User Guide

AWS PCS

AWS PCS: User Guide

Table of Contents

What is AWS Parallel Computing Service?

AWS Parallel Computing Service (AWS PCS) is a managed service that makes it easier to run and scale high performance computing (HPC) workloads, and build scientific and engineering models on AWS using Slurm. Use AWS PCS to build compute clusters that integrate best in class AWS compute, storage, networking, and visualization. Run simulations or build scientific and engineering models. Streamline and simplify your cluster operations using built-in management and observability capabilities. Empower your users to focus on research and innovation by enabling them to run their applications and jobs in a familiar environment.

Topics

- Concepts in AWS PCS

Concepts in AWS PCS

A cluster in AWS PCS has 1 or more queues, associated with at least 1 compute node group. Jobs are submitted to queues and run on EC2 instances defined by compute node groups. You can use these foundations to implement sophisticated HPC architectures.

Cluster

A cluster is a resource for managing resources and running workloads. A cluster is an AWS PCS resource that defines an assembly of compute, networking, storage, identity, and job scheduler configuration. You create a cluster by specifying which job scheduler you want to use (Slurm currently), what scheduler configuration you want, what service controller you want to manage the cluster, and in which VPC you want the cluster resources to be launched. The scheduler accepts and schedules jobs, and also launches the compute nodes (EC2 instances) that process those jobs.

Compute node group

A compute node group is a collection of compute nodes that AWS PCS uses to run jobs or provide interactive access to a cluster. When you define a compute node group, you specify common traits such as Amazon EC2 instance types, minimum and maximum instance count, target VPC subnets, Amazon Machine Image (AMI), purchase option, and custom launch configuration. AWS PCS uses these settings to efficiently launch, manage, and terminate compute nodes in a compute node group.

Queue

When you want to run a job on a specific cluster, you submit it to a particular queue (also sometimes called a *partition*). The job remains in the queue until AWS PCS schedules it to run on a compute node group. You associate one or more compute node groups with each queue. A queue is required to schedule and execute jobs on the underlying compute node group resources using various scheduling policies offered by the job scheduler. Users don’t submit jobs directly to a compute node or compute node group.

System administrator

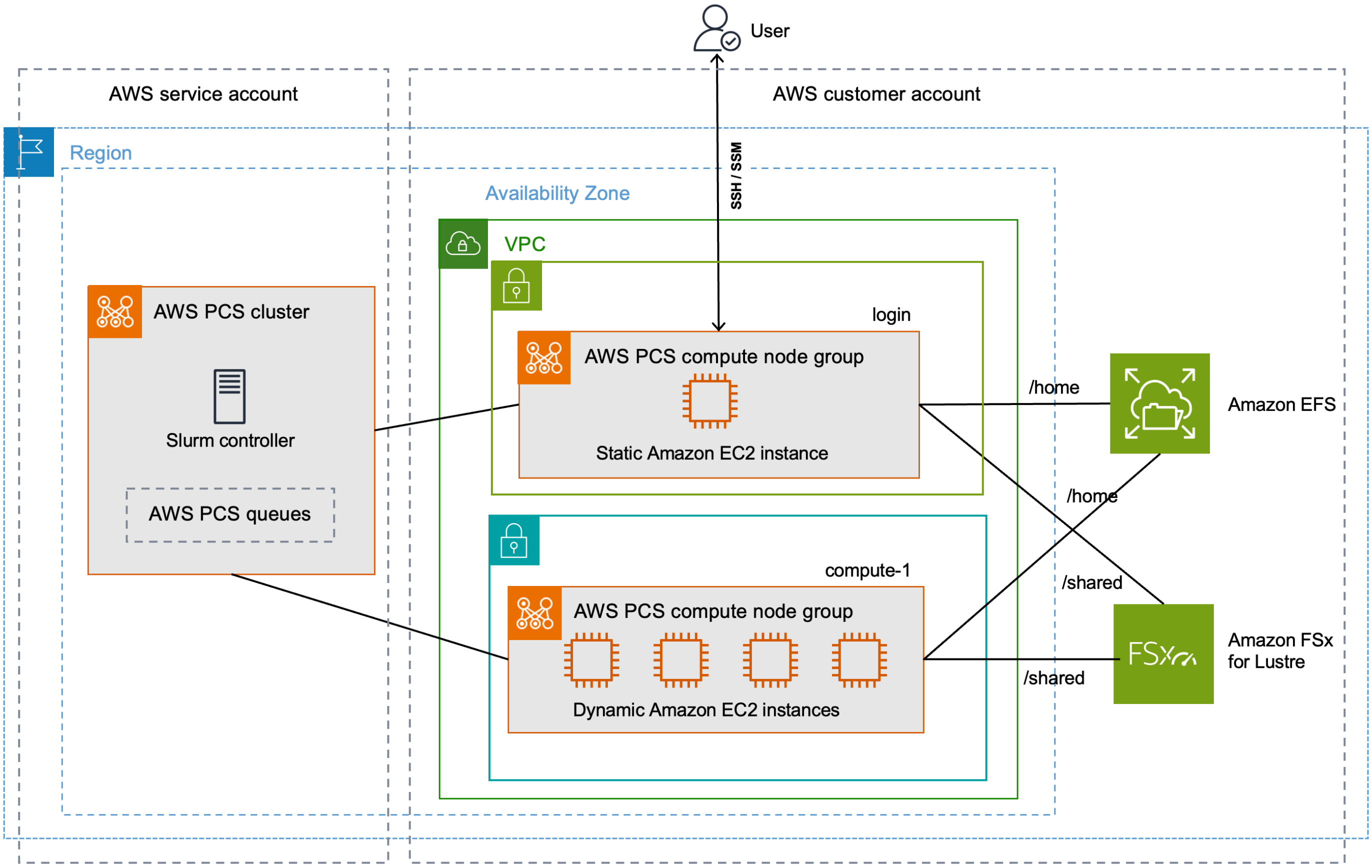
A system administrator deploys, maintains, and operates a cluster. They can access AWS PCS through the AWS Management Console, AWS PCS API, and AWS SDK. They have access to specific clusters through SSH or AWS Systems Manager, where they can run administrative tasks, run jobs, manage data, and perform other shell-based activities. For more information, see *AWS Systems Manager Documentation*.

End user

An end user doesn't have day-to-day responsibility to deploy or operate a cluster. They use a terminal interface (such as SSH) to access cluster resources, run jobs, manage data, and perform other shell-based activities.

Get started with AWS Parallel Computing Service

This is a tutorial to create a simple cluster that you can use to try AWS PCS. The following figure shows the design of the cluster.



The tutorial cluster design has the following key components:

- A VPC and subnets that meet AWS PCS networking requirements.
- An Amazon EFS file system, which will be used as a shared home directory.
- An Amazon FSx for Lustre file system, which provides a shared high performance directory.
- An AWS PCS cluster, which provides a Slurm controller.
- 2 AWS PCS compute node groups.
  - The `login` node group, which provides shell-based interactive access to the system.
  - The `compute-1` node group provides elastically-scaling instances to run jobs.
- 1 queue that sends jobs to EC2 instances in the `compute-1` node group.

The cluster requires additional AWS resources, such as security groups, IAM roles, and EC2 launch templates, which aren't shown in the diagram.

> **Note**
>
> We recommend that you complete the command line steps in this topic in a Bash shell. If you aren't using a Bash shell, some script commands such as line continuation characters and the way variables are set and used require adjustment for your shell. Additionally, the quoting and escaping rules for your shell might be different. For more information, see Quotation marks and literals with strings in the AWS CLI in the *AWS Command Line Interface User Guide for Version 2*.

Topics

Prerequisites for getting started with AWS PCS

Refer to the following topics to prepare your AWS account and local development environment for AWS PCS.

Topics

Sign up for AWS and create an administrative user

Complete the following tasks to set up for AWS Parallel Computing Service (AWS PCS).

Topics

Sign up for an AWS account

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 or text message 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 a user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.
2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

Install the AWS CLI for AWS PCS

You must use the latest version of the AWS CLI. For information, see Install or update to the latest version of the AWS CLI in the *AWS Command Line Interface User Guide for Version 2*.

You must configure the AWS CLI. For more information, see Configure the AWS CLI in the *AWS Command Line Interface User Guide for Version 2*.

Enter the following command at a command prompt to check your AWS CLI; it should display help information.

```
aws pcs help
```

Required IAM permissions for AWS PCS

The IAM security principal that you're using must have permissions to work with AWS PCS IAM roles, service linked roles, AWS CloudFormation, a VPC, and related resources. For more information, see Identity and Access Management for AWS Parallel Computing Service, and Create a service-linked role in the *AWS Identity and Access Management User Guide*. You must complete all steps in this guide as the same user. To check the current user, run the following command:

```
aws sts get-caller-identity
```

Using AWS CloudFormation with the AWS PCS tutorial

The AWS PCS tutorial has many steps and is intended to help you understand the parts of an AWS PCS cluster and the procedures required to create it. We recommend that you go through the tutorial steps at least 1 time. After you have a good understanding of what is involved, you can use AWS CloudFormation to create the sample cluster quickly with automation.

CloudFormation is an AWS service that enables you to create and provision AWS infrastructure deployments predictably and repeatedly. You can use a CloudFormation template to automatically provision the AWS resources for the sample cluster as a single unit, called a **stack**. You can delete the stack when you are done with it.

For more information, see Get started with CloudFormation and AWS PCS.

Create a VPC and subnets for AWS PCS

You can create a VPC and subnets with a CloudFormation template. Use the following URL to download the CloudFormation template, then upload the template in the CloudFormation console to create a new CloudFormation stack. For more information, see Using the CloudFormation console in the *AWS CloudFormation User Guide*.

```
https://aws-hpc-recipes.s3.amazonaws.com/main/recipes/net/hpc_large_scale/assets/
main.yaml
```

With the template open in the CloudFormation console, enter the following options. You can use the default values provided in the template.

- Under **Provide a stack name**:
  - Under **Stack name**, enter:

    ```
    hpc-networking
    ```
- Under **Parameters**:
  - Under **VPC**:
    - Under **CidrBlock**, enter:

      ```
      10.3.0.0/16
      ```
  - Under **Subnets A**:
    - Under **CidrPublicSubnetA**, enter:

      ```
      10.3.0.0/20
      ```

    - Under **CidrPrivateSubnetA**, enter:

      ```
      10.3.128.0/20
      ```

  - Under **Subnets B**:
    - Under **CidrPublicSubnetB**, enter:

      ```
      10.3.16.0/20
      ```

    - Under **CidrPrivateSubnetB**, enter:

      ```
      10.3.144.0/20
      ```

  - Under **Subnets C**:
    - For **ProvisionSubnetsC**, select **True**
    - Under **CidrPublicSubnetC**, enter:

      ```
      10.3.32.0/20
      ```

    - Under **CidrPrivateSubnetC**, enter:

      ```
      10.3.160.0/20
      ```

- Under **Capabilities**:
  - Check the box for **I acknowledge that AWS CloudFormation might create IAM resources**.

Monitor the status of the CloudFormation stack. When it reaches `CREATE_COMPLETE`, find the ID for the default security group in the new VPC. You use the ID later in the tutorial.

Find the default security group for the cluster VPC

To find the ID for the default security group in the new VPC, follow this procedure:

- Navigate to the Amazon VPC console.
- Under the **VPC Dashboard**, select **Filter by VPC**.
  - Choose the VPC where the name starts with `hpc-networking`.
  - Under **Security**, choose **Security groups**.

- Find the **Security group ID** for the group named `default`. It has the description `default VPC security group`. You use the ID later to configure EC2 launch templates.

Create security groups for AWS PCS

AWS PCS relies on security groups to manage network traffic into and out of a cluster and its compute node groups. For detailed information on this topic, see Security group requirements and considerations.

In this step, you will use an CloudFormation template to create two security groups.

- A cluster security group, which enables communications between AWS PCS controller, compute nodes, and login nodes.
- An inbound SSH security group, which you can optionally add to your login nodes to support SSH access

Create the security groups for AWS PCS

You can use a CloudFormation template to create the security groups. Use the following URL to download the CloudFormation template, then upload the template in the CloudFormation console to create a new CloudFormation stack. For more information, see Using the CloudFormation console in the *AWS CloudFormation User Guide*.

```
https://aws-hpc-recipes.s3.amazonaws.com/main/recipes/pcs/getting_started/assets/pcs-
cluster-sg.yaml
```

With the template open in the AWS CloudFormation console, enter the following options. Note that some options will be pre-populated in the template — you can simply leave them as the default values.

- Under **Provide a stack name**
  - Under **Stack name**, enter:

    ```
    getstarted-sg
    ```
- Under **Parameters**
  - Under **VpcId**, choose the VPC where the name starts with `hpc-networking`.

- (Optional) Under **ClientIpCidr**, enter a more restrictive IP range for the inbound SSH security group. We recommend that you restrict this with your own IP/subnet (x.x.x.x/32 for your own ip or x.x.x.x/24 for range. Replace x.x.x.x with your own PUBLIC IP. You can get your public IP using tools such as https://ifconfig.co/)

Monitor the status of the CloudFormation stack. When it reaches `CREATE_COMPLETE` the security group resources are ready.

There are two security groups created, with the names:

- `cluster-getstarted-sg` – this is the cluster security group
- `inbound-ssh-getstarted-sg` – this is a security group to allow inbound SSH access

Create a cluster in AWS PCS

In AWS PCS, a cluster is a persistent resource for managing resources and running workloads. You create a cluster for a specific scheduler (AWS PCS currently supports Slurm) in a subnet of a new or existing VPC. The cluster accepts and schedules jobs, and also launches the compute nodes (EC2 instances) that process those jobs.

To create your cluster

1. Open the AWS PCS console and choose **Create cluster**.
2. In the **Cluster details** section, enter the following fields:
   - **Cluster name** – Enter `get-started`
   - **Scheduler** – Select **Slurm Version 25.05**
   - **Controller size** – Select **Small**
3. In the **Networking** section, select values for the following fields:
   - **VPC** – Choose the VPC named `hpc-networking:Large-Scale-HPC`
   - **Subnet** – Select the subnet where the name starts with `hpc-networking:PrivateSubnetA`
   - **Security groups** – Select the cluster security group named `cluster-getstarted-sg`
4. Choose **Create cluster**.

> **Note**
>
> The **Status** field shows **Creating** while the cluster is being provisioned. Cluster creation can take several minutes.

Create shared storage for AWS PCS in Amazon Elastic File System

Amazon Elastic File System (Amazon EFS) is an AWS service that provides serverless, fully elastic file storage so that you can share file data without provisioning or managing storage capacity and performance. For more information, see [What is Amazon Elastic File System?]() in the *Amazon Elastic File System User Guide*.

The AWS PCS demonstration cluster uses an EFS file system to provide a shared home directory between the cluster nodes. Create an EFS file system in the same VPC as your cluster.

To create your Amazon EFS file system

1. Go to the [Amazon EFS console]().
2. Make sure it's set to the same AWS Region where you will try AWS PCS.
3. Choose **Create file system**.
4. On the **Create file system** page, set the following parameters:
    - For **Name**, enter `getstarted-efs`
    - Under **Virtual Private Cloud (VPC)**, choose the VPC named `hpc-networking:Large-Scale-HPC`
    - Choose **Create**. This returns you to the **File systems** page.
5. Make a note of the **File system ID** for the `getstarted-efs` file system. You use this information later.

Create shared storage for AWS PCS in Amazon FSx for Lustre

Amazon FSx for Lustre makes it easy and cost-effective to launch and run the popular, high-performance Lustre file system. You use Lustre for workloads where speed matters, such as machine learning, high performance computing (HPC), video processing, and financial modeling. For more information, see What is Amazon FSx for Lustre? in the *Amazon FSx for Lustre User Guide*.

The AWS PCS demonstration cluster can use an FSx for Lustre file system to provide a high-performance shared directory between the cluster nodes. Create an FSx for Lustre file system in the same VPC as your cluster.

To create your FSx for Lustre file system

1. Go to the Amazon FSx console.
2. Make sure the console is set to use the same AWS Region as your cluster.
3. Choose **Create file system**.
    - For **Select file system type**, choose **Amazon FSx for Lustre**, then choose **Next**.
4. On the **Specify file system details** page, set the following parameters:
    - Under **File system details**
        - For **Name**, enter `getstarted-fsx`
        - For **Deployment and storage type**, choose **Persistent, SSD**
        - For **Throughput per unit of storage**, choose **125 MB/s/TiB**
        - For **Storage capacity**, enter 1.2 TiB
        - For **Metadata Configuration**, choose **Automatic**
        - For **Data compression type**, choose **LZ4**
    - Under **Network & security**
        - For **Virtual Private Cloud (VPC)**, choose the VPC named `hpc-networking:Large-Scale-HPC`
        - For **VPC Security Groups**, leave the security group named `default`
        - For **Subnet**, choose the subnet where the name starts with `hpc-networking:PrivateSubnetA`
    - Leave the other options set to their default values.
    - Choose **Next**.
5. On the **Review and create** page, choose **Create file system**. This returns you to the **File systems** page.
6. Navigate to the details page for the FSx for Lustre file system you created.
7. Make a note of the **File system ID** and the **Mount name**. You use this information later.

> **Note**
>
> The **Status** field shows **Creating** while the file system is being provisioned. File system creation can take several minutes. Wait until it completes before proceeding with the rest of the tutorial.

Create compute node groups in AWS PCS

A compute node group is virtual collection of compute nodes (EC2 instances) that AWS PCS launches and manages. When you define a compute node group, you specify common traits such as EC2 instance types, minimum and maximum instance count, target VPC subnets, preferred purchase option, and custom launch configuration. AWS PCS efficiently launches, manages, and terminates compute nodes in a compute node group, according to these settings. The demonstration cluster uses a compute node group to provide login nodes for user access, and a separate compute node group to process jobs. The following topics describe the procedures to set up these compute node groups in your cluster.

Topics

Create an instance profile for AWS PCS

Compute node groups require an instance profile when they are created. If you use the AWS Management Console to create a role for Amazon EC2, the console automatically creates an instance profile and gives it the same name as the role. For more information, see Using instance profiles in the *AWS Identity and Access Management User Guide*.

In the following procedure, you use the AWS Management Console to create a role for Amazon EC2, which also creates the instance profile for your compute node groups.

To create the role and instance profile

- Navigate to the IAM console.

- Under **Access management**, choose **Policies**.
  - Choose **Create policy**.
  - Under **Specify permissions**, for **Policy editor**, choose **JSON**.
  - Replace the contents of the text editor with the following:

    JSON

    ```
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Action": [
                    "pcs:RegisterComputeNodeGroupInstance"
                ],
                "Resource": "*",
                "Effect": "Allow"
            }
        ]
    }
    ```
  - Choose **Next**.
  - Under **Review and create**, for **Policy name**, enter `AWSPCS-getstarted-policy`.
  - Choose **Create policy**.
- Under **Access management**, choose **Roles**.
- Choose **Create role**.
- Under **Select trusted entity**:
  - For **Trusted entity type**, select **AWS service**
  - Under **Use case**, select **EC2**.
    - Then, under **Choose a use case** for the specified service, choose **EC2**.
  - Choose **Next**.
- Under **Add permissions**:
  - In **Permissions policies**, search for **AWSPCS-getstarted-policy**.
  - Check the box beside **AWSPCS-getstarted-policy** to add it to the role.
  - In **Permissions policies**, search for **AmazonSSMManagedInstanceCore**.
  - Check the box beside **AmazonSSMManagedInstanceCore** to add it to the role.
  - Choose **Next**.

- Under **Name, review, and create**:
  - Under **Role details**:
    - For **Role name**, enter `AWSPCS-getstarted-role`.
  - Choose **Create role**.

Create launch templates for AWS PCS

When you create a compute node group, you provide an EC2 launch template that AWS PCS uses to configure EC2 instances it launches. This includes settings such as security groups and scripts that run when the instance launches.

In this step, one CloudFormation template will be used to create two EC2 launch templates. One template will be used to create login nodes, and the other will be used to create compute nodes. The key difference between them is that the login nodes can be configured to allow inbound SSH access.

Access the CloudFormation template

Use the following URL to download the CloudFormation template, then upload the template in the CloudFormation console to create a new CloudFormation stack. For more information, see Using the CloudFormation console in the *AWS CloudFormation User Guide*.

```
https://aws-hpc-recipes.s3.amazonaws.com/main/recipes/pcs/getting_started/assets/pcs-lt-efs-fsxl.yaml
```

Use the CloudFormation template to create EC2 launch templates

Use the following procedure to complete the CloudFormation template in the CloudFormation console

- Under **Provide a stack name**:
  - Under **Stack name**, enter `getstarted-lt`.
- Under **Parameters**:
  - Under **Security**
    - For **VpcSecurityGroupId**, select the security group named `default` in your cluster VPC.
    - For **ClusterSecurityGroupId**, select the group named `cluster-getstarted-sg`
    - For **SshSecurityGroupId**, select the group named `inbound-ssh-getstarted-sg`

    - For **SshKeyName**, select your preferred SSH key pair.
  - Under **File systems**
    - For **EfsFilesystemId**, enter the file system ID from the EFS file system you created earlier in the tutorial.
    - For **FSxLustreFilesystemId**, enter the file system ID from the FSx for Lustre file system you created earlier in the tutorial.
    - For **FSxLustreFilesystemMountName**, enter the mount name for that same FSx for Lustre file system.
- Choose **Next**, then choose **Next** again.
- Choose **Submit**.

Monitor the status of the CloudFormation stack. When it reaches `CREATE_COMPLETE` the launch template is ready to be used.

> **Note**
>
> To see all the resources the CloudFormation template created, open the [CloudFormation console](). Choose the `getstarted-lt` stack and then choose the **Resources** tab.

Create compute node group for login nodes in AWS PCS

A compute node group is virtual collection of compute nodes (EC2 instances) that AWS PCS launches and manages. When you define a compute node group, you specify common traits such as EC2 instance types, minimum and maximum instance count, target VPC subnets, preferred purchase option, and custom launch configuration. AWS PCS efficiently launches, manages, and terminates compute nodes in a compute node group, according to these settings.

In this step, you will launch a static compute node group that provides interactive access to the cluster. You can use SSH or Amazon EC2 Systems Manager (SSM) to log in to it, then run shell commands and manage Slurm jobs.

To create the compute node group

- Open the [AWS PCS console]() and navigate to **Clusters**.
- Select the cluster named `get-started`

- Navigate to **Compute node groups** and choose **Create**.
- In the **Compute node group setup** section, provide the following:
  - **Compute node group name –** Enter `login`.
- Under **Computing configuration**, enter or select these values:
  - **EC2 launch template –** Choose the launch template where the name is `login-getstarted-lt`
  - **IAM instance profile –** Choose the instance profile named `AWSPCS-getstarted-role`
  - **Subnets –** Select the subnet where the name starts with `hpc-networking:PublicSubnetA`.
  - **Instances –** Select `c6i.xlarge`.
  - **Scaling configuration –** For **Min. instance count**, enter 1. For **Max. instance count**, enter 1.
- Under **Additional settings**, specify the following:
  - **AMI ID –** Select an AMI you want to use, that has a name in the following format:

    ```
    aws-pcs-sample_ami-amzn2-platform-slurm-version
    ```

    For more information about the sample AMIs, see Using sample Amazon Machine Images (AMIs) with AWS PCS.
- Choose **Create compute node group**.

The **Status** field shows **Creating** while the compute node group is being provisioned. You can proceed to the next step in the tutorial while it is in progress.

Create compute node group for running compute jobs in AWS PCS

In this step, you will launch a compute node group that scales elastically to run jobs submitted to the cluster.

To create the compute node group

- Open the AWS PCS console and navigate to **Clusters**.
- Select the cluster named `get-started`
- Navigate to **Compute node groups** and choose **Create**.
- In the **Compute node group setup** section, provide the following:
  - **Compute node group name –** Enter `compute-1`.

- Under **Computing configuration**, enter or select these values:
  - **EC2 launch template** – Choose the launch template where the name is `compute-getstarted-lt`
  - **IAM instance profile** – Choose the instance profile named `AWSPCS-getstarted-role`
  - **Subnets** – Select the subnet where the name starts with `hpc-networking:PrivateSubnetA`.
  - **Instances** – Select `c6i.xlarge`.
  - **Scaling configuration** – For **Min. instance count**, enter `0`. For **Max. instance count**, enter `4`.
- Under **Additional settings**, specify the following:
  - **AMI ID** – Select an AMI you want to use, that has a name in the following format:

    ```
    aws-pcs-sample_ami-amzn2-platform-slurm-version
    ```

    For more information about the sample AMIs, see Using sample Amazon Machine Images (AMIs) with AWS PCS.
- Choose **Create compute node group**.

The **Status** field shows **Creating** while the compute node group is being provisioned.

> ⚠ **Important**
>
> Wait for the **Status** field to show **Active** before proceeding to the next step in this tutorial.

Create a queue to manage jobs in AWS PCS

You submit a job to a queue to run it. The job remains in the queue until AWS PCS schedules it to run on a compute node group. Each queue is associated with one or more compute node groups, which provide the necessary EC2 instances to do the processing.

In this step, you will create a queue that uses the compute node group to process jobs.

To create a queue

- Open the AWS PCS console.
- Select the cluster named `get-started`.

- Navigate to **Compute node groups** and make sure the status of the `compute-1` group is **Active**.

  > ⚠ **Important**
  >
  > The status of the `compute-1` group must be **Active** before you proceed to the next step.

- Navigate to **Queues** and choose **Create queue**.
  - In the **Queue configuration** section, provide the following values:
    - **Queue name** – Enter the following: `demo`
    - **Compute node groups** – Select the compute node group named `compute-1`.
- Choose **Create queue**.

The **Status** field shows **Creating** while the queue is being created.

> ⚠ **Important**
>
> Wait for the **Status** field to show **Active** before proceeding to the next step in this tutorial.

Connect to your AWS PCS cluster

After the status of the `login` compute node group becomes **Active**, you can connect to the EC2 instance it created.

To connect to the login node

- Open the [AWS PCS console]() and navigate to **Clusters**.
- Select the cluster named `get-started`.
- Choose **Compute node groups**.
- Navigate to the compute node group named `login`.
- Find the **Compute node group ID**.
- In another browser window or tab, open the [Amazon EC2 console]().
  - Choose **Instances**.
  - Search for EC2 instances with the following tag. Replace *`node-group-id`* with the value of the **Compute node group ID** from the previous step. There should be 1 instance.

```
aws:pcs:compute-node-group-id=node-group-id
```

- Connect to the EC2 instance. You can use Session Manager or SSH.

  Session Manager

  - Select the instance.
  - Choose **Connect**.
  - Under **Connect to instance**, select **Session Manager**.
  - Choose **Connect**.
  - Choose **Connect**. An interactive terminal launches in your browser.

  SSH

  - Select the instance.
  - Choose **Connect**.
  - Under **Connect to instance**, select **SSH client**.
  - Follow the instructions provided by the console.

  > **Note**
  >
  > The user name for the instance is **`ec2-user`** not `root`.

Explore the cluster environment in AWS PCS

After you have logged into the cluster, you can run shell commands. For instance, you can change users, work with data on shared filesystems, and interact with Slurm.

Change user

If you have logged in to the cluster using Session Manager, you may be connected as `ssm-user`. This is an special user that is created for Session Manager. Switch to the default user on Amazon Linux 2 using the following command. You will not need to do this if you connected using SSH.

```
sudo su - ec2-user
```

Work with shared file systems

You can confirm that the EFS filesystem and FSx for Lustre file systems are available with the command `df -h`. Output on your cluster should resemble the following:

```
[ec2-user@ip-10-3-6-103 ~]$ df -h
Filesystem                 Size  Used Avail Use% Mounted on
devtmpfs                   3.8G     0  3.8G   0% /dev
tmpfs                      3.9G     0  3.9G   0% /dev/shm
tmpfs                      3.9G  556K  3.9G   1% /run
tmpfs                      3.9G     0  3.9G   0% /sys/fs/cgroup
/dev/nvme0n1p1              24G   18G  6.6G  73% /
127.0.0.1:/                8.0E     0  8.0E   0% /home
10.3.132.79@tcp:/zlshxbev  1.2T  7.5M  1.2T   1% /shared
tmpfs                      780M     0  780M   0% /run/user/0
tmpfs                      780M     0  780M   0% /run/user/1000
```

The /home filesystem mounts 127.0.0.1 and has a very large capacity. This is the EFS file system that you created earlier in the tutorial. Any files written here will be available under /home on all nodes in the cluster.

The `/shared` filesystem mounts a private IP and has a capacity of 1.2 TB. This is the FSx for Lustre file system that you created earlier in the tutorial. Any files written here will be available under `/shared` on all nodes in the cluster.

Interact with Slurm

Topics

- List queues and nodes
- Show jobs

List queues and nodes

You can list the queues and the nodes they are associated with using `sinfo`. Output from your cluster should resemble the following:

```
[ec2-user@ip-10-3-6-103 ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
demo         up   infinite      4  idle~ compute-1-[1-4]
```

```
[ec2-user@ip-10-3-6-103 ~]$
```

Note the partition named demo. Its status is up and it has a maximum of 4 nodes. It is associated with nodes in the `compute-1` node group. If you edit the compute node group and increase the maximum number of instances to 8, the number of nodes would read 8 and the node list would read `compute-1-[1-8]`. If you created a second compute node group named `test` with 4 nodes, and added it to the demo queue, those nodes would show up in the node list as well.

Show jobs

You can list all jobs, in any state, on the system with `squeue`. Output from your cluster should resemble the following:

```
[ec2-user@ip-10-3-6-103 ~]$ squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
```

Try running `squeue` again later, when you have a Slurm job pending or running.

Run a single node job in AWS PCS

To run a job using Slurm, you prepare a submission script specifying job requirements and submit it to a queue with the `sbatch` command. Typically, this is done from a shared directory so the login and compute nodes have a common space for accessing files.

Connect to the login node of your cluster and run the following commands at its shell prompt.

- Become the default user. Change to the shared directory.

  ```
  sudo su - ec2-user
  cd /shared
  ```

- Use the following commands to create an example job script:

  ```
  cat << EOF > job.sh
  #!/bin/bash
  #SBATCH -J single
  #SBATCH -o single.%j.out
  #SBATCH -e single.%j.err
  ```

```
echo "This is job \${SLURM_JOB_NAME} [\${SLURM_JOB_ID}] running on \
${SLURMD_NODENAME}, submitted from \${SLURM_SUBMIT_HOST}" && sleep 60 && echo "Job
 complete"
EOF
```

- Submit the job script to the Slurm scheduler:

```
sbatch -p demo job.sh
```

- When the job is submitted, it will return a job ID as a number. Use that ID to check the job status. Replace *job-id* in the following command with the number returned from `sbatch`.

```
squeue --job job-id
```

Example

```
squeue --job 1
```

The `squeue` command returns output similar to the following:

```
JOBID PARTITION NAME USER     ST TIME NODES NODELIST(REASON)
1     demo      test ec2-user CF 0:47 1     compute-1
```

- Continue to check the status of the job until it reaches the `R` (running) status. The job is done when `squeue` doesn't return anything.
- Inspect the contents of the `/shared` directory.

```
ls -alth /shared
```

The command output is similar to the following:

```
-rw-rw-r- 1 ec2-user ec2-user 107 Mar 19 18:33 single.1.out
-rw-rw-r- 1 ec2-user ec2-user 0 Mar 19 18:32 single.1.err
-rw-rw-r- 1 ec2-user ec2-user 381 Mar 19 18:29 job.sh
```

The files named `single.1.out` and `single.1.err` were written by one of your cluster's compute nodes. Because the job was run in a shared directory (`/shared`), they are also available on your login node. This is why you configured an FSx for Lustre file system for this cluster.

- Inspect the contents of the `single.1.out` file.

```
cat /shared/single.1.out
```

The output is similar to the following:

```
This is job test [1] running on compute-1, submitted from ip-10-3-13-181
Job complete
```

Run a multi-node MPI job with Slurm in AWS PCS

These instructions demonstrate using Slurm to run a message passing interface (MPI) job in AWS PCS.

Run the following commands at a shell prompt of your login node.

- Become the default user. Change to its home directory.

```
sudo su - ec2-user
cd ~/
```

- Create source code in the C programming language.

```
cat > hello.c << EOF
// * mpi-hello-world - https://www.mpitutorial.com
// Released under MIT License
//
// Copyright (c) 2014 MPI Tutorial.
//
// Permission is hereby granted, free of charge, to any person obtaining a copy
// of this software and associated documentation files (the "Software"), to
// deal in the Software without restriction, including without limitation the
// rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
// sell copies of the Software, and to permit persons to whom the Software is
// furnished to do so, subject to the following conditions:
// The above copyright notice and this permission notice shall be included in
// all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
// FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
// DEALINGS IN THE SOFTWARE.

#include <mpi.h>
#include <stdio.h>
#include <stddef.h>

int main(int argc, char** argv) {
  // Initialize the MPI environment. The two arguments to MPI Init are not
  // currently used by MPI implementations, but are there in case future
  // implementations might need the arguments.
  MPI_Init(NULL, NULL);

  // Get the number of processes
  int world_size;
  MPI_Comm_size(MPI_COMM_WORLD, &world_size);

  // Get the rank of the process
  int world_rank;
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

  // Get the name of the processor
  char processor_name[MPI_MAX_PROCESSOR_NAME];
  int name_len;
  MPI_Get_processor_name(processor_name, &name_len);

  // Print off a hello world message
  printf("Hello world from processor %s, rank %d out of %d processors\n",
         processor_name, world_rank, world_size);

  // Finalize the MPI environment. No more MPI calls can be made after this
  MPI_Finalize();
}
EOF
```

- Load the OpenMPI module.

```
module load openmpi
```

- Compile the C program.

```
mpicc -o hello hello.c
```

- Write a Slurm job submission script.

```
cat > hello.sh << EOF
#!/bin/bash
#SBATCH -J multi
#SBATCH -o multi.out
#SBATCH -e multi.err
#SBATCH --exclusive
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=1

srun $HOME/hello
EOF
```

- Change to the shared directory.

```
cd /shared
```

- Submit the job script.

```
sbatch -p demo ~/hello.sh
```

- Use `squeue` to monitor the job until it's done.
- Check the contents of `multi.out`:

```
cat multi.out
```

The output is similar to the following. Note that each rank has its own IP address because it ran on a different node.

```
Hello world from processor ip-10-3-133-204, rank 0 out of 4 processors
Hello world from processor ip-10-3-128-219, rank 2 out of 4 processors
Hello world from processor ip-10-3-141-26, rank 3 out of 4 processors
Hello world from processor ip-10-3-143-52, rank 1 out of 4 processor
```

Delete your AWS resources for AWS PCS

After you are done with the cluster and node groups that you created for this tutorial, you should delete the resources that you created.

> ⚠ **Important**
>
> You get billing charges for all resources running in your AWS account

To delete AWS PCS resources that you created for this tutorial

- Open the AWS PCS console.
- Navigate to the cluster named **get-started**.
- Navigate to the **Queues** section.
- Select the queue named **demo**.
- Choose **Delete**.

> ⚠ **Important**
>
> Wait until the queue has been deleted before proceeding.

- Navigate to the **Compute node groups** section.
- Select the compute node group named **compute-1**.
- Choose **Delete**.
- Select the compute node group named **login**.
- Choose **Delete**.

> ⚠ **Important**
>
> Wait until both compute node groups have been deleted before proceeding.

- In the cluster detail page for **get-started**, choose **Delete**.

> ⚠ **Important**
>
> Wait until the cluster has been deleted before proceeding with subsequent steps.

To delete other AWS resources you created for this tutorial

- Open the IAM console.
  - Choose **Roles**.
  - Select the role named **AWSPCS-getstarted-role** then choose **Delete**.
  - After the role has been deleted, choose **Policies**.
  - Select the policy named **AWSPCS-getstarted-policy** then choose **Delete**.
- Open the CloudFormation console.
  - Select the stack named **getstarted-lt**.
  - Choose **Delete**.

  > ⚠ **Important**
  >
  > Wait for the stack to delete before proceeding.
- Open the Amazon EFS console.
  - Choose **File systems**.
  - Select the file system named **getstarted-efs**.
  - Choose **Delete**.

  > ⚠ **Important**
  >
  > Wait for the file system to delete before proceeding.
- Open the Amazon FSx console.
  - Choose **File systems**.
  - Select the file system named **getstarted-fsx**.
  - Choose **Delete**.

> ⚠ **Important**
>
> Wait for the file system to delete before proceeding.

- Open the CloudFormation console.
  - Select the stack named **getstarted-sg**.
  - Choose **Delete**.
- Open the CloudFormation console.
  - Select the stack named **hpc-networking**.
  - Choose **Delete**.

Get started with CloudFormation and AWS PCS

You can use AWS CloudFormation to create an AWS PCS cluster. CloudFormation enables you to create and provision AWS infrastructure deployments predictably and repeatedly. You can use CloudFormation to automatically provision resources from many AWS services to build highly reliable, scalable, cost-effective applications in the AWS Cloud without creating and configuring the underlying AWS infrastructure. CloudFormation enables you to use a template file to create and delete a collection of resources together as a single unit, called a **stack**. For more information about CloudFormation, see What is CloudFormation? in the *AWS CloudFormation User Guide*. For more information about AWS PCS resource types in CloudFormation, see AWS PCS resource type reference in the *AWS CloudFormation User Guide*.

Topics

Use CloudFormation to create a sample AWS PCS cluster

The following procedure uses a CloudFormation template in the AWS Management Console to create a sample AWS PCS cluster. For more information about CloudFormation, see What is CloudFormation? in the *AWS CloudFormation User Guide*. For more information about AWS PCS resource types in CloudFormation, see AWS PCS resource type reference in the *AWS CloudFormation User Guide*.

To create the sample cluster

1. Choose the AWS Region to create the cluster in (the link opens the CloudFormation console with the template):

   - US East (N. Virginia) (us-east-1)
   - US East (Ohio) (us-east-2)
   - US West (Oregon) (us-west-2)

- Asia Pacific (Singapore) (ap-southeast-1)
- Asia Pacific (Sydney) (ap-southeast-2)
- Asia Pacific (Tokyo) (ap-northeast-1)
- Europe (Frankfurt) (eu-central-1)
- Europe (Ireland) (eu-west-1)
- Europe (London) (eu-west-2)
- Europe (Stockholm) (eu-north-1)
- AWS GovCloud (US-East) (us-gov-east-1)
- AWS GovCloud (US-West) (us-gov-west-1)

2. Under **Provide a stack name**, enter a descriptive name. This is the name for your CloudFormation stack. The template uses this value as the name for your AWS PCS cluster.
3. Under **Parameters**:

   a. Under **SlurmVersion**, choose the version of Slurm you want your cluster to use.

   b. Under **NodeArchitecture**, choose **x86** to deploy a cluster that uses x86_64-compatible instances, or choose **Graviton** to use Arm64 instances.

   c. For **KeyName**, choose an SSH key pair to access the cluster login nodes. Make sure that you have the PEM file for the key pair that you choose.

   d. For **ClientIpCidr**, enter an IP range in CIDR format to control access to the login nodes.

   > ⚠ **Warning**
   >
   > The default value of `0.0.0.0/0` allows access from all IP addresses.

   e. Leave the values for **HpcRecipesS3Bucket** and **HpcRecipesBranch** as their default values.
4. Under **Capabilities and transforms**:

   a. Select the checkbox to acknowledge that CloudFormation will create IAM resources.

   b. Select the checkbox to acknowledge that CloudFormation will create IAM resources with custom names.

   c. Select the checkbox to acknowledge `CAPABILITY_AUTO_EXPAND` for the new stack. For more information, see CreateStack in the *AWS CloudFormation API Reference*.
5. Choose **Create stack**.

6. Monitor the status of your stack. You can connect to the cluster after the status of the stack is `CREATE_COMPLETE`.

Connect to a AWS PCS cluster created with CloudFormation

After you create an AWS PCS cluster from a CloudFormation template, you can use the AWS PCS console (in the AWS Management Console) to administer the cluster. You can also connect to 1 of the cluster's login nodes to administer the cluster, run jobs, and manage data. The CloudFormation stack provides links you can use to connect to your cluster.

To connect to your cluster

1. Open the CloudFormation console
2. Choose the stack you created.
3. Choose the **Outputs** tab of the stack.

   The stack provides the following links:

   - **PcsConsoleUrl** — Choose this link to open the AWS PCS console with the cluster selected. You can use it to explore the cluster, node group, and queue configurations.
   - **Ec2ConsoleUrl** — Choose this link to open the Amazon EC2 console, filtered to show the instances that the cluster's login node group manages.

     From this view, you can select an instance and choose **Connect**. The sample cluster's instance supports inbound SSH and AWS Systems Manager connections in a web browser. For more information, see Connect to your AWS PCS cluster.

     After you connect to a login instance, you can follow the tutorial at Explore the cluster environment in AWS PCS.

Clean up an AWS PCS cluster in CloudFormation

If you used CloudFormation to create your AWS PCS cluster, you can open the CloudFormation console and delete the stack to delete the cluster and all its associated resources.

⚠ **Important**

For the sample cluster, if you created additional compute node groups or queues in your cluster (beyond the `login` and `compute-1` groups that the sample CloudFormation template created), you must use the AWS PCS console or AWS CLI to delete those resources before you delete the CloudFormation stack. For more information, see Deleting a cluster in AWS PCS.

Parts of a CloudFormation template for AWS PCS

A CloudFormation template has 1 or more sections that each serve a specific purpose. CloudFormation defines standard format, syntax, and language in a template. For more information, see Working with CloudFormation templates in the *AWS CloudFormation User Guide*.

CloudFormation templates are highly customizable and therefore their formats can vary. To understand the necessary parts of a CloudFormation template to create an AWS PCS cluster, we recommend you examine the sample template we provide to create a sample cluster. This topic briefly explains the sections of that sample template.

⚠ **Important**

The code samples in this topic are **not complete**. The presence of ellipsis (`[...]`) indicates that there is additional code that isn't displayed. To download the complete YAML-formatted CloudFormation template, see CloudFormation templates to create a sample AWS PCS cluster.

Contents

Header

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: AWS Parallel Computing Service "getting started" cluster
```

`AWSTemplateFormatVersion` identifies the template format version that the template conforms to. For more information, see CloudFormation template format version syntax in the *AWS CloudFormation User Guide*.

`Transform` specifies a macro that CloudFormation uses to process the template. For more information, see CloudFormation template Transform section in the *AWS CloudFormation User Guide*. The `AWS::Serverless-2016-10-31` transform enables CloudFormation to process a template written in the AWS Serverless Application Model (AWS SAM) syntax. For more information, see `AWS::Serverless` transform in the *AWS CloudFormation User Guide*.

Metadata

```
### Stack metadata
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label:
          default: PCS Cluster configuration
        Parameters:
          - SlurmVersion
          - ManagedAccounting
          - AccountingPolicyEnforcement
      - Label:
          default: PCS ComputeNodeGroups configuration
        Parameters:
          - NodeArchitecture
          - KeyName
          - ClientIpCidr
      - Label:
          default: HPC Recipes configuration
        Parameters:
          - HpcRecipesS3Bucket
          - HpcRecipesBranch
```

The `metadata` section of a CloudFormation template provides information about the template itself. The sample template creates a complete high performance computing (HPC) cluster that uses AWS PCS. The metadata section of the sample template declares parameters that control how CloudFormation launches (provisions) the corresponding stack. There are parameters that control architecture choice (`NodeArchitecture`), Slurm version (`SlurmVersion`), and access controls (`KeyName` and `ClientIpCidr`).

Parameters

The `Parameters` section defines the custom parameters for the template. CloudFormation uses these parameter definitions to construct and validate the form that you interact with when you launch a stack from this template.

```
Parameters:

  NodeArchitecture:
    Type: String
    Default: x86
    AllowedValues:
      - x86
      - Graviton
    Description: Processor architecture for the login and compute node instances

  SlurmVersion:
    Type: String
    Default: 25.05
    Description: Version of Slurm to use
    AllowedValues:
         - 24.11
         - 25.05

  ManagedAccounting:
    Type: String
    Default: 'disabled'
    AllowedValues:
      - 'enabled'
      - 'disabled'
    Description: Monitor cluster usage, manage access control, and enforce resource
 limits with Slurm accounting. Requires Slurm 24.11 or newer.

  AccountingPolicyEnforcement:
    Description: Specify which Slurm accounting policies to enforce
    Type: String
    Default: none
    AllowedValues:
      - none
      - 'associations,limits,safe'

  KeyName:
    Description: SSH keypair to log in to the head node
    Type: AWS::EC2::KeyPair::KeyName
    AllowedPattern: ".+"  # Required

  ClientIpCidr:
    Description: IP(s) allowed to access the login node over SSH. We recommend that you restrict it with your own IP/subnet (x.x.x.x/32 for your own ip or x.x.x.x/24 for range. Replace x.x.x.x with your own PUBLIC IP. You can get your public IP using tools such as https://ifconfig.co/)
    Default: 127.0.0.1/32
    Type: String
    AllowedPattern: (\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})/(\d{1,2})
    ConstraintDescription: Value must be a valid IP or network range of the form x.x.x.x/x.

  HpcRecipesS3Bucket:
    Type: String
    Default: aws-hpc-recipes
    Description: HPC Recipes for AWS S3 bucket
    AllowedValues:
         - aws-hpc-recipes
         - aws-hpc-recipes-dev
  HpcRecipesBranch:
    Type: String
    Default: main
    Description: HPC Recipes for AWS release branch
    AllowedPattern: '^(?!.*/\.git$)(?!.*/\.)(?!.*\\.\.)[a-zA-Z0-9-_\.]+$'
```

Mappings

The `Mappings` section defines key-value pairs that specify values based on certain conditions or dependencies.

```
Mappings:

  Architecture:
    AmiArchParameter:
      Graviton: arm64
      x86: x86_64
    LoginNodeInstances:
      Graviton: c7g.xlarge
      x86: c6i.xlarge
    ComputeNodeInstances:
      Graviton: c7g.xlarge
      x86: c6i.xlarge
```

Resources

The `Resources` section declares the AWS resources to provision and configure as part of the stack.

```
Resources:

  [...]
```

The template provisions the sample cluster infrastructure in layers. It starts with `Networking` for VPC configuration. Storage is provided by dual systems: `EfsStorage` for shared storage and `FSxLStorage` for high-performance storage. The core cluster is established through `PCSCluster`.

```

  Networking:
    Type: AWS::CloudFormation::Stack
    Properties:
      Parameters:
        ProvisionSubnetsC: "False"
      TemplateURL: !Sub 'https://${HpcRecipesS3Bucket}.s3.amazonaws.com/
${HpcRecipesBranch}/recipes/net/hpc_large_scale/assets/main.yaml'

  EfsStorage:
    Type: AWS::CloudFormation::Stack
    Properties:
      Parameters:
        SubnetIds: !GetAtt [ Networking, Outputs.DefaultPrivateSubnet ]
        SubnetCount: 1
        VpcId: !GetAtt [ Networking, Outputs.VPC ]
      TemplateURL: !Sub 'https://${HpcRecipesS3Bucket}.s3.amazonaws.com/
${HpcRecipesBranch}/recipes/storage/efs_simple/assets/main.yaml'
```

```
  FSxLStorage:
    Type: AWS::CloudFormation::Stack
    Properties:
      Parameters:
        PerUnitStorageThroughput: 125
        SubnetId: !GetAtt [ Networking, Outputs.DefaultPrivateSubnet ]
        VpcId: !GetAtt [ Networking, Outputs.VPC ]
      TemplateURL: !Sub 'https://${HpcRecipesS3Bucket}.s3.amazonaws.com/
${HpcRecipesBranch}/recipes/storage/fsx_lustre/assets/persistent.yaml'

  [...]

  # Cluster
  PCSCluster:
    Type: AWS::PCS::Cluster
    Properties:
      Name: !Sub '${AWS::StackName}'
      Size: SMALL
      Scheduler:
        Type: SLURM
        Version: !Ref SlurmVersion
      Networking:
        SubnetIds:
          - !GetAtt [ Networking, Outputs.DefaultPrivateSubnet ]
        SecurityGroupIds:
          - !GetAtt [ PCSSecurityGroup, Outputs.ClusterSecurityGroupId ]
```

For compute resources, the template creates two node groups: `PCSNodeGroupLogin` for a single login node and `PCSNodeGroupCompute` for up to four compute nodes. These node groups are supported by `PCSInstanceProfile` for permissions and `PCSLaunchTemplate` for instance configurations.

```

  # Compute Node groups
  PCSInstanceProfile:
    Type: AWS::CloudFormation::Stack
    Properties:
      Parameters:
        # We have to regionalize this in case CX use the template in more than one
 region. Otherwise,
        # the create action will fail since instance-role-${AWS::StackName} already
 exists!
        RoleName: !Sub '${AWS::StackName}-${AWS::Region}'
      TemplateURL: !Sub 'https://${HpcRecipesS3Bucket}.s3.amazonaws.com/
${HpcRecipesBranch}/recipes/pcs/getting_started/assets/pcs-iip-minimal.yaml'

  PCSLaunchTemplate:
    Type: AWS::CloudFormation::Stack
    Properties:
      Parameters:
        VpcDefaultSecurityGroupId: !GetAtt [ Networking, Outputs.SecurityGroup ]
        ClusterSecurityGroupId: !GetAtt [ PCSSecurityGroup,
 Outputs.ClusterSecurityGroupId ]
        SshSecurityGroupId: !GetAtt [ PCSSecurityGroup,
 Outputs.InboundSshSecurityGroupId ]
        EfsFilesystemSecurityGroupId: !GetAtt [ EfsStorage, Outputs.SecurityGroupId ]
        FSxLustreFilesystemSecurityGroupId: !GetAtt [ FSxLStorage,
 Outputs.FSxLustreSecurityGroupId ]
        SshKeyName: !Ref KeyName
        EfsFilesystemId: !GetAtt [ EfsStorage, Outputs.EFSFilesystemId ]
        FSxLustreFilesystemId: !GetAtt [ FSxLStorage, Outputs.FSxLustreFilesystemId ]
        FSxLustreFilesystemMountName: !GetAtt [ FSxLStorage,
 Outputs.FSxLustreMountName ]
      TemplateURL: !Sub 'https://${HpcRecipesS3Bucket}.s3.amazonaws.com/
${HpcRecipesBranch}/recipes/pcs/getting_started/assets/cfn-pcs-lt-efs-fsxl.yaml'

  # Compute Node groups - Login Nodes
  PCSNodeGroupLogin:
    Type: AWS::PCS::ComputeNodeGroup
    Properties:
      ClusterId: !GetAtt [PCSCluster, Id]
      Name: login
      ScalingConfiguration:
        MinInstanceCount: 1
        MaxInstanceCount: 1
      IamInstanceProfileArn: !GetAtt [ PCSInstanceProfile, Outputs.InstanceProfileArn ]
      CustomLaunchTemplate:
        TemplateId: !GetAtt [ PCSLaunchTemplate, Outputs.LoginLaunchTemplateId ]
        Version: 1
      SubnetIds:
        - !GetAtt [ Networking, Outputs.DefaultPublicSubnet ]
      AmiId: !GetAtt [PcsSampleAmi, AmiId]
      InstanceConfigs:
        - InstanceType: !FindInMap [ Architecture, LoginNodeInstances, !Ref
 NodeArchitecture ]

  # Compute Node groups - Compute Nodes
  PCSNodeGroupCompute:
    Type: AWS::PCS::ComputeNodeGroup
    Properties:
      ClusterId: !GetAtt [PCSCluster, Id]
      Name: compute-1
      ScalingConfiguration:
        MinInstanceCount: 0
        MaxInstanceCount: 4
      IamInstanceProfileArn: !GetAtt [ PCSInstanceProfile, Outputs.InstanceProfileArn ]
      CustomLaunchTemplate:
        TemplateId: !GetAtt [ PCSLaunchTemplate, Outputs.ComputeLaunchTemplateId ]
        Version: 1
      SubnetIds:
        - !GetAtt [ Networking, Outputs.DefaultPrivateSubnet ]
      AmiId: !GetAtt [PcsSampleAmi, AmiId]
      InstanceConfigs:
        - InstanceType: !FindInMap [ Architecture, ComputeNodeInstances, !Ref 
 NodeArchitecture ]
```

Job scheduling is handled through `PCSQueueCompute`.

```
 PCSQueueCompute:
   Type: AWS::PCS::Queue
   Properties:
     ClusterId: !GetAtt [PCSCluster, Id]
     Name: demo
     ComputeNodeGroupConfigurations:
       - ComputeNodeGroupId: !GetAtt [PCSNodeGroupCompute, Id]
```

AMI selection happens automatically through the PcsAMILookupFn Lambda function and related resources.

```
 PcsAMILookupRole:
   Type: AWS::IAM::Role
   [...]

 PcsAMILookupFn:
   Type: AWS::Lambda::Function
   Properties:
     Runtime: python3.12
```

```
      Handler: index.handler
      Role: !GetAtt PcsAMILookupRole.Arn
      Code:
        [...]
      Timeout: 30
      MemorySize: 128

  # Example of using the custom resource to look up an AMI
  PcsSampleAmi:
    Type: Custom::AMILookup
    Properties:
      ServiceToken: !GetAtt PcsAMILookupFn.Arn
      OperatingSystem: 'amzn2'
      Architecture: !FindInMap [ Architecture, AmