deadline:AssumeQueueRoleForRead`) to read job logs from the queue's log group. The Deadline Cloud CLI and monitor use the queue role (through `deadline:AssumeQueueRoleForUser`) to upload or download job attachments data.

Monitor role

Configure a monitor role to give the Deadline Cloud monitor web and desktop applications access to your Deadline Cloud resources.

When creating or updating monitors programmatically, specify the monitor role ARN using the `CreateMonitor` or `UpdateMonitor` API operations.

What the monitor role does

The monitor role enables Deadline Cloud monitor to provide end users with access to:

- Basic functionality required for the Deadline Cloud Integrated Submitters, CLI and monitor
- Custom functionality for end users

Set up the monitor role trust policy

Your monitor role must trust the Deadline Cloud service.

As a best practice, the trust policy should include security conditions for Confused Deputy protection. To learn more about Confused Deputy protection, see Confused Deputy in the *Deadline Cloud User Guide*.

`aws:SourceAccount` ensures only resources from the same AWS account can assume this role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        }
      }
    }
  ]
}
```

Attach monitor role permissions

Attach all of the following AWS managed policies to your monitor role for basic operation:

- AWSDeadlineCloud-UserAccessFarms
- AWSDeadlineCloud-UserAccessFleets
- AWSDeadlineCloud-UserAccessJobs
- AWSDeadlineCloud-UserAccessQueues

How the monitor role works

When using the Deadline Cloud monitor, a service user signs in using AWS IAM Identity Center (IAM Identity Center), and the monitor role is assumed. The assumed role credentials are used by the

monitor application to display the monitor UI, including the list of farms, fleets, queues, and other information.

When using the Deadline Cloud monitor desktop application, these credentials are additionally made available on the workstation using a named AWS credential profile corresponding to the profile name provided by the end user. Learn more about named profiles in the AWS SDK and Tools reference guide.

This named profile is how the Deadline CLI and submitters access Deadline Cloud resources.

Customizing the monitor role for advanced use cases

You can customize the monitor role to modify what users can do at each access level (Viewer, Contributor, Manager, Owner) or to add permissions for advanced workflows.

Customizing access level permissions

The four AWS managed policies attached to the monitor role control what each access level can do. You can add custom policies to the monitor role to grant or restrict permissions for specific access levels using the `deadline:MembershipLevel` condition key.

For example, to allow Contributors to update and cancel jobs (which is normally restricted to Managers and Owners), add a policy like the following:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "deadline:UpdateJob",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "deadline:MembershipLevel": "CONTRIBUTOR"
        }
      }
    }
  ]
}
```

With this policy, Contributors can update and cancel jobs in addition to submitting them.

Adding permissions for advanced workflows

You can add custom IAM policies to the monitor role to grant additional permissions to all monitor users. Custom policies are useful for advanced scripting workflows where users need access to AWS services beyond the standard Deadline Cloud functionality.

Follow these guidelines when modifying your monitor role:

- Don't remove any of the managed policies. Removing these policies breaks monitor functionality.

How Deadline Cloud monitor uses monitor role credentials

Deadline Cloud monitor automatically obtains monitor role credentials when you authenticate. These temporary credentials last for 15 minutes and automatically refresh as long as you are signed in to IAM Identity Center. This capability enables the desktop application to provide monitoring capabilities beyond what's available in a standard web browser.

When you log in with Deadline Cloud monitor, it automatically creates a profile that you can use with the AWS CLI or any other AWS tool. This profile uses the monitor role credentials, giving you programmatic access to AWS services based on the permissions in your monitor role.

Deadline Cloud submitters work the same way - they use the profile created by Deadline Cloud monitor to access AWS services with the appropriate role permissions.

Advanced customization of Deadline Cloud roles

You can extend Deadline Cloud roles with additional permissions to enable advanced use cases beyond basic rendering workflows. This approach uses the access management system in Deadline Cloud to control access to additional AWS services based on queue membership.

Team collaboration with AWS CodeCommit

Add AWS CodeCommit permissions to your Queue role to enable team collaboration on project repositories. This approach uses the access management system in Deadline Cloud for additional use cases. Only users with access to the specific queue receive these AWS CodeCommit permissions, so you can manage per-project repository access through Deadline Cloud queue membership.

Repository access through queue membership is useful when artists need to access project-specific assets, scripts, or configuration files stored in AWS CodeCommit repositories as part of their rendering workflow.

Add AWS CodeCommit permissions to queue role

Add the following permissions to your queue role to enable AWS CodeCommit access:

```
{
  "Effect": "Allow",
  "Action": [
    "codecommit:GitPull",
    "codecommit:GitPush",
    "codecommit:GetRepository",
    "codecommit:ListRepositories"
  ],
  "Resource": "arn:aws:codecommit:REGION:YOUR_ACCOUNT_ID:PROJECT_REPOSITORY"
}
```

Set up credential provider on artist workstations

Configure each artist workstation to use Deadline Cloud queue credentials for AWS CodeCommit access. This setup is done once per workstation.

To configure the credential provider

1. Add a credential provider profile to your AWS config file (`~/.aws/config`):

   ```
   [profile queue-codecommit]
   credential_process = deadline queue export-credentials --farm-id farm-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX --queue-id queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   ```

2. Configure Git to use this profile for AWS CodeCommit repositories:

   ```
   git config --global credential.https://git-codecommit.REGION.amazonaws.com.helper
    '!aws codecommit credential-helper --profile queue-codecommit $@'
   git config --global credential.https://git-
   codecommit.REGION.amazonaws.com.UseHttpPath true
   ```

Replace *farm-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX* and *queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX* with your actual farm and queue IDs. Replace *REGION* with your AWS region (for example, `us-west-2`).

Using AWS CodeCommit with queue credentials

Once configured, Git operations will automatically use the queue role credentials when accessing AWS CodeCommit repositories. The `deadline queue export-credentials` command returns temporary credentials that look like this:

```
{
  "Version": 1,
  "AccessKeyId": "ASIA...",
  "SecretAccessKey": "...",
  "SessionToken": "...",
  "Expiration": "2025-11-10T23:02:23+00:00"
}
```

Deadline Cloud automatically refreshes these credentials as needed, and your Git operations work without additional configuration:

