



User Guide

# AWS Deadline Cloud



**Version latest**

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

# AWS Deadline Cloud: User Guide

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

---

# Table of Contents

<b>What is Deadline Cloud?</b> .....	<b>1</b>
Features of Deadline Cloud .....	1
Concepts and terminology .....	2
Getting started with Deadline Cloud .....	4
Accessing Deadline Cloud .....	4
Related services .....	5
How Deadline Cloud works .....	6
.....	6
Permissions in Deadline Cloud .....	6
Software support with Deadline Cloud .....	7
<b>Getting started</b> .....	<b>9</b>
Set up your AWS account .....	9
Set up your monitor .....	10
Step 1: Set up your monitor .....	10
Step 2: Define farm details .....	13
Step 3: Define queue details .....	14
Step 4: Define fleet details .....	15
Step 5: Configure worker requirements .....	16
Step 6: Define access levels .....	16
Step 7: Review and create .....	16
Set up a developer workstation .....	16
Step 1: Create farm .....	17
Step 2: Run the worker agent .....	21
Step 3: Submit and run jobs .....	23
Step 4: Run jobs with attachments .....	31
Step 5: Add a service-managed fleet .....	40
Step 6: Clean up farm resources .....	42
Set up the submitter .....	45
Step 1: Install the Deadline Cloud submitter .....	46
Step 2: Install and set up Deadline Cloud monitor .....	51
Step 3: Launch the Deadline Cloud submitter .....	52
Use the farm .....	55
<b>Using the monitor</b> .....	<b>56</b>
Share the Deadline Cloud monitor URL .....	56

Open the Deadline Cloud monitor .....	57
View queue and fleet details .....	58
View and manage jobs, steps, and tasks .....	59
View job details .....	60
View a step .....	61
View a task .....	62
View logs .....	62
Download finished output .....	64
<b>Farms .....</b>	<b>65</b>
Create a farm .....	65
Delete a farm .....	65
Edit a farm .....	66
<b>Queues .....</b>	<b>67</b>
Create a queue .....	67
Create a queue environment .....	69
Default Conda queue environment .....	69
Delete a queue .....	71
Edit a queue .....	71
Associate a queue and fleet .....	71
<b>Managing fleets .....</b>	<b>72</b>
Service-managed fleets .....	72
VFX platform .....	73
Customer-managed fleets .....	74
Create a CMF .....	75
Worker host setup .....	80
Manage access .....	86
Install software for jobs .....	88
Configure credentials .....	89
Create an AMI .....	90
Create fleet infrastructure .....	93
Connect to a license endpoint .....	103
<b>Managing users .....</b>	<b>107</b>
Manage users and groups for the monitor .....	107
Manage users and groups for farms, queues, and fleets .....	109
<b>Jobs .....</b>	<b>111</b>
Submitting jobs .....	112

More options for submitting jobs .....	114
Scheduling jobs .....	115
Determine fleet compatibility .....	116
Fleet scaling .....	117
Sessions .....	118
Step dependencies .....	119
Job states .....	121
Modifying jobs .....	124
Processing jobs .....	128
Troubleshooting jobs .....	129
Why did creating my job fail? .....	129
Why is my job not compatible? .....	130
Why is my job stuck in ready? .....	130
Why did my job fail? .....	130
Why is my step pending? .....	131
<b>Storage .....</b>	<b>132</b>
Job attachments .....	132
Encryption for job attachment S3 buckets .....	133
Managing job attachments in S3 buckets .....	134
Virtual file system .....	134
Shared storage .....	137
Storage profiles in Deadline Cloud .....	137
<b>Managing budgets and usage .....</b>	<b>139</b>
Cost assumptions .....	139
Using the budget manager .....	140
Prerequisite .....	141
Access budget manager .....	141
Create a budget .....	141
View a budget .....	142
Edit a budget .....	143
Deactivate a budget .....	143
Using the usage explorer .....	144
Prerequisite .....	144
Open the usage explorer .....	144
Use the usage explorer .....	144
<b>Security .....</b>	<b>148</b>

Data protection .....	149
Encryption at rest .....	150
Encryption in transit .....	150
Key management .....	150
Inter-network traffic privacy .....	160
Identity and Access Management .....	160
Audience .....	161
Authenticating with identities .....	161
Managing access using policies .....	165
How Deadline Cloud works with IAM .....	167
Identity-based policy examples .....	174
AWS managed policies .....	178
Troubleshooting .....	181
Compliance validation .....	183
Resilience .....	184
Infrastructure security .....	185
Configuration and vulnerability analysis .....	185
Cross-service confused deputy prevention .....	186
Security best practices .....	187
Data protection .....	188
IAM permissions .....	188
Run jobs as users and groups .....	188
Networking .....	189
Job data .....	189
Farm structure .....	190
Job attachment queues .....	190
Custom software buckets .....	192
Worker hosts .....	193
Workstations .....	194
<b>Monitoring .....</b>	<b>196</b>
Logging with CloudTrail .....	197
Deadline Cloud information in CloudTrail .....	197
Understanding Deadline Cloud log file entries .....	201
Monitoring with CloudWatch .....	203
<b>Quotas .....</b>	<b>205</b>
<b>AWS CloudFormation resources .....</b>	<b>206</b>

---

Deadline Cloud and AWS CloudFormation templates .....	206
Learn more about AWS CloudFormation .....	206
<b>Document history .....</b>	<b>207</b>
<b>AWS Glossary .....</b>	<b>208</b>

# What is AWS Deadline Cloud?

Deadline Cloud is an AWS service you can use to create and manage rendering projects and jobs on Amazon Elastic Compute Cloud (Amazon EC2) instances directly from digital content creation pipelines and workstations.

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 requirements, create environments for specific workloads, and integrate the content creation tools that your production 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](#)
- [Concepts and terminology for Deadline Cloud](#)
- [Getting started with Deadline Cloud](#)
- [Accessing Deadline Cloud](#)
- [Related services](#)
- [How Deadline Cloud works](#)

## Features of Deadline Cloud

Here are some of the key ways Deadline Cloud can help you run and manage visual compute 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 a wide range of 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.

### 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 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.

### Digital content creation application (DCC)

Digital content creation applications (DCCs) are third-party products where you create digital content. Examples of DCCs are Maya, Nuke, and Houdini. Deadline Cloud provides job submitter integrated plugins for specific DCCs.

### Farm

A farm is a where your project resources are located. It consists of queues and fleets.

### Fleet

A fleet is a group of worker nodes that do the rendering. Worker nodes process jobs. A fleet can be associated to multiple queues, and a queue can be associated to multiple fleets.

### Job

A job is a rendering request. Users submit jobs. Jobs contain specific job properties that are outlined as steps and tasks.

## Job attachments

A job attachment is a Deadline Cloud feature that you can use to manage inputs and outputs for jobs. Job files are uploaded as job attachments during the rendering process. These files can be textures, 3D models, lighting rigs, and other similar items.

## 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.

## Job template

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

## Queue

A queue is where submitted jobs are located and scheduled to be rendered. A queue must be associated with a fleet to create a successful render. A queue can be associated with multiple fleets.

## Queue-fleet association

When a queue is associated with a fleet, there is a queue-fleet association. Use an association to schedule workers from a fleet to jobs in that queue. You can start and stop associations to control scheduling of work.

## Step

A step is one particular process to run in the job.

## Deadline Cloud submitter

A Deadline Cloud submitter is a digital content creation (DCC) plugin. Artists use it to submit jobs from a third-party DCC interface that they are familiar with.

## 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. For example, you could define a set of tags for your account's Amazon EC2 instances that help you track each instance's owner and stack level.

You can also categorize your AWS resources by purpose, owner, or environment. This approach is useful when you have many resources of the same type. You can quickly identify a specific resources based on the tags that you've assigned to it.

## Task

A task is a single component of a render step.

## 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 you go, and you are charged for the number of hours and minutes that you use.

## Usage explorer

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

## Worker

Workers belong to fleets and run Deadline Cloud assigned tasks to complete steps and jobs. Workers store the logs from task operations in Amazon CloudWatch Logs. Workers can also use the job attachments feature to sync inputs and outputs to an Amazon Simple Storage Service (Amazon S3) bucket.

# 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.

## 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](#).

## 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 the [Amazon CloudWatch User 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 instances as the demand on your instances changes. Auto Scaling helps to make sure that you're running your desired number of instances, even if an instance fails. If you enable Auto Scaling with Deadline Cloud, instances that are launched by Auto Scaling are automatically registered with the workload. Likewise, instances that are terminated by Auto Scaling are automatically de-registered from the workload. For more information, see the [Amazon EC2 Auto Scaling User 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.
- **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.

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 it can be connected to your existing identity provider (IdP) like 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.

# 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 without setting up the Deadline Cloud monitor infrastructure, set up a developer workstation for Deadline Cloud.

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](#)
- [Set up the Deadline Cloud monitor](#)
- [Setting up a developer workstation for Deadline Cloud](#)
- [Set up Deadline Cloud submitters](#)
- [Use the farm](#)

## Set up your AWS account

Set up your AWS account to use AWS Deadline Cloud.

If you do not have an AWS account, complete the following steps to create one.

### To sign up for an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.



Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, [assign administrative access to an administrative user](#), and use only the root user to perform [tasks that require root user access](#).

When you first create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the *AWS account root user* and is accessed by signing in with the email address and password that you used to create the account.

### Important

We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see [Tasks that require root user credentials](#) in the *IAM User Guide*.

## Set up the Deadline Cloud monitor

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

### Step 1: Set up your monitor

The Deadline Cloud monitor uses AWS IAM Identity Center to authorize users. The IAM Identity Center instance that you use for Deadline Cloud must be in the same AWS Region as the monitor. 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 display name:** The **Monitor display name** is how you can identify your monitor — for example *AnyCompany monitor*. Your monitor's name also determines your **monitor URL**.

**⚠ Important**

You can't change the **monitor display name** after you finish setting up.

- **Monitor URL:** You can access your monitor by using the **Monitor URL**. The URL is based on the **Monitor display name** — for example *https://anycompanymonitor.awsapps.com*.

**⚠ Important**

You can't change the **Monitor URL** after you finish setting up.

- **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 reduces lag and improves data transfer speeds. AWS IAM Identity Center must be enabled in the same AWS Region as Deadline Cloud.

**⚠ 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 display name** — for example **AnyCompany Monitor**.
  3. **(Optional)** To change the **Monitor name**, 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**.
- **(Optional)** To add groups and users, select [\(Optional\) Add groups and users](#).
  - **(Optional)** To further customize your monitor setup, select [Additional settings](#).

5. If you are ready to [Step 2: Define farm details](#), choose Next.

## (Optional) Add groups and users

Before you complete Deadline Cloud monitor setup, you can add monitor users and add them to a group.

After setup is complete, you can create new users and groups, and manage users such as to assign them groups, permissions, and applications, or delete users from your monitor.

## 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. 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 allow users 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 clear description can help you quickly identify your farm's purpose.
3. **(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**.

4. **(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-[MONITORNAME]**.

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. If you are using customer-managed fleets, select **Enable association with customer-managed fleets**.
    - 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.
  5. Your queue requires permission to access Amazon S3 on your behalf. We recommend you create a new service role for every queue.
    - a. For a new role, complete the following steps.
      - i. Select **Create and use a new service role**.
      - ii. Enter a **Role name** for your queue role or use the provided role name.
      - iii. *(Optional)* Add a queue role **Description**.
      - iv. You can view the IAM permissions for the queue role by choosing **View permission details**.
    - b. Alternatively, you can choose an existing service role.
  6. *(Optional)* Add environment variables for the queue environment using name and value pairs.

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

After you enter all the queue details, select **Next**.

## **(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. Select the way your compute resources should scale. The **Service-managed** option allows Deadline Cloud to auto scale your compute resources. The **Customer-managed** option leaves you in control of your own compute scaling.
2. In the **Instance option** section, choose either **Spot** or **On-demand**. Amazon EC2 On-demand instances provide faster availability and Amazon EC2 Spot 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.

- a. For a new role, complete the following steps.
    - i. Select **Create and use a new service role**.
    - ii. Enter a **Role name** for your fleet role or use the provided role name.
    - iii. *(Optional)* Add a fleet role **Description**.
    - iv. You can view the IAM permissions for the fleet role by choosing **View permission details**.
  - b. Alternatively, you can use an existing service role.
5. *(Optional)* Add tags for the fleet using key and value pairs.

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

## (Optional) Step 5: Configure worker requirements

Define the requirements for your worker instances.

1. Review the operating system (OS) and CPU architecture settings for awareness.
2. Update the minimum and maximum number of vCPUs for your hardware requirements.
3. Update the minimum and maximum number of memory (GiB) for your hardware requirements.
4. You can filter instance types by either allowing or excluding types of worker instances. In both filtering options, you can filter up to 10 Amazon EC2 instance types.
5. Under **Additional requirements (Optional)**, you can define the root EBS volume by **Size (GiB)**, **IOPS**, and **Throughput (MiB/s)**.
6. After all worker requirement are set, choose **Next** to define the access level of your groups.

## (Optional) Step 6: Define access levels

If you have groups connected to your monitor, you can define their access level. Permission to use Deadline Cloud features is managed by access levels. You can assign different access levels to groups of users.

1. Use the **Deadline Cloud farm access level** menu to select the level of permission for the group.
2. Choose **Next** to continue and review all farm details entered.

## Step 7: Review and create

Review all of 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.

## Setting up a developer workstation for Deadline Cloud

In this tutorial, you will use AWS CloudShell to create a simple developer farm and run the worker agent. You can then submit and run a simple job with parameters and attachments, add a service managed fleet, and clean up your farm resources when you're done.

The following sections introduce you to the different features of Deadline Cloud, and how they function and work together. Following these steps is useful for developing and testing new workloads and customizations.

## Topics

- [Step 1: Create a Deadline Cloud farm](#)
- [Step 2: Run the worker agent in developer mode in Deadline Cloud](#)
- [Step 3: Submit and run jobs with Deadline Cloud](#)
- [Step 4: Run jobs with job attachments in Deadline Cloud](#)
- [Step 5: Add a service managed fleet to your developer farm in Deadline Cloud](#)
- [Step 6: Clean up your farm resources in Deadline Cloud](#)

## Step 1: Create a Deadline Cloud farm

To create your developer farm and queue resources in AWS Deadline Cloud, use the AWS Command Line Interface (AWS CLI), as shown in the following procedure. You will also create an AWS Identity and Access Management (IAM) role and a customer managed fleet (CMF) and associate the fleet with your queue. Then you can configure the AWS CLI and confirm that your farm is set up and working as specified.

You can use this farm to explore the features of Deadline Cloud, then develop and test new workloads, customizations, and pipeline integrations.

### To create a farm

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already. For information, see [Install or update to the latest version of the AWS CLI](#).
2. Create a name for your farm, and add that farm name to `~/.bashrc`. This will make it available for other terminal sessions.

```
echo "DEV_FARM_NAME=DeveloperFarm" >> ~/.bashrc
source ~/.bashrc
```

3. Create the farm resource, and add its farm ID to `~/.bashrc`.

```
aws deadline create-farm \  
  --display-name "$DEV_FARM_NAME"
```



```
echo "DEV_FARM_ID=\$(aws deadline list-farms \
  --query \"farms[?displayName=='\${DEV_FARM_NAME}'].farmId \
  | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

4. Create the queue resource, and add its queue ID to ~/.bashrc.

```
aws deadline create-queue \
  --farm-id $DEV_FARM_ID \
  --display-name "$DEV_FARM_NAME Queue" \
  --job-run-as-user '{"posix": {"user": "job-user", "group": "job-group"},
  "runAs": "QUEUE_CONFIGURED_USER"}'

echo "DEV_QUEUE_ID=\$(aws deadline list-queues \
  --farm-id \${DEV_FARM_ID} \
  --query \"queues[?displayName=='\${DEV_FARM_NAME} Queue'].queueId \
  | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

5. Create an IAM role for the fleet. This role provides worker hosts in your fleet with the necessary security credentials to run jobs from your queue.

```
aws iam create-role \
  --role-name "${DEV_FARM_NAME}FleetRole" \
  --assume-role-policy-document \
    '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "credentials.deadline.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }'
```

```
aws iam put-role-policy \
  --role-name "${DEV_FARM_NAME}FleetRole" \
  --policy-name WorkerPermissions \
  --policy-document \
    '{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "deadline:AssumeFleetRoleForWorker",
      "deadline:UpdateWorker",
      "deadline>DeleteWorker",
      "deadline:UpdateWorkerSchedule",
      "deadline:BatchGetJobEntity",
      "deadline:AssumeQueueRoleForWorker"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws:PrincipalAccount": "${aws:ResourceAccount}"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "logs>CreateLogStream"
    ],
    "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
    "Condition": {
      "StringEquals": {
        "aws:PrincipalAccount": "${aws:ResourceAccount}"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "logs:PutLogEvents",
      "logs:GetLogEvents"
    ],
    "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
    "Condition": {
      "StringEquals": {
        "aws:PrincipalAccount": "${aws:ResourceAccount}"
      }
    }
  }
]
```

```
    ]
  }'
```

6. Create the customer managed fleet (CMF), and add its fleet ID to ~/.bashrc.

```
FLEET_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
  --query "Account" --output text):role/${DEV_FARM_NAME}FleetRole"
aws deadline create-fleet \
  --farm-id $DEV_FARM_ID \
  --display-name "$DEV_FARM_NAME CMF" \
  --role-arn $FLEET_ROLE_ARN \
  --max-worker-count 5 \
  --configuration \
  '{
    "customerManaged": {
      "mode": "NO_SCALING",
      "workerCapabilities": {
        "vCpuCount": {"min": 1},
        "memoryMiB": {"min": 512},
        "osFamily": "linux",
        "cpuArchitectureType": "x86_64"
      }
    }
  }'
```

```
echo "DEV_CMF_ID=$(aws deadline list-fleets \
  --farm-id \$$DEV_FARM_ID \
  --query \"fleets[?displayName=='\$$DEV_FARM_NAME CMF'].fleetId \
  | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

7. Ensure you can access Deadline Cloud.

```
pip install deadline
```

8. Associate the CMF with your queue.

```
aws deadline create-queue-fleet-association \
  --farm-id $DEV_FARM_ID \
  --queue-id $DEV_QUEUE_ID \
  --fleet-id $DEV_CMF_ID
```

- To set the default farm to the farm ID and the queue to the queue ID that you created earlier, use the following command.

```
deadline config set defaults.farm_id $DEV_FARM_ID
deadline config set defaults.queue_id $DEV_QUEUE_ID
```

- (Optional) To confirm that your farm is set up according to your specifications, use the following commands:

- List all farms – **deadline farm list**
- List all queues in the default farm – **deadline queue list**
- List all fleets in the default farm – **deadline fleet list**
- Get the default farm – **deadline farm get**
- Get the default queue – **deadline queue get**
- Get all the fleets associated with the default queue – **deadline fleet get**

## Step 2: Run the worker agent in developer mode in Deadline Cloud

Before you can run the jobs you submit to the queue on your developer farm, you must run the AWS Deadline Cloud worker agent in developer mode on a worker host.

Throughout the remainder of this tutorial, you will perform AWS CLI operations on your developer farm using two AWS CloudShell tabs. In the first tab, you can submit jobs. In the second tab, you can run the worker agent.

### Note

If you leave your CloudShell session idle for more than 20 minutes, it will timeout and stop the worker agent. To restart the worker agent, follow the instructions in the following procedure.


### To run the worker agent in developer mode

- Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already. For information, see [Install or update to the latest version of the AWS CLI](#).

2. With your farm still open in the first CloudShell tab, open a second CloudShell tab, then create the `demoenv-logs` and `demoenv-persist` directories.

```
mkdir ~/demoenv-logs
mkdir ~/demoenv-persist
```

3. Download and install the Deadline Cloud worker agent packages from PyPI:

 **Note**

On Windows, it is required that the agent files are installed into Python's global site-packages directory. Python virtual environments are not currently supported.

```
python -m pip install deadline-cloud-worker-agent
```

4. To allow the worker agent to create the temporary directories for running jobs, create a directory:

```
sudo mkdir /sessions
sudo chmod 750 /sessions
sudo chown cloudshell-user /sessions
```

5. Run the Deadline Cloud worker agent in developer mode with the variables `DEV_FARM_ID` and `DEV_CMF_ID` that you added to the `~/ .bashrc`.

```
deadline-worker-agent \
  --farm-id $DEV_FARM_ID \
  --fleet-id $DEV_CMF_ID \
  --run-jobs-as-agent-user \
  --logs-dir ~/demoenv-logs \
  --persistence-dir ~/demoenv-persist
```

As the worker agent initializes and then polls the `UpdateWorkerSchedule` API operation the following output is displayed:

```
INFO    Worker Agent starting
[2024-03-27 15:51:01,292][INFO    ] # Worker Agent starting
[2024-03-27 15:51:01,292][INFO    ] AgentInfo
Python Interpreter: /usr/bin/python3
```

```
Python Version: 3.9.16 (main, Sep 8 2023, 00:00:00) - [GCC 11.4.1 20230605 (Red Hat 11.4.1-2)]
Platform: linux
...
[2024-03-27 15:51:02,528][INFO ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params={'assignedSessions': {}, 'cancelSessionActions': {},
'updateIntervalSeconds': 15} ...
[2024-03-27 15:51:17,635][INFO ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params=(Duplicate removed, see previous response) ...
[2024-03-27 15:51:32,756][INFO ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params=(Duplicate removed, see previous response) ...
...
```

6. Select your first CloudShell tab, then list the workers in the fleet.

```
deadline worker list --fleet-id $DEV_CMF_ID
```

Output such as the following is displayed:

```
Displaying 1 of 1 workers starting at 0

- workerId: worker-8c9af877c8734e89914047111f
  status: STARTED
  createdAt: 2023-12-13 20:43:06+00:00
```

In a production configuration, the Deadline Cloud worker agent requires setting up multiple users and configuration directories as an administrative user on the host machine. You can override these settings because you're running jobs in your own development farm, which only you can access.

### Step 3: Submit and run jobs with Deadline Cloud

To use AWS Deadline Cloud to run jobs, use the following procedures. Use the first AWS CloudShell tab to submit jobs to your developer farm. Use the second CloudShell tab to view the worker agent output.

#### Topics

- [Submit the simple\\_job sample](#)
- [Submit a simple\\_job with a parameter](#)
- [Create a simple\\_file\\_job job bundle with file I/O](#)

## Submit the `simple_job` sample

After you create a farm and run the worker agent, you can submit the `simple_job` sample to Deadline Cloud.

### To submit the `simple_job` sample to Deadline Cloud

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already. For information, see [Install or update to the latest version of the AWS CLI](#).

2. Download the sample from GitHub.

```
cd ~
git clone https://github.com/aws-deadline/deadline-cloud-
samples.git
```

3. Choose your first CloudShell tab, then navigate to the job bundle samples directory.

```
cd ~/deadline-cloud-samples/job_bundles/
```

4. Submit the `simple_job` sample.

```
deadline bundle submit simple_job
```

5. Choose your second CloudShell tab to view the logging output about calling `BatchGetJobEntities`, getting a session, and running a session action.

```
...
[2024-03-27 16:00:21,846][INFO    ] # Session.Starting
# [session-053d77cef82648fe2] Starting new Session.
[queue-3ba4ff683ff54db09b851a2ed8327d7b/job-d34cc98a6e234b6f82577940ab4f76c6]
[2024-03-27 16:00:21,853][INFO    ] # API.Req # [deadline:BatchGetJobEntity]
resource={'farm-id': 'farm-3e24cfc9bbcd423e9c1b6754bc1',
'fleet-id': 'fleet-246ee60f46d44559b6cce010d05', 'worker-id':
'worker-75e0fce9c3c344a69bff57fcd83'} params={'identifiers': [{'jobDetails':
{'jobId': 'job-d34cc98a6e234b6f82577940ab4'}}]} request_url=https://
scheduling.deadline.us-west-2.amazonaws.com/2023-10-12/farms/
farm-3e24cfc9bbcd423e /fleets/fleet-246ee60f46d44559b1 /workers/worker-
75e0fce9c3c344a69b /batchGetJobEntity
[2024-03-27 16:00:22,013][INFO    ] # API.Resp # [deadline:BatchGetJobEntity](200)
params={'entities': [{'jobDetails': {'jobId': 'job-d34cc98a6e234b6f82577940ab6',
'jobRunAsUser': {'posix': {'user': 'job-user', 'group': 'job-group'}},
'runAs': 'QUEUE_CONFIGURED_USER'}, 'logGroupName': '/aws/deadline/
```

```
farm-3e24cfc9bbcd423e9c1b6754bc1/queue-3ba4ff683ff54db09b851a2ed83', 'parameters':
  '*REDACTED*', 'schemaVersion': 'jobtemplate-2023-09'}}], 'errors': []}
request_id=a3f55914-6470-439e-89e5-313f0c6
[2024-03-27 16:00:22,013][INFO    ] # Session.Add #
[session-053d77cef82648fea9c69827182] Appended new SessionActions.
(ActionIds: ['sessionaction-053d77cef82648fea9c69827182-0'])
[queue-3ba4ff683ff54db09b851a2ed8b/job-d34cc98a6e234b6f82577940ab6]
[2024-03-27 16:00:22,014][WARNING ] # Session.User #
[session-053d77cef82648fea9c69827182] Running as the Worker Agent's
user. (User: cloudshell-user) [queue-3ba4ff683ff54db09b851a2ed8b/job-
d34cc98a6e234b6f82577940ac6]
[2024-03-27 16:00:22,015][WARNING ] # Session.AWSCreds #
[session-053d77cef82648fea9c69827182] AWS Credentials are not available: Queue has
no IAM Role. [queue-3ba4ff683ff54db09b851a2ed8b/job-d34cc98a6e234b6f82577940ab6]
[2024-03-27 16:00:22,026][INFO    ] # Session.Logs #
[session-053d77cef82648fea9c69827182] Logs streamed to: AWS CloudWatch
Logs. (LogDestination: /aws/deadline/farm-3e24cfc9bbcd423e9c1b6754bc1/
queue-3ba4ff683ff54db09b851a2ed83/session-053d77cef82648fea9c69827181)
[queue-3ba4ff683ff54db09b851a2ed83/job-d34cc98a6e234b6f82577940ab4]
[2024-03-27 16:00:22,026][INFO    ] # Session.Logs #
[session-053d77cef82648fea9c69827182] Logs streamed to: local
file. (LogDestination: /home/cloudshell-user/demoenv-logs/
queue-3ba4ff683ff54db09b851a2ed8b/session-053d77cef82648fea9c69827182.log)
[queue-3ba4ff683ff54db09b851a2ed83/job-d34cc98a6e234b6f82577940ab4]
...
```

**Note**

Only the logging output from the worker agent is shown. There is a separate log for the session that runs the job.

6. Choose your first tab, then inspect the log files that the worker agent writes.
  - a. Navigate to the worker agent logs directory and view its contents.

```
cd ~/demoenv-logs
ls
```

- b. Print the first log file that the worker agent creates.

```
cat worker-agent-bootstrap.log
```



This file contains worker agent output about how it called the Deadline Cloud API to create a worker resource in your fleet, and then assumed the fleet role.

- c. Print the log file output when the worker agent joins the fleet.

```
cat worker-agent.log
```

This log contains outputs about all the actions that the worker agent takes, but doesn't contain output about the queues it runs jobs from, except for the IDs of those resources.

- d. Print the log files for each session in a directory that is named the same as the queue resource id.

```
cat $DEV_QUEUE_ID/session-*.log
```

If the job is successful, the log file output will be similar to the following:

```
cat $DEV_QUEUE_ID/$(ls -t $DEV_QUEUE_ID | head -1)
2024-03-27 16:00:22,026 WARNING Session running with no AWS Credentials.
2024-03-27 16:00:22,404 INFO
2024-03-27 16:00:22,405 INFO =====
2024-03-27 16:00:22,405 INFO ----- Running Task
2024-03-27 16:00:22,405 INFO =====
2024-03-27 16:00:22,406 INFO -----
2024-03-27 16:00:22,406 INFO Phase: Setup
2024-03-27 16:00:22,406 INFO -----
2024-03-27 16:00:22,406 INFO Writing embedded files for Task to disk.
2024-03-27 16:00:22,406 INFO Mapping: Task.File.runScript -> /sessions/
session-053d77cef82648fea9c698271812a/embedded_files/gj55_/tmp2u9yqtsz
2024-03-27 16:00:22,406 INFO Wrote: runScript -> /sessions/
session-053d77cef82648fea9c698271812a/embedded_files/gj55_/tmp2u9yqtsz
2024-03-27 16:00:22,407 INFO -----
2024-03-27 16:00:22,407 INFO Phase: Running action
2024-03-27 16:00:22,407 INFO -----
2024-03-27 16:00:22,407 INFO Running command /sessions/
session-053d77cef82648fea9c698271812a/tmpzuzxpslm.sh
2024-03-27 16:00:22,414 INFO Command started as pid: 471
2024-03-27 16:00:22,415 INFO Output:
2024-03-27 16:00:22,420 INFO Welcome to AWS Deadline Cloud!
2024-03-27 16:00:22,571 INFO
2024-03-27 16:00:22,572 INFO =====
2024-03-27 16:00:22,572 INFO ----- Session Cleanup
```

```
2024-03-27 16:00:22,572 INFO =====  
2024-03-27 16:00:22,572 INFO Deleting working directory: /sessions/  
session-053d77cef82648fea9c698271812a
```

## 7. Print information about the job.

```
deadline job get
```

When you submit the job, the system saves it as the default so you don't have to enter the job ID.

## Submit a simple\_job with a parameter

You can submit jobs with parameters. In the following procedure, you edit the `simple_job` template to include a custom message, submit the `simple_job`, then print the session log file to view the message.

### To submit the `simple_job` sample with a parameter

1. Select your first CloudShell tab, then navigate to the job bundle samples directory.

```
cd ~/deadline-cloud-samples/job_bundles/
```

2. Print the contents of the `simple_job` template.

```
cat simple_job/template.yaml
```

The `parameterDefinitions` section with the `Message` parameter should look like the following:

```
parameterDefinitions:  
- name: Message  
  type: STRING  
  default: Welcome to AWS Deadline Cloud!
```

3. Submit the `simple_job` sample with a parameter value, then wait for the job to finish running.

```
deadline bundle submit simple_job \  
-p "Message=Greetings from the developer getting started guide."
```

4. To see the custom message, view the most recent session log file.

```
cd ~/demoenv-logs
cat $DEV_QUEUE_ID/$(ls -t $DEV_QUEUE_ID | head -1)
```

## Create a `simple_file_job` job bundle with file I/O

A render job needs to read the scene definition, render an image from it, and then save that image to an output file. You can simulate this action by making the job compute the hash of the input instead of rendering an image.

### To create a `simple_file_job` job bundle with file I/O

1. Select your first CloudShell tab, then navigate to the job bundle samples directory.

```
cd ~/deadline-cloud-samples/job_bundles/
```

2. Make a copy of `simple_job` with the new name `simple_file_job`.

```
cp -r simple_job simple_file_job
```

3. Edit the job template as follows:

#### Note

We recommend that you use `nano` for these steps. If you prefer to use `Vim`, you must set its paste mode using `:set paste`.

- a. Open the template in a text editor.

```
nano simple_file_job/template.yaml
```

- b. Add the following type, `objectType`, and `dataFlow parameterDefinitions`.

```
- name: InFile
  type: PATH
  objectType: FILE
```

```

dataFlow: IN
- name: OutFile
  type: PATH
  objectType: FILE
  dataFlow: OUT

```

- c. Add the following bash script command to the end of the file that reads from the input file and writes to the output file.

```

# hash the input file, and write that to the output
sha256sum "{{Param.InFile}}" > "{{Param.OutFile}}"

```

The updated `template.yaml` should exactly match the following:

```

specificationVersion: 'jobtemplate-2023-09'
name: Simple File Job Bundle Example
parameterDefinitions:
- name: Message
  type: STRING
  default: Welcome to AWS Deadline Cloud!
- name: InFile
  type: PATH
  objectType: FILE
  dataFlow: IN
- name: OutFile
  type: PATH
  objectType: FILE
  dataFlow: OUT
steps:
- name: WelcomeToDeadlineCloud
  script:
    actions:
      onRun:
        command: '{{Task.File.runScript}}'
    embeddedFiles:
      - name: runScript
        type: TEXT
        runnable: true
        data: |
          #!/usr/bin/env bash
          echo "{{Param.Message}}"

          # hash the input file, and write that to the output

```

```
sha256sum "{{Param.InFile}}" > "{{Param.OutFile}}"
```

**Note**

If you want to adjust the spacing in the `template.yaml`, make sure that you use spaces instead of indentations.

- d. Save the file, and exit the editor.
4. Provide parameter values for the input and output files to submit the `simple_file_job`.

```
deadline bundle submit simple_file_job \  
  -p "InFile=simple_job/template.yaml" \  
  -p "OutFile=hash.txt"
```

5. Print information about the job.

```
deadline job get
```

- You will see output such as the following:

```
parameters:  
  Message:  
    string: Welcome to AWS Deadline Cloud!  
  InFile:  
    path: /local/home/cloudshell-user/BundleFiles/JobBundle-Examples/simple_job/  
template.yaml  
  OutFile:  
    path: /local/home/cloudshell-user/BundleFiles/JobBundle-Examples/hash.txt
```

- Although you only provided relative paths, the parameters have the full path set. The AWS CLI joins the current working directory to any paths that are provided as parameters when the paths have the type `PATH`.
- The worker agent running in the other terminal window picks up and runs the job. This action creates the `hash.txt` file, which you can view with the following command.

```
cat hash.txt
```

This command will print output similar to the following.

```
eea2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3 /local/home/  
cloudshell-user/BundleFiles/JobBundle-Examples/simple_job/template.yaml
```

## Step 4: Run jobs with job attachments in Deadline Cloud

Many farms use shared filesystems to share files between the hosts that submit jobs and those that run jobs. For example, in the previous `simple_file_job` example, the local filesystem is shared between the AWS CloudShell terminal windows, which run in tab one where you submit the job, and tab two where you run the worker agent.

A shared filesystem is advantageous when the submitter workstation and the worker hosts are on the same local area network. If you store your data on premises near the workstations that access it, then using a cloud-based farm means you have to share your filesystems over a high-latency VPN or synchronize your filesystems in the cloud. Neither of these options are easy to set up or operate.

AWS Deadline Cloud offers a simple solution with *job attachments*, which are similar to email attachments. With job attachments, you attach data to your job. Then, Deadline Cloud handles the details of transferring and storing your job data in Amazon Simple Storage Service (Amazon S3) buckets.

Content creation workflows are often iterative, meaning a user submits jobs with a small subset of modified files. Because Amazon S3 buckets store job attachments in a content-addressable storage, the name of each object is based on the hash of the object's data and the contents of a directory tree are stored in a manifest file format attached to a job.

To run jobs with job attachments, complete the following steps.

### Topics

- [Add a job attachments configuration to your queue](#)
- [Submit simple\\_file\\_job with job attachments](#)
- [Understanding how job attachments are stored in Amazon S3](#)

## Add a job attachments configuration to your queue

To enable job attachments in your queue, add a job attachments configuration to the queue resource in your account.

## To add a job attachments configuration to your queue

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already. For information, see [Install or update to the latest version of the AWS CLI](#).
2. Choose your first CloudShell tab, then enter one of the following commands to use an Amazon S3 bucket for job attachments.
  - If you don't have an existing private Amazon S3 bucket, you can create and use a new S3 bucket.

```
DEV_FARM_BUCKET=$(echo $DEV_FARM_NAME \
  | tr '[:upper:]' '[:lower:]')-$(xxd -l 16 -p /dev/urandom)
if [ "$AWS_REGION" == "us-east-1" ]; then LOCATION_CONSTRAINT=
else LOCATION_CONSTRAINT="--create-bucket-configuration \
  LocationConstraint=${AWS_REGION}"
fi
aws s3api create-bucket \
  $LOCATION_CONSTRAINT \
  --acl private \
  --bucket ${DEV_FARM_BUCKET}
```

- If you already have a private Amazon S3 bucket, you can use it by replacing *MY\_BUCKET\_NAME* with the name of your bucket.

```
DEV_FARM_BUCKET=MY_BUCKET_NAME
```

3. After you create or choose your Amazon S3 bucket, add the bucket name to `~/.bashrc` to make the bucket available for other terminal sessions.

```
echo "DEV_FARM_BUCKET=${DEV_FARM_BUCKET}" >> ~/.bashrc
```

4. Create an AWS Identity and Access Management (IAM) role for the queue.

```
aws iam create-role --role-name "${DEV_FARM_NAME}QueueRole" \
  --assume-role-policy-document \
  '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": "credentials.deadline.amazonaws.com"
```

```

        },
        "Action": "sts:AssumeRole"
    }
]
}'
aws iam put-role-policy \
--role-name "${DEV_FARM_NAME}QueueRole" \
--policy-name S3BucketsAccess \
--policy-document \
'{'
    "Version": "2012-10-17",
    "Statement": [
    {
        "Action": [
            "s3:GetObject*",
            "s3:GetBucket*",
            "s3:List*",
            "s3:DeleteObject*",
            "s3:PutObject",
            "s3:PutObjectLegalHold",
            "s3:PutObjectRetention",
            "s3:PutObjectTagging",
            "s3:PutObjectVersionTagging",
            "s3:Abort*"
        ],
        "Resource": [
            "arn:aws:s3:::$DEV_FARM_BUCKET",
            "arn:aws:s3:::$DEV_FARM_BUCKET/*"
        ],
        "Effect": "Allow"
    }
]
}'

```

5. Update your queue to include the job attachments settings and the IAM role.

```

QUEUE_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
--query "Account" --output text):role/${DEV_FARM_NAME}QueueRole"
aws deadline update-queue \
--farm-id $DEV_FARM_ID \
--queue-id $DEV_QUEUE_ID \
--role-arn $QUEUE_ROLE_ARN \
--job-attachment-settings \
'{'

```



```
"s3BucketName": "'$DEV_FARM_BUCKET'",  
  "rootPrefix": "JobAttachments"  
}'
```

6. Confirm that you updated your queue.

```
deadline queue get
```

Output such as the following is shown:

```
...  
jobAttachmentSettings:  
  s3BucketName: DEV_FARM_BUCKET  
  rootPrefix: JobAttachments  
roleArn: arn:aws:iam::ACCOUNT_NUMBER:role/DeveloperFarmQueueRole  
...
```

## Submit `simple_file_job` with job attachments

When you use job attachments, job bundles must give Deadline Cloud enough information to determine the job's data flow, such as using PATH parameters. In the case of the `simple_file_job`, you edited the `template.yaml` file to tell Deadline Cloud that the data flow is in the input file and output file.

After you've added the job attachments configuration to your queue, you can submit the `simple_file_job` sample with job attachments. After you do this, you can view the logging and job output to confirm that the `simple_file_job` with job attachments is working.

### To submit the `simple_file_job` job bundle with job attachments

1. Choose your first CloudShell tab, then open the `JobBundle-Samples` directory.
2. 

```
cd ~/AmazonDeadlineCloud-DocumentationAndSamples/JobBundle-Samples
```
3. Submit `simple_file_job` to the queue. When prompted to confirm the upload, enter `y`.

```
deadline bundle submit simple_file_job \  
  -p InFile=simple_job/template.yaml \  
  -p OutFile=hash-jobattachments.txt
```

- To view the job attachments data transfer session log output, choose your second CloudShell tab.

```
JOB_ID=$(deadline config get defaults.job_id)
SESSION_ID=$(aws deadline list-sessions \
  --farm-id $DEV_FARM_ID \
  --queue-id $DEV_QUEUE_ID \
  --job-id $JOB_ID \
  --query "sessions[0].sessionId" \
  --output text)
cat ~/demoenv-logs/$DEV_QUEUE_ID/$SESSION_ID.log
```

- List the session actions that were run within the session.

```
aws deadline list-session-actions \
  --farm-id $DEV_FARM_ID \
  --queue-id $DEV_QUEUE_ID \
  --job-id $JOB_ID \
  --session-id $SESSION_ID
```

Output such as the following is shown:

```
{
  "sessionactions": [
    {
      "sessionActionId": "sessionaction-123-0",
      "status": "SUCCEEDED",
      "startedAt": "<timestamp>",
      "endedAt": "<timestamp>",
      "progressPercent": 100.0,
      "definition": {
        "syncInputJobAttachments": {}
      }
    },
    {
      "sessionActionId": "sessionaction-123-1",
      "status": "SUCCEEDED",
      "startedAt": "<timestamp>",
      "endedAt": "<timestamp>",
      "progressPercent": 100.0,
      "definition": {
        "taskRun": {
```

```
        "taskId": "task-abc-0",  
        "stepId": "step-def"  
      }  
    }  
  ]  
}
```

The first session action downloaded the input job attachments, while the second action runs the task like before and then uploaded the output job attachments.

6. List the output directory.

```
ls *.txt
```

Output such as `hash.txt` is shown, but `hash-jobattachments.txt` doesn't exist.

7. Download the output from the most recent job.

```
deadline job download-output
```

8. View the output of the downloaded file.

```
cat hash-jobattachments.txt
```

Output such as the following is shown:

```
eea2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3 /tmp/openjd/  
session-123/assetroot-abc/simple_job/template.yaml
```

## Understanding how job attachments are stored in Amazon S3

You can use the AWS Command Line Interface (AWS CLI) to upload or download data for job attachments, which are stored in Amazon S3 buckets. Understanding how Deadline Cloud stores job attachments on Amazon S3 will help when you develop workloads and pipeline integrations.

### To inspect how Deadline Cloud job attachments are stored in Amazon S3

1. Choose your first CloudShell tab, then open the job bundle samples directory.

```
cd ~/AmazonDeadlineCloud-DocumentationAndSamples/JobBundle-Samples
```

## 2. Inspect the job properties.

```
deadline job get
```

Output such as the following is shown:

```
parameters:
  Message:
    string: Welcome to Amazon Deadline Cloud!
  InFile:
    path: /home/cloudshell-user/AmazonDeadlineCloud-DocumentationAndSamples/
JobBundle-Samples/simple_job/template.yaml
  OutFile:
    path: /home/cloudshell-user/AmazonDeadlineCloud-DocumentationAndSamples/
JobBundle-Samples/hash-jobattachments.txt
attachments:
  manifests:
    - rootPath: /home/cloudshell-user/AmazonDeadlineCloud-DocumentationAndSamples/
JobBundle-Samples
      rootPathFormat: posix
      outputRelativeDirectories:
        - .
      inputManifestPath: farm-3040c59a5b9943d58052c29d907a645d/queue-
cde9977c9f4d4018a1d85f3e6c1a4e6e/Inputs/
f46af01ca8904cd8b514586671c79303/0d69cd94523ba617c731f29c019d16e8_input.xxh128
      inputManifestHash: f95ef91b5dab1fc1341b75637fe987ee
      fileSystem: COPIED
```

The attachments field contains a list of manifest structures that describe input and output data paths that the job uses when it runs. Look at `rootPath` to see the local directory path on the machine that submitted the job. To see the Amazon S3 object suffix that contains a manifest file, look at `inputManifestFile`. The manifest file contains metadata for a directory tree snapshot of the job's input data.

## 3. Pretty-print the Amazon S3 manifest object to see the input directory structure for the job.

```
MANIFEST_SUFFIX=$(aws deadline get-job \
  --farm-id $DEV_FARM_ID \
  --queue-id $DEV_QUEUE_ID \
```

```

--job-id $JOB_ID \
--query "attachments.manifests[0].inputManifestPath" \
--output text)
aws s3 cp s3://$DEV_FARM_BUCKET/JobAttachments/Manifests/$MANIFEST_SUFFIX - | jq .

```

Output such as the following is shown:

```

{
  "hashAlg": "xxh128",
  "manifestVersion": "2023-03-03",
  "paths": [
    {
      "hash": "2ec297b04c59c4741ed97ac8fb83080c",
      "mtime": 1698186190000000,
      "path": "simple_job/template.yaml",
      "size": 445
    }
  ],
  "totalSize": 445
}

```

- Construct the Amazon S3 prefix that holds manifests for the output job attachments and list the object under it.

```

SESSION_ACTION=$(aws deadline list-session-actions \
  --farm-id $DEV_FARM_ID \
  --queue-id $DEV_QUEUE_ID \
  --job-id $JOB_ID \
  --session-id $SESSION_ID \
  --query "sessionActions[?definition.taskRun != null] | [0]")
STEP_ID=$(echo $SESSION_ACTION | jq -r .definition.taskRun.stepId)
TASK_ID=$(echo $SESSION_ACTION | jq -r .definition.taskRun.taskId)
TASK_OUTPUT_PREFIX=JobAttachments/Manifests/$DEV_FARM_ID/$DEV_QUEUE_ID/$JOB_ID/
$STEP_ID/$TASK_ID/
aws s3api list-objects-v2 --bucket $DEV_FARM_BUCKET --prefix $TASK_OUTPUT_PREFIX

```

The output job attachments are not directly referenced from the job resource but are instead placed in an Amazon S3 bucket based on farm resource IDs.

- Get the newest manifest object key for the specific session action id, then pretty-print the manifest objects.

```
SESSION_ACTION_ID=$(echo $SESSION_ACTION | jq -r .sessionId)
MANIFEST_KEY=$(aws s3api list-objects-v2 \
  --bucket $DEV_FARM_BUCKET \
  --prefix $TASK_OUTPUT_PREFIX \
  --query "Contents[*].Key" --output text \
  | grep $SESSION_ACTION_ID \
  | sort | tail -1)
MANIFEST_OBJECT=$(aws s3 cp s3://$DEV_FARM_BUCKET/$MANIFEST_KEY -)
echo $MANIFEST_OBJECT | jq .
```

You'll see properties of the file `hash-jobattachments.txt` in the output such as the following:

```
{
  "hashAlg": "xxh128",
  "manifestVersion": "2023-03-03",
  "paths": [
    {
      "hash": "f60b8e7d0fabf7214ba0b6822e82e08b",
      "mtime": 1698785252554950,
      "path": "hash-jobattachments.txt",
      "size": 182
    }
  ],
  "totalSize": 182
}
```

Your job will only have a single manifest object per task run, but in general it is possible to have more of objects per task run.

6. View content-addressible Amazon S3 storage output under the Data prefix.

```
FILE_HASH=$(echo $MANIFEST_OBJECT | jq -r .paths[0].hash)
FILE_PATH=$(echo $MANIFEST_OBJECT | jq -r .paths[0].path)
aws s3 cp s3://$DEV_FARM_BUCKET/JobAttachments/Data/$FILE_HASH -
```

Output such as the following is shown:

```
eea2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3 /tmp/openjd/  
session-123/assetroot-abc/simple_job/template.yaml
```

## Step 5: Add a service managed fleet to your developer farm in Deadline Cloud

AWS CloudShell does not provide enough compute capacity to test larger workloads. It's also not configured to work with jobs that distribute tasks on multiple worker hosts.

Instead of using CloudShell, you can add an Auto Scaling service managed fleet (SMF) to your developer farm. An SMF provides sufficient compute capacity for larger workloads and can handle jobs that need to distribute job tasks across multiple worker hosts. The scheduler will use both the SMF and CMF workers to run jobs, unless you shut down the CMF worker.

### To add a service-managed fleet to your developer farm

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already. For information, see [Install or update to the latest version of the AWS CLI](#).
2. Choose your first AWS CloudShell tab, then create the service managed fleet and add its fleet ID to `.bashrc`. This action makes it available for other terminal sessions.

```
FLEET_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \  
    --query "Account" --output text):role/${DEV_FARM_NAME}FleetRole"  
aws deadline create-fleet \  
  --farm-id $DEV_FARM_ID \  
  --display-name "$DEV_FARM_NAME SMF" \  
  --role-arn $FLEET_ROLE_ARN \  
  --max-worker-count 5 \  
  --configuration \  
    '{  
      "serviceManagedEc2": {  
        "instanceCapabilities": {  
          "vCpuCount": {  
            "min": 2,  
            "max": 4  
          },  
          "memoryMiB": {  
            "min": 512  
          },  
        },  
      },  
    }'
```

```

        "osFamily": "linux",
        "cpuArchitectureType": "x86_64"
    },
    "instanceMarketOptions": {
        "type": "spot"
    }
}
}'

echo "DEV_SMF_ID=$(aws deadline list-fleets \
    --farm-id $DEV_FARM_ID \
    --query "fleets[?displayName=='$DEV_FARM_NAME SMF'].fleetId \
    | [0]" --output text)" >> ~/.bashrc
source ~/.bashrc

```


3. Associate the SMF with your queue.

```

aws deadline create-queue-fleet-association \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --fleet-id $DEV_SMF_ID

```

- 4.

 **Note**

The scheduler will use both the SMF and CMF workers to run jobs, unless you shut down the CMF worker.

Submit `simple_file_job` to the queue. When prompted to confirm the upload, enter `y`.

```

deadline bundle submit simple_file_job \
    -p InFile=simple_job/template.yaml \
    -p OutFile=hash-jobattachments.txt

```

5. Confirm the SMF is working correctly.

```

deadline fleet get

```

- The worker may take a few minutes to start.
- The `queueFleetAssociationsStatus` for your customer managed fleet and service managed fleet will be `ACTIVE`.



- The SMF `autoScalingStatus` will change from `GROWING` to `STEADY`.

Your status will look similar to the following:

```
fleetId: fleet-2cc78e0dd3f04d1db427e7dc1d51ea44
farmId: farm-63ee8d77cdab4a578b685be8c5561c4a
displayName: DeveloperFarm SMF
description: ''
status: ACTIVE
autoScalingStatus: STEADY
targetWorkerCount: 0
workerCount: 0
minWorkerCount: 0
maxWorkerCount: 5
```

6. View the log for the job that you submitted. This log is stored in a log in Amazon CloudWatch Logs, not the CloudShell file system.

```
JOB_ID=$(deadline config get defaults.job_id)
SESSION_ID=$(aws deadline list-sessions \
  --farm-id $DEV_FARM_ID \
  --queue-id $DEV_QUEUE_ID \
  --job-id $JOB_ID \
  --query "sessions[0].sessionId" \
  --output text)
aws logs tail /aws/deadline/$DEV_FARM_ID/$DEV_QUEUE_ID \
  --log-stream-names $SESSION_ID
```

## Step 6: Clean up your farm resources in Deadline Cloud

To develop and test new workloads and pipeline integrations, you can continue to use the Deadline Cloud developer farm that you created for this tutorial. If you no longer need your developer farm, you can delete its resources including farm, fleet, queue, AWS Identity and Access Management (IAM) roles, and logs in Amazon CloudWatch Logs. After you delete these resources, you will need to begin the tutorial again to use the resources. For more information, see [Setting up a developer workstation for Deadline Cloud](#).

## To clean up developer farm resources

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already. For information, see [Install or update to the latest version of the AWS CLI](#).
2. Choose your first CloudShell tab, then stop all the queue-fleet associations for your queue.

```
FLEETS=$(aws deadline list-queue-fleet-associations \
  --farm-id $DEV_FARM_ID \
  --queue-id $DEV_QUEUE_ID \
  --query "queueFleetAssociations[].fleetId" \
  --output text)
for FLEET_ID in $FLEETS; do
  aws deadline update-queue-fleet-association \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --fleet-id $FLEET_ID \
    --status STOP_SCHEDULING_AND_CANCEL_TASKS
done
```

3. List the queue fleet associations.

```
aws deadline list-queue-fleet-associations \
  --farm-id $DEV_FARM_ID \
  --queue-id $DEV_QUEUE_ID
```

You might need to rerun the command until the output reports "status": "STOPPED", then you can proceed to the next step. This process can take several minutes to complete.

```
{
  "queueFleetAssociations": [
    {
      "queueId": "queue-a08c6dd0e60a460db895360e2186229d",
      "fleetId": "fleet-421e6141c304405089e8a9c792ec7bb8",
      "status": "STOPPED",
      "createdAt": "2023-11-21T20:49:19+00:00",
      "createdBy": "arn:aws:sts::576194808517:assumed-role/admin/mcklein-Isengard",
      "updatedAt": "2023-11-21T20:49:38+00:00",
      "updatedBy": "arn:aws:sts::576194808517:assumed-role/admin/mcklein-Isengard"
    },
    {
```

```

        "queueId": "queue-a08c6dd0e60a460db895360e2186229d",
        "fleetId": "fleet-687acf47a0944a58ab2aedb0d0361db5",
        "status": "STOPPED",
        "createdAt": "2023-11-21T20:32:06+00:00",
        "createdBy": "arn:aws:sts::576194808517:assumed-role/admin/mcklein-
Isengard",
        "updatedAt": "2023-11-21T20:49:39+00:00",
        "updatedBy": "arn:aws:sts::576194808517:assumed-role/admin/mcklein-
Isengard"
    }
]
}

```

4. Delete all of the queue-fleet associations for your queue.

```

for FLEET_ID in $FLEETS; do
    aws deadline delete-queue-fleet-association \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --fleet-id $FLEET_ID
done

```

5. Delete all of the fleets associated with your queue.

```

for FLEET_ID in $FLEETS; do
    aws deadline delete-fleet \
        --farm-id $DEV_FARM_ID \
        --fleet-id $FLEET_ID
done

```

6. Delete the queue.

```

aws deadline delete-queue \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID

```

7. Delete the farm.

```

aws deadline delete-farm \
    --farm-id $DEV_FARM_ID

```

8. Delete other AWS resources for your farm.

- a. Delete the fleet AWS Identity and Access Management (IAM) role.

```
aws iam delete-role-policy \  
  --role-name "${DEV_FARM_NAME}FleetRole" \  
  --policy-name WorkerPermissions  
aws iam delete-role \  
  --role-name "${DEV_FARM_NAME}FleetRole"
```

- b. Delete the queue IAM role.

```
aws iam delete-role-policy \  
  --role-name "${DEV_FARM_NAME}QueueRole" \  
  --policy-name S3BucketsAccess  
aws iam delete-role \  
  --role-name "${DEV_FARM_NAME}QueueRole"
```

- c. Delete the Amazon CloudWatch Logs log groups. Each queue and fleet has their own log group.

```
aws logs delete-log-group \  
  --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_QUEUE_ID"  
aws logs delete-log-group \  
  --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_CMF_ID"  
aws logs delete-log-group \  
  --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_SMF_ID"
```

## Set up Deadline Cloud submitters

This process is for administrators and artists who want to install, set up, and launch the AWS Deadline Cloud submitter. A Deadline Cloud *submitter* is a digital content creation (DCC) plugin. Artists use it to submit jobs from a third-party DCC interface that they're familiar with.

### Note

This process must be completed on all workstations that artists will use for submitting renders.

## Topics

- [Step 1: Install the Deadline Cloud submitter](#)
- [Step 2: Install and set up Deadline Cloud monitor](#)
- [Step 3: Launch the Deadline Cloud submitter](#)

## Step 1: Install the Deadline Cloud submitter

The following sections guide you through the steps to install the Deadline Cloud submitter.

### Download the submitter installer

Before you can install the Deadline Cloud submitter, you must download the submitter installer. Currently, the Deadline Cloud submitter installer only supports Windows and Linux.

1. Sign in to the AWS Management Console and open the Deadline Cloud [console](#).
2. From the side navigation pane, choose **Downloads**.
3. Locate the **Deadline Cloud submitter installer** section.
4. Select the **installer** for your computer's operating system, and then choose **Download**.

### (Optional) Verify the authenticity of the downloaded software

To verify that the software you downloaded is authentic, use the following procedure for either Windows or Linux.

#### Windows

Verify the authenticity of your downloaded files by completing 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 /vfile
```

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
```

## Linux

Verify the authenticity of your downloaded files by using the gpg command line tool.

1. Import the OpenPGP key for the Deadline Cloud submitter installer by running the following command:

```
gpg --import --armor <<EOF
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGX6GQsBEADduUtJgqSXI+q7606fsFwEYKmbnlyL0xKvlq32EZuyv0otZo5L
le4m5Gg52AzrvPvDiUTLooAlvYeozaYyirIGsK08Ydz0Ftdjroiuh/mw9JSJDJRI
rnRn5yKet1JFzckjopA3pjsTBP6lW/mb1bDBDEwwwtH0x9lV7A03FJ9T7Uzu/qSh
q0/UYdkafro3cPASvkkqgDt2tCvURfBcUCAjZVFcLZcVD5iwXacxvKsxxS/e7kuVV
I1+VGT8Hj8XzWYhjCZx0LZk/fvpYPMYEEujN0fYUp6RtMIXve0C9awwMCy5nBG2J
eE2015DsCpTaBd4Fdr3LWcSs8JFA/YfP9auL3Ncz0ozPoVJt+fw8CB1VIX00J715
hvHDjcC+5v0wxqAlMG6+f/SX7CT8FXK+L3i0J5gBYUNXqHSxUdv8kt76/KVmQa1B
Ak1+MPKpMq+1hw++S3G/1XqwWadNQBRRw7dSZHymQVXvPp1nsgc3hV7K10M+6s6g
1g4mvFY41f6DhptwZLWyQXU8rBQpojvQfiSmDFrFPWF5BexesuVnkGIo1Qok1Kx
AVUSdJPVEJCteyy7td4FPhBaSqT5vW3+ANbr9b/uoRYWJvn17dN0cc9HuRh/Ai+I
nkfECo2WUDLZ0fEKGjGyFX+todWvJXjvc5kmE9Ty5vJp+M9Vvb8jd6t+mwARAQAB
tCxBV1MgRGVhZGxpbnUgQ2xvdWQgPGF3cy1kZWZkbGluZUBhbWF6b24uY29tPokC
VwQTAQgAQRyhbLhAwIwpqQeWoHH6pfbNP0a3bzzvBQJl+hkLAXsvBAUJA8JnAAUL
CQgHAgIiAgYVCgkICwIDFgIBAh4HAheAAAoJEPbNP0a3bzzvKswQAjXzKSAY8sY8
F6Eas2oYwIDDdDurs8FiEnFghjUE06MTt9AykF/jw+CQg2UzFtEy0bHBymhgmhXE
3buVeom96tgM3ZDfZu+sxi5pGX6oAQnZ6riztN+VpkpQmLgwtMGpSML13KLwnv2k
WK8mrR/fPMkfaewB7A6RIUYiW33GAL4KfMIs8/vIwIJw99NxHpZQVoU6dFpuDtE
10uxGcCqGJ7mAmo6H/YawSNp2Ns80gyqIKYo7o3LJ+WRroIR1Qyctq8gnR9JvYXX
42ASqLq5+0XKo4qh81blXKYqtc176BbbSNFjWnzIQgKDgNiHFZCdc0VgqDhw015r
NICbqqwwNLj/Fr2kecYx180Ktp10j00w5I0yh3bf3MVGWnYRdjvA1v+/CO+55N4g
z0kf50Lcdu5RtqV10XBCifn28pecqPaSdYcssYSR15DLiFktGbNzTGcZZwITTKQc
af8PPdTGtnnb6P+cdbw3bt9MvtN5/dgSHLThnS8MPEuNCtkTnpXshuVuBGgwBMdb
qUC+HjqvhZzbwns8dr5WI+6HWNBFgGANn6ageY158vVp0UkuNP8wcWjRARciHXZx
ku6W2jPTHDWGNrBQ02Fx7fd2QYJheIPPASHcfJ0+XgWCof45D0vAxAJ8gGg9Eq+
gFWhsx4NSHn2gh1gDZ410u/4exJ1lwPM
=uVaX
-----END PGP PUBLIC KEY BLOCK-----
```

EOF

2. Determine whether to trust the OpenPGP key. Some factors to consider when deciding whether to trust the above key include the following:
  - The internet connection you've used to obtain the GPG key from this website is secure.
  - The device that you are accessing this website on is secure.
  - AWS has taken measures to secure the hosting of the OpenPGP public key on this website.
3. If you decide to trust the OpenPGP key, edit the key to trust with gpg similar to the following example:

```
$ gpg --edit-key 0xB840C08C29A90796A071FAA5F6CD3CE6B76F3CEF
```

```
gpg (GnuPG) 2.0.22; Copyright (C) 2013 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 4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA  
                    trust: unknown      validity: unknown  
[ unknown] (1). AWS Deadline Cloud example@example.com
```

```
gpg> trust
```

```
pub 4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  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 4096R/4BF0B8D2 created: 2023-06-23 expires: 2025-06-22 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. Return to the Deadline Cloud [console](#) **Downloads** page and download the signature file for the Deadline Cloud submitter installer.
5. Verify the signature of the Deadline Cloud submitter installer by running:

```
gpg --verify ./DeadlineCloudSubmitter-linux-x64-installer.run.sig ./
DeadlineCloudSubmitter-linux-x64-installer.run
```

## Install the Deadline Cloud submitter

You can install a Deadline Cloud submitter with Windows or Linux. With the installer, you can install the following submitters:

- Maya 2024
- Nuke 14.0 - 15.0
- Houdini 19.5
- Keyshot 12
- Blender 3.6
- Unreal Engine 5

### 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 choosing the installation scope, choose **Next**.
  5. Choose **Next** again to accept the installation directory.
  6. Select **Integrated submitter for Nuke** 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 submitter 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. After the installer opens, follow the prompts on your screen.

You can install other submitters not listed here. We use Deadline Cloud libraries to build submitters. You can find the source code for these libraries and submitters in the [aws-deadline GitHub](#) organization.

## Step 2: Install and set up Deadline Cloud monitor

### To install the Deadline Cloud monitor desktop application

1. If you haven't already, sign in to the AWS Management Console and open the Deadline Cloud [console](#).
2. From the left navigation pane, choose **Downloads**.
3. In the **Deadline Cloud monitor** section, select the file for your computer's operating system.
4. To download the Deadline Cloud monitor, choose **Download**.

After you complete the download, 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.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 sync your credentials every 15 minutes.
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 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 following sections guide you through the steps to launch the Deadline Cloud submitter plugin in Blender, Nuke, Maya, and Houdini.

#### To launch the Deadline Cloud submitter in Blender

##### Note

Support for Blender is provided using the Conda environment for service-managed fleets. For more information, see [Default Conda queue environment](#).

1. Open **Blender**.
2. Open a Blender scene with dependencies that exist within the asset root directory.
3. In the **Render** menu, select the **Deadline Cloud Dialog**.
  - a. If you are not already authenticated in the Deadline Cloud submitter, the **Credentials Status** displays **NEEDS\_LOGIN**.
  - b. Choose **Login**.
  - c. A login browser window displays. Log in with your user credentials.

- d. Choose **Allow**. You are now logged in and the **Credentials Status** will show as **AUTHENTICATED**.
4. Choose **Submit**.

### To launch the Deadline Cloud submitter in Foundry Nuke

#### Note

Support for Nuke is provided using the Conda environment for service-managed fleets. For more information, see [Default Conda queue environment](#).

1. Open **Nuke**.
2. Open a Nuke script with dependencies that exist within the asset root directory.
3. Choose **Thinkbox**, and then choose **Submit to Deadline Cloud** to launch the submitter.
  - a. If you are not already authenticated in the Deadline Cloud submitter, the **Credentials Status** will show as **NEEDS\_LOGIN**.
  - b. Choose **Login**.
  - c. A login browser window displays. Log in with your user credentials.
  - d. Choose **Allow**. You are now logged in and the **Credentials Status** will show as **AUTHENTICATED**.
4. Choose **Submit**.

### To launch the Deadline Cloud submitter in Maya

#### Note

Support for Maya and Arnold for Maya(MtoA) is provided using the Conda environment for service-managed fleets. For more information, see [Default Conda queue environment](#).

1. Open **Maya**.
2. Set your project, and open a file that exists within the asset root directory.
3. Choose **Windows** → **Settings/Preferences** → **Plugin Manager**.

4. Search for **DeadlineCloudSubmitter**.
5. To load the Deadline Cloud submitter plugin, select **Loaded**.
  - a. If you are not already authenticated in the Deadline Cloud submitter, the **Credentials Status** will show as **NEEDS\_LOGIN**.
  - b. Choose **Login**.
  - c. A login browser window displays. Log in with your user credentials.
  - d. Choose **Allow**. You are now logged in and the **Credentials Status** displays as **AUTHENTICATED**.
6. (Optional) To load the Deadline Cloud submitter plugin every time you open Maya, choose **Auto-load**.
7. Select the Deadline Cloud shelf, then select the green button to launch the submitter.

### To launch the Deadline Cloud submitter in Houdini

#### Note

Support for Houdini is provided using the Conda environment for service-managed fleets. For more information, see [Default Conda queue environment](#).

1. Open **Houdini**.
2. In the **Network Editor**, select the **/out** network.
3. Press **tab**, and enter **deadline**.
4. Select the Deadline Cloud option, and connect it to your existing network.
5. Double-click the **Deadline Cloud node**.

### To launch the Deadline Cloud submitter in Keyshot

This assumes you've already downloaded Deadline Cloud and PySide2.

1. Copy or link the file **deadline-cloud-for-keyshot/keyshot\_script/Submit to AWS Deadline Cloud.py** to the KeyShot scripts folder.

For example, on Windows, the scripts folder location would be **C:/Users/*USER*/Documents/KeyShot 12/Scripts**.

2. Set the following environment variables.
  - a. Set the environment variable **DEADLINE\_PYTHON** as the path to the Python installation where deadline-cloud and PySide2 are located.  
  
For example, on Windows, if using Python 3.10, the command might be **set DEADLINE\_PYTHON=C:/Users/*USER*/AppData/Local/Programs/Python/Python310/python**.
  - b. Set the environment variable **DEADLINE\_KEYSHOT** as the path to the **keyshot\_submitter** folder.  
  
For example, on Windows, if the source is on your desktop the command might be **set DEADLINE\_KEYSHOT=C:/Users/*USER*/Desktop/deadline-cloud-for-keyshot/src/deadline/keyshot\_submitter**.
3. With the environment variables set, launch **KeyShot**.
4. To launch the submitter from KeyShot, choose **Windows, Scripting console, Submit to AWS Deadline Cloud**, and **Run**.

## Use the farm

If you have followed all of the getting started instructions, you have set up everything you need to start submitting jobs from your local workstation to your farm, and then monitoring those jobs and resources. For more information about submitting all kinds of jobs or monitoring, see the related topics below.

- [Jobs](#)
- [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.

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, and requeue jobs.

The 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 [Managing budgets and usage for Deadline Cloud](#).

## Topics

- [Share the Deadline Cloud monitor URL](#)
- [Open the Deadline Cloud monitor](#)
- [View queue and fleet details in Deadline Cloud](#)
- [View and manage jobs, steps, and tasks in Deadline Cloud](#)
- [View job details in Deadline Cloud](#)
- [View a step in Deadline Cloud](#)
- [View a task in Deadline Cloud](#)
- [View logs in Deadline Cloud](#)
- [Download finished output in Deadline Cloud](#)

## 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 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 at <https://console.aws.amazon.com/deadline-cloud/>.
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 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 at <https://console.aws.amazon.com/deadline-cloud/>.
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 at <https://console.aws.amazon.com/deadline-cloud/>.

-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 **AWS 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.
  - If this is your first time opening the 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.

## View queue and fleet details in Deadline Cloud

You can use the AWS 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 see queue and fleet details. If the details aren't showing, 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 the queue to see details about. 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 that you want 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 the fleet to see details about. 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 that you want from the **Fleets** list in the left navigation pane.

# View and manage jobs, steps, and tasks in Deadline Cloud

When you select a queue, the AWS Deadline Cloud job 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, such as changing prioritization or viewing step to step dependencies.
- View additional details using the job's parameters.

For more information, see [View 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 job details in Deadline Cloud

The **Job monitor** page in AWS Deadline Cloud 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 see a list of steps for the job, and then choose a step from the list of steps to see the tasks for the job. After you choose an item, you can use the **Actions** menu for that item to see 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. From the **Jobs** list, select a job to view its details.

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.
- View and modify the properties of a job. You can view the dependencies between steps in the job, and change the priority of the job. Generally, jobs with a higher priority complete sooner.
- 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.

## 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 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.
- 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.

- 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 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 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.
- View task logs. For more information, see [View logs in Deadline Cloud](#).
- View that parameters that were set when the task was created.
- Download the output of the task. When you download the output of a task, it only contains the output generated by the selected task.

## View 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.

### To view session logs

1. Follow the steps in [View 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 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**.

# 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 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.

# Deadline Cloud farms

A farm is a container for queues that manage jobs and fleets of compute resources that perform tasks.

## Topics

- [Create a farm](#)
- [Delete a farm](#)
- [Edit a farm](#)

## 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)** Choose **Add new tag** to add one or more tags to your farm.
7. Choose **Create farm**. After creation, your farm displays.

## Delete a farm

1. From the Deadline Cloud dashboard, choose **Farms and other resources**.
2. In the farms list, select the farm or farms you want to delete and then choose **Delete**.



## Edit a farm

1. From the Deadline Cloud dashboard, choose **Farms and other resources**.
2. In the farms list, select the farm or farms you want to delete and then choose **Edit**.
3. In the edit window that displays, change the farm name or description, and then choose **Save changes**.

# 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](#)
- [Create a queue environment](#)
- [Delete a queue](#)
- [Edit a queue](#)
- [Associate a queue and fleet](#)

## 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 need to complete the following steps.

**⚠ Important**

We strongly recommend specifying users and groups for run-as functionality. If you bypass it, 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 users and groups](#).

- 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.
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. 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.

Conda provides packages from *channels*. A channel is a location where packages are stored. Deadline Cloud provides a channel, `deadline-cloud`, that hosts packages that support partner DCC applications and renderers. The packages are:

- Blender
  - `blender=3.6`
  - `blender-openjd`
- Houdini
  - `houdini=19.5`
  - `houdini-openjd`
- Maya
  - `maya=2024`
  - `maya-mtoa=2024.5.3`
  - `maya-openjd`
- Nuke
  - `nuke=15`
  - `nuke-openjd`

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.
- `CondaChannels` – a space separated list of [Conda channels](#) such as `deadline-cloud`, `conda-forge`, or `s3://DOC-EXAMPLE-BUCKET/conda/channel`. The default is `deadline-cloud`, a channel available to service-managed fleets that provides partner DCC applications and renderers.

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.*`.

## Delete a queue

### Warning

You can't recover the jobs in a queue if you delete the queue. Deleting the queue also deletes the jobs in that queue.

1. From the Deadline Cloud dashboard, choose **Farms and other resources**.
2. In the farms list, select the farm that contains the queue to delete.
3. Select the queue, and then choose **Delete**.
4. In the confirmation window, choose **Delete**. Your queue and all of the jobs in the queue are deleted.

## Edit a queue

1. From the Deadline Cloud dashboard, choose **Farms and other resources**.
2. In the farms list, select the farm that contains the queue to edit.
3. Select the queue, and then choose **Edit**.
4. You can edit the name, description, budget requirement, Run as user option, and assigned service role. You can also associate an existing fleet with your queue.
5. Choose **Save changes**.

## Associate a queue and fleet

1. Select the **Queue** you to associate with a fleet.
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 **Associated**.

# Manage Deadline Cloud fleets

This section explains how to manage service-managed fleets (SMF) 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 this service, Deadline Cloud. These default settings are designed to be the efficient and cost-effective.
- Customer-managed fleets (CMFs) are fleets of workers that you manage. A CMF may reside within AWS infrastructure, on-premise, or in a co-located data center. A CMF provides full control and responsibility of the fleet. This includes provisioning, operations, management, and decommissioning workers in the fleet.

## Topics

- [Manage Deadline Cloud service-managed fleets](#)
- [Manage Deadline Cloud customer-managed fleets](#)

## Manage Deadline Cloud service-managed 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.

1. To create a service-managed fleet (SMF), navigate to the farm you want to create the fleet in.
2. Select the **Fleets** tab.
3. Choose **Create fleet**.
4. Enter a **Name** for your fleet.
5. Enter a **Description**. A clear description can help you quickly identify your fleet's purpose.
6. Select **Service-managed** fleet type.
7. Choose either the **Spot** or **On-demand** instance market option for your fleet. Spot instances are unreserved capacity that you can use at a discounted price, but may be interrupted by On-demand requests. On-demand instances are priced by the second, but have no long-term commitment, and will not be interrupted. By default, fleets use Spot instances.

8. **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.
9. 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.
10. Choose **Next**.
11. Enter the minimum and maximum **vCPU's** that you require for you fleet.
12. Enter the minimum and maximum **memory** that you require for you fleet.
13. **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.
14. **Optional** You can specify the size of the Amazon Elastic Block Store (Amazon EBS) gp3 volume that will be attached to the workers in this fleet. For more information, see the [EBS user guide](#).
15. Choose **Next**.
16. **Optional** Define custom worker requirements that define features of this fleet that can be combined with custom host requirements specified on job submissions. One example is a particular license type if you plan to connect your fleet to your own license server.
17. Choose **Next**.
18. **Optional** To associate your fleet with a queue, select a **queue** from the dropdown. If the queue 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](#).
19. Choose **Next**.
20. **Optional** To add a tag to your fleet, choose **Add new tag**, and then enter the **key** and **value** for that tag.
21. Choose **Next**.
22. Review your fleet settings, and then choose **Create fleet**. After creation, your fleet displays.

## 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, you should keep in mind 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](#).

**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 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](#).

## Manage Deadline Cloud customer-managed fleets

This section explains how to manage a customer-managed fleet (CMF) for Deadline Cloud.

CMFs are fleets of workers that you manage. A CMF may reside within AWS infrastructure, on-premise, or in a co-located data center. A CMF provides full control and responsibility of the fleet. This includes provisioning, operations, management, and decommissioning workers in the fleet.

## Topics

- [Create a customer-managed fleet](#)
- [Worker host setup and configuration](#)
- [Manage access to Windows job user secrets](#)
- [Install and configure software required for jobs](#)
- [Configuring AWS credentials](#)
- [Create an Amazon Machine Image](#)
- [Create fleet infrastructure with an Amazon EC2 Auto Scaling group](#)
- [Connect customer-managed fleets to a license endpoint](#)

## Create a customer-managed fleet


Complete the tasks in this section to create a customer-managed fleet (CMF) using the Deadline Cloud command line interface (Deadline Cloud CLI)

Deadline Cloud console

### To use the Deadline Cloud console to create a customer-managed fleet

1. Open the Deadline Cloud [console](#).
2. Select **Farms**. A list of available farms displays.
3. Select the name of the **Farm** you want to work in.
4. Select the **Fleets** tab.
5. Choose **Create fleet**.
6. Enter a **Name** for your fleet.
7. **(Optional)** Enter a **Description** for your fleet.
8. Select **Customer managed** for **Fleet type**.
9. Select an Auto Scaling type. For more information, see [Use EventBridge to handle Auto Scaling events](#).

- **No scaling:** You are creating an on-premise fleet and want opt out of Deadline Cloud Auto Scaling.
  - **Scaling recommendations:** You are creating an Amazon Elastic Compute Cloud (Amazon EC2) fleet.
10. Select your fleet's service access.
    - a. If you previously created a fleet, select Choose a service role, then select the role you created.
    - b. If you haven't previously created a fleet, select Create and use an new service role, then provide the new role name and description.
  11. Review your selections, then choose **Next**.
  12. Select an **operating system** for your fleet. All of a fleet's workers must have a common operating system.
  13. Select the **host CPU architecture**.
  14. Select the following hardware requirements for the worker hosts in this fleet.
    - a. Select the minimum and maximum vCPU and memory **Hardware requirements** to meet the workload demands of your fleets.
    - b. **(Optional)** Select **GPU requirement** then enter the minimum and maximum GPUs.
  15. Review your selections, then choose **Next**.
  16. **(Optional)** Define custom worker requirements.
  17. Using the dropdown, select one or more **queues** to associate with the fleet.

 **Note**

We recommend associating a fleet only with queues that are all in the same trust boundary. This ensures a strong security boundary between running jobs on the same worker.

18. Review the queue associations, then select **Next**.
19. **(Optional)** For Default Conda queue environment, we'll create an environment for your queue that will install Conda packages requested by jobs.

**Note**

Since you are creating a queue for a CMF, be sure to uncheck this option.

20. **(Optional)** Add tags to your CMF. For more information, see [Tagging your AWS resources](#).
21. Review fleet configuration and make any changes.
22. Select **Create fleet**.
23. Select the **Fleets** tab, then note the **Fleet ID**.

## AWS CLI

**To use the Deadline Cloud CLI to create a customer-managed fleet**

1. Open the AWS CLI.
2. Edit `fleet-trust-policy.json`.
  - a. Add the following IAM policy, replacing the *ITALICIZED* text with your AWS account ID and Deadline Cloud farm ID.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "ACCOUNT_ID"
        },
        "ArnEquals": {
          "aws:SourceArn":
            "arn:aws:deadline:*:ACCOUNT_ID:farm/FARM_ID"
        }
      }
    }
  ]
}
```

```
}

```

- b. Save your changes.
3. Edit `create-cmf-fleet.json`.
    - a. Add the following IAM policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "deadline:AssumeFleetRoleForWorker",
        "deadline:UpdateWorker",
        "deadline>DeleteWorker",
        "deadline:UpdateWorkerSchedule",
        "deadline:BatchGetJobEntity",
        "deadline:AssumeQueueRoleForWorker"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalAccount": "${aws:ResourceAccount}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalAccount": "${aws:ResourceAccount}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",

```

```

        "logs:GetLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalAccount": "${aws:ResourceAccount}"
        }
      }
    }
  ]
}

```

b. Save your changes.

4. Add an IAM role for the workers in your fleet to use.

```

aws iam create-role --role-name FleetWorkerRoleName --assume-role-policy-
document file://fleet-trust-policy.json
aws iam put-role-policy --role-name FleetWorkerRoleName --policy-name
FleetWorkerPolicy --policy-document file://fleet-policy.json

```

5. Edit `create-fleet-request.json`.

a. Add the following IAM policy, replacing the *ITALICIZED* text with your CMF's values.

**Note**

You can find the *ROLE\_ARN* in the `create-cmf-fleet.json`.  
For the *OS\_FAMILY*, you must choose one of `linux`, `macos` or `windows`.

```

{
  "farmId": "FARM_ID",
  "displayName": "FLEET_NAME",
  "description": "FLEET_DESCRIPTION",
  "roleArn": "ROLE_ARN",
  "minWorkerCount": 0,
  "maxWorkerCount": 10,
  "configuration": {
    "customerManaged": {
      "mode": "NO_SCALING",
      "workerCapabilities": {
        "vCpuCount": {

```

```
        "min": 1,
        "max": 4
    },
    "memoryMiB": {
        "min": 1024,
        "max": 4096
    },
    "osFamily": "OS_FAMILY",
    "cpuArchitectureType": "x86_64",
  },
},
}
```

b. Save your changes.

6. Create your fleet.

```
aws deadline create-fleet --cli-input-json file://create-fleet-request.json
```

## Worker host setup and configuration

A worker host refers to a host machine that runs an Deadline Cloud worker. The tasks in this section explains how to set up the worker host and configure it to your specific needs. Each worker host runs a program call a worker agent. The worker agent is responsible for the following:

- Managing the worker life cycle.
- Synchronizing assigned work and its progress and results.
- Monitoring running work.
- Forwarding logs to configured destinations.

We recommend that you use the provided Deadline Cloud worker agent. The worker agent is open source and we encourage feature requests, but you can also develop and customize to fit your needs.

In order to complete the tasks in the following sections, you will need the following prerequisites:

## Linux

- A Linux-based Amazon Elastic Compute Cloud (Amazon EC2) instance. We recommend Amazon Linux 2023.
- sudo privileges.
- Python 3.9 or above.

## Windows

- A Windows-based Amazon Elastic Compute Cloud (Amazon EC2) instance. We recommend Windows Server 2022.
- Administrator access to the worker host
- Python 3.9 or above installed for all-users

## Create and configure a Python virtual environment

You can create a Python virtual environment if you have installed Python 3.9 or greater and placed it in your PATH.

### To create and activate a Python virtual environment

1. Open the AWS CLI.
2. Create and activate a Python virtual environment.

```
python3 -m venv /opt/deadline/worker
source /opt/deadline/worker/bin/activate
pip install --upgrade pip
```

## Install Deadline Cloud worker agent

After you've set up your Python virtual environment, you can install the Deadline Cloud worker agent Python packages.

To install the worker agent Python packages



1. Open a terminal.
  - a. On Linux, open a terminal as the root user (or use `sudo / su`)
  - b. On Windows, open an administrator command prompt or PowerShell terminal.
2. Download and install the Deadline Cloud worker agent packages from PyPI:

**Note**

On Windows, it is required that the agent files are installed into Python's global site-packages directory. Python virtual environments are not currently supported.

```
python -m pip install deadline-cloud-worker-agent
```

## Configure the Deadline Cloud worker agent

You can configure the Deadline Cloud worker agent settings in three ways. We recommend your use the operating system set up through `install-deadline-worker`.

**Command line arguments** — You can specify arguments when you run the Deadline Cloud worker agent from the command line. Some configuration settings are not available through command line arguments. To see all the available command line arguments, enter `deadline-worker-agent --help` to see all the available command line arguments.

**Environment variables** — You can configure the Deadline Cloud worker agent by setting environment variable beginning with `DEADLINE_WORKER_`. For example, you can use `export DEADLINE_WORKER_VERBOSE=true` to set the worker agent's output to verbose. For more examples and information, see `/etc/amazon/deadline/worker.toml.example` on Linux or `C:\ProgramData\Amazon\Deadline\Config\worker.toml.example` on Windows.

**Configuration file** — When you install the worker agent, it creates a configuration file located at `/etc/amazon/deadline/worker.toml` on Linux or `C:\ProgramData\Amazon\Deadline\Config\worker.toml` on Windows. The worker agent then loads this configuration file when it starts. You can use the example configuration file (`/etc/amazon/deadline/worker.toml.example` on Linux or `C:\ProgramData\Amazon\Deadline\Config\worker.toml.example` on Windows) to tailor the default worker agent configuration file to meet your specific needs.

Finally, we recommend you enable the worker agent auto shutdown. This allows the worker fleet to scale up when needed and to shutdown when the rendering job finishes. This auto scaling helps ensure you're only using resources as needed.

- To enable auto shutdown. As a **root** user:
  - First, install the worker agent with parameters **--allow-shutdown**.

#### Linux

Enter:

```
/opt/deadline/worker/bin/install-deadline-worker \  
  --farm-id FARM_ID \  
  --fleet-id FLEET_ID \  
  --region REGION \  
  --allow-shutdown
```

#### Windows

Enter:

```
install-deadline-worker ^  
  --farm-id FARM_ID ^  
  --fleet-id FLEET_ID ^  
  --region REGION ^  
  --allow-shutdown
```

## Create job users and groups

This section describes the required user and group relationship between the agent user and the `jobRunAsUser` defined on your queues.

The Deadline Cloud worker agent should run as a dedicated agent-specific user on the host. You should configure the `jobRunAsUser` property of Deadline Cloud queues so that workers will run the queue jobs as a specific operating system user and group. This means you can control the shared filesystem permissions that your jobs have. It also provides as an important security boundary between your jobs and the worker agent user.

### To create Linux job users and groups

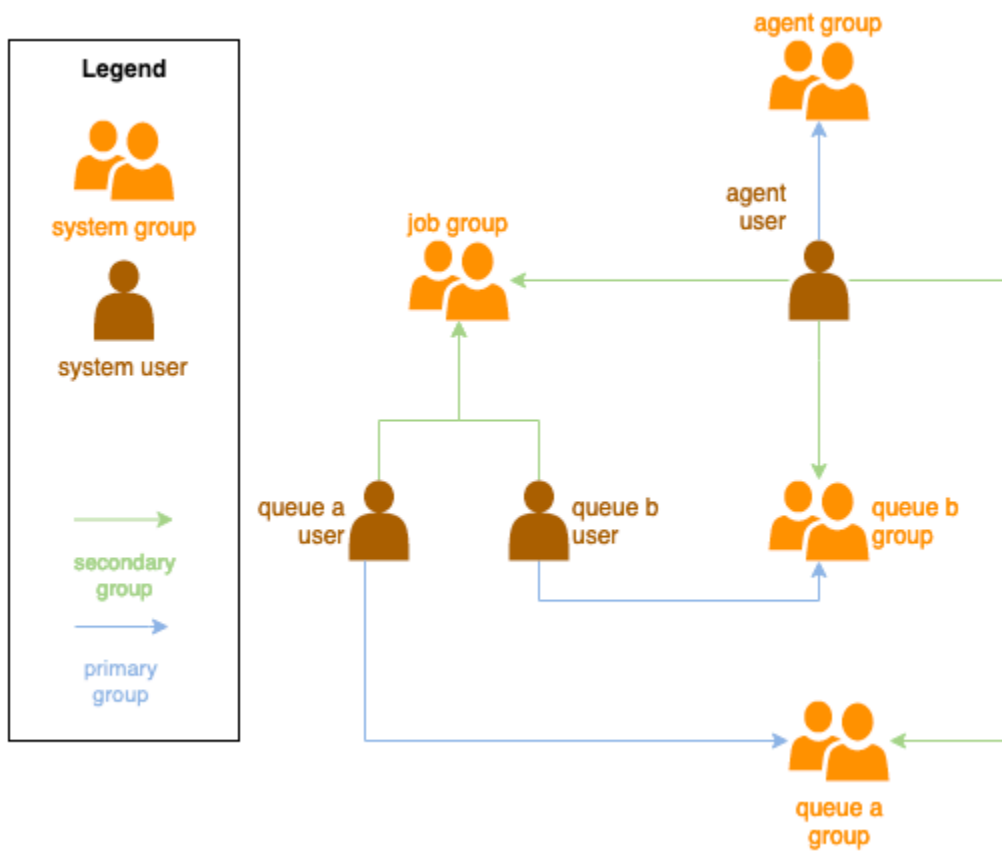
To set up your agent-user and `jobRunAsUser`, ensure you meet the following requirements:

- There is a group for each `jobRunAsUser`, and it is the primary group for its corresponding `jobRunAsUser`.
- The agent-user belongs to the primary group of the `jobRunAsUser` for the queues where the worker obtains work. For security best-practices, we recommend this as a secondary group of the agent-user. This shared group allows the worker agent to make files available for the job while it is running.
- A `jobRunAsUser` *doesn't* belong to the agent-user's primary group. For security best-practices:
  - Sensitive files written by the worker agent are owned by the agent's primary group.
  - If a `jobRunAsUser` belongs to this group, and files the worker agent writes may be accessible by the jobs submitted to the queue running on the worker.
- The default AWS Region should match the Region of the farm the worker belongs to. For more information, see [Configuration and credential file settings](#).

This should be applied to:

- The agent-user
- All queue `jobRunAsUser` accounts on the worker
- The agent-user can run `sudo` commands as the `jobRunAsUser`.

The following diagram illustrates the relationship between the agent user and the `jobRunAsUser` users and groups for queues associated with the fleet.



## To create Windows users

All queue `jobRunAsUser` users must exist, their passwords must match the value of the secret specified in their queue's `JobRunAsUser` field, and the agent-user must be able to log on as those users.

The following instructions describe how to add a local Windows user account for a queue `jobRunAsUser`.

1. Open PowerShell or command-prompt as administrator.
2. Create the user.

```
net user JOB_USER /add
```

3. Set the password. This password must match the secret specified in the queue's `jobRunAsUser` field.

```
net user JOB_USER *
```

4. Create a local profile and home directory for the user. Run the following command and enter the password for the user when prompted.

```
runas /profile /user:JOB_USER "cmd.exe /C"
```

## Manage access to Windows job user secrets

When you configure a queue with a Windows `jobRunAsUser`, you must specify an AWS Secrets Manager secret. The value of this secret is expected to be JSON-encoded object of the form:

```
{
  "password": "JOB_USER_PASSWORD"
}
```

For Workers to run jobs as the queue's configured `jobRunAsUser`, the fleet's IAM role must have permissions to get the value of the secret. If the secret is encrypted using a customer-managed KMS key, then the fleet's IAM role must also have permissions to decrypt using the KMS key.

It is highly recommended to follow the principle of least-privilege for these secrets. This means that access to fetch the secret value of a queue's `jobRunAsUser` → `windows` → `passwordArn` should be:

- granted to a fleet role when a queue-fleet association is created between the fleet and queue
- revoked from a fleet role when a queue-fleet association is deleted between the fleet and queue

Further, the AWS Secrets Manager secret containing the `jobRunAsUser` password should be deleted when it is no longer being used.

## Grant access to a password secret

Deadline Cloud fleets require access to the `jobRunAsUser` password stored in the queue's password secret when the queue and fleet are associated. We recommend using the AWS Secrets Manager resource policy to grant access to the fleet roles. If you strictly adhere to this guideline, it is easier to determine which fleet roles have access to the secret.

### To grant access to the secret

1. Open the AWS Secret Manager console to the secret.

2. In the “Resource permissions” section, add a policy statement of the form:

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    // ...
    {
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "FLEET_ROLE_ARN"
      },
      "Action" : "secretsmanager:GetSecretValue",
      "Resource" : "*"
    }
    // ...
  ]
}
```

## Revoke access to a password secret

When a fleet no longer requires access to a queue, remove access to the password secret for the queue `jobRunAsUser`. We recommend using the AWS Secrets Manager resource policy to grant access to the fleet roles. If you strictly adhere to this guideline, it is easier to determine which fleet roles have access to the secret.

### To revoke access to the secret

1. Open the AWS Secret Manager console to the secret.
2. In the Resource permissions section, remove the policy statement of the form:

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    // ...
    {
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "FLEET_ROLE_ARN"
      },
      "Action" : "secretsmanager:GetSecretValue",
      "Resource" : "*"
    }
  ]
}
```

```
    }  
    // ...  
  ]  
}
```

## Install and configure software required for jobs

After you set up the Deadline Cloud worker agent, you can prepare the worker host with any software that is required to run jobs.

When you submit a job to a queue with an associated `jobRunAsUser`, the job runs as that user. All commands must be available in the `PATH` of that user.

On Linux, you might specify the `PATH` for a user in one of the following:

- their `~/.bashrc` or `~/.bash_profile`
- system configuration files such as `/etc/profile.d/*` and `/etc/profile`
- shell startup scripts: `/etc/bashrc`.

On Windows, you might specify the `PATH` for a user in one of the following:

- their user-specific environment variables
- the system-wide environment variables

## Install digital content creation tool adapters

Deadline Cloud provides digital content creation (DCC) applications with first-party integration support. To use these integrations on a customer-managed fleet, you must install the DCC software and the adapters.

To install DCC adapters on a customer-managed fleet

1. Open the a terminal.
  - a. On Linux, open a terminal as the `root` user (or use `sudo / su`)
  - b. On Windows, open an administrator command prompt or PowerShell terminal.
2. Install the Deadline Cloud adapter packages.

```
pip install deadline deadline-cloud-for-maya deadline-cloud-for-nuke deadline-  
cloud-for-blender
```

## Configuring AWS credentials

This section explains how to configure AWS credentials.

This initial phase of the worker life cycle is bootstrapping. In this phase the worker agent software creates a worker in your fleet, and obtains AWS credentials from your fleet's role for further operation.

### AWS credentials for Amazon EC2

#### To configure AWS credentials for Amazon EC2

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Select Roles in the navigation pane, then Create role.
3. Select **AWS service**.
4. Select **EC2** as the **Service or use case**, then select **Next**.
5. Attach the AWSDeadlineCloud-WorkerHost AWS managed policy.


### On-premise AWS credentials

#### To configure AWS on-premise credentials

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Select Roles in the navigation pane, then Create role.
3. Select **AWS account**, then select **Next**.
4. Attach the AWSDeadlineCloud-WorkerHost AWS managed policy.
5. Generate AWS IAM access and secret keys for the IAM user:
  - a. For IAM Role Anywhere, see [IAM Roles Anywhere](#).
  - b. For the most secure way to set up credentials on the host, see [Obtaining temporary security credentials from AWS Identity and Access Management Roles Anywhere](#).




- c. You can also use CLI as alternative authentication, for more information see [Authenticate with IAM user credentials](#).
6. Store these keys in the agent-user's AWS credentials file on the worker host filesystem.
    - a. On Linux, this is located at `~/.aws/credentials`
    - b. On Windows, this is located at `%USERPROFILE%\aws\credentials`

 **Note**

Credentials should only be accessible by the OS user name (`deadline-worker-agent`) who installed the worker agent.

```
# Replace keys below
[default]
aws_access_key_id=ACCESS_KEY_ID
aws_secret_access_key=SECRET_ACCESSSS_KEY
```

7. Change the `deadline-worker-agent` owner and permissions.

 **Note**

If you changed the OS user (`deadline-worker-agent`) name when you installed the worker agent, use that name instead.

## Create an Amazon Machine Image

Complete the tasks in this section to create an Amazon Machine Image (AMI). to use in an Amazon Elastic Compute Cloud (Amazon EC2) customer-managed fleet (CMF). You must create an Amazon EC2 instance before proceeding. For more information see: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/LaunchingAndUsingInstances.html>.

 **Important**

Creating an AMI creates a snapshot of the Amazon EC2 instance's attached volumes. Any software installed on the instance persists so instances, which are reused when you

launch instances from the AMI. We recommend adopting a patching strategy and regularly updating any new AMI with updated software before applying to your fleet.

## Prepare the Amazon EC2 instance

Before you build an AMI, you must delete the worker state. The worker state persists between worker agent launches. If this state persists onto the AMI, then all instances launched from it will share the same state.

We also recommend you delete any existing log files. Log files can remain on an Amazon EC2 instance when you prepare the AMI. Deleting these files minimizes confusion when diagnosing possible issue in worker fleets that use the AMI.

You should also enable the worker agent system service so the Deadline Cloud worker agent launch when the Amazon EC2 is started.

Finally, we recommend you enable the worker agent auto shutdown. This allows the worker fleet to scale up when needed and to shutdown when the rendering job finishes. This auto scaling helps ensure you're only using resources as needed.

### To prepare the Amazon EC2 instance

1. Open the Amazon EC2 console.
2. Launch an Amazon EC2 instance. For more information, see [Launch your instance](#).
3. Set up the host to connect to your identity provider (IdP), then mount any shared filesystem it needs.
4. Follow the tutorials to [Install Deadline Cloud worker agent](#), then [Configure worker agent](#), and [Create job users and groups](#).
5. If you are preparing an AMI based on Amazon Linux 2023 to run software compatible with the VFX Reference Platform, you need to update several requirements. For information, see [VFX Reference Platform compatibility](#).
6. Open a terminal.
  - a. On Linux, open a terminal as the root user (or use `sudo / su`)
  - b. On Windows, open an administrator command prompt or PowerShell terminal.
7. Ensure the worker service is not running and configured to start on boot:

- a. On Linux, run

```
systemctl stop deadline-worker  
systemctl enable deadline-worker
```

- b. On Windows, run

```
sc.exe stop DeadlineWorker  
sc.exe config DeadlineWorker start= auto
```

8. Delete the worker state.

- a. On Linux, run

```
rm -rf /var/lib/deadline/*
```

- b. On Windows, run

```
del /Q /S %PROGRAMDATA%\Amazon\Deadline\Cache\*
```

9. Delete the log files.


- a. On Linux, run

```
rm -rf /var/log/amazon/deadline/*
```

- b. On Windows, run

```
del /Q /S %PROGRAMDATA%\Amazon\Deadline\Logs\*
```

10. On Windows, it is recommended to run the Amazon EC2Launch Settings application found in the Start menu to complete the final host preparation and shutdown of the instance.

 **Note**

You **MUST** choose Shutdown without Sysprep and never choose Shutdown with Sysprep. Shutting down with Sysprep will cause all local users to become unusable. For more information, see [Before you Begin section of the Create a custom AMI topic of the User Guide for Windows Instances](#).

## Build the AMI

### To build the AMI

1. Open the Amazon EC2 console.
2. Select **Instances** in the navigation pane, then select your instance.
3. Choose **Instance state**, then **Stop instance**.
4. After the instance is **Stopped**, choose **Actions**.
5. Choose **Image and templates**, then **Create image**.
6. Enter an **Image name**.
7. **(Optional)** Enter a description for your image.
8. Choose **Create image**.

## Create fleet infrastructure with an Amazon EC2 Auto Scaling group

This section explains how to create an Amazon EC2 Auto Scaling fleet.

Use the AWS CloudFormation YAML template below to create an Amazon EC2 Auto Scaling (Auto Scaling) group, an Amazon Virtual Private Cloud (Amazon VPC) with two subnets, an instance profile, and an instance access role. These are required to launch instance using Auto Scaling in the subnets.

You should review and update the list of instance types to fit your rendering needs.

### To create an Amazon EC2 Auto Scaling fleet

1. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
2. Create a CloudFormation template with parameters `Farm ID`, `Fleet ID`, and `AMI ID`.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Amazon Deadline Cloud customer-managed fleet
Parameters:
  FarmId:
    Type: String
    Description: Farm ID
  FleetId:
    Type: String
    Description: Fleet ID
```

```
AMIId:
  Type: String
  Description: AMI ID for launching Workers
Resources:
  deadlineVPC:
    Type: 'AWS::EC2::VPC'
    Properties:
      CidrBlock: 100.100.0.0/16
  deadlineWorkerSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: !Join
        - ' '
        - - Security Group created for deadline workers in fleet
          - !Ref FleetId
      GroupName: !Join
        - ''
        - - deadlineWorkerSecurityGroup-
          - !Ref FleetId
      SecurityGroupEgress:
        - CidrIp: 0.0.0.0/0
          IpProtocol: '-1'
      SecurityGroupIngress: []
      VpcId: !Ref deadlineVPC
  deadlineIGW:
    Type: 'AWS::EC2::InternetGateway'
    Properties: {}
  deadlineVPCGatewayAttachment:
    Type: 'AWS::EC2::VPCGatewayAttachment'
    Properties:
      VpcId: !Ref deadlineVPC
      InternetGatewayId: !Ref deadlineIGW
  deadlinePublicRouteTable:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref deadlineVPC
  deadlinePublicRoute:
    Type: 'AWS::EC2::Route'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref deadlineIGW
  DependsOn:
    - deadlineIGW
```

```
- deadlineVPCGatewayAttachment
deadlinePublicSubnet0:
  Type: 'AWS::EC2::Subnet'
  Properties:
    VpcId: !Ref deadlineVPC
    CidrBlock: 100.100.16.0/22
    AvailabilityZone: !Join
      - ''
      - - !Ref 'AWS::Region'
        - a
deadlineSubnetRouteTableAssociation0:
  Type: 'AWS::EC2::SubnetRouteTableAssociation'
  Properties:
    RouteTableId: !Ref deadlinePublicRouteTable
    SubnetId: !Ref deadlinePublicSubnet0
deadlinePublicSubnet1:
  Type: 'AWS::EC2::Subnet'
  Properties:
    VpcId: !Ref deadlineVPC
    CidrBlock: 100.100.20.0/22
    AvailabilityZone: !Join
      - ''
      - - !Ref 'AWS::Region'
        - c
deadlineSubnetRouteTableAssociation1:
  Type: 'AWS::EC2::SubnetRouteTableAssociation'
  Properties:
    RouteTableId: !Ref deadlinePublicRouteTable
    SubnetId: !Ref deadlinePublicSubnet1
deadlineInstanceAccessAccessRole:
  Type: 'AWS::IAM::Role'
  Properties:
    RoleName: !Join
      - '-'
      - - deadline
        - InstanceAccess
        - !Ref FleetId
    AssumeRolePolicyDocument:
      Statement:
        - Effect: Allow
          Principal:
            Service: ec2.amazonaws.com
          Action:
            - 'sts:AssumeRole'
```

```

    Path: /
    ManagedPolicyArns:
      - 'arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy'
      - 'arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore'
      - 'arn:aws:iam::aws:policy/AWSDeadlineCloud-WorkerHost'
    deadlineInstanceProfile:
      Type: 'AWS::IAM::InstanceProfile'
      Properties:
        Path: /
        Roles:
          - !Ref deadlineInstanceAccessAccessRole
    deadlineLaunchTemplate:
      Type: 'AWS::EC2::LaunchTemplate'
      Properties:
        LaunchTemplateName: !Join
          - ''
          - - deadline-LT-
            - !Ref FleetId
        LaunchTemplateData:
          NetworkInterfaces:
            - DeviceIndex: 0
              AssociatePublicIpAddress: true
              Groups:
                - !Ref deadlineWorkerSecurityGroup
              DeleteOnTermination: true
          ImageId: !Ref AMIID
          InstanceInitiatedShutdownBehavior: terminate
          IamInstanceProfile:
            Arn: !GetAtt
              - deadlineInstanceProfile
              - Arn
          MetadataOptions:
            HttpTokens: required
            HttpEndpoint: enabled
    deadlineAutoScalingGroup:
      Type: 'AWS::AutoScaling::AutoScalingGroup'
      Properties:
        AutoScalingGroupName: !Join
          - ''
          - - deadline-ASG-autoscalable-
            - !Ref FleetId
        MinSize: 0
        MaxSize: 10

```

```
VPCZoneIdentifier:
  - !Ref deadlinePublicSubnet0
  - !Ref deadlinePublicSubnet1
NewInstancesProtectedFromScaleIn: true
MixedInstancesPolicy:
  InstancesDistribution:
    OnDemandBaseCapacity: 0
    OnDemandPercentageAboveBaseCapacity: 100
    SpotAllocationStrategy: capacity-optimized
    OnDemandAllocationStrategy: lowest-price
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateId: !Ref deadlineLaunchTemplate
      Version: !GetAtt
        - deadlineLaunchTemplate
        - LatestVersionNumber
    Overrides:
      - InstanceType: m5.large
      - InstanceType: m5d.large
      - InstanceType: m5a.large
      - InstanceType: m5ad.large
      - InstanceType: m5n.large
      - InstanceType: m5dn.large
      - InstanceType: m4.large
      - InstanceType: m3.large
      - InstanceType: r5.large
      - InstanceType: r5d.large
      - InstanceType: r5a.large
      - InstanceType: r5ad.large
      - InstanceType: r5n.large
      - InstanceType: r5dn.large
      - InstanceType: r4.large
MetricsCollection:
  - Granularity: 1Minute
  Metrics:
    - GroupMinSize
    - GroupMaxSize
    - GroupDesiredCapacity
    - GroupInServiceInstances
    - GroupTotalInstances
    - GroupInServiceCapacity
    - GroupTotalCapacity
```

### 3. After you create the IAM roles, you need to acknowledge the following:



- Credentials from the IAM role that are attached to your worker's Amazon EC2 instance are available to *all* processes running on that worker, which includes jobs. The worker should have the least privileges to operate: `deadline:CreateWorker` and `deadline:AssumeFleetRoleForWorker`.
- The worker agent obtains credentials for the queue role and configures them for use by running jobs. The Amazon EC2 instance profile role shouldn't include permissions that are needed by your jobs.

## Auto scale your Amazon EC2 fleet with Deadline Cloud scale recommendation feature

Deadline Cloud leverages an Amazon EC2 Auto Scaling (Auto Scaling) group to scale the Amazon EC2 customer-managed fleet (CMF) automatically. You need to configure the fleet mode as well as deploy the required infrastructure in your account in order to make your fleet auto scale. The infrastructure you deployed will work for all fleets, so you only need to set it up once.

The basic workflow is: you configure your fleet mode to auto scale, and then Deadline Cloud will send out an EventBridge event for that fleet whenever recommended fleet size changes (one event contains fleet id, recommended fleet size, and other metadata). You will have an EventBridge rule to filter the relevant events and have a Lambda to consume them. The Lambda will integrate with Amazon EC2 Auto Scaling `AutoScalingGroup` to scale the Amazon EC2 fleet automatically.

### Set fleet mode to `EVENT_BASED_AUTO_SCALING`

Configure your fleet mode to `EVENT_BASED_AUTO_SCALING`. You can use the console to do this, or use the AWS CLI to directly call the `CreateFleet` or `UpdateFleet` API. After the mode is configured, Deadline Cloud starts sending EventBridge events whenever the recommended fleet size changes.

- Example `UpdateFleet` command:

```
aws deadline update-fleet \  
  --farm-id FARM_ID \  
  --fleet-id FLEET_ID \  
  --configuration file://configuration.json
```

- Example `CreateFleet` command:

```
aws deadline create-fleet \  
  --farm-id FARM_ID \  
  --display-name "Fleet name" \  
  --max-worker-count 10 \  
  --configuration file://configuration.json
```

The following is an example of `configuration.json` used in the CLI commands above (`--configuration file://configuration.json`).

- To enable Auto Scaling on your fleet, you should set the mode to `EVENT_BASED_AUTO_SCALING`.
- The `workerCapabilities` are the default values assigned to the CMF when you created it. You can change these values if you need to increase resources available to your CMF.

After you configure the fleet mode, Deadline Cloud starts emitting fleet size recommendation events for that fleet.

```
{  
  "customerManaged": {  
    "mode": "EVENT_BASED_AUTO_SCALING",  
    "workerCapabilities": {  
      "vCpuCount": {  
        "min": 1,  
        "max": 4  
      },  
      "memoryMiB": {  
        "min": 1024,  
        "max": 4096  
      },  
      "osFamily": "linux",  
      "cpuArchitectureType": "x86_64",  
    }  
  }  
}
```

## Deploy Auto Scaling stack using the AWS CloudFormation template

You can set up an EventBridge rule to filter events, a Lambda to consume the events and control Auto Scaling, and an SQS queue to store unprocessed events. Use the following AWS

CloudFormation template to deploy everything in a stack. After you deploy the resources successfully, you can submit a job and the fleet will automatically scale up.

**Resources:****AutoScalingLambda:**

Type: 'AWS::Lambda::Function'

**Properties:****Code:**

ZipFile: |-

"""

This lambda is configured to handle "Fleet Size Recommendation Change" messages. It will handle all such events, and requires that the ASG is named based on the fleet id. It will scale up/down the fleet based on the recommended fleet size in the message.

**Example EventBridge message:**

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "Fleet Size Recommendation Change",
  "source": "aws.deadline",
  "account": "111122223333",
  "time": "2017-12-22T18:43:48Z",
  "region": "us-west-1",
  "resources": [],
  "detail": {
    "farmId": "farm-12345678900000000000000000000000",
    "fleetId": "fleet-12345678900000000000000000000000",
    "oldFleetSize": 1,
    "newFleetSize": 5,
  }
}
```

"""

```
import json
import boto3
import logging
```

```
logger = logging.getLogger()
logger.setLevel(logging.INFO)
```

```
auto_scaling_client = boto3.client("autoscaling")
```

```
def lambda_handler(event, context):
    logger.info(event)
    event_detail = event["detail"]
    fleet_id = event_detail["fleetId"]
    desired_capacity = event_detail["newFleetSize"]

    asg_name = f"deadline-ASG-autoscalable-{fleet_id}"
    auto_scaling_client.set_desired_capacity(
        AutoScalingGroupName=asg_name,
        DesiredCapacity=desired_capacity,
        HonorCooldown=False,
    )

    return {
        'statusCode': 200,
        'body': json.dumps(f'Successfully set desired_capacity for {asg_name}
to {desired_capacity}')
    }
Handler: index.lambda_handler
Role: !GetAtt
- AutoScalingLambdaServiceRole
- Arn
Runtime: python3.11
DependsOn:
- AutoScalingLambdaServiceRoleDefaultPolicy
- AutoScalingLambdaServiceRole
AutoScalingEventRule:
Type: 'AWS::Events::Rule'
Properties:
EventPattern:
source:
- aws.deadline
detail-type:
- Fleet Size Recommendation Change
State: ENABLED
Targets:
- Arn: !GetAtt
- AutoScalingLambda
- Arn
DeadLetterConfig:
Arn: !GetAtt
- UnprocessedAutoScalingEventQueue
- Arn
Id: Target0
```

```
    RetryPolicy:
      MaximumRetryAttempts: 15
  AutoScalingEventRuleTargetPermission:
    Type: 'AWS::Lambda::Permission'
    Properties:
      Action: 'lambda:InvokeFunction'
      FunctionName: !GetAtt
        - AutoScalingLambda
        - Arn
      Principal: events.amazonaws.com
      SourceArn: !GetAtt
        - AutoScalingEventRule
        - Arn
  AutoScalingLambdaServiceRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: 'sts:AssumeRole'
            Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
        Version: 2012-10-17
      ManagedPolicyArns:
        - !Join
          - ''
          - - 'arn:'
            - !Ref 'AWS::Partition'
            - ':iam::aws:policy/service-role/AWSLambdaBasicExecutionRole'
  AutoScalingLambdaServiceRoleDefaultPolicy:
    Type: 'AWS::IAM::Policy'
    Properties:
      PolicyDocument:
        Statement:
          - Action: 'autoscaling:SetDesiredCapacity'
            Effect: Allow
            Resource: '*'
        Version: 2012-10-17
      PolicyName: AutoScalingLambdaServiceRoleDefaultPolicy
    Roles:
      - !Ref AutoScalingLambdaServiceRole
  UnprocessedAutoScalingEventQueue:
    Type: 'AWS::SQS::Queue'
    Properties:
```

```
QueueName: deadline-unprocessed-autoscaling-events
UpdateReplacePolicy: Delete
DeletionPolicy: Delete
UnprocessedAutoScalingEventQueuePolicy:
  Type: 'AWS::SQS::QueuePolicy'
  Properties:
    PolicyDocument:
      Statement:
        - Action: 'sqs:SendMessage'
          Condition:
            ArnEquals:
              'aws:SourceArn': !GetAtt
                - AutoScalingEventRule
                - Arn
          Effect: Allow
          Principal:
            Service: events.amazonaws.com
          Resource: !GetAtt
            - UnprocessedAutoScalingEventQueue
            - Arn
      Version: 2012-10-17
    Queues:
      - !Ref UnprocessedAutoScalingEventQueue
```

## Connect customer-managed fleets to a license endpoint

The AWS Deadline Cloud (Deadline Cloud) usage-based license server provides on-demand licenses for select third-party products. This enables you to pay as you go. You are only charged for the time you use.

The Deadline Cloud usage-based license server can be used with any fleet type as long as the Deadline Cloud workers can communicate with the license server. This is automatically set up in service managed fleets. This setup is only needed for customer managed fleets.

To create the license server, you need the following:

- A security group for your fleet's VPC that allows traffic for third-party licenses.
- An AWS Identity and Access Management (IAM) role with an attached policy that allows access to the Deadline Cloud license endpoint operations.

### Topics

- [Step 1: Create a security group](#)
- [Step 2: Set up the license endpoint](#)
- [Step 3: Connect a rendering application to an endpoint](#)

## Step 1: Create a security group

Use the Amazon VPC Console (<https://console.aws.amazon.com/vpc/>) to create a security group for your farm's VPC. Configure the security group to allow the following inbound rules:

- Autodesk Maya and Arnold – 2701 - 2702, TCP, IPv4
- Foundry Nuke – 6101, TCP, IPv4
- SideFX Houdini, Mantra, and Karma – 1715 - 1717, TCP, IPv4

The source for each inbound rule is the fleet's worker security group.

For more information about creating a security group, see [Create a security group](#) in the *Amazon Virtual Private Cloud user guide*.

## Step 2: Set up the license endpoint

A *license endpoint* provides access to license servers for third-party products. License requests are sent to the license endpoint. The endpoint routes them to the appropriate license server. The license server tracks usage limits and entitlements.

You can create your license endpoint from the AWS Command Line Interface with the appropriate permissions. For the required policy to create a license endpoint, see [Policy to allow creating a license endpoint](#).

You can use the AWS CloudShell (<https://console.aws.amazon.com/cloudshell/>) or any other AWS CLI environment to configure the license endpoint using the following AWS Command Line Interface commands.

1. Create the license endpoint. Replace the security group ID, subnet ID, and VPC ID with the values you created earlier. If you use multiple subnets, separate them with spaces.

```
aws deadline create-license-endpoint \  
  --security-group-id SECURITY_GROUP_ID \  
  --subnet-id SUBNET_ID \  
  --vpc-id VPC_ID
```

```
--subnet-ids SUBNET_ID1 SUBNET_ID2 \  
--vpc-id VPC_ID
```

2. Confirm that the endpoint was created successfully with the following command. Remember the DNS name of the VPC endpoint.

```
aws deadline get-license-endpoint \  
--license-endpoint-id LICENSE_ENDPOINT_ID
```

3. View a list of available metered products:

```
aws deadline list-available-metered-products
```

4. Add metered products to the license endpoint with the following command.

```
aws deadline put-metered-product \  
--license-endpoint-id LICENSE_ENDPOINT_ID \  
--product-id PRODUCT_ID
```

You can remove a product from a license endpoint with the `remove-metered-product` command:

```
aws deadline remove-metered-product \  
--license-endpoint-id LICENSE_ENDPOINT_ID \  
--productId PRODUCT_ID
```

You can delete a license endpoint with the `delete-license-endpoint` command:

```
aws deadline delete-license-endpoint \  
--license-endpoint-id LICENSE_ENDPOINT_ID
```

### Step 3: Connect a rendering application to an endpoint

After the license endpoint is set up, applications use it the same as they use a third-party license server. You typically configure the license server for the application by setting an environment variable or other system setting, such as a Microsoft Windows registry key, to a license server port and address.

To get the license endpoint DNS name, use the following AWS CLI command.



```
aws deadline get-license-endpoint
```

Or you can use the Amazon VPC Console (<https://console.aws.amazon.com/vpc/>) to identify the VPC endpoint created by the Deadline Cloud API in the previous step.

## Configuration examples

### Example – Autodesk Maya and Arnold

Set the environment variable `ADSKFLEX_LICENSE_FILE` to:

```
2702@VPC_Endpoint_DNS_Name:2701@VPC_Endpoint_DNS_Name
```

### Example – Foundry Nuke

Set the environment variable `foundry_LICENSE` to `6101@VPC_Endpoint_DNS_Name` To test that licensing is working properly, you can run Nuke in a terminal:

```
~/nuke/Nuke14.0v5/Nuke14.0 -x
```

### Example – SideFX Houdini, Mantra, and Karma

Run the following command:

```
/opt/hfs19.5.640/bin/hserver -S  
"http://VPC_Endpoint_DNS_Name:1715;http://VPC_Endpoint_DNS_Name:1716;http://  
VPC_Endpoint_DNS_Name:1717;"
```

To test that licensing is working properly, you can render a Houdini scene via this command:

```
/opt/hfs19.5.640/bin/hython ~/forpentest.hip -c "hou.node('/out/mantra1').render()"
```

# 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 it is required to set up and use Deadline Cloud. For more information, see [Manage your identity source](#).

An organization owner for your AWS Organizations is responsible for managing the users and groups that have access to your Deadline Cloud monitor. You can create and manage these users and groups using IAM Identity Center or the Deadline Cloud console. For more information, see [What is AWS Organizations](#).

You create and remove users and groups that can use the monitor to manage farms, queues, and fleets using the Deadline Cloud console. When you add a user to Deadline Cloud, they must reset their password using IAM Identity Center before they get access.

## Topics

- [Manage users and groups for the monitor](#)
- [Manage users and groups for farms, queues, and fleets](#)

## Manage users and groups for the monitor

An Organizations owner can use the Deadline Cloud console to manage the users and groups that have access to the Deadline Cloud monitor. You can choose from existing IAM Identity Center users and groups, or you can add new users and groups from the console.

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.

## Monitor 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.

## Monitor 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. If desired, 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 from your monitor.
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 and groups for farms, queues, and fleets

1. If you haven't already, 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. To manage a queue or fleet, choose the **Queues** or **Fleets** tab, and then choose the queue or fleet to manage.
5. Choose the **Access management** tab. By default, the **Groups** tab is selected. To manage users, move the toggle to **Users**.

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**.

## Users

### To add users

1. To add a user, choose **Add user**.
2. From the dropdown, select the users to add to your farm.
3. For the user access level, choose one of the following options:
  - **Viewer**
  - **Contributor**
  - **Manager**
  - **Owner**
4. Choose **Add**. The users are added to your farm.

### To remove users

1. Select the user to remove.
2. In the **Remove** confirmation dialog, choose **Remove**. The user is then removed from the selected farm.

You can also add or remove farm permissions for users and groups by using the IAM Identity Center console at <https://console.aws.amazon.com/singlesignon/>.

# 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. You also provide a JSON or YAML file that provides the instructions for workers to process. Deadline Cloud accepts job templates that follow the Open Job Description (OpenJD) specification for describing jobs. For more information, see the [Open Job Description Documentation](#) on the GitHub website.

A job consists of:

- *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. 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 frame number, that are used in the script. The job is complete when all tasks are complete for all steps.
- *Environments* – Set up and tear down instructions shared by multiple steps or tasks.

You can create a job in any of the following ways:

- Use a Deadline Cloud submitter.
- Create a job bundle and use the [Deadline Cloud command line interface](#) (Deadline Cloud CLI).
- Use the AWS SDK.
- Use the AWS Command Line Interface (AWS CLI).

A *submitter* is a plugin for your digital content creation (DCC) software that manages creating a job in the interface to your DCC software. After you create the job, you use the submitter to send it to Deadline Cloud for processing. Behind the scenes, the submitter creates an OpenJD job 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 the time it takes to send files, only files that have changed since the last time you uploaded files are sent to Amazon S3.

To create your own scripts and pipelines to submit jobs to Deadline Cloud, you can use the Deadline Cloud CLI, the AWS SDK, or the AWS CLI to call operations to create, get, view, and list jobs. The following topics explain how to use the Deadline Cloud CLI.

The Deadline Cloud CLI is installed along with the Deadline Cloud submitter. For more information, see [Set up Deadline Cloud submitters](#).

## Topics

- [Submitting jobs with the Deadline Cloud CLI](#)
- [Scheduling jobs in Deadline Cloud](#)
- [Job states in the Deadline Cloud CLI](#)
- [Modifying jobs in Deadline Cloud](#)
- [How Deadline Cloud processes jobs](#)
- [Troubleshooting Deadline Cloud jobs](#)

## Submitting jobs with the Deadline Cloud CLI

To submit a job using the Deadline Cloud command line interface (Deadline Cloud CLI), use the `deadline bundle submit` command.

Jobs are submitted to queues. If you haven't set up a farm and queue yet, use the Deadline Cloud console (<https://console.aws.amazon.com/deadline-cloud/>) to set up a farm and queue and to see the farm and queue ID. For more information, see [Define farm details](#) and [Define queue details](#).

To set the default farm and queue for the Deadline Cloud CLI, use the following command. When you set the defaults, you can use Deadline Cloud CLI commands without specifying a farm or queue. In the following example, replace *farmId* and *queueId* with your own information:

```
deadline config set defaults.farm_id farmId
deadline config set defaults.queue_id queueId
```

To specify the steps and tasks in a job, create an OpenJD job template. For more information, see [Template Schemas \[Version: 2023-09\]](#) in the *Open Job Description specification* GitHub repository.

The following example is a YAML job template. It defines a job with two steps and five tasks per step.

```
name: Sample Job
specificationVersion: jobtemplate-2023-09
steps:
```

```
- name: Sample Step 1
  parameterSpace:
    taskParameterDefinitions:
      - name: var
        range: 1-5
        type: INT
  script:
    actions:
      onRun:
        args:
          - '1'
        command: /usr/bin/sleep
- name: Sample Step 2
  parameterSpace:
    taskParameterDefinitions:
      - name: var
        range: 1-5
        type: INT
  script:
    actions:
      onRun:
        args:
          - '1'
        command: /usr/bin/sleep
```

To create a job, create a new folder named `sample_job`, then save the template file in the new folder as `template.yaml`. You submit the job with the following Deadline Cloud CLI command:

```
deadline bundle submit path/to/sample_job
```

The response from the command contains an identifier for the job. Remember the ID so that you can check the job's status later.

```
Submitting to Queue: test-queue
Waiting for Job to be created...
Submitted job bundle:
  sample_job
Job creation completed successfully
jobId
```

There are additional options that you can use when submitting a job. For more information, see [More options for submitting jobs with the Deadline Cloud CLI](#).



## More options for submitting jobs with the Deadline Cloud CLI

The `deadline bundle submit` Deadline Cloud CLI command provides options that you can use to specify additional information for a job. The following examples show you how to:

- Specify parameters used when processing the job template.
- Attach files and folders in a shared environment to a job.
- Set the maximum number of task failures before a job is canceled.
- Set the maximum number of retries for a task.

### Job parameters

The `parameters` option sets the value of a job parameter when you create the job. The job template defines the field, and the `parameters` option sets the value. A parameter can have a default value. If a value is specified for the parameter, the specified value overrides the default value.

The following job template defines the `TestParameter` field:

```
name: Sample Job With Job Parameter
parameterDefinitions:
- default: test
  name: TestParameter
  type: STRING
specificationVersion: jobtemplate-2023-09
steps:
- description: step description
  name: MyStep
  parameterSpace:
    taskParameterDefinitions:
    - name: var
      range: 1-5
      type: INT
  script:
    actions:
    onRun:
      args:
      - '1'
      command: /usr/bin/sleep
```

The following command sets the value of the TestParameter to "Hello AWS":

```
deadline bundle submit sample_job --parameter "TestParameter=Hello AWS"
```

## Storage profiles

Storage profiles help with sharing files between workers with different operating systems. Create a storage profile using the Deadline Cloud console. Then, use the `storage-profile-id` parameter to use the storage profile. For more information, see [Shared storage in Deadline Cloud](#).

To set the storage profile for job submissions, using the Deadline Cloud CLI, use the following command to set the `storage-profile-id` configuration parameter:

```
deadline config set settings.storage_profile_id storageProfileId
```

## Maximum failed tasks

The `max-failed-tasks-count` option sets the maximum number of tasks that can fail before the entire job fails and all remaining tasks are marked CANCELED. The default value is 100.

```
deadline bundle submit sample_job --max-failed-tasks-count 10
```

## Maximum failed task retries

The `max-retries-per-task` option sets the maximum number of times that a task is retried before it fails. When a task is retried, it is put in the READY state. The default value is 5.

```
deadline bundle submit sample_job --max-retries-per-task 10
```

## Scheduling jobs in Deadline Cloud

After a job is created, AWS Deadline Cloud schedules it to be processed on one or more of the fleets associated with a queue. The fleet that processes a particular task is chosen based on the capabilities configured for the fleet and the host requirements of a specific step.

Jobs are scheduled in a best-effort priority order, highest to lowest. When two jobs have the same priority, the oldest job is scheduled first.

The following sections provide details of the process of scheduling a job.

## Determine fleet compatibility

After a job is created, Deadline Cloud checks the host requirements for each step in the job against the capabilities of the fleets associated with the queue the job was submitted to. If a fleet meets the host requirements, the job is put into the READY state.

If any step in the job has requirements that can't be met by a fleet associated with the queue, the step's status is set to NOT\_COMPATIBLE. In addition, the rest of the steps in the job are canceled.

Capabilities for a fleet are set at the fleet level. Even if a worker in a fleet meets the job's requirements, it won't be assigned tasks from the job if its fleet doesn't meet the job's requirements.

The following job template has a step that specifies host requirements for the step:

```
name: Sample Job With Host Requirements
specificationVersion: jobtemplate-2023-09
steps:
- name: Step 1
  script:
    actions:
      onRun:
        args:
          - '1'
        command: /usr/bin/sleep
  hostRequirements:
    amounts:
      # Capabilities starting with "amount." are amount capabilities. If they start with
      "amount.worker.",
      # they are defined by the OpenJD specification. Other names are free for custom
      usage.
      - name: amount.worker.vcpu
        min: 4
        max: 8
    attributes:
      - name: attr.worker.os.family
        anyOf:
          - linux
```

This job can be scheduled to a fleet with the following capabilities:

```
{
  "vCpuCount": {"min": 4, "max": 8},
  "memoryMiB": {"min": 1024},
  "osFamily": "linux",
  "cpuArchitectureType": "x86_64"
}
```

This job can't be scheduled to a fleet with any of the following capabilities:

```
{
  "vCpuCount": {"min": 4},
  "memoryMiB": {"min": 1024},
  "osFamily": "linux",
  "cpuArchitectureType": "x86_64"
}
```

The vCpuCount has no maximum, so it exceeds the maximum vCPU host requirement.

```
{
  "vCpuCount": {"max": 8},
  "memoryMiB": {"min": 1024},
  "osFamily": "linux",
  "cpuArchitectureType": "x86_64"
}
```

The vCpuCount has no minimum, so it doesn't satisfy the minimum vCPU host requirement.

```
{
  "vCpuCount": {"min": 4, "max": 8},
  "memoryMiB": {"min": 1024},
  "osFamily": "windows",
  "cpuArchitectureType": "x86_64"
}
```

The osFamily doesn't match.

## Fleet scaling

When a job is assigned to a compatible service managed fleet, the fleet is auto scaled. The number of workers in the fleet fluctuates based on the number of tasks available for the fleet to run.

When a job is assigned to a customer managed fleet, workers might already exist or can be created using event-based auto scaling. For more information, see [Use EventBridge to handle auto scaling events](#) in the *Amazon EC2 Auto Scaling User Guide*.

## Sessions

The tasks in a job are divided into one or more sessions. Workers run the sessions to set up the environment, run the tasks, and then tear down the environment. Each session is composed of one or more actions that a worker must take.

As a worker completes section actions, additional session actions can be sent to the worker. The worker reuses existing environments and job attachments in the session to complete tasks more efficiently.

Job attachments are created by the submitter that you use, as part of your Deadline Cloud CLI job bundle. You can also create job attachments by using the `--attachments` option for the `create-job` AWS CLI command. Environments are defined in two places: queue environments attached to a specific queue, and job step environments defined in the job template.

There are four session action types:

- `syncInputJobAttachments` – Downloads the input job attachments to the worker.
- `envEnter` – Performs the `onEnter` actions for an environment.
- `taskRun` – Performs the `onRun` actions for a task.
- `envExit` – Performs the `onExit` actions for an environment.

The following job template has a step environment. It has an `onEnter` definition to set up the step environment, an `onRun` definition that defines the task to run, and an `onExit` definition to tear down the step environment. The sessions created for this job will include an `envEnter` action, one or more `taskRun` actions, and then an `envExit` action.

```
name: Sample Job with Maya Environment
specificationVersion: jobtemplate-2023-09
steps:
- name: Maya Step
  stepEnvironments:
  - name: Maya
    description: Runs Maya in the background.
    script:
      embeddedFiles:
      - name: initData
        filename: init-data.yaml
        type: TEXT
```

```
    data: |
      scene_file: MyAwesomeSceneFile
      renderer: arnold
      camera: persp
  actions:
    onEnter:
      command: MayaAdaptor
      args:
        - daemon
        - start
        - --init-data
        - file://{{Env.File.initData}}
    onExit:
      command: MayaAdaptor
      args:
        - daemon
        - stop
  parameterSpace:
    taskParameterDefinitions:
      - name: Frame
        range: 1-5
        type: INT
  script:
    embeddedFiles:
      - name: runData
        filename: run-data.yaml
        type: TEXT
        data: |
          frame: {{Task.Param.Frame}}
  actions:
    onRun:
      command: MayaAdaptor
      args:
        - daemon
        - run
        - --run-data
        - file://{{ Task.File.runData }}
```

## Step dependencies

Deadline Cloud supports defining dependencies between steps so that one step waits until another step is complete before starting. You can define more than one dependency for a step. A step with a dependency isn't scheduled until all of its dependencies are complete.

If the job template defines a circular dependency, the job is rejected and the job status is set to `CREATE_FAILED`.

The following job template creates a job with two steps. StepB depends on StepA. StepB only runs after StepA completes successfully.

After the job is created, StepA is in the `READY` state and StepB is in the `PENDING` state. After StepA finishes, StepB moves to the `READY` state. If StepA fails, or if StepA is canceled, StepB moves to the `CANCELED` state.

You can set a dependency on multiple steps. For example, if StepC depends on both StepA and StepB, StepC won't start until the other two steps finish.

```
name: Step-Step Dependency Test
specificationVersion: 'jobtemplate-2023-09'
steps:
- name: A
  script:
    actions:
      onRun:
        command: bash
        args: ['{{ Task.File.run }}']
    embeddedFiles:
      - name: run
        type: TEXT
        data: |
          #!/bin/env bash

          set -euo pipefail

          sleep 1
          echo Task A Done!
- name: B
  dependencies:
    - dependsOn: A # This means Step B depends on Step A
  script:
    actions:
      onRun:
        command: bash
        args: ['{{ Task.File.run }}']
    embeddedFiles:
      - name: run
        type: TEXT
```

```
data: |
  #!/bin/env bash

  set -euo pipefail

  sleep 1
  echo Task B Done!
```

## Job states in the Deadline Cloud CLI

This topic describes how to use the AWS Deadline Cloud command line interface (Deadline Cloud CLI) to view the status of a job or step. If you want to use the Deadline Cloud monitor to view the status of jobs or steps, see [View and manage jobs, steps, and tasks in Deadline Cloud](#).

You can see the status of a job using the `deadline job get --job-id` Deadline Cloud CLI command. The response to the commands include the status of the job or step and the number of tasks in each processing status.

When you first submit a job, the status is `CREATE_IN_PROGRESS`. If the job passes the validation checks, its status changes to `CREATE_COMPLETE`. If not, the status changes to `CREATE_FAILED`.

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.

To see the quotas for the maximum number of steps and tasks in a job, use the Service Quotas console. For more information, see [Quotas for Deadline Cloud](#).

There may also be an internal service error that prevents a job from being created. If this happens, the job's status code is `INTERNAL_ERROR` and the status message field provides a more detailed explanation.

Use the following Deadline Cloud CLI command to view the details for a job. In the following example, replace *jobID* with your own information:

```
deadline job get --job-id jobId
```



The response from the `deadline job get` command is as follows:

```
jobId: jobId
name: Sample Job
lifecycleStatus: CREATE_COMPLETE
lifecycleStatusMessage: Job creation completed successfully
priority: 50
createdAt: 2024-03-26 18:11:19.065000+00:00
createdBy: Test User
startedAt: 2024-03-26 18:12:50.710000+00:00
taskRunStatus: STARTING
taskRunStatusCounts:
  PENDING: 0
  READY: 5
  RUNNING: 0
  ASSIGNED: 0
  STARTING: 0
  SCHEDULED: 0
  INTERRUPTING: 0
  SUSPENDED: 0
  CANCELED: 0
  FAILED: 0
  SUCCEEDED: 0
  NOT_COMPATIBLE: 0
maxFailedTasksCount: 100
maxRetriesPerTask: 5
```

Each task in a job or step has a status. The task statuses are combined to give an overall status for jobs and steps. The number of tasks in each state is reported in the `taskRunStatusCounts` field of the response.

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 is 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. Interruptions can happen when you manually update the job's status. It can also happen in response to an interruption due to Amazon Elastic Compute Cloud (Amazon EC2) Spot price changes.

## 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.

# Modifying jobs in Deadline Cloud

You can use the following AWS Command Line Interface (AWS CLI) update commands to modify the configuration of a job, or to set the target status of a job, step, or task:

- `aws deadline update-job`
- `aws deadline update-step`
- `aws deadline update-task`

In the following examples of the update commands, replace each *user input placeholder* with your own information.

You can also use the Deadline Cloud monitor to modify the configuration of a job. For more information, see [View and manage jobs, steps, and tasks in Deadline Cloud](#).

## Example – Requeue a job

All tasks in the job switch to the READY status, unless there are step dependencies. Steps with dependencies switch to either READY or PENDING as they are restored.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--target-task-run-status PENDING
```

## Example – Cancel a job

All tasks in the job that don't have the status SUCCEEDED or FAILED are marked CANCELED.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--target-task-run-status CANCELED
```

## Example – Mark a job failed

All tasks in the job that have the status SUCCEEDED are left unchanged. All other tasks are marked FAILED.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--target-task-run-status FAILED
```

### Example – Mark a job successful

All tasks in the job move to the SUCCEEDED state.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--target-task-run-status SUCCEEDED
```

### Example – Suspend a job

Tasks in the job in the SUCCEEDED, CANCELED, or FAILED state don't change. All other tasks are marked SUSPENDED.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--target-task-run-status SUSPENDED
```

### Example – Change the priority of a job

Updates the priority of a job to change the order that it is scheduled. Higher priority jobs are generally scheduled first.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--priority 100
```

## Example – Change the number of failed tasks allowed

Updates the maximum number of failed tasks that the job can have before the remaining tasks are canceled.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--max-failed-tasks-count 200
```

## Example – Change the number of task retries allowed

Updates the maximum number of retries for a task before the task fails. A task that has reached the maximum number of retries can't be requeued until this value is increased.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--max-retries-per-task 10
```

## Example – Archive a job

Updates the job's lifecycle status to ARCHIVED. Archived jobs can't be scheduled or modified. You can only archive a job that is in the FAILED, CANCELED, SUCCEEDED, or SUSPENDED state.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--lifecycle-status ARCHIVED
```

## Example – Requeue a step

All tasks in the step switch to the READY state, unless there are step dependencies. Tasks in steps with dependencies switch to either READY or PENDING, and the task is restored.

```
aws deadline update-step \  
--farm-id farmID \  
--queue-id queueID \  
--step-id stepID
```

```
--job-id jobID \  
--step-id stepID \  
--target-task-run-status PENDING
```

### Example – Cancel a step

All tasks in the step that don't have the status SUCCEEDED or FAILED are marked CANCELED.

```
aws deadline update-step \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--step-id stepID \  
--target-task-run-status CANCELED
```

### Example – Mark a step failed

All tasks in the step that have the status SUCCEEDED are left unchanged. All other tasks are marked FAILED.

```
aws deadline update-step \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--step-id stepID \  
--target-task-run-status FAILED
```

### Example – Mark a step successful

All tasks in the step are marked SUCCEEDED.

```
aws deadline update-step \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--step-id stepID \  
--target-task-run-status SUCCEEDED
```

### Example – Suspend a step

Tasks in the step in the SUCCEEDED, CANCELED, or FAILED state don't change. All other tasks are marked SUSPENDED.

```
aws deadline update-step \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--step-id stepID \  
--target-task-run-status SUSPENDED
```

## Example – Change the status of a task

When you use the `update-task` Deadline Cloud CLI command, the task switches to the specified status.

```
aws deadline update-task \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--step-id stepID \  
--task-id taskID \  
--target-task-run-status SUCCEEDED | SUSPENDED | CANCELED | FAILED | PENDING
```

## How Deadline Cloud processes jobs

To process a job, AWS Deadline Cloud uses the Open Job Description (OpenJD) job template to determine the resources needed. Deadline Cloud selects a suitable worker for a step from the fleets associated with your queue. The selected worker meets all of the capability attributes required for the step.

Next, Deadline Cloud sends instructions to the workers to set up a session for the step. The software required for the step must be available on the worker instance for the job to run. The service can open sessions on multiple workers if the scaling settings for the fleet have capacity.

You can set up the software in an Amazon Machine Image (AMI), or your worker can load the software at runtime from a repository or package manager. You can use a queue, job, or step environments to deploy the software that you prefer.

The Deadline Cloud service uses the OpenJD template to determine the steps required for the job, and the tasks required for each step. Some steps have dependencies on other steps, so Deadline Cloud determines the order to complete the steps. Then, Deadline Cloud sends the tasks for each step to workers to process. When a task is finished, the service sends another task in the same session, or the worker can start a new session.

You can track the progress of the job in the Deadline Cloud monitor, the Deadline Cloud command line interface (Deadline Cloud CLI) or the AWS CLI. For more information about using the monitor, see [Using the Deadline Cloud monitor](#). For more information about using the Deadline Cloud CLI, see [Job states in the Deadline Cloud CLI](#).

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.

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.

## Troubleshooting Deadline Cloud jobs

For information about common problems with jobs in AWS Deadline Cloud, see the following topics.

### Topics

- [Why did creating my job fail?](#)
- [Why is my job not compatible?](#)
- [Why is my job stuck in ready?](#)
- [Why did my job fail?](#)
- [Why is my step pending?](#)

## Why did creating my job fail?

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 [Quotas for Deadline Cloud](#).



## 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](#).

## 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 [Auto scale your Amazon EC2 fleet with Deadline Cloud scale recommendation feature](#).

## 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 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](#).

## 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](#).

# 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 two options for storage locations:

- 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 service managed fleet, you can set up a virtual file system (VFS) in your virtual private network (VPN). Then workers can load files only when needed.

- 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.

## Topics

- [Job attachments in Deadline Cloud](#)
- [Shared storage in Deadline Cloud](#)

## Job attachments in Deadline Cloud

*Job attachments* enable you to 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 automatically encrypted in your S3 bucket by default. This approach 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:Region:accountID:key/keyID"
]
}
```

5. Choose **Next**.
6. Review the changes to the policy, and then when you're satisfied, choose **Save changes**.

## 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.

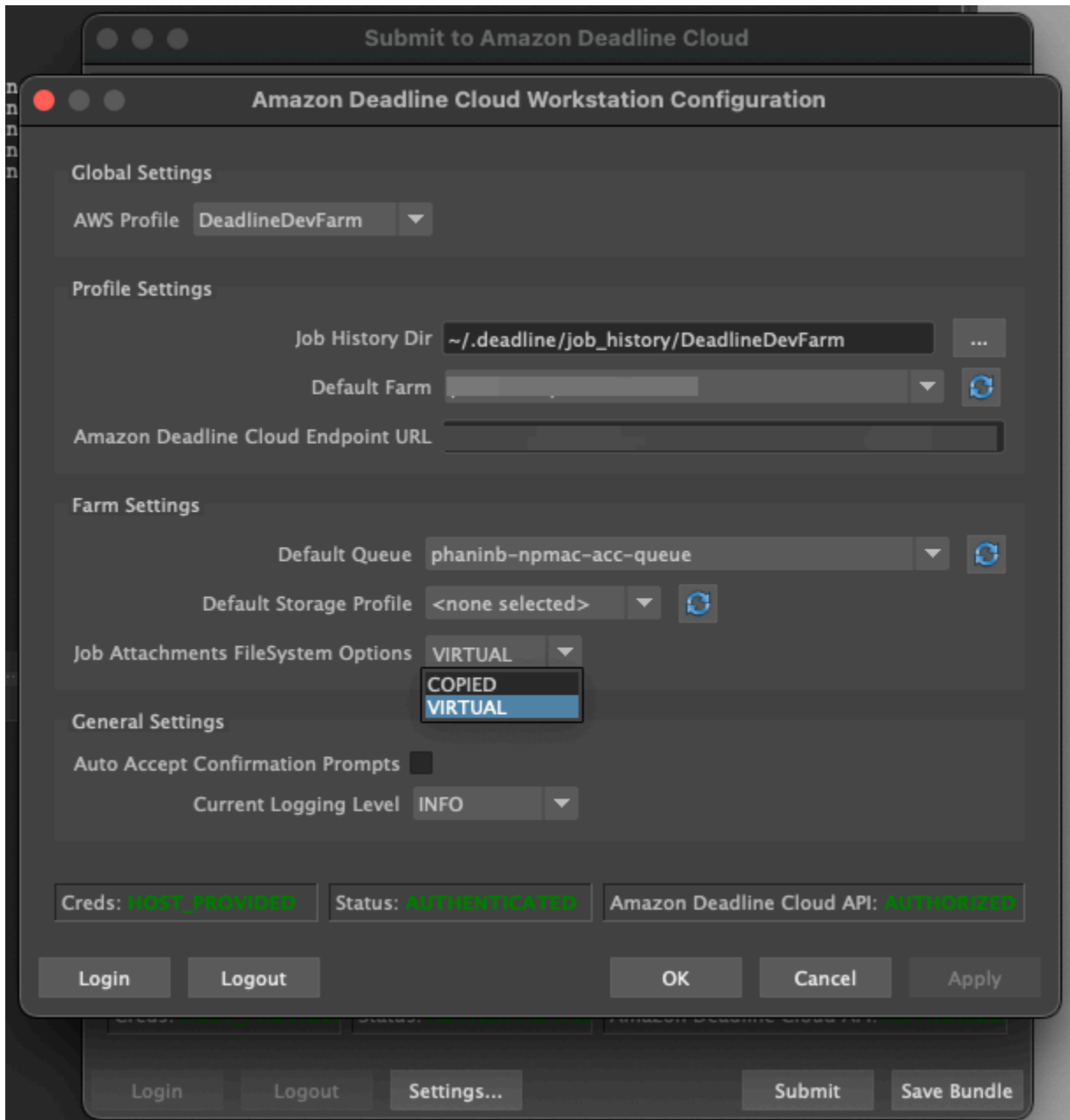
## Enable VFS support

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 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.

## Shared storage in Deadline Cloud

To use *shared storage*, workers use the operating system file sharing system for access to a shared storage space for the input and output of your jobs.

The actual method that you use to share files depends on your operating system and the way that you implement shared storage on your network. You're responsible for how you configure file sharing and ensuring that it meets your needs.

If you're using a cross-system file sharing solution, you can use storage profiles to map file locations between Linux and Windows file systems.

## Storage profiles in Deadline Cloud

A *storage profile* enables you to set up farms using cross-platform shared storage. A storage profile maps paths across operating systems for jobs processed on workers with a different operating system than the workstation they were submitted from.

Storage profiles are required when you use a customer managed fleet with a mixture of operating systems between workstations and workers. Storage profiles are not supported on service managed fleets.



After you create a storage profile, you must grant access to the queues and fleets that use the profile.

### To create a storage profile

1. Open the Deadline Cloud console at <https://console.aws.amazon.com/deadline-cloud/>.
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. Choose an **Operating system** from the dropdown.
6. Provide a **Name** for the profile. A clear name helps you choose the storage profile to use when submitting jobs.
7. For the **Path name**, enter the root location of job data on the workstation that you submit jobs from.
8. Choose a **Storage type**:
  - **Local** refers to file locations that are not shared between the worker and the workstation. They are uploaded as job attachments.
  - **Shared** refers to storage that is shared between the worker and the workstation. Files in shared storage are not uploaded as job attachments.
9. Provide a **File system location path**. This is the root directory for your job data.
10. Choose **Create**.

After you create a storage profile, you must modify your queues and customer managed fleets to use the new profile. To allow access to a storage profile, use the following procedure after you complete the previous procedure.

### To allow queues and customer managed fleets to use a storage profile

1. Choose either the **Queues** or **Fleets** tab.
2. Choose the queue or fleet to modify.
3. Choose **Modify storage profiles**.
4. Select the storage profile to allow, and the file system locations from that profile.
5. Choose **Save changes**.

# Managing budgets and usage for Deadline Cloud

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 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. 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-premise or other cloud providers. Actual costs of these resources are not calculated.

- *Idle worker costs.* For fleets with a minimum instance count greater than zero, idle workers are not accounted for in calculations.
- *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.
- *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 public ally 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 (bring your own license), Deadline Cloud cost management tools can't account for this cost.

## Using the Deadline Cloud budget manager

The Deadline Cloud budget manager helps you control spending on a given resource, such as a queue, fleet, or farm. You can create budget amounts and limits, and set automated actions to help reduce or stop additional spending against the budget.

The following sections provide you with the steps for using the Deadline Cloud budget manager.

### Topics

- [Prerequisite](#)
- [Access budget manager](#)
- [Create a budget](#)
- [View a budget](#)
- [Edit a budget](#)
- [Deactivate a budget](#)

## 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](#).

## Access budget manager

To access 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**. The Deadline Cloud monitor opens in a new tab.
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.

## Create a budget

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 clear, brief description for the budget.
5. From **Resource** choose the **Queue** dropdown to find and 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.
  - d. To create additional limit alerts, choose **Add new action** and repeat the previous two steps.
9. Choose **Create budget**. The budget manager page appears. The newly created budget displays in the **Active budgets** tab.

## View a 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

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 that you want to edit.
3. From the **Actions** dropdown menu in the upper right corner, select **Edit budget**.
4. Make the changes that you want, and then choose **Update budget**.

## Deactivate a budget

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 in the upper right corner, 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.

# Using 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, 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 [Manage users and groups for farms, queues, and fleets](#).

## 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, job, compute usage, instance type, or license product. If you choose license product, costs are calculated for specific licenses. For all other groups the time is calculated by adding up the time taken for each task to run.

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
- Select a different grouping, such as grouping by queue instead of job

## Topics

- [Use visual graphs to review data](#)
- [View a breakdown of metrics](#)
- [View approximate runtime of queues](#)

## 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.

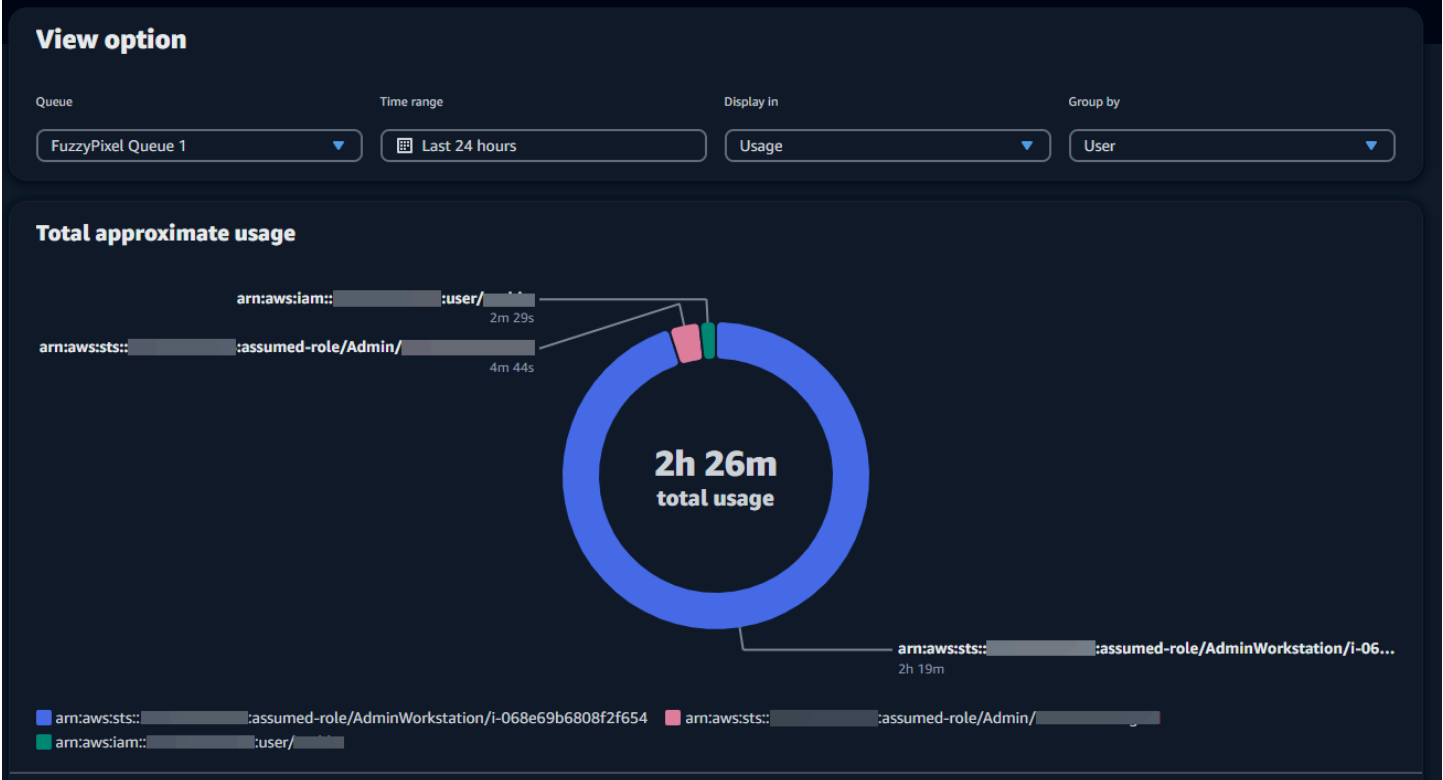
### Note

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.



## Cost Explorer

Visualize and understand costs incurred in FuzzyPixelFarm-M8-1025. The numbers displayed here are estimation and may be different from the AWS Cost Explorer.



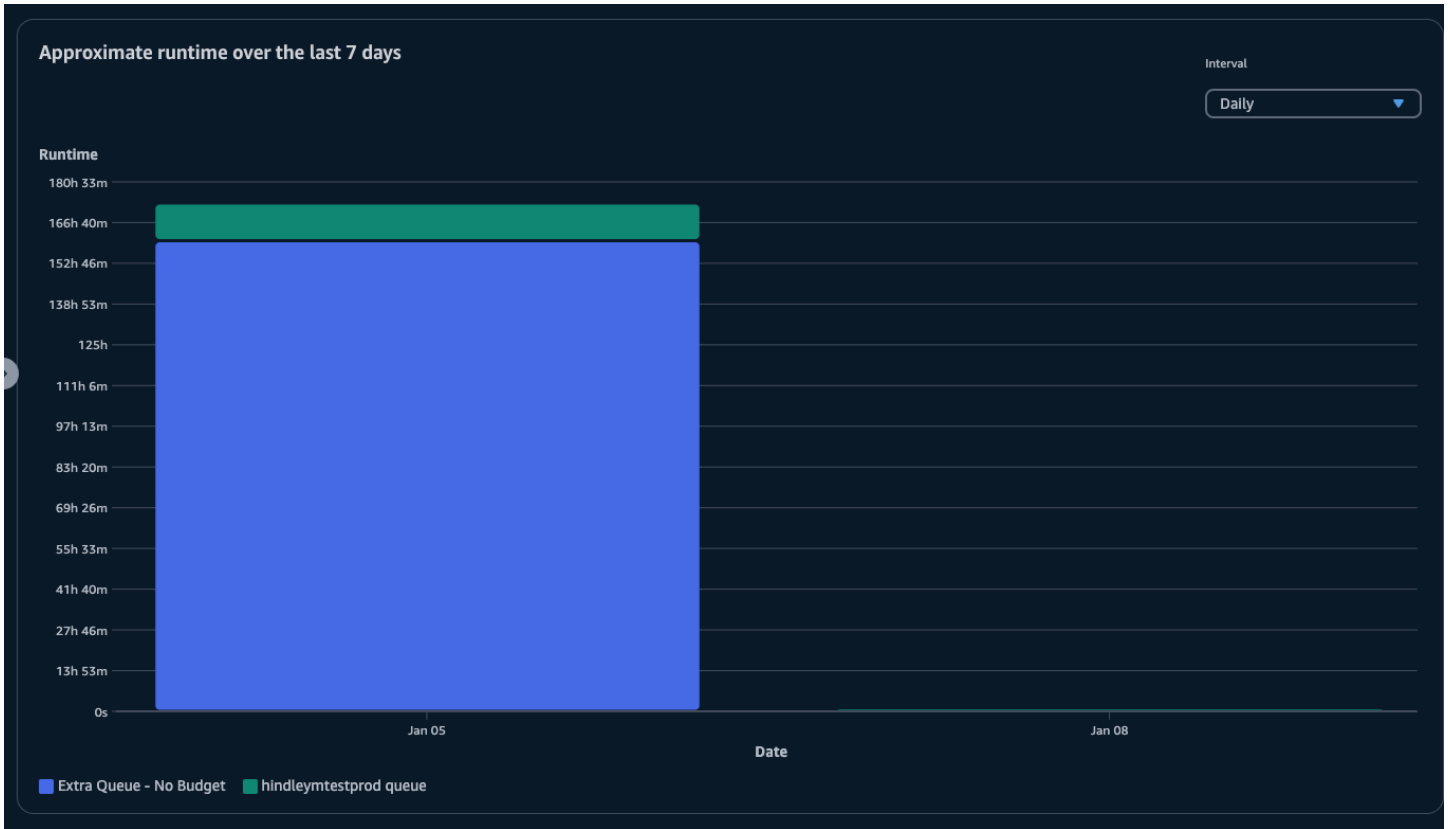
## View a breakdown of metrics

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

You can also view the approximate runtime of your queues 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.



# Security in Deadline Cloud

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

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](#).
- **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.

This documentation helps you understand how to apply the shared responsibility model when using Deadline Cloud. The following topics show you how to configure Deadline Cloud to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Deadline Cloud resources.

## Topics

- [Data protection in Deadline Cloud](#)
- [Identity and Access Management in Deadline Cloud](#)
- [Compliance validation for Deadline Cloud](#)
- [Resilience in Deadline Cloud](#)
- [Infrastructure security in Deadline Cloud](#)
- [Configuration and vulnerability analysis in Deadline Cloud](#)
- [Cross-service confused deputy prevention](#)
- [Security best practices for Deadline Cloud](#)

# 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 the [Data Privacy FAQ](#). For information about data protection in Europe, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the *AWS Security Blog*.

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.
- 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 140-2 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 \(FIPS\) 140-2](#).

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.

## Topics

- [Encryption at rest](#)
- [Encryption in transit](#)

- [Key management](#)
- [Inter-network traffic privacy](#)

## 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.

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 policies
  - 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 use 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, this 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 your 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 information, 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.

```
{
  "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 as 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.

```
{
  "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.

```
{
  "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.

### 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": "AROAIQDTESTANDEXAMPLE:SampleUser01",
    "arn": "arn:aws::sts::111122223333:assumed-role/Admin/SampleUser01",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAIQDTESTANDEXAMPLE",

```

```
    "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-abcdef12345678900987654321fedcba",
      "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-
EXAMPLE444444"
  }
],
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "111122223333",
"eventCategory": "Management"
}

```

## CloudTrail event for decryption

The following example CloudTrail event occurs when decrypting values using the customer managed KMS key.

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAIKDTESTANDEXAMPLE:SampleUser01",
    "arn": "arn:aws::sts::111122223333:assumed-role/SampleRole/SampleUser01",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAIKDTESTANDEXAMPLE",
        "arn": "arn:aws::iam::111122223333:role/SampleRole",
        "accountId": "111122223333",
        "userName": "SampleRole"
      },
      "webIdFederationData": {},
      "attributes": {
        "creationDate": "2024-04-23T18:46:51Z",
        "mfaAuthenticated": "false"
      }
    },
    "invokedBy": "deadline.amazonaws.com"
  }
}

```

```

    },
    "eventTime": "2024-04-23T18:51:44Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
      "encryptionContext": {
        "aws:deadline:farmId": "farm-abcdef12345678900987654321fedcba",
        "aws:deadline:accountId": "111122223333",
        "aws-crypto-public-key": "AotL+SAMPLEVALUEiOMEXAMPLEEaaqNOTREALaGTESTONLY
+p/5H+EuKd4Q=="
      },
      "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
      "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "responseElements": null,
    "requestID": "aaaaaaaa-bbbb-cccc-dddd-eeeeefffffff",
    "eventID": "ffffffff-eeee-dddd-cccc-bbbbbbaaaaaa",
    "readOnly": true,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
      }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
  }

```

## CloudTrail event for encryption

The following example CloudTrail event occurs when encrypting values using the customer managed KMS key.

```

{
  "eventVersion": "1.08",

```

```

"userIdentity": {
  "type": "AssumedRole",
  "principalId": "AROAIQDTESTANDEXAMPLE:SampleUser01",
  "arn": "arn:aws::sts::111122223333:assumed-role/SampleRole/SampleUser01",
  "accountId": "111122223333",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
    "sessionIssuer": {
      "type": "Role",
      "principalId": "AROAIQDTESTANDEXAMPLE",
      "arn": "arn:aws::iam::111122223333:role/SampleRole",
      "accountId": "111122223333",
      "userName": "SampleRole"
    },
    "webIdFederationData": {},
    "attributes": {
      "creationDate": "2024-04-23T18:46:51Z",
      "mfaAuthenticated": "false"
    }
  },
  "invokedBy": "deadline.amazonaws.com"
},
"eventTime": "2024-04-23T18:52:40Z",
"eventSource": "kms.amazonaws.com",
"eventName": "GenerateDataKey",
"awsRegion": "us-west-2",
"sourceIPAddress": "deadline.amazonaws.com",
"userAgent": "deadline.amazonaws.com",
"requestParameters": {
  "numberOfBytes": 32,
  "encryptionContext": {
    "aws:deadline:farmId": "farm-abcdef12345678900987654321fedcba",
    "aws:deadline:accountId": "111122223333",
    "aws-crypto-public-key": "AotL+SAMPLEVALUEiOMEXAMPLEEaaqNOTREALaGTESTONLY
+p/5H+EuKd4Q=="
  },
  "keyId": "arn:aws::kms:us-
west-2:111122223333:key/abcdef12-3456-7890-0987-654321fedcba"
},
"responseElements": null,
"requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
"eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
"readOnly": true,
"resources": [

```

```
{
  "accountId": "111122223333",
  "type": "AWS::KMS::Key",
  "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE33333"
},
"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. This means that the data becomes unrecoverable.

This is why 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.

## 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](#)
- [Authenticating with identities](#)
- [Managing access using policies](#)
- [How Deadline Cloud works with IAM](#)
- [Identity-based policy examples for Deadline Cloud](#)
- [AWS managed policies for Deadline Cloud](#)
- [Troubleshooting AWS Deadline Cloud identity and access](#)

## Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Deadline Cloud.

**Service user** – If you use the Deadline Cloud service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Deadline Cloud features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Deadline Cloud, see [Troubleshooting AWS Deadline Cloud identity and access](#).

**Service administrator** – If you're in charge of Deadline Cloud resources at your company, you probably have full access to Deadline Cloud. It's your job to determine which Deadline Cloud features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Deadline Cloud, see [How Deadline Cloud works with IAM](#).

**IAM administrator** – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Deadline Cloud. To view example Deadline Cloud identity-based policies that you can use in IAM, 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* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see [How to sign in to your AWS account](#) in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If



you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see [Signing AWS API requests](#) in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see [Multi-factor authentication](#) in the *AWS IAM Identity Center User Guide* and [Using multi-factor authentication \(MFA\) in AWS](#) in the *IAM User Guide*.

## AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see [Tasks that require root user credentials](#) in the *IAM User Guide*.

## Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see [What is IAM Identity Center?](#) in the *AWS IAM Identity Center User Guide*.

## IAM users and groups

An [IAM user](#) is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating

IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see [Rotate access keys regularly for use cases that require long-term credentials](#) in the *IAM User Guide*.

An [IAM group](#) is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see [When to create an IAM user \(instead of a role\)](#) in the *IAM User Guide*.

## IAM roles

An [IAM role](#) is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by [switching roles](#). You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see [Using IAM roles](#) in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see [Creating a role for a third-party Identity Provider](#) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permission sets, see [Permission sets](#) in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource

(instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see [How IAM roles differ from resource-based policies](#) in the *IAM User Guide*.

- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see [Forward access sessions](#).
  - **Service role** – 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 [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.
  - **Service-linked role** – 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.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see [Using an IAM role to grant permissions to applications running on Amazon EC2 instances](#) in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see [When to create an IAM role \(instead of a user\)](#) 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 is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see [Overview of JSON policies](#) in the *IAM User Guide*.

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**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

### Identity-based policies

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 [Creating IAM policies](#) in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see [Choosing 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 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.

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.

## Access control lists (ACLs)

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.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see [Access control list \(ACL\) overview](#) in the *Amazon Simple Storage Service Developer Guide*.

## Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see [Permissions boundaries for IAM entities](#) in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see [How SCPs work](#) in the *AWS Organizations User Guide*.

- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. 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
<a href="#">Identity-based policies</a>	Yes
<a href="#">Resource-based policies</a>	No
<a href="#">Policy actions</a>	Yes
<a href="#">Policy resources</a>	Yes
<a href="#">Policy condition keys (service-specific)</a>	Yes
<a href="#">ACLs</a>	No
<a href="#">ABAC (tags in policies)</a>	Yes
<a href="#">Temporary credentials</a>	Yes
<a href="#">Forward access sessions (FAS)</a>	Yes
<a href="#">Service roles</a>	Yes

IAM feature	Deadline Cloud support
<a href="#">Service-linked roles</a>	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 [Creating IAM 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. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. 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. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see [How IAM roles differ from resource-based policies](#) 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. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

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. Statements must include either a Resource or a NotResource element. As a best practice, specify a resource using its [Amazon Resource Name \(ARN\)](#). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, 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 (or *Condition block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. 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.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see [IAM policy elements: variables and tags](#) in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. 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. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

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 [What is ABAC?](#) 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
--------------------------------	-----

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see [AWS services that work with IAM](#) in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see [Switching to a role \(console\)](#) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate

temporary credentials instead of using long-term access keys. For more information, see [Temporary security credentials in IAM](#).

## Forward access sessions for Deadline Cloud

Supports forward access sessions (FAS)	Yes
--	-----

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. 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 [Creating 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. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see [Creating IAM policies](#) 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](#)
- [Using the Deadline Cloud console](#)
- [Policy to submit jobs to a queue](#)
- [Policy to allow creating a license endpoint](#)
- [Policy to allow monitoring a specific farm queue](#)

### 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 AWS 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 [IAM Access Analyzer policy validation](#) 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 [Configuring MFA-protected API access](#) 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 ensure that users and roles can still use the Deadline Cloud console, also attach the Deadline Cloud *ConsoleAccess* or *ReadOnly* AWS managed policy to the entities. For more information, see [Adding permissions to a user](#) in the *IAM User Guide*.

## 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.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SubmitJobsFarmAndQueue",
      "Effect": "Allow",
      "Action": "deadline:CreateJob",
      "Resource": "arn:aws:deadline:REGION:ACCOUNT_ID: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.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "SID": "CreateLicenseEndpoint",
    "Effect": "Allow",
    "Action": [
      "deadline:CreateLicenseEndpoint",
      "deadline>DeleteLicenseEndpoint",
      "deadline:GetLicenseEndpoint",

```

```

        "deadline:UpdateLicenseEndpoint",
        "deadline:ListLicenseEndpoints",
        "deadline:PutMeteredProduct",
        "deadline>DeleteMeteredProduct",
        "deadline:ListMeteredProducts",
        "deadline:ListAvailableMeteredProducts",
        "ec2:CreateVpcEndpoint",
        "ec2:DescribeVpcEndpoints",
        "ec2>DeleteVpcEndpoints"
    ],
    "Resource": "*"
}]
}

```

## 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.

```

{
  "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:REGION:123456789012:farm/FARM_A/queue/QUEUE_B",

```



```
    "arn:aws:deadline:REGION:123456789012:farm/FARM_A/queue/QUEUE_B/*"  
  ]  
}]  
}
```

## 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 principals to create 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.

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
Deadline Cloud started tracking changes	Deadline Cloud started tracking changes to its AWS managed policies.	April 2, 2024

## 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](#)
- [I am not authorized to perform iam:PassRole](#)

- [I want to allow people outside of my AWS account to access my Deadline Cloud resources](#)

## 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 [How IAM roles differ from resource-based policies](#) in the *IAM User Guide*.

## Compliance validation for Deadline Cloud

To learn whether an AWS service is within the scope of specific compliance programs, see [AWS services in Scope by Compliance Program](#) and choose the compliance program that you are interested in. For general information, see [AWS Compliance Programs](#).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- [Security and Compliance Quick Start Guides](#) – These deployment guides discuss architectural considerations and provide steps for deploying baseline environments on AWS that are security and compliance focused.
- [Architecting for HIPAA Security and Compliance on Amazon Web Services](#) – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.

**Note**

Not all AWS services are HIPAA eligible. For more information, see the [HIPAA Eligible Services Reference](#).

- [AWS Compliance Resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [AWS Customer Compliance Guides](#) – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- [Evaluating Resources with Rules](#) in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see [Security Hub controls reference](#).
- [AWS Audit Manager](#) – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

## 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.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the [AWS Security Token Service](#) (AWS STS) to generate temporary security credentials to sign requests.

Deadline Cloud doesn't support using AWS PrivateLink virtual private cloud (VPC) endpoint policies. It uses the AWS PrivateLink default policy, which grants full access to the endpoint. For more information, see [Default endpoint policy](#) in the *AWS PrivateLink user guide*.

## 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](#)

## 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.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "deadline.amazonaws.com"
    },
    "Action": "deadline:ActionName",
    "Resource": [
      "*"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:deadline:*:123456789012:*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

## Security best practices for Deadline Cloud

AWS Deadline Cloud (Deadline Cloud) provides a number of security features to consider as you develop and implement your own security policies. The following best practices are general guidelines and don't represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful considerations rather than prescriptions.

### Note

For more information about the importance of many security topics, see the [Shared Responsibility Model](#).

## Data protection

For data protection purposes, we recommend that you protect AWS account credentials and set up individual accounts with 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.
- 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 personal data that is stored in Amazon Simple Storage Service (Amazon S3).
- If you require FIPS 140-2 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 \(FIPS\) 140-2](#).

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with AWS Deadline Cloud or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into Deadline Cloud or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

## AWS Identity and Access Management permissions

Manage access to AWS resources using users, AWS Identity and Access Management (IAM) roles, and by granting the least privilege to users. Establish credential management policies and procedures for creating, distributing, rotating, and revoking AWS access credentials. For more information, see [IAM Best Practices](#) in the *IAM User Guide*.

## Run jobs as users and groups

When using queue functionality in Deadline Cloud, it's a best practice to specify an operating system (OS) user and its primary group so that the OS user has least-privilege permissions for the queue's jobs.

When you specify a “Run as user” (and group), any processes for jobs submitted to the queue will be run using that OS user and will inherit that user’s associated OS permissions.

The fleet and queue configurations combine to establish a security posture. On the queue side, the “Job run as user” and IAM role can be specified to use the OS and AWS permissions for the queue’s jobs. The fleet defines the infrastructure (worker hosts, networks, mounted shared storage) that, when associated to a particular queue, run jobs within the queue. The data available on the worker hosts needs to be accessed by jobs from one or more associated queues. Specifying a user or group helps protect the data in jobs from other queues, other installed software, or other users with access to the worker hosts. When a queue is without a user, it runs as the agent user which can impersonate (sudo) any queue user. In this way, a queue without a user can escalate privileges to another queue.

## Networking

To prevent traffic from being intercepted or redirected, it's essential to secure how and where your network traffic is routed.

We recommend that you secure your networking environment in the following ways:

- Secure Amazon Virtual Private Cloud (Amazon VPC) subnet route tables to control how IP layer traffic is routed.
- If you are using Amazon Route 53 (Route 53) as a DNS provider in your farm or workstation setup, secure access to the Route 53 API.
- If you connect to Deadline Cloud outside of AWS such as by using on-premises workstations or other data centers, secure any on-premises networking infrastructure. This includes DNS servers and route tables on routers, switches, and other networking devices.

## Jobs and job data

Deadline Cloud jobs run within sessions on worker hosts. Each session runs one or more processes on the worker host, which generally require that you input data to produce output.

To secure this data, you can configure operating system users with queues. The worker agent uses the queue OS user to run session sub-processes. These sub-processes inherit the queue OS user's permissions.

We recommend that you follow best practices to secure access to the data these sub-processes access. For more information, see [Shared responsibility model](#).

## Farm structure

You can arrange Deadline Cloud fleets and queues many ways. However, there are security implications with certain arrangements.

A farm has one of the most secure boundaries because it can't share Deadline Cloud resources with other farms, including fleets, queues, and storage profiles. However, you can share external AWS resources within a farm, which compromises the security boundary.

You can also establish security boundaries between queues within the same farm using the appropriate configuration.

Follow these best practices to create secure queues in the same farm:

- Associate a fleet only with queues within the same security boundary. Note the following:
  - After job runs on the worker host, data may remain behind, such as in a temporary directory or the queue user's home directory.
  - The same OS user runs all the jobs on a service-owned fleet worker host, regardless of which queue you submit the job to.
  - A job might leave processes running on a worker host, making it possible for jobs from other queues to observe other running processes.
- Ensure that only queues within the same security boundary share an Amazon S3 bucket for job attachments.
- Ensure that only queues within the same security boundary share an OS user.
- Secure any other AWS resources that are integrated into the farm to the boundary.

## Job attachment queues

Job attachments are associated with a queue, which uses your Amazon S3 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`.

If you create a queue and reuse an Amazon S3 bucket and root prefix, there is a risk of information being disclosed to unauthorized parties. For example, `QueueA` and `QueueB` share the same bucket and root prefix. In a secure workflow, `ArtistA` has access to `QueueA` but not `QueueB`. However, when multiple queues share a bucket, `ArtistA` can access the data in `QueueB` data because it uses the same bucket and root prefix as `QueueA`.

The console sets up queues that are secure by default. Ensure that the queues have a distinct combination of Amazon S3 bucket and root prefix unless they're part of a common security boundary.

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.

```
{
  "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": "ACCOUNT_ID" }
      }
    },
    {
      "Action": ["logs:GetLogEvents"],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:REGION:ACCOUNT_ID: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.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["sts:AssumeRole"],
      "Effect": "Allow",
      "Principal": { "Service": "deadline.amazonaws.com" },
      "Condition": {
        "StringEquals": { "aws:SourceAccount": "ACCOUNT_ID" },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:deadline:REGION:ACCOUNT_ID:farm/FARM_ID"
        }
      }
    },
    {
      "Action": ["sts:AssumeRole"],
      "Effect": "Allow",
      "Principal": { "Service": "credentials.deadline.amazonaws.com" },
      "Condition": {
        "StringEquals": { "aws:SourceAccount": "ACCOUNT_ID" },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:deadline:REGION:ACCOUNT_ID: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.

```

"Statement": [

```

```
{
  "Action": [
    "s3:GetObject",
    "s3:ListBucket"
  ],
  "Effect": "Allow",
  "Resource": [
    "arn:aws:s3:::SOFTWARE_BUCKET_NAME",
    "arn:aws:s3:::SOFTWARE_BUCKET_NAME/*"
  ]
}
```

For more information about Amazon S3 security best practices, see [Security best practices for Amazon S3](#) in the *Amazon Simple Storage Service User Guide*.

## Worker hosts

Secure worker hosts to help ensure that each user can only perform operations for their assigned role.

We recommend the following best practices to secure worker hosts:

- 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.
- Grant queue users least-privileged OS permissions required for the intended queue workloads. Ensure that they don't have filesystem write permissions to work agent program files or other shared software.
- Ensure only the root user on Linux and the Administrator owns 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.
- Grant trusted individuals least-privileged access to worker hosts.
- 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 workstations and worker host operating systems).



- Restrict permissions to DNS configuration on workstations and worker host operating systems.
- 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`.
- Regularly rotate the passwords for your queue `jobRunAsUser`.
- Ensure least privilege access to the Windows password secretes and delete unused secrets.
- Don't give the queue `jobRunAsUser` permission the 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.

### Note

For more information about the importance of regularly patching the operating system and installed software, see the [Shared Responsibility Model](#).

## Workstations

It's important to secure workstations with access to Deadline Cloud. This approach helps ensure that any jobs you submit to Deadline Cloud can't run arbitrary workloads billed to your AWS account.

We recommend the following best practice to secure artist workstations. For more information, see the [Shared Responsibility Model](#).

- Secure any persisted credentials that provide access to AWS, including Deadline Cloud. For more information, see [Managing access keys for IAM users](#) in the *IAM User Guide*.
- Only install trusted, secure software.
- Require users federate with an identity provider to access AWS with temporary credentials.
- Use secure permissions on Deadline Cloud submitter program files to prevent tampering.
- Grant trusted individuals least-privileged access to artist workstations.
- Only use submitters and adaptors that you obtain through the Deadline Cloud Monitor.
- Restrict permissions to `/etc/hosts` and route tables on workstations and worker host operating systems.

- Restrict permissions to `/etc/resolv.conf` on workstations and worker host operating systems.
- 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.

# 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 has three CloudWatch metrics.

- *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](#).

## Topics

- [Logging calls with CloudTrail](#)
- [Monitoring with CloudWatch](#)

## Logging calls with CloudTrail

AWS Deadline Cloud is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Deadline Cloud. CloudTrail captures all API calls for Deadline Cloud as events. The calls captured include calls from the Deadline Cloud console and code calls to the Deadline Cloud API operations.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Deadline Cloud. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Deadline Cloud, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the [AWS CloudTrail User Guide](#).

## Deadline Cloud information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Deadline Cloud, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see [Viewing Events with CloudTrail Event History](#).

CloudTrail also records events when users sign in to the Deadline Cloud monitor and receive AWS credentials. When a user signs in, there is a CloudTrail event with the source `signin.amazonaws.com` and the name `UserAuthentication`. There is a second event when

the signed-in user is given AWS credentials from the source `sts.amazonaws.com` and the name `AssumeRole`. The user's ID is recorded in second event inside the role session name.

For an ongoing record of events in your AWS account, including events for Deadline Cloud, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs.

For more information, see the following:

[Overview for creating a trail](#)

[CloudTrail supported services and integrations](#)

[Configuring Amazon SNS notifications for CloudTrail](#)

[Receiving CloudTrail log files from multiple Regions](#)

[Receiving CloudTrail log files from multiple accounts](#)

Deadline Cloud supports logging the following actions as events in CloudTrail log files:

- [associate-member-to-farm](#)
- [associate-member-to-fleet](#)
- [associate-member-to-job](#)
- [associate-member-to-queue](#)
- [assume-fleet-role-for-read](#)
- [assume-fleet-role-for-worker](#)
- [assume-queue-role-for-read](#)
- [assume-queue-role-for-user](#)
- [assume-queue-role-for-worker](#)
- [create-budget](#)
- [create-farm](#)
- [create-fleet](#)
- [create-license-endpoint](#)

- [create-monitor](#)
- [create-queue](#)
- [create-queue-environment](#)
- [create-queue-fleet-association](#)
- [create-storage-profile](#)
- [create-worker](#)
- [delete-budget](#)
- [delete-farm](#)
- [delete-fleet](#)
- [delete-license-endpoint](#)
- [delete-metered-product](#)
- [delete-monitor](#)
- [delete-queue](#)
- [delete-queue-environment](#)
- [delete-queue-fleet-association](#)
- [delete-storage-profile](#)
- [delete-worker](#)
- [disassociate-member-from-farm](#)
- [disassociate-member-from-fleet](#)
- [disassociate-member-from-job](#)
- [disassociate-member-from-queue](#)
- [get-application-version](#)
- [get-budget](#)
- [get-farm](#)
- [get-feature-map](#)
- [get-fleet](#)
- [get-license-endpoint](#)
- [get-monitor](#)
- [get-queue](#)

- [get-queue-environment](#)
- [get-queue-fleet-association](#)
- [get-sessions-statistics-aggregation](#)
- [get-storage-profile](#)
- [get-storage-profile-for-queue](#)
- [list-available-metered-products](#)
- [list-budgets](#)
- [list-farm-members](#)
- [list-farms](#)
- [list-fleet-members](#)
- [list-fleets](#)
- [list-job-members](#)
- [list-license-endpoints](#)
- [list-metered-products](#)
- [list-monitors](#)
- [list-queue-environments](#)
- [list-queue-fleet-associations](#)
- [list-queue-members](#)
- [list-queues](#)
- [list-storage-profiles](#)
- [list-storage-profiles-for-queue](#)
- [list-tags-for-resource](#)
- [put-metered-product](#)
- [start-sessions-statistics-aggregation](#)
- [tag-resource](#)
- [untag-resource](#)
- [update-budget](#)
- [update-farm](#)
- [update-fleet](#)
- [update-monitor](#)

- [update-queue](#)
- [update-queue-environment](#)
- [update-queue-fleet-association](#)
- [update-storage-profile](#)
- [update-worker](#)

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another service.

For more information, see the [CloudTrail user Identity element](#).

## Understanding Deadline Cloud log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

**This JSON example shows the log generated by a call to the CreateFarm API:**

```
{
  "eventVersion": "0",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE-PrincipalID:EXAMPLE-Session",
    "arn": "arn:aws:sts::111122223333:assumed-role/EXAMPLE-UserName/EXAMPLE-Session",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE-accessKeyId",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
```



```
        "principalId": "EXAMPLE-PrincipalID",
        "arn": "arn:aws:iam::111122223333:role/EXAMPLE-UserName",
        "accountId": "111122223333",
        "userName": "EXAMPLE-UserName"
    },
    "webIdFederationData": {},
    "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2021-03-08T23:25:49Z"
    }
}
},
"eventTime": "2021-03-08T23:25:49Z",
"eventSource": "deadline.amazonaws.com",
"eventName": "CreateFarm",
"awsRegion": "us-west-2",
"sourceIPAddress": "192.0.2.0",
"userAgent": "EXAMPLE-userAgent",
"requestParameters": {
    "displayName": "example-farm",
    "kmsKeyArn": "arn:aws:kms:us-west-2:111122223333:key/111122223333",
    "X-Amz-Client-Token": "12abc12a-1234-1abc-123a-1a11bc1111a",
    "description": "example-description",
    "tags": {
        "purpose_1": "e2e"
        "purpose_2": "tag_test"
    }
},
"responseElements": {
    "farmId": "EXAMPLE-farmID"
},
"requestID": "EXAMPLE-requestID",
"eventID": "EXAMPLE-eventID",
"readOnly": false,
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "111122223333"
"eventCategory": "Management",
}
```

The example shows the AWS Region, IP address, and other "requestParameters" such as the "displayName" and "kmsKeyArn" that can help you identify the event.

# Monitoring with CloudWatch

Amazon CloudWatch (CloudWatch) collects raw data and processes it into readable, near real-time metrics. You can open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/> to view and filter Deadline Cloud metrics.

- In a Deadline Cloud customer-managed fleet, CloudWatch sends you two metrics `UnhealthyWorkerCount` and `RecommendedFleetSize`:
- The namespace for these metrics is `AWS/DeadlineCloud`.
- You can use the dimensions `farmID` and `fleetID` to filter metrics.
- Both metrics use the unit count.

These statistics are kept for 15 months so you can access historical information to gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information, see the [Amazon CloudWatch User Guide](#).

Deadline Cloud has two kinds of logs – task logs and worker logs. A task log is when you run execution logs as a script or as DCC runs. A task log might show events such as assets loading, tiles rendering, or textures not being found.

A worker log shows worker agent processes. These might include things such as when the worker agents starts up, registers itself, reports progress, loads configurations, or completes tasks.

For Deadline Cloud, workers upload these logs to CloudWatch Logs. By default, logs never expire. If a job outputs a high volume of data, you can incur extra costs. For more information, see [Amazon CloudWatch pricing](#).

You can adjust the retention policy for each log group. A shorter retention removes old logs and can help reduce storage costs. To keep logs, you can archive them to Amazon Simple Storage Service before removing the log. For more information, see [Export log data to Amazon S3 using the console](#) in the *Amazon CloudWatch user guide*.

## Note

CloudWatch log reads are limited by AWS. If you plan to onboard many artists, we suggest you contact AWS customer support and request an increase for the `GetLogEvents` quota

in CloudWatch. Additionally, we recommend you close the log tailing portal when you are not debugging.

For more information, see [CloudWatch Logs quotas](#) in the *Amazon CloudWatch user guide*.

# Quotas 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.

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](#).

# 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 AWS CloudFormation provisions and configures those resources for you.

When you use AWS 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 AWS CloudFormation templates

To provision and configure resources for Deadline Cloud and related services, you must understand [AWS CloudFormation templates](#). Templates are formatted text files in JSON or YAML. These templates describe the resources that you want to provision in your AWS CloudFormation stacks. If you're unfamiliar with JSON or YAML, you can use AWS CloudFormation Designer to help you get started with AWS CloudFormation templates. For more information, see [What is AWS CloudFormation Designer?](#) in the *AWS CloudFormation User Guide*.

Deadline Cloud supports creating farms, queues, and fleets in AWS 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 AWS CloudFormation

To learn more about AWS CloudFormation, see the following resources:

- [AWS CloudFormation](#)
- [AWS CloudFormation User Guide](#)
- [AWS CloudFormation API Reference](#)
- [AWS CloudFormation Command Line Interface User Guide](#)

# Document history for the Deadline Cloud user guide

The following table describes important changes in each release of the *AWS Deadline Cloud user guide*.

Change	Description	Date
<a href="#">Initial release</a>	This is the initial release of the Deadline Cloud user guide.	April 2, 2024

# AWS Glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS Glossary Reference*.