```
git clone https://git-codecommit.REGION.amazonaws.com/v1/repos/PROJECT_REPOSITORY
git pull
git push
```

Artists can now access project repositories using their queue permissions without needing separate AWS CodeCommit credentials. Only users with access to the specific queue will be able to access the associated repository, so you can control repository access through queue membership in Deadline Cloud.

Troubleshooting AWS Deadline Cloud identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Deadline Cloud and IAM.

Topics

I am not authorized to perform an action in Deadline Cloud

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `deadline:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 deadline:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `deadline:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Deadline Cloud.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Deadline Cloud. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my Deadline Cloud resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Deadline Cloud supports these features, see How Deadline Cloud works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Data protection in Deadline Cloud

The AWS shared responsibility model applies to data protection in AWS Deadline Cloud. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see Data Privacy FAQ. For information about data protection in Europe, see the General Data Protection Regulation (GDPR) Center.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.

- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with Deadline Cloud or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

The data entered into name fields in Deadline Cloud job templates may also be included in billing or diagnostic logs and should not contain confidential or sensitive information.

Topics

Encryption at rest

AWS Deadline Cloud protects sensitive data by encrypting it at rest using encryption keys stored in AWS Key Management Service (AWS KMS). Encryption at rest is available in all AWS Regions where Deadline Cloud is available.

Encrypting data means sensitive data saved on disks isn't readable by a user or application without a valid key. Only a party with a valid managed key can decrypt the data.

Deadline Cloud deletes Amazon Elastic Block Store volumes when service-managed fleet worker instances terminate.

For information about how Deadline Cloud uses AWS KMS for encrypting data at rest, see Key management.

Encryption in transit

For data in transit, AWS Deadline Cloud uses Transport Layer Security (TLS) 1.2 or 1.3 to encrypt data sent between the service and workers. We require TLS 1.2 and recommend TLS 1.3. Additionally, if you use a virtual private cloud (VPC), you can use AWS PrivateLink to establish a private connection between your VPC and Deadline Cloud.

Key management

When creating a new farm, you can choose one of the following keys to encrypt your farm data:

- **AWS owned KMS key** – Default encryption type if you don't specify a key when you create the farm. The KMS key is owned by AWS Deadline Cloud. You can't view, manage, or use AWS owned keys. However, you don't need to take any action to protect the keys that encrypt your data. For more information, see AWS owned keys in the *AWS Key Management Service developer guide*.
- **Customer managed KMS key** – You specify a customer managed key when you create a farm. All of the content within the farm is encrypted with the KMS key. The key is stored in your account and is created, owned, and managed by you and AWS KMS charges apply. You have full control over the KMS key. You can perform such tasks as:
  - Establishing and maintaining key polices
  - Establishing and maintaining IAM policies and grants
  - Enabling and disabling key policies
  - Adding tags
  - Creating key aliases

  You can't manually rotate a customer owned key used with a Deadline Cloud farm. Automatic rotation of the key is supported.

For more information, see Customer owned keys in the *AWS Key Management Service Developer Guide*.

To create a customer managed key, follow the steps for Creating symmetric customer managed keys in the *AWS Key Management Service Developer Guide*.

How Deadline Cloud uses AWS KMS grants

Deadline Cloud requires a grant to use your customer managed key. When you create a farm encrypted with a customer managed key, Deadline Cloud creates a grant on your behalf by sending a `CreateGrant` request to AWS KMS to get access to the KMS key that you specified.

Deadline Cloud uses multiple grants. Each grant is used by a different part of Deadline Cloud that needs to encrypt or decrypt your data. Deadline Cloud also uses grants to allow access to other AWS services used to store data on your behalf, such as Amazon Simple Storage Service, Amazon Elastic Block Store, or OpenSearch.

Grants that enable Deadline Cloud to manage machines in a service-managed fleet include a Deadline Cloud account number and role in the `GranteePrincipal` instead of a service principal. While not typical, the grant configuration is necessary to encrypt Amazon EBS volumes for workers in service-managed fleets using the customer managed KMS key specified for the farm.

Customer managed key policy

Key policies control access to your customer managed key. Each key must have exactly one key policy that contains statements that determine who can use the key and how they can use it. When you create you customer managed key, you can specify a key policy. For more information, see Managing access to customer managed keys in the *AWS Key Management Service Developer Guide*.

Minimal IAM policy for CreateFarm

To use your customer managed key to create farms using the console or the `CreateFarm` API operation, the following AWS KMS API operations must be permitted:

- `kms:CreateGrant` – Adds a grant to a customer managed key. Grants console access to a specified AWS KMS key. For more informations, see Using grants in the *AWS Key Management Service developer guide*.
- `kms:Decrypt` – Allows Deadline Cloud to decrypt data in the farm.

- kms:DescribeKey – Provides the customer managed key details to allow Deadline Cloud to validate the key.
- kms:GenerateDataKey – Allows Deadline Cloud to encrypt data using a unique data key.

The following policy statement grants the necessary permissions for the `CreateFarm` operation.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineCreateGrants",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1234567890abcdef0",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

Minimal IAM policy for read-only operations

To use your customer managed key for read-only Deadline Cloud operations, such getting information about farms, queues, and fleets. The following AWS KMS API operations must be permitted:

- kms:Decrypt – Allows Deadline Cloud to decrypt data in the farm.

- kms:DescribeKey – Provides the customer managed key details to allow Deadline Cloud to validate the key.

The following policy statement grants the necessary permissions for read-only operations.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineReadOnly",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

Minimal IAM policy for read-write operations

To use your customer managed key for read-write Deadline Cloud operations, such as creating and updating farms, queues, and fleets. The following AWS KMS API operations must be permitted:

- kms:Decrypt – Allows Deadline Cloud to decrypt data in the farm.
- kms:DescribeKey – Provides the customer managed key details to allow Deadline Cloud to validate the key.
- kms:GenerateDataKey – Allows Deadline Cloud to encrypt data using a unique data key.

The following policy statement grants the necessary permissions for the `CreateFarm` operation.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineReadWrite",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

Monitoring your encryption keys

When you use an AWS KMS customer managed key with your Deadline Cloud farms, you can use AWS CloudTrail or Amazon CloudWatch Logs to track requests that Deadline Cloud sends to AWS KMS.

Deadline Cloud generates a `CreateGrant` event when it sets up access to your key, and `Decrypt` and `GenerateDataKey` events when it uses the key. The events include an `encryptionContext` that identifies the farm. For more information about the events, see Logging AWS KMS API calls with AWS CloudTrail in the *AWS Key Management Service Developer Guide*.

CloudTrail event for grants

The following example CloudTrail event occurs when grants are created, typically when you call the `CreateFarm`, `CreateMonitor`, or `CreateFleet` operation.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/Admin/SampleUser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-04-23T02:05:26Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T02:05:35Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "operations": [
            "CreateGrant",
            "Decrypt",
            "DescribeKey",
            "Encrypt",
            "GenerateDataKey"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:deadline:farmId": "farm-abcdef1234567890098765432 1fedcba",
                "aws:deadline:accountId": "111122223333"
            }
```

```
        },
        "granteePrincipal": "deadline.amazonaws.com",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "retiringPrincipal": "deadline.amazonaws.com"
    },
    "responseElements": {
        "grantId": "6bbe819394822a400fe5e3a75d0e9ef16c1733143fff0c1fc00dc7ac282a18a0",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE44444"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


Deleting a customer managed KMS key

Deleting a customer managed KMS key in AWS Key Management Service (AWS KMS) is destructive and potentially dangerous. It irreversibly deletes the key material and all metadata associated with the key. After a customer managed KMS key is deleted, you can no longer decrypt the data that was encrypted by that key. Deleting the key means that the data becomes unrecoverable.

For safety, AWS KMS gives customers a waiting period of up to 30 days before deleting the KMS key. The default waiting period is 30 days.

About the waiting period

Because it's destructive and potentially dangerous to delete a customer managed KMS key, we require that you set a waiting period of 7–30 days. The default waiting period is 30 days.

However, the actual waiting period might be up to 24 hours longer than the period you scheduled. To get the actual date and time when the key will be deleted, use the DescribeKey operation. You can also see the scheduled deletion date of a key in the AWS KMS console on the key's detail page, in the **General configuration** section. Notice the time zone.

During the waiting period, the customer managed key's status and key state is **Pending deletion**.

- A customer managed KMS key that is pending deletion can't be used in any cryptographic operations.
- AWS KMS doesn't rotate the backing keys of customer managed KMS keys that are pending deletion.

For more information about deleting a customer managed KMS key, see Deleting customer master keys in the *AWS Key Management Service Developer Guide*.

Inter-network traffic privacy

AWS Deadline Cloud supports Amazon Virtual Private Cloud (Amazon VPC) to secure connections. Amazon VPC provides features that you can use to increase and monitor the security for your virtual private cloud (VPC).

You can set up a customer-managed fleet (CMF) with Amazon Elastic Compute Cloud (Amazon EC2) instances that run inside a VPC. By deploying Amazon VPC endpoints to use AWS PrivateLink, traffic between workers in your CMF and the Deadline Cloud endpoint stays within your VPC. Furthermore, you can configure your VPC to restrict internet access to your instances.

In service-managed fleets, workers aren't reachable from the internet, but they do have internet access and connect to the Deadline Cloud service over the internet. Each service-managed fleet runs in its own isolated network, and worker instances remain dedicated to individual customers.

Opt out

AWS Deadline Cloud collects certain operational information to help us develop and improve Deadline Cloud. The collected data includes things such as your AWS account ID and user ID, so that we can correctly identify you if you have an issue with the Deadline Cloud. We also collect Deadline Cloud specific information, such as Resource IDs (a FarmID or QueueID when applicable), the product name (for example, JobAttachments, WorkerAgent, and more) and the product version.

You can choose to opt out from this data collection using application configuration. Each computer interacting with Deadline Cloud, both client workstations and fleet workers, needs to opt out separately.

Deadline Cloud monitor - desktop

Deadline Cloud monitor - desktop collects operational information, such as when crashes occur and when the application is opened, to help us know when you are having problems with the application. To opt out from the collection of this operational information, go to the settings page and clear **Turn on data collection to measure Deadline Cloud Monitor's performance**.

After you opt out, the desktop monitor no longer sends the operational data. Any previously collected data is retained and may still be used to improve the service. For more information, see Data Privacy FAQ.

AWS Deadline Cloud CLI and Tools

The AWS Deadline Cloud CLI, submitters, and worker agent all collect operational information such as when crashes occur and when jobs are submitted to help us know when you are having problems with these applications. To opt out from the collection of this operational information, use any of the following methods:

- In the terminal, enter **`deadline config set telemetry.opt_out true`**.

  The command opts out the CLI, submitters, and worker agent when running as the current user.
- When installing the Deadline Cloud worker agent, add the **`--telemetry-opt-out`** command line argument. For example, **`./install.sh --farm-id $FARM_ID --fleet-id $FLEET_ID --telemetry-opt-out`**.
- Before running the worker agent, CLI, or submitter, set an environment variable: **`DEADLINE_CLOUD_TELEMETRY_OPT_OUT=true`**

After you opt out, the Deadline Cloud tools no longer send the operational data. Any previously collected data is retained and may still be used to improve the service. For more information, see Data Privacy FAQ.

Access AWS Deadline Cloud using an interface endpoint (AWS PrivateLink)

You can use AWS PrivateLink to create a private connection between your VPC and AWS Deadline Cloud. You can access Deadline Cloud as if it were in your VPC, without the use of an internet gateway, NAT device, VPN connection, or Direct Connect connection. Instances in your VPC don't need public IP addresses to access Deadline Cloud.

You establish this private connection by creating an *interface endpoint*, powered by AWS PrivateLink. We create an endpoint network interface in each subnet that you enable for the interface endpoint. These are requester-managed network interfaces that serve as the entry point for traffic destined for Deadline Cloud.

Deadline Cloud also has dual-stack endpoints available. Dual-stack endpoints support requests over IPv6 and IPv4.

For more information, see Access AWS services through AWS PrivateLink in the *AWS PrivateLink Guide*.

Considerations for Deadline Cloud

Before you set up an interface endpoint for Deadline Cloud, see Access an AWS service using an interface VPC endpoint in the *AWS PrivateLink Guide*.

Deadline Cloud supports making calls to all of its API actions through the interface endpoint.

By default, full access to Deadline Cloud is allowed through the interface endpoint. Alternatively, you can associate a security group with the endpoint network interfaces to control traffic to Deadline Cloud through the interface endpoint.

Deadline Cloud also supports VPC endpoint policies. For more information, see Control access to VPC endpoints using endpoint policies in the *AWS PrivateLink Guide*.

Deadline Cloud endpoints

Deadline Cloud uses four endpoints for access to the service using AWS PrivateLink - two for IPv4 and two for IPv6.

Workers use the `scheduling.deadline.region.amazonaws.com` endpoint to get tasks from the queue, report progress to Deadline Cloud, and to send task output back. If you are using a

customer-managed fleet, the scheduling endpoint is the only endpoint that you need to create unless you are using management operations. For example, if a job creates more jobs, you need to enable the management endpoint to call the `CreateJob` operation.

The Deadline Cloud monitor uses the `management.deadline.region.amazonaws.com` to manage the resources in your farm, such as creating and modifying queues and fleets or getting lists of jobs, steps, and tasks.

The AWS SDKs and CLI automatically add the `management` and `scheduling` prefixes to the endpoint. If you want to disable this behavior, see the host prefix injection section in the *AWS SDKs and Tools Reference Guide*.

Deadline Cloud also requires endpoints for the following AWS service endpoints:

- If you set up your customer-managed fleet in a subnet with no internet connection you must create a VPC endpoint for Amazon CloudWatch Logs so that workers can write logs. For more information, see Monitoring with CloudWatch.
- If you use job attachments, you must create a VPC endpoint for Amazon Simple Storage Service (Amazon S3) so that workers can access the attachments. For more information, see Job attachments in Deadline Cloud.

Create endpoints for Deadline Cloud

You can create interface endpoints for Deadline Cloud using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Create an interface endpoint in the *AWS PrivateLink Guide*.

Create management and scheduling endpoints for Deadline Cloud using the following service names. Replace *region* with the AWS Region where you've deployed Deadline Cloud.

```
com.amazonaws.region.deadline.management
```

```
com.amazonaws.region.deadline.scheduling
```

Deadline Cloud supports dual-stack endpoints.

If you enable private DNS for the interface endpoints, you can make API requests to Deadline Cloud using its default Regional DNS name. For example, `scheduling.deadline.us-`

`east-1.amazonaws.com` for worker operations, or `management.deadline.us-east-1.amazonaws.com` for all other operations.

If your customer-managed fleet is on a subnet without an internet connection, you must create a CloudWatch Logs endpoint using the following service name:

```
com.amazonaws.region.logs
```

If you use job attachments to transfer files, you must create an Amazon S3 endpoint using the following service name:

```
com.amazonaws.region.s3
```

Restricted network environments

Deadline Cloud provides tools that are used by artists or other users on their local workstations. These tools require access to AWS API and web endpoints to perform their function. If you filter access to specific AWS domains or URL endpoints by using a web content filtering solution such as next-generation firewalls (NGFW) or Secure Web Gateways (SWG), you must add the following domains or URL endpoints to your web-content filtering solution allowlists.

AWS API endpoints to allowlist

Deadline Cloud client tools, such as the AWS Management Console, monitor, CLI, and integrated submitters, require access to AWS APIs in addition to Deadline Cloud. These endpoints only support IPv4.

- `scheduling.deadline.[Region].amazonaws.com`
- `management.deadline.[Region].amazonaws.com`
- `logs.[Region].amazonaws.com`
- `ec2.[Region].amazonaws.com`
- `s3.[Region].amazonaws.com`
- `sts.[Region].amazonaws.com`
- `identitystore.[Region].amazonaws.com`

Web domains to allowlist

The Deadline Cloud monitor requires access to the following domains to operate.

For additional information about allowlisting domains for AWS Sign-In, see Domains to add to your allow list in the *AWS Sign-In User Guide*.

- `downloads.deadlinecloud.amazonaws.com`
- `d2ev1rdnjzhmnr.cloudfront.net`
- `prod.log.shortbread.aws.dev`
- `prod.tools.shortbread.aws.dev`
- `prod.log.shortbread.analytics.console.aws.a2z.com`
- `prod.tools.shortbread.analytics.console.aws.a2z.com`
- `global.help-panel.docs.aws.a2z.com`
- *`[Region]`*`.signin.aws`
- *`[Region]`*`.signin.aws.amazon.com`
- `sso.`*`[Region]`*`.amazonaws.com`
- `portal.sso.`*`[Region]`*`.amazonaws.com`
- `oidc.`*`[Region]`*`.amazonaws.com`
- `assets.sso-portal.`*`[Region]`*`.amazonaws.com`

Environment-specific endpoints to allowlist

These domains vary depending on the specific configuration of Deadline Cloud. If additional Deadline Cloud monitors or queues are created, additional domains will need to be allowlisted.

- *`[Directory ID or alias]`*`.awsapps.com`

  This domain is tied to the IAM Identity Center setup and should be the same for all setups in this using the same IAM Identity Center instance. The exact value can be found by the enterprise admin in the IAM Identity Center console under *Settings → AWS access portal URL*.
- *`[Monitor alias]`*`.`*`[Region]`*`.deadlinecloud.amazonaws.com`

  This domain is for the Monitor setup in Deadline Cloud. Artists enter this link into their browser or Deadline Cloud monitor application. If Deadline Cloud is set up in additional accounts or

regions in the future, this domain will change. You can find this value in the Deadline Cloud console in the *Dashboard → Monitor overview → Monitor details → URL*. The URL is reachable from any location on the internet. For information about what the URL exposes and how Deadline Cloud authorizes access to farm data, see Access to the monitor web application.

- `[Bucket name].[Region].s3.amazonaws.com`

  This is the domain for the job attachments bucket used by Deadline Cloud queues. Each queue can have its own job attachments bucket configured. The exact bucket name can be found in the Deadline Cloud console under *Queues → Queue details → Job attachments*. For more information about job attachments, see the queues documentation.

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the aws:SourceArn and aws:SourceAccount global condition context keys in resource policies to limit the permissions that AWS Deadline Cloud gives another service to the resource. Use `aws:SourceArn` if you want only one resource to be associated with the cross-service access. Use `aws:SourceAccount` if you want to allow any resource in that account to be associated with the cross-service use.

The most effective way to protect against the confused deputy problem is to use the `aws:SourceArn` global condition context key with the full Amazon Resource Name (ARN) of the resource. If you don't know the full ARN of the resource or if you are specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcard characters (*) for the unknown portions of the ARN. For example, `arn:aws:deadline:*:123456789012:*`.

If the `aws:SourceArn` value does not contain the account ID, such as an Amazon S3 bucket ARN, you must use both global condition context keys to limit permissions.

The following example shows how you can use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in Deadline Cloud to prevent the confused deputy problem.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "deadline.amazonaws.com"
    },
    "Action": "deadline:CreateFarm",
    "Resource": [
      "*"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:deadline:*:111122223333:*"
      },
      "StringEquals": {
        "aws:SourceAccount": "111122223333"
      }
    }
  }
}
```

Compliance validation for Deadline Cloud

Third-party auditors assess the security and compliance of AWS Deadline Cloud as part of the AWS System and Organization Controls (SOC) compliance program.

AWS provides a frequently updated list of AWS services in scope of specific compliance programs at AWS services in Scope by Compliance Program.

Third-party audit reports are available for you to download using AWS Artifact (AWS Artifact). For more information, see Downloading Reports in AWS Artifact.

For more information about AWS compliance programs, see AWS Compliance Programs.

The sensitivity of your data, your company's compliance objectives, and applicable laws and regulations determine your compliance responsibility when you use AWS services. If your use of Deadline Cloud is subject to compliance with standards such as SOC, AWS provides resources to help:

- Security and Compliance Quick Start Guides – Architectural considerations and steps for deploying security- and compliance-focused baseline environments on AWS.
- AWS Compliance Resources – Workbooks and guides that might apply to your industry and location.
- AWS Config – Assess how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub CSPM – A comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in Deadline Cloud

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

AWS Deadline Cloud does not back up data stored in your job attachments S3 bucket. You can enable backups of your job attachments data using any standard Amazon S3 backup mechanism, such as S3 Versioning or AWS Backup.

Infrastructure security in Deadline Cloud

As a managed service, AWS Deadline Cloud is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access Deadline Cloud through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Deadline Cloud supports AWS PrivateLink VPC endpoint policies on its management and scheduling endpoints. An endpoint policy controls which Deadline Cloud API actions your VPC allows through the interface endpoint. For more information, see Access AWS Deadline Cloud using an interface endpoint (AWS PrivateLink).

Configuration and vulnerability analysis in Deadline Cloud

AWS handles basic security tasks like guest operating system (OS) and database patching, firewall configuration, and disaster recovery. These procedures have been reviewed and certified by the appropriate third parties. For more details, see the following resources:

- Shared Responsibility Model
- Amazon Web Services: Overview of Security Processes (whitepaper)

AWS Deadline Cloud manages tasks on service-managed or customer-managed fleets:

- For service-managed fleets, Deadline Cloud manages the guest operating system.
- For customer-managed fleets, you are responsible for managing the operating system.

For additional information about configuration and vulnerability analysis for AWS Deadline Cloud, see

- Security best practices for Deadline Cloud

Deadline Cloud assistant

The Deadline Cloud assistant is an AI-powered troubleshooting tool built into the Deadline Cloud monitor. It uses generative AI through Amazon Bedrock to help you diagnose render job failures by analyzing job configurations, task statuses, session logs, and CloudWatch data. The assistant runs in your browser and provides intelligent root cause analysis with actionable recommendations.

> **Important**
>
> The Deadline Cloud assistant is powered by generative AI. AI models generate the responses, and the responses might be inaccurate, incomplete, or outdated. Verify all recommendations before acting on them. Use of this feature is subject to the AWS Service Terms and the AWS Responsible AI Policy.

Topics

How the assistant works

When you interact with the assistant, it uses Amazon Bedrock foundation models to reason about your render job issues. The assistant has read-only access to your Deadline Cloud resources and CloudWatch logs, and follows a structured troubleshooting workflow:

1. Analyzes job configuration and lifecycle status
2. Identifies failed tasks and examines failure patterns

3. Retrieves session information and session action details
4. Analyzes CloudWatch logs for error patterns
5. Provides a root cause analysis with specific recommendations

The assistant can also help with the following tasks:

- Summarizing logs on the current page
- Navigating to relevant resources in the monitor (workers, logs, tasks)
- Answering questions about Deadline Cloud concepts and terminology
- Troubleshooting renderer-specific issues

All inference occurs within your AWS account using your own Amazon Bedrock service quotas. For more information, see Amazon Bedrock quotas for the Deadline Cloud assistant. Refreshing the page clears conversation history.

Important considerations

Before you use the Deadline Cloud assistant, be aware of the following considerations about AI-generated content and data handling:

- **AI-generated responses** – The assistant uses generative AI to produce responses. As with any generative AI, responses might be inaccurate, incomplete, or outdated. Always verify recommendations before making changes to your environment.
- **Content safeguards** – The assistant's instructions and tools are scanned for abusive or dangerous content as part of the deployment process. However, the assistant does not apply runtime content filtering on outputs. Responses might occasionally contain unexpected or inappropriate content.
- **Feedback** – You can provide feedback on individual responses by using the thumbs up and thumbs down icons. A general feedback form is also available from the assistant panel (non-EU and non-UK regions only). You can also contact AWS Support to report issues.

Enabling the Deadline Cloud assistant

Only Deadline Cloud administrators can enable or disable the assistant. The assistant is disabled by default.

Prerequisites

To use the assistant, your monitor must have the following:

- An administrator who enables the assistant through the Deadline Cloud console
- The Amazon Bedrock IAM policy attached to the monitor user role

Enabling the assistant

You can turn on the assistant when you first set up your monitor, or at any time afterward from the monitor settings. Use the following procedure to enable the assistant for all users of an existing monitor.

To enable the assistant

1. Open the Deadline Cloud console.
2. In the navigation pane, choose **Dashboard**.
3. In the **Monitor overview** section, choose **Actions**, and then choose **Edit** to open the monitor settings page.
4. In the **Deadline Cloud Assistant** section, select **Enable Deadline Cloud Assistant**.
5. Choose **Update**.

To verify that the assistant is enabled, check that the **Deadline Cloud Assistant** field in the **Monitor overview** section shows **Enabled**.

When you enable the assistant, Deadline Cloud:

- Attaches an Amazon Bedrock IAM policy to the monitor user role. The policy name begins with `DeadlineCloudAssistantBedrockPolicy` and grants the `bedrock:InvokeModelWithResponseStream` permission scoped to your Region's cross-region inference profile.
- Persists the enabled state through the monitor settings API.

Disabling the assistant

Use the following procedure to disable the assistant.

To disable the assistant

1. Open the Deadline Cloud console.
2. In the navigation pane, choose **Dashboard**.
3. In the **Monitor overview** section, choose **Actions**, and then choose **Edit** to open the monitor settings page.
4. In the **Deadline Cloud Assistant** section, clear **Enable Deadline Cloud Assistant**.
5. Choose **Update**.

Disabling the assistant turns it off for all monitor users, but the Amazon Bedrock IAM policy remains attached to the monitor user role. To also remove the Amazon Bedrock permissions, detach the policy whose name begins with `DeadlineCloudAssistantBedrockPolicy` from the monitor user role in the IAM console.

Required permissions

Monitor users need the `bedrock:InvokeModelWithResponseStream` permission to use the assistant. When an administrator enables the assistant, Deadline Cloud automatically attaches the required Amazon Bedrock IAM policy to the monitor user role.

For information about Deadline Cloud IAM permissions, see Identity-based policy examples for Deadline Cloud.

Amazon Bedrock IAM policy

When an administrator enables the assistant, the following IAM policy is attached to the monitor user role. The policy grants permission to invoke Amazon Bedrock models through cross-region inference profiles scoped to your monitor's geographic Region.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GrantCrisInferenceProfileAccess",
            "Effect": "Allow",
            "Action": "bedrock:InvokeModelWithResponseStream",
            "Resource": "arn:aws:bedrock:Region:AccountId:inference-
profile/InferenceProfilePrefix.*",
```

```
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": RequestedRegions
                }
            }
        },
        {
            "Sid": "GrantCrisModelAccess",
            "Effect": "Allow",
            "Action": "bedrock:InvokeModelWithResponseStream",
            "Resource": "arn:aws:bedrock:*::foundation-model/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": RequestedRegions
                },
                "ArnLike": {
                    "bedrock:InferenceProfileArn":
 "arn:aws:bedrock:Region:AccountId:inference-profile/InferenceProfilePrefix.*"
                }
            }
        }
    ]
}
```

The policy only grants `bedrock:InvokeModelWithResponseStream` – no other Amazon Bedrock actions are permitted.

Cross-region inference

The assistant uses Amazon Bedrock cross-region inference to optimize model availability and throughput. When you invoke the assistant, Amazon Bedrock might route your request to a different AWS Region within the same geographic area to process the inference request.

- Requests are routed to AWS Regions within a geographic boundary determined by your monitor's Region.
- All data transmitted between Regions remains on the AWS network and does not traverse the public internet.
- Data is encrypted in transit between AWS Regions.
- There is no additional routing cost for cross-region inference. Pricing is based on the Region from which you call the inference profile.

- Cross-region inference requests are logged in CloudTrail in your source Region. The `additionalEventData.inferenceRegion` field identifies where the request was processed.

The following table shows which geographic inference profile and destination Regions are used based on your monitor's Region:

Cross-region inference profile mapping

Monitor Region	Inference profile prefix	Destination Regions
us-east-1	`us`	us-east-1, us-east-2, us-west-2
us-east-2	`us`	us-east-1, us-east-2, us-west-2
us-west-2	`us`	us-east-1, us-east-2, us-west-2
eu-central-1	`eu`	eu-central-1, eu-north-1, eu-south-1, eu-south-2, eu-west-1, eu-west-3
eu-west-1	`eu`	eu-central-1, eu-north-1, eu-south-1, eu-south-2, eu-west-1, eu-west-3
eu-west-2	`eu`	eu-central-1, eu-north-1, eu-south-1, eu-south-2, eu-west-1, eu-west-2, eu-west-3
ap-northeast-1	`jp`	ap-northeast-1, ap-northeast-3
ap-southeast-2	`au`	ap-southeast-2, ap-southeast-4
ap-northeast-2	`global`	ap-northeast-2
ap-southeast-1	`global`	ap-southeast-1

For Regions using the `global` inference profile prefix, Amazon Bedrock might route requests to any supported commercial AWS Region worldwide.

Security

The Deadline Cloud assistant operates within the existing Deadline Cloud security model:

- **Read-only access** – The assistant only performs read operations (Get, List, Search) on Deadline Cloud resources and CloudWatch logs. It cannot modify your resources.
- **Customer-account execution** – All Amazon Bedrock model invocations occur in your AWS account using your credentials and service quotas.
- **Scoped permissions** – The Amazon Bedrock policy is scoped to cross-region inference profiles for your geographic region. Monitor users cannot access Amazon Bedrock actions beyond `InvokeModelWithResponseStream`.
- **Session isolation** – Conversations are isolated to individual browser sessions and are not persisted or shared.
- **Fail closed** – If the assistant cannot determine whether it is enabled (for example, if the `GetMonitorSettings` call fails), the assistant UI is not displayed.
- **Admin control** – Only administrators can enable or disable the assistant. Monitor users cannot self-escalate access.
- **Abuse detection** – Amazon Bedrock abuse detection capabilities apply to assistant usage. For more information, see Abuse detection in the *Amazon Bedrock User Guide*.

Model information

The Deadline Cloud assistant uses Anthropic Claude Sonnet 4.5 (`anthropic.claude-sonnet-4-5-20250929-v1:0`) as its foundation model, accessed through Amazon Bedrock cross-region inference profiles. The assistant also includes a knowledge base built from public Deadline Cloud documentation, public AWS documentation, and public documentation for popular digital content creation applications. This knowledge base is fetched by the assistant at invocation time. AWS did not use customer data from any Deadline Cloud account to build or fine-tune the assistant.

Data privacy

The Deadline Cloud assistant is subject to the Amazon Bedrock data protection policies. For more information about Amazon Bedrock data protection, see Data protection in the *Amazon Bedrock User Guide*.

The assistant holds conversation history in browser memory only. Refreshing or closing the page permanently deletes the conversation. The assistant doesn't persist any conversation data to disk, databases, or AWS services.

If you have Amazon Bedrock model invocation logging enabled in your account, your assistant conversations (including log content sent to the model) are captured in your configured logging destination (your Amazon S3 bucket or CloudWatch Logs log group). Model invocation logging is disabled by default and is entirely under your control. For more information, see Model invocation logging in the *Amazon Bedrock User Guide*.

Network path

The Deadline Cloud assistant runs in your browser as part of the Deadline Cloud monitor application. When you interact with the assistant, your browser makes Amazon Bedrock API calls (`InvokeModelWithResponseStream`) directly to the Amazon Bedrock service endpoint by using your monitor user credentials. These calls travel over HTTPS (TLS 1.2 or higher) to the public Amazon Bedrock endpoint in your Region.

Because the assistant runs in the browser, Amazon VPC interface endpoints (AWS PrivateLink) do not apply to assistant traffic. Amazon Bedrock PrivateLink support is designed for server-side workloads running within your Amazon VPC, not browser-based applications.

Organization-level controls

In addition to the per-monitor admin toggle, you can enforce organization-wide control over the assistant by using AWS Organizations (Organizations) service control policies (SCPs). An SCP that denies `bedrock:InvokeModelWithResponseStream` prevents the assistant from functioning, even if a monitor administrator enables the feature.

The following example SCP denies all Amazon Bedrock model invocations, which disables the assistant across all accounts in the organization or organizational unit (OU) where the policy is attached:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyBedrockInvocations",
            "Effect": "Deny",
```

```
            "Action": "bedrock:InvokeModelWithResponseStream",
            "Resource": "*"
        }
    ]
}
```

For more information about SCPs, see Service control policies in the *Organizations User Guide*.

> **Note**
>
> This SCP blocks all Amazon Bedrock model invocations in the affected accounts, including those not related to Deadline Cloud. To block only the assistant, disable it through the monitor settings instead.

Audit trail

The assistant's activities are auditable through AWS CloudTrail (CloudTrail):

- **Amazon Bedrock invocations** – CloudTrail logs each `InvokeModelWithResponseStream` call as a management event. The log entry records the model ID, user identity, timestamp, and source IP. The `additionalEventData.inferenceRegion` field identifies where the request was processed. The CloudTrail event doesn't include prompt or response content.
- **Deadline Cloud resource reads** – The assistant's read operations on Deadline Cloud resources (such as `GetJob`, `ListTasks`, `ListSessions`, and `SearchTasks`) are logged in CloudTrail as standard Deadline Cloud API calls. You can query these logs to determine which specific jobs, tasks, and sessions the assistant accessed during a conversation.
- **CloudWatch Logs reads** – The assistant reads worker and task logs by assuming the queue role (using `deadline:AssumeQueueRoleForRead`) or fleet role (using `deadline:AssumeFleetRoleForRead`). These role assumption events are logged in CloudTrail.

Abuse detection

The Amazon Bedrock automated abuse detection mechanisms apply to all assistant usage. For more information, see Abuse detection in the *Amazon Bedrock User Guide*.

Feedback data

The assistant provides two feedback mechanisms. Each mechanism transmits different data:

- **Thumbs up/down buttons** – When you click a thumbs up or thumbs down icon on an assistant response, only a sentiment indicator (positive or negative) and a session identifier are recorded as a telemetry event. No conversation content, log data, or prompts are included in the feedback event.
- **General feedback form** (non-EU and non-UK regions only) – When you submit general feedback through the speech bubble icon, the form transmits only the information that you explicitly enter. This includes a category selection, a subject line, a description, and an optional email address. The form also includes your monitor's Region and the current page path as metadata. No conversation content or log data is included unless you manually type it into the form fields. General feedback is submitted to an AWS feedback service.

General feedback is unavailable in EU and UK regions because of data residency requirements. The thumbs up/down feedback is available in all regions because the telemetry event contains no customer content.

Costs

The Deadline Cloud assistant incurs Amazon Bedrock usage costs in your AWS account. Costs are based on the number of input and output tokens processed during each interaction. Because the assistant uses cross-region inference, pricing is calculated based on your monitor's source Region.

For current pricing, see Amazon Bedrock pricing.

> **Tip**
>
> The assistant displays context window usage during interactions to help you monitor token consumption.

Tracking assistant costs

Amazon Bedrock supports cost allocation by AWS Identity and Access Management (IAM) principal, which you can use to track and attribute assistant inference costs across teams, projects, or cost

centers. To set up cost tracking for the assistant, tag the monitor user role with attributes that represent your organizational structure, then activate those tags for cost allocation.

To complete this procedure, you need IAM permissions to tag roles and access to the Billing console.

To set up cost tracking for the assistant

1. Tag the IAM role that your monitor users assume. You can add tags in the IAM console or by using the AWS CLI. The following example uses the AWS CLI:

   ```
   aws iam tag-role \
       --role-name YourMonitorUserRole \
       --tags Key=team,Value=YourTeam Key=project,Value=YourProject
   ```

   Choose tag keys that align with your organizational structure, such as `team`, `project`, or `cost-center`.
2. Activate the tags for cost allocation:

   a. Open the Billing console.

   b. In the navigation pane, under **Cost Organization**, choose **Cost Allocation Tags**.

   c. Find and select your IAM principal tags, then choose **Activate**.
3. In (), filter or group by your activated IAM principal tags to view costs.

   a. Choose **Tag** as the grouping dimension.

   b. Select your tag key.

After you apply tags to your IAM roles, it can take up to 24 hours for the tag keys to appear on the cost allocation tags page. It can then take up to 24 hours for the tags to activate.

For line-item cost attribution, create a AWS Cost and Usage Report () data export and select **Include caller identity (IAM principal) allocation data**. The export includes a `line_item_iam_principal` column that records the IAM ARN for each Amazon Bedrock request, along with your IAM principal tags prefixed with `iamPrincipal/`.

For more information, see Using IAM principal for cost allocation in the *AWS Billing User Guide*.

Troubleshooting

This section describes common issues you might encounter when using the Deadline Cloud assistant and how to resolve them.

Assistant returns errors or is slow to respond

The assistant displays error messages or takes a long time to generate responses.

Common cause

Errors or slow responses typically occur when your Amazon Bedrock service quotas are exceeded or when the monitor user role is missing required permissions.

Resolution

- Check your Amazon Bedrock service quotas in your Region. See Amazon Bedrock quotas for the Deadline Cloud assistant for details on checking and increasing your quotas.
- If you receive throttling errors, request a service quota increase for Amazon Bedrock in your Region.
- Verify that your monitor user role has the required permissions listed in Required permissions.

Assistant cannot access Amazon CloudWatch logs

The assistant cannot retrieve or analyze log data from CloudWatch.

Common cause

The assistant cannot retrieve logs when the queue role or fleet role lacks the required CloudWatch Logs read permissions.

Resolution

The assistant assumes the queue role (by using `deadline:AssumeQueueRoleForRead`) to read task logs and the fleet role (by using `deadline:AssumeFleetRoleForRead`) to read worker logs. Verify that these roles have CloudWatch Logs read permissions.

Additional resources

The following resources provide additional information related to the Deadline Cloud assistant:

- What is Amazon Bedrock? in the *Amazon Bedrock User Guide*
- Data protection in the *Amazon Bedrock User Guide*
- Abuse detection in the *Amazon Bedrock User Guide*
- Model invocation logging in the *Amazon Bedrock User Guide*
- Cross-region inference in the *Amazon Bedrock User Guide*
- Monitor Amazon Bedrock API calls using CloudTrail in the *Amazon Bedrock User Guide*
- Amazon Bedrock pricing
- Using IAM principal for cost allocation in the *AWS Billing User Guide*
- Service control policies in the *Organizations User Guide*
- Identity-based policy examples for Deadline Cloud
- AWS Responsible AI Policy
- AWS Data Privacy FAQ

Monitoring AWS Deadline Cloud

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS Deadline Cloud (Deadline Cloud) and your AWS solutions. Collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. Before you start monitoring Deadline Cloud, you should create a monitoring plan that includes answers to the following questions:

- What are your monitoring goals?
- Which resources will you monitor?
- How often will you monitor these resources?
- Which monitoring tools will you use?
- Who will perform the monitoring tasks?
- Who should be notified when something goes wrong?

AWS and Deadline Cloud provide tools that you can use to monitor your resources and respond to potential incidents. Some of these tools do the monitoring for you, some of the tools require manual intervention. You should automate monitoring tasks as much as possible.

- *Amazon CloudWatch* monitors your AWS resources and the applications you run on AWS in real time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. For example, you can have CloudWatch track CPU usage or other metrics of your Amazon EC2 instances and automatically launch new instances when needed. For more information, see the Amazon CloudWatch User Guide.

  Deadline Cloud publishes CloudWatch metrics for customer-managed fleet sizing, usage-based licensing, and resource limits. For the full list of metrics and recommended alarms, see Monitoring with CloudWatch in the *Deadline Cloud Developer Guide*.
- *Amazon CloudWatch Logs* enables you to monitor, store, and access your log files from Amazon EC2 instances, CloudTrail, and other sources. CloudWatch Logs can monitor information in the log files and notify you when certain thresholds are met. You can also archive your log data in highly durable storage. For more information, see the Amazon CloudWatch Logs User Guide.
- *Amazon EventBridge* can be used to automate your AWS services and respond automatically to system events, such as application availability issues or resource changes. Events from AWS

services are delivered to EventBridge in near real time. You can write simple rules to indicate which events are of interest to you and which automated actions to take when an event matches a rule. For more information, see Amazon EventBridge User Guide.

- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the AWS CloudTrail User Guide.

For more information, see the following topics in the *Deadline Cloud Developer Guide*:

- CloudTrail logs
- Managing events using EventBridge
- Monitoring with CloudWatch

Service quotas and throttling for Deadline Cloud

AWS Deadline Cloud provides resources, such as farms, fleets, and queues, that you can use to process jobs. When you create your AWS account, we set default quotas on these resources for each AWS Region. Deadline Cloud also limits the rate of API requests. For more information, see API request throttling.

Service Quotas is a central location where you can view and manage your quotas for AWS services. You can also request a quota increase for many of the resources that you use.

To view the quotas for Deadline Cloud, open the Service Quotas console. In the navigation pane, choose **AWS services** and select **Deadline Cloud**.

To request a quota increase, see Requesting a quota increase in the *Service Quotas User Guide*. If the quota is not yet available in Service Quotas, use the service quota increase form.

Your AWS account has the following quotas related to Deadline Cloud. If you're scaling up an existing farm, see Plan quota increases as you scale your farm for the quotas to review together before you request increases.

The *associated members* quotas count the memberships that you assign to a farm, fleet, queue, or job. Each user grant and each group grant counts as one membership, so a group counts as one member no matter how many users it contains. Deadline Cloud doesn't limit the number of users in a group or the number of groups that a user belongs to; the quotas for AWS IAM Identity Center apply instead. To grant access to more users within the membership quota, assign groups instead of individual users. For more information, see How permissions work in Deadline Cloud.

Compute for service-managed fleets counts against the Deadline Cloud vCPU and GPU quotas in the following table, not against your Amazon Elastic Compute Cloud (Amazon EC2) service quotas. For more information, see Quotas for related services.

The following table includes quotas for persistent storage volumes used by service-managed fleets. For more information about persistent storage, see Persistent storage for service-managed fleets.

Name	Default	Adjuse	Description
Associated members per farm	Each supported Region: 75	No	The maximum number of principals (users or

Name	Default	Adjuse	Description
			groups) that can be associated to each farm in the current AWS Region. Each associated group counts as one member, regardless of how many users it contains.
Associated members per fleet	Each supported Region: 75	No	The maximum number of principals (users or groups) that can be associated to each fleet in the current AWS Region. Each associated group counts as one member, regardless of how many users it contains.
Associated members per job	Each supported Region: 75	No	The maximum number of principals (users or groups) that can be associated to each job in the current AWS Region. Each associated group counts as one member, regardless of how many users it contains.

Name	Default	Adjuse	Description
Associated members per queue	Each supported Region: 75	No	The maximum number of principals (users or groups) that can be associated to each queue in the current AWS Region. Each associated group counts as one member, regardless of how many users it contains.
Budgets per farm	Each supported Region: 20	Yes	The maximum number of budgets per farm in the current AWS Region
Farms per region	Each supported Region: 2	Yes	The maximum number of farms that can be created in the current AWS Region.
Fleets per farm	Each supported Region: 5	Yes	The maximum number of fleets that can be created for each farm in the current AWS Region.
Jobs per farm	Each supported Region: 100,000	Yes	The maximum number of jobs per farm in the current AWS Region.
License endpoints per region	Each supported Region: 5	Yes	The maximum number of license endpoints in the current AWS Region.

Name	Default	Adjustable	Description
License sessions per license endpoint	Each supported Region: 500	Yes	The maximum number of license sessions per license endpoint in the current AWS Region.
Limits per farm	Each supported Region: 50	Yes	The maximum number of limits that can be created for each farm in the current AWS Region.
Monitors per region	Each supported Region: 1	No	The maximum number of monitors in the current AWS Region.
OnDemand G instance GPUs per region	Each supported Region: 0	Yes	The maximum number of on-demand G instance GPUs that can be provisioned across all service-managed fleets in the current AWS Region.
OnDemand vCPUs per region	Each supported Region: 50	Yes	The maximum number of on-demand vCPUs that can be provisioned across all service-managed fleets in the current AWS Region.
Queue environments per queue	Each supported Region: 10	No	The maximum number of queue environments that can be created for each queue in the current AWS Region.

Name	Default	Adjustable	Description
Queue fleet associations per farm	Each supported Region: 100	Yes	The maximum number of queue fleet associations per farm in the current AWS Region
Queue limit associations per queue	Each supported Region: 10	Yes	The maximum number of limits that can be associated with each queue in the current AWS Region.
Queues per farm	Each supported Region: 20	Yes	The maximum number of queues that can be created for each farm in the current AWS Region.
Resource configurations per fleet	Each supported Region: 1	Yes	The maximum number of VPC Lattice resource configurations that can be added to each fleet.
Spot G Instance GPUs per region	Each supported Region: 0	Yes	The maximum number of spot G instance GPUs that can be provisioned across all service-managed fleets in the current AWS Region.
Spot vCPUs per region	Each supported Region: 50	Yes	The maximum number of spot vCPUs that can be provisioned across all service-managed fleets in the current AWS Region.

Name	Default	Adjustable	Description
Step consumers per step	Each supported Region: 32	Yes	The maximum number of steps that may declare a dependency on (consume) a single step within a job in the current AWS Region.
Step dependencies per step	Each supported Region: 128	Yes	The maximum number of steps that a single step within a job may declare a dependency on in the current AWS Region.
Steps per job	Each supported Region: 200	Yes	The maximum number of steps per job in the current AWS Region.
Storage for General Purpose SSD (gp3) volumes, in TiB	Each supported Region: 1	Yes	The maximum aggregated amount of EBS storage, measured in TiB, that can be used across all fleets in the current AWS Region.
Storage for persistent General Purpose SSD (gp3) volumes, in TiB	Each supported Region: 4	Yes	The maximum aggregated amount of storage, in TiB, that can be provisioned across persistent General Purpose SSD (gp3) volumes in the current AWS Region.

Name	Default	Adjustable	Description
Storage profiles per farm	Each supported Region: 50	No	The maximum number of storage profiles that can be created for each farm in the current AWS Region.
Tasks per chunk	Each supported Region: 150	No	The maximum number of tasks that can be combined into a single chunk when submitting a job.
Tasks per job	Each supported Region: 10,000	Yes	The maximum number of tasks per job in the current AWS Region.
Tasks per step	Each supported Region: 10,000	Yes	The maximum number of tasks per step in the current AWS Region.
Wait-and-save vCPUs per region	Each supported Region: 50	Yes	The maximum number of wait-and-save vCPUs that can be provisioned across all service-managed fleets in the current AWS Region.
Workers per farm	Each supported Region: 7,500	Yes	The maximum number of workers per farm in the current AWS Region.

Plan quota increases as you scale your farm

Scaling up a farm can require increases to several related quotas. Review these quotas as a set and request any needed increases together. Quota limits fall into two groups: limits that block fleet creation or updates, and limits that appear only when jobs run. The second group includes license session limits, which are easy to miss because fleet setup can succeed even when a job can't get a license.

The following table maps what you're scaling to the quotas that apply and when each limit takes effect. For the full list of quotas and their current values in your account, see the table in Service quotas and throttling for Deadline Cloud.

What you're scaling	Quotas that apply	When the limit takes effect
Compute in service-managed fleets	vCPU and GPU quotas. Each purchasing option (on-demand, spot, and wait and save) has its own quota.	Creating or updating the fleet fails.
Worker storage in service-managed fleets	Root volume storage, and persistent volume storage if the fleet uses Persistent storage. Both grow with the same worker count as the compute quotas.	Creating or updating the fleet fails.
Workers in a farm	Workers for each farm. Counts workers in both service-managed and customer-managed fleets.	Creating or updating the fleet fails.
Jobs that use usage-based licensing	License sessions for each license endpoint.	Tasks fail to get a license at render time.
Compute in customer-managed fleets	Your Amazon Elastic Compute Cloud (Amazon EC2) quotas, because those workers run in your own account.	Instance launches fail in your account.

Request related increases together

The compute and storage quotas grow with the same maximum worker count, so a fleet that needs a vCPU or GPU increase often needs a storage increase as well. If you request only the compute increase, creating the fleet fails again on the storage quota and you wait through a second request. When you plan an increase, review the following quotas as a set:

- The vCPU or GPU quota for the fleet's purchasing option.
- Root volume storage.
- Persistent volume storage, if the fleet uses persistent volumes.
- Workers for each farm.
- License sessions for each license endpoint, if your jobs use usage-based licensing.

The Deadline Cloud console helps you catch these limits early. When a fleet configuration exceeds a quota, the create and edit fleet pages show a warning that names the quota, explains the calculation, and links to the request page.

Quotas apply for each AWS Region, so a farm in a new Region starts from the default values. Increase requests can take time to process, so request them ahead of a deadline rather than during one. To view your applied values and request increases, see Service quotas and throttling for Deadline Cloud.

How Deadline Cloud sizes a fleet against quotas

Deadline Cloud counts a fleet against the compute quotas at its worst case: the maximum worker count multiplied by the largest worker the configuration can launch. If the fleet allows specific instance types, the largest allowed instance type that fits the fleet's other capability ranges sets the size for each worker. Otherwise, the configured maximum vCPU or GPU count sets it. A fleet that sets neither is assumed to use the largest supported worker, so it can count far more vCPUs against your quota than its workers ever use.

Usage is then added up across all of your service-managed fleets in the Region. To reduce how much a fleet counts against your quotas, restrict its allowed instance types or narrow its vCPU, memory, and GPU ranges. For more information about fleet configuration, see Deadline Cloud fleets.

License sessions appear only at render time

If your jobs use usage-based licensing, each worker that renders with a licensed product checks out a license session. Deadline Cloud doesn't check the license session quota when you create a fleet, so a fleet sized past it is created successfully. When the fleet scales up, workers beyond the limit start but can't get a license, and their tasks fail.

The license session quota applies only to usage-based licensing. If your workers check out licenses from your own license server, or run software that needs no license, the quota doesn't affect them. Customer-managed fleets usually bring their own licenses.

If your fleet can run more workers than the quota allows, request an increase or lower the fleet's maximum worker count. For a comparison of usage-based licensing and bringing your own licenses, see Using software licenses with Deadline Cloud in the *AWS Deadline Cloud Developer Guide*.

API request throttling

Deadline Cloud limits the rate of API requests for each AWS account in each AWS Region. The default request rates support large-scale workloads and usage. When your requests exceed the allowed rate, Deadline Cloud rejects the request with an HTTP 429 status code and a `ThrottlingException` error.

The AWS SDKs and the AWS CLI automatically retry throttled requests using exponential backoff. Occasional throttling resolves without requiring changes to your application. If you experience sustained throttling, contact AWS Support to request higher request rates.

Quotas for related services

Some Deadline Cloud features use other AWS services, and the quotas for those services also apply.

Amazon EC2 quotas for customer-managed fleets

Your Amazon Elastic Compute Cloud (Amazon EC2) quotas apply to customer-managed fleets, because those workers run on instances in your own account. Compute for service-managed fleets counts against the Deadline Cloud vCPU and GPU quotas in the preceding table instead.

To view or increase your Amazon EC2 quotas, open the Service Quotas console and choose **Amazon Elastic Compute Cloud**. For more information, see Amazon EC2 service quotas in the *Amazon EC2 User Guide*.

Amazon Bedrock quotas for the Deadline Cloud assistant

The Deadline Cloud assistant uses Amazon Bedrock on-demand inference in your AWS account, so your account's Amazon Bedrock service quotas apply. The two primary constraints are:

- **Requests per minute (RPM)** – The number of model invocation requests allowed per minute.
- **Tokens per minute (TPM)** – The total number of input and output tokens processed per minute.

Default quotas vary by Region. Some Regions have lower default limits, as low as 20 RPM, which might result in throttling during heavy assistant usage. The assistant uses cross-region inference profiles, which support a minimum of 200 RPM. Cross-region inference can help alleviate throttling in Regions with lower single-Region limits. If your Region uses cross-region inference, the service quotas in the destination Regions also apply.

If you experience throttling errors when using the assistant, you can request an Amazon Bedrock service quota increase:

To request a quota increase

1. Open the Service Quotas console.
2. In the navigation pane, choose **AWS services**, then choose **Amazon Bedrock**.
3. Find the quota for the model used by the assistant (look for quotas related to `InvokeModelWithResponseStream` for the relevant model).
4. Choose the quota name, then choose **Request increase at account level**.
5. Enter your desired quota value and submit the request.

You can monitor your Amazon Bedrock quota usage through CloudWatch metrics. Set up CloudWatch alarms on Amazon Bedrock throttling metrics to identify when you are approaching your quota limits. For more information, see Monitoring Amazon Bedrock in the *Amazon Bedrock User Guide*.

Creating AWS Deadline Cloud resources with AWS CloudFormation

AWS Deadline Cloud is integrated with AWS CloudFormation, a service that helps you to model and set up your AWS resources so that you can spend less time creating and managing your resources and infrastructure. You create a template that describes all the AWS resources that you want (such as farms, queues, and fleets), and CloudFormation provisions and configures those resources for you.

When you use CloudFormation, you can reuse your template to set up your Deadline Cloud resources consistently and repeatedly. Describe your resources once, and then provision the same resources over and over in multiple AWS accounts and Regions.

Deadline Cloud and CloudFormation templates

To provision and configure resources for Deadline Cloud and related services, you must understand CloudFormation templates. Templates are formatted text files in JSON or YAML. These templates describe the resources that you want to provision in your CloudFormation stacks. If you're unfamiliar with JSON or YAML, you can use CloudFormation Designer to help you get started with CloudFormation templates. For more information, see What is CloudFormation Designer? in the *AWS CloudFormation User Guide*.

Deadline Cloud supports creating farms, queues, and fleets in CloudFormation. For more information, including examples of JSON and YAML templates for farms, queues, and fleets, see the AWS Deadline Cloud in the *AWS CloudFormation User Guide*.

Learn more about CloudFormation

To learn more about CloudFormation, see the following resources:

- AWS CloudFormation
- AWS CloudFormation User Guide
- CloudFormation API Reference
- AWS CloudFormation Command Line Interface User Guide

Troubleshooting

The following procedures and tips can help you troubleshoot issues with your AWS Deadline Cloud farms and resources.

Topics

Why can a user not see my farm, fleet, or queue?

User access

When your users are not seeing your farms, fleets, or queues in the Deadline Cloud monitor, there might be an issue with their access to your farm and resources.

Users without access to any farms receive the message "No farms available" in the Deadline Cloud monitor.

To confirm you have the correct user or group assigned to your farm, fleet, or queue

1. In the AWS Deadline Cloud console, find your farm, fleet, or queue, and then choose **Access management**.
2. The groups tab is selected by default. If you're assigning permissions by groups, which is recommended, your group should display in the list and have an assigned access level.

   If the group is not in the list, choose **Add group** to assign permission for the group.
3. If you're assigning permissions by user, select the **Users** tab. Your user should display in the list and have an assigned access level.

   If your user is not in the list, choose **Add user** to assign permission for the user.

To confirm you have the user assigned to your group

1. In the AWS Deadline Cloud console, find your farm, fleet, or queue, and then choose **Access management**.
2. The groups tab is selected by default. Select the group name to view its members.
3. If the user is not listed in the group, they must be added.

   If you're using the default identity setup, you can directly add the user to the group in the Identity Center console. If you're connected to an external identity provider such as Okta or Google Workspace, you can add your user to the group in your identity provider.

   > **Note**
   >
   > Some external identity providers sync users but not groups to Identity Center. In this case, consider assigning permissions to a user directly instead of by group.

For more information about managing user access to Deadline Cloud, see Managing users in Deadline Cloud.

Why are workers not picking up my jobs?

Fleet role configuration

Sometimes when workers are created but do not complete initialization and do not start working on jobs, it's because the fleet role was not configured correctly.

To verify this cause, check your CloudTrail logs for any access denied errors. After you confirm the access denied issue, go to your fleet and update the role configuration to the correct permissions. For more information, see CloudTrail logs in the Deadline Cloud developer guide.

Why is my worker stuck running?

Worker stuck exiting OpenJD environment

Workers can get stuck in long-running `envExit` session actions. This problem might happen if you use a job template that overrides the OpenJD template and sets the environment exit actions timeout to more than 5 minutes. The Deadline Cloud monitor provides some visibility into workers

stuck in this situation, but it requires cross-referencing RUNNING workers against available work in the associated queues.

To find stuck workers, go through all fleets in the Deadline Cloud monitor and complete the following steps:

1. In the worker status column, find RUNNING workers.
2. From the Fleet details section, navigate to each associated queue.
3. In each associated queue, search for jobs that are RUNNING, READY, or PENDING. If all associated queues don't have any jobs in those states, then the worker is running an environment exit.

To stop a worker stuck in this state, use the following AWS CLI command:

```
aws deadline update-worker \
    --farm-id $FARM_ID     \
    --fleet-id $FLEET_ID   \
    --worker-id $WORKER_ID \
    --status STOPPED
```

After running the command, the worker agent restarts when the program exits. Workers then come back online and run more jobs from associated queues. If the queue contains more jobs with environment exit action timeouts longer than 5 minutes, the worker will get stuck again. If this happens, you will need to repeat this process until no more workers are stuck exiting.

To avoid this issue, set the timeout option to no more than 5 minutes when using a job template.

Troubleshooting Deadline Cloud jobs

For information about common problems with jobs in AWS Deadline Cloud, see the following topics.

Why did creating my job fail?

Quota validation

Some possible reasons that a job can fail validation checks include the following:

- The job template doesn't follow the OpenJD specification.

- The job contains too many steps.
- The job contains too many total tasks.
- There was an internal service error that prevents the job from being created.

To see the quotas for the maximum number of steps and tasks in a job, use the Service Quotas console. For more information, see Service quotas and throttling for Deadline Cloud.

CHUNK[INT] task parameter error

If job creation fails with the following error message, you need to add the `TASK_CHUNKING` extension to your job template.

```
The CHUNK[INT] task parameter requires the TASK_CHUNKING extension.
```

To resolve this issue, add the following to your job template:

```
extensions:
  - TASK_CHUNKING
```

For more information, see Add task chunking to a job template in the *AWS Deadline Cloud Developer Guide*.

Why is my job not compatible?

Common reasons that jobs are not compatible with queues include the following:

- No fleets are associated with the queue that the job was submitted to. Open the Deadline Cloud monitor, and check that the queue has associated fleets. For more information about how to view queues, see View queue and fleet details in Deadline Cloud.
- The job has host requirements that are not satisfied by any of the fleets associated with the queue. To check, compare the `hostRequirements` entry in the job template with the configuration of the fleets in your farm. Make sure that one of the fleets satisfies the host requirements. For more information about fleet compatibility, see Determine fleet compatibility. To view fleet configuration, see View queue and fleet details in Deadline Cloud.

Deadline Cloud checks compatibility when you submit the job. If you associate a compatible fleet with the queue later, existing `NOT_COMPATIBLE` jobs don't automatically restart. To run those jobs, requeue them. For more information, see Requeue a job.

Why is my job stuck in ready?

Possible reasons for your job appearing to be stuck in the READY state include the following:

- The maximum worker count for fleets associated with the queue is set to zero. To check, see View queue and fleet details in Deadline Cloud.
- There is a higher priority job in the queue. To check, see View queue and fleet details in Deadline Cloud.
- For customer-managed fleets, check the auto scaling configuration. For more information, see Create fleet infrastructure with an Amazon EC2 Auto Scaling group in the *Deadline Cloud Developer Guide*.

Why did my job fail?

A job can fail for many reasons. To search for the issue, open the Deadline Cloud monitor and choose the failing job. Choose a task that failed and then view the logs for the task. For instructions, see View session and worker logs in Deadline Cloud.

- If you see license errors or if you get a watermark that occurs because the software doesn't have a valid license, make sure that the worker can connect to the required license server. For more information, see Connect customer-managed fleets to a license endpoint in the *Deadline Cloud Developer Guide*.
- The last session action message or the process exit code may provide information about why you job failed. If you are using Windows and your exit code is negative, try searching for the unsigned version of your exit code:

```
2,147,483,647 - |your exit code|
```

Why does my job fail on Windows when my file paths are long?

Jobs can fail on Windows workers when your project uses long paths, even though the same files render on your workstation. The files transfer correctly, and the failure happens when the rendering application opens one. Symptoms include the following:

- The rendering application can't find the scene file or one of its dependencies, even though the file was uploaded with the job.

- A task log contains `FileNotFoundError`, or a `[WinError 3] The system cannot find the path specified` message naming a long path.
- The job succeeds when you shorten the scene file name and change nothing else.

Windows limits most file paths to 260 characters. Two things make that limit easier to reach on a render farm than on a workstation:

- Long path support is per application, not per machine. Windows honors the `LongPathsEnabled` registry setting only for applications that declare `longPathAware` in their application manifest, so enabling it on a worker host doesn't lift the limit for an application that doesn't declare it.
- macOS and Linux allow up to 1,024 characters, so submitting from one of those workstations to a Windows fleet can produce paths that are valid where you created them and too long where they render.

Deadline Cloud applies the Windows extended-length path prefix to its own file operations, so transfer and upload aren't affected. It can't apply that prefix inside a rendering application that opens files with its own code, which is why the failure appears at render time. Session and job attachment directory names also use part of the 260 characters.

To resolve the failure, use one or more of the following approaches, starting with the most reliable:

1. Shorten the paths in your project. This approach is the only one that works for every application, because it avoids the limit instead of working around it.
2. Use a queue environment that shortens the path the application sees. Deadline Cloud publishes a sample that junctions the job attachment directory to a short path and supplies matching path mapping rules. See windows_path_limit_junction_fix.yaml in the `deadline-cloud-samples` repository on the GitHub website. It takes effect only for integrations that read those rules.
3. Render on a Linux fleet if your software is available there. Linux isn't subject to the limit.

> **Note**
>
> These approaches enlarge the path budget available to your project rather than removing the 260-character limit. A long enough path still fails.

Why is my step pending?

Steps may stay in the PENDING state when one or more of their dependencies are not complete. You can check the state of dependencies using the Deadline Cloud monitor. For instructions, see View a step in Deadline Cloud.

Deadline Cloud monitor desktop application logs

The Deadline Cloud monitor desktop application writes diagnostic logs that you can use to investigate crashes or other unexpected behavior. When reporting an issue with the desktop application, include the relevant log files to help with diagnosis.

The location of the log files depends on your operating system:

Windows

```
%APPDATA%\com.amazonaws.deadline.monitor\logs
```

macOS

```
~/Library/Logs/com.amazonaws.deadline.monitor/
```

Linux

```
~/.config/com.amazonaws.deadline.monitor/logs
```

Additional resources

You can find additional information and resources in the aws-deadline repositories on the GitHub website.

Deadline Cloud release notes

This page contains information about the latest releases and updates to AWS Deadline Cloud.

Date	Title	Description
2026-08-31	Deadline Cloud Monitor adds a step dependency graph view	In the Deadline Cloud Monitor, you can now switch the Steps panel between a table view and a graph view. The graph shows each step as a node and its dependencies as edges, so you can quickly see the order of execution. Choose a node to select its step, or choose View step dependencies to filter the panel to related steps.
2026-08-24	Share and browse job bundles in the Deadline Cloud client	You can now share job bundles with everyone on a queue using Deadline Cloud client 0.60.5. A new job bundle browser in the job submitter lets you browse and preview bundles from your queue, your local filesystem, or your job submission history. New deadline bundle CLI commands let you upload, download, and list shared bundles so you can integrate bundle sharing into pipeline scripts.

Date	Title	Description
2026-08-21	Automatic download status tracking in Deadline Cloud Monitor	With the Deadline Cloud Monitor desktop applicati on, you can now see the progress, status, and health of automatic file downloads from your jobs. A new Download status column at the job and task level shows download progress. It confirms when all output files are available on your destination drive, and an indicator displays how recently the status was updated. A queue-level output sync indicator also shows the overall download status for your queue and surfaces any problems. If files are unavailable for any reason, the monitor provides guidance on next steps. You can also choose Troublesh oot with AI to diagnose download problems with the Deadline Cloud Assistant. You no longer need to manually verify files on your drive before you start the next steps in your workflow.

Date	Title	Description
2026-08-13	Deadline Cloud monitor adds an inline logs panel	In the Deadline Cloud monitor, you can now add a Logs panel to the job monitor page. To add it, choose Add panel, then choose Logs. The panel shows logs for the selected task, and you can switch to worker logs, choose a retry attempt, or jump to a session action.
2026-08-12	Job and task table improvements in Deadline Cloud monitor	The tasks table in the Deadline Cloud monitor now includes columns for each task's latest run. These columns show run status, status message, progress, exit code, worker, fleet, and instance type. You can identify and address retried or failed tasks without opening each task's run details. Additional columns such as host name, IP addresses, vCPU, memory, operating system, and GPU details are available in the table preferences. The jobs table also adds an optional Description column. Job, step, and task details throughout the job panel also load faster.

Date	Title	Description
2026-08-05	Usage explorer adds persistent volume storage costs	Deadline Cloud now reports persistent volume costs alongside compute and license costs. You can view per-fleet storage costs by selecting the Usage Type grouping in the Deadline Cloud monitor usage explorer. This helps you better understand your infrastructure costs.
2026-06-03	Plugin sync for Blender and Autodesk Maya	Service-managed fleets now support plugin sync for Blender and Autodesk Maya. Upload your plugin files to a prescribed path in your queue's job attachments S3 bucket. Deadline Cloud automatically syncs them to workers when a job session starts.
2026-06-02	Blender 5.1 support and LTS updates	Deadline Cloud now supports Blender 5.1. You can submit render and compositing jobs from Blender 5.1 to your Deadline Cloud render farm. Additionally, Blender 4.5 LTS has been updated from 4.5.5 to 4.5.10 and Blender 4.2 LTS from 4.2.16 to 4.2.21, incorporating upstream rendering fixes and stability improvements.

Date	Title	Description
2026-05-28	Persistent storage for service-managed fleets	You can now configure persistent storage on service-managed fleets to preserve data across worker lifecycle events. Persistent storage uses dedicated Amazon EBS gp3 volumes, separate from the root boot volume. These volumes maintain applicati on caches, conda package installations, and workspace s when the fleet recycles workers. For more informati on, see Persistent storage for service-managed fleets in the Deadline Cloud User Guide.
2026-05-28	SOC 1, 2, and 3 compliance	Deadline Cloud is now listed on the AWS Services in Scope page for System and Organization Controls (SOC) 1, 2, and 3 complianc e. Third-party audit reports are available for download using AWS Artifact. For more information, see Compliance validation for Deadline Cloud.

Date	Title	Description
2026-05-18	Faster conda queue environment setup on service-managed fleets	The default conda queue environment on service-managed fleets has been improved. New conda-queue-env-enter and conda-queue-env-exit commands have been added, which provide faster environment creation, with even greater improvements on repeated runs. For more details, see https://github.com/aws-deadline/deadline-cloud-samples/tree/mainline/queue_environments.

Date	Title	Description
2026-05-14	Deadline Cloud Monitor 1.1.9	This release of the Deadline Cloud Monitor desktop application includes the following changes: • Upgrades the Deadline client to support filtering job attachment input and output. • Fixes a bug that prevented some users from being prompted to update to the latest version upon logging in. • Improves error messaging when the Deadline Cloud Monitor fails to start due to a missing system dependency.
2026-05-11	In-app release notes in Deadline Cloud Monitor	You can now browse product updates directly in the Deadline Cloud Monitor. The release notes page displays new features, fixes, and documentation updates with category filtering and search, so you can find relevant changes without leaving the Monitor. Access the release notes from the secondary navigation menu at the bottom of the side navigation in the Monitor.

Date	Title	Description
2026-04-29	NVIDIA RTX PRO Server 6000 GPU accelerator support	You can now select the RTX PRO Server 6000 GPU accelerator when you create service-managed fleets. This accelerator uses the G7e instance family and provides 96 GiB of GPU memory per accelerator. To use this accelerator, choose rtx-pro-server-6000 for the accelerator name and grid:r580 for the driver version when you configure your fleet.
2026-04-20	Deadline Cloud Monitor 1.1.8	This release of the Deadline Cloud Monitor desktop application includes the following changes: • Upgrades the Deadline client so that you can work with tasks in chunks. • Bundles runtime dependencies for faster job submission. • Removes the long splash animation for a quicker startup. • Fixes several minor bugs. • Includes improved translations for non-English languages.

Date	Title	Description
2026-04-17	Deadline Cloud Assistant for intelligent job troubleshooting	You can now use the Deadline Cloud Assistant, an AI-powered troubleshooting assistant built into the Deadline Cloud monitor, to diagnose render job failures. The assistant analyzes job configurations, related Amazon CloudWatch logs, and documentation for supported digital content creation applications. It runs in your browser and uses Amazon Bedrock in your AWS account, so your data stays within your security boundary. To get started, select Enable Deadline Cloud Assistant in Monitor settings in the AWS Management Console.
2026-04-10	Support for monitor creation in multiple Regions	You can now connect a Deadline Cloud monitor to an IAM Identity Center instance in a different Region without any additional IAM Identity Center configuration. Deadline Cloud reads identity data from the Region where your instance is located.

Date	Title	Description
2026-04-06	New batch APIs simplify bulk resource operations	With 8 new Batch APIs, you can retrieve or update multiple resources in a single API call, reducing overhead for bulk operations. The following are the new Batch Get APIs: • BatchGetJob • BatchGetSession • BatchGetSessionAction • BatchGetStep • BatchGetTask • BatchGetWorker The following are the new Batch Update APIs: • BatchUpdateJob • BatchUpdateTask
2026-04-03	Foundry Nuke 17 support	Deadline Cloud now supports Foundry Nuke 17. You can submit render and compositi ng jobs from Nuke 17 to your Deadline Cloud render farm.

Date	Title	Description
2026-04-03	NVIDIA GRID R580 support for GPU fleets	You can now use NVIDIA Virtual GPU Software v19.0 (GRID R580) on Deadline Cloud service-managed fleets with both Windows Server 2022 and Amazon Linux 2023. To use this software version, when you create a GPU-accelerated service-managed fleet, choose grid:r580 for Driver version.
2026-04-02	Unreal Engine 5.7 support	You can now use Epic Games Unreal Engine 5.7 with Deadline Cloud. You can submit jobs from Windows and use conda packages on Windows service-managed fleets.
2026-04-02	Balanced scheduling support	You can now control how Deadline Cloud distributes jobs to workers by setting a scheduling configuration on each queue.

Date	Title	Description
2026-03-30	Auto scaling configuration for fleets	Deadline Cloud now supports three new fleet auto scaling settings. With scale out rate, you can configure how quickly workers launch. With worker idle duration, you can set how long workers wait before shutting down. With standby worker count, you can keep idle workers ready so that jobs start quickly.
2026-03-25	Cost scale factors for farms	You can now configure a cost scale factor on your farms to model costs in Usage Explorer and Budget Manager. You can apply discounts or premiums to your farm's cost calculations to align Deadline Cloud usage data with your organization's actual costs.
2026-03-23	Submitter installer v2026-03-23 updates Maya submitter	A new submitter installer updates the following components: • Maya: 0.15.13 → 0.15.14 (release notes) The installer now bundles GUI dependencies for the Deadline Cloud client, enabling a full installation without internet access.

Date	Title	Description
2026-03-16	Estimated time remaining for in-progress jobs	To display estimated time remaining for in-progre ss jobs, use the --show-es timates flag with the deadline job list and deadline job get commands. The estimate uses the average task completion time and displays the result as a human-readable duration.
2026-03-13	After Effects 25.6 and 26.0 support	Adobe After Effects versions 25.6 and 26.0 are now supported. Submitter support is available for Windows and macOS, with conda packages available for Windows service-managed fleets.
2026-03-11	Submitter installer v2026-03-11 released	A new submitter installer has been released which updates the following components: • 3ds-max: 0.1.9 → 0.1.10 (release notes) • after-effects: 0.4.4 → 0.4.5 (release notes) • cinema-4d: 0.9.2 → 0.10.0 (release notes) • deadline-cloud: 0.54.1 → 0.54.2 (release notes) • houdini: 0.7.10 → 0.7.11 (release notes)

Date	Title	Description
2026-03-10	Blender 5.0 support	Deadline Cloud now supports Blender 5.0 with all built-in render engines including Cycles, Eevee, and Workbench . Submitter support is available for Windows, macOS, and Linux, with conda packages available for Linux service-managed fleets.
2026-03-02	Submitter installer v2026-03-02 updates Blender and core components	A new submitter installer has been released which updates the following components: • blender: 0.6.0 → 0.6.1 (release notes) • deadline-cloud: 0.54.0 → 0.54.1 (release notes)
2026-02-24	Deadline Cloud documentation now includes supported software user guide	Deadline Cloud user guide now includes dedicated subpages for each supported application, providing in-depth details on version compatibility and feature support.
2026-02-24	Deadline Cloud Monitor usage explorer groups usage by user	Analyze usage patterns per user and attribute costs across your team with the usage explorer.

Date	Title	Description
2026-02-24	Task chunking improves rendering efficiency	Deadline Cloud now supports task chunking, which groups multiple frames into a single task run. This feature reduces overhead by loading the application and scene once per chunk instead of once per frame. You can specify a default chunk size or let Deadline Cloud dynamically adjust chunk sizes based on a target runtime.
2026-02-19	Submitter installer v2026-02-19 updates Maya submitter	A new submitter installer has been released which updates the Autodesk Maya submitter from 0.15.12 to 0.15.13.
2026-02-13	OpenJD specifications add Claude and Kiro skill for RFC review	The Open Job Description specifications repository now includes a Kiro skill for AI-assisted review of RFC proposals, checking for completeness, clarity, tenet alignment, and compatibility with existing specifications.
2026-02-13	Deadline Cloud for 3ds Max introduces Kiro powers for AI-assisted development	The Deadline Cloud for 3ds Max repository now includes Kiro powers that provide AI-assisted setup, design, and development workflows with built-in guardrails and best practices.

Date	Title	Description
2026-02-06	Jobs support tagging for access control	Job resources now support tagging and Attribute-Based Access Control (ABAC). IAM policies can reference job tags using condition keys, enabling tag-based authorization patterns — for example, restricting GetJob API calls to jobs with a particular team tag.
2026-02-05	Multi-region replication support through IAM Identity Center	Deadline Cloud now supports IAM Identity Center's multi-region replication feature, giving studios more flexibility in where they set up Deadline Cloud relative to their Identity Center instance. Studios can create a Deadline Cloud farm in regions that align with their rendering needs while administrators continue managing Identity Center from a primary region.

Date	Title	Description
2026-02-04	New sample job bundle demonstrates FLUX.2 Klein LoRA training	A sample job bundle is now available that demonstrates how to train custom LoRA adapters on the FLUX.2 Klein model using 20-50 images. This enables you to create personalized image generators for products, characters, or brand assets without requiring deep machine learning expertise. The LoRA fine-tuning approach creates small, portable model adapters that are efficient to train and easy to share across your team.
2026-01-29	New V-Ray Standalone tiled rendering job bundle available	A new job bundle for tiled rendering of exported V-Ray scenes is now available. This job bundle enables efficient rendering of high-resolution images by splitting them into tiles that can be processed in parallel across your render farm. You can now use 3ds Max and V-Ray to export V-Ray scenes locally and submit them using this bundle to Linux workers instead of needing to use Windows.

Date	Title	Description
2026-01-27	Edit job names and descriptions after submission	Deadline Cloud now supports editing job names and descriptions after submission. This new feature makes it easier to organize and identify jobs after submission by updating names or adding useful tracking details in the description field.
2026-01-22	Redshift 2026 support for Maya on service-managed fleets	Redshift 2026 is now supported on Linux service-managed fleets with Deadline Cloud for Maya.
2026-01-22	Train machine learning models with Foundry Nuke CopyCat	Deadline Cloud now integrates with Foundry Nuke CopyCat, enabling you to run ML training jobs for visual effects in the cloud. CopyCat learns adjustments from sample frames and applies them across entire sequences. Submit training jobs to your Deadline Cloud render farm, scale workloads in parallel, and free up your artist workstations.

Date	Title	Description
2026-01-15	SDKs introduce waiters for job completion	Deadline Cloud SDKs now include JobComplete and JobSucceeded waiters that simplify polling for job status. The JobComplete waiter polls until a job reaches any terminal state (SUCCEEDED, FAILED, or CANCELED), while the JobSucceeded waiter polls until a job succeeds. These waiters eliminate the need to write custom polling logic, making it easier to build automation workflows that depend on job completion.

Date	Title	Description
2026-01-15	Budgets support tagging for access control	Deadline Cloud customers can now apply tags to Budget resources and use Attribute-Based Access Control (ABAC) for fine-grained permissio ns management. This new capability allows customers to organize, manage, and control access to their Deadline Cloud budgets using tags, enabling consisten t authorization patterns across their AWS resources . Customers can now tag budgets during creation and use these tags in IAM policies to control who can access specific budgets based on tag values.
2026-01-15	Deadline Cloud Monitor search adds multi-select filtering	When using the Deadline Cloud monitor, you can now select up to 16 values for any search filter, including user names and job status. This allows you to quickly find jobs across multiple users or filter for several statuses at once. This functionality is also available in the Deadline Cloud API through the new StringListFilterExpression for Jobs, Steps, Tasks, and Workers.

Date	Title	Description
2026-01-07	Direct Deadline Cloud Monitor and submitter installer download links available	Users can now download the Deadline Cloud Monitor desktop application and submitter installer directly from the Deadline Cloud documentation. This enables users without AWS console access to download the software they need to get started with Deadline Cloud.
2025-12-19	Submitter installer v2025-12-19 released	A new submitter installer has been released which updates the following components: • cinema-4d: 0.9.0 → 0.9.2 (release notes) • deadline-cloud: 0.53.3 → 0.54.0 (release notes) • nuke: 0.18.13 → 0.18.14 (release notes)

Date	Title	Description
2025-12-17	Deadline Cloud Monitor 1.1.7 introduces integrated job submission	The latest Deadline Cloud Monitor desktop application release includes: • Support for submitting jobs directly from the Deadline Cloud Monitor desktop application. • Simpler workstation setup. • Improved proxy support. • Bug fixes for edge cases when reading and writing to and from the Deadline Cloud profile config file.
2025-12-11	Developer guide adds AI agent guidance	The Deadline Cloud developer guide now includes best practices for using AI agents to write job bundles, develop conda packages, and troubleshoot jobs more efficiently.
2025-12-10	Autodesk VRED submitter user guide launches	Documentation for the Deadline Cloud submitter for Autodesk VRED is now available. The guide covers how to install the submitter and submit rendering jobs to Deadline Cloud. This helps VRED users get started quickly with cloud rendering.

Date	Title	Description
2025-12-10	Developer guide adds LicensesInUse metric section	The Deadline Cloud developer guide now includes informati on about the LicensesInUse metric. This metric helps you monitor how many licenses your jobs are currently consuming across your fleets. You can use this informati on to optimize license usage and to avoid running out of licenses when scaling out workloads.
2025-12-10	Service-managed fleets add Cinema 4D 2026.1 support	Maxon Cinema 4D 2026.1 is now supported on Linux and Windows service-managed fleets. This release includes Redshift 2026.2.0. Cross-platform font rendering support has also been added for all versions of Cinema 4D. This release allows customers to use the latest Cinema 4D features. It also enables customers to use custom fonts on cross-platform setups, such as when submitting jobs from Windows while using the faster startup time and lower cost of Linux workers.

Date	Title	Description
2025-12-09	Autodesk Maya submitter expands setup and usage documentation	New setup and usage documentation added for the Deadline Cloud Submitter for Autodesk Maya.
2025-12-09	After Effects submitter 0.4.4 improves macOS installation and font support	The After Effects submitter now automatically installs to the user preferences directory on macOS, removing the need for manual installation. This release also adds support for most TrueType Collectio n (TTC) font files, allowing you to submit and render jobs that use these fonts. These improvements simplify setup and expand font compatibility for After Effects users.
2025-12-08	Release notes page launches in user guide	All major changes to Deadline Cloud features, applicati ons, integrations, samples and documentation will now be listed on the Release Notes page in the User Guide going forward. You can find previous major Deadline Cloud releases at AWS What's New and CLI/ Worker/integration-specific release notes in the repositor ies of the Deadline Cloud github organization

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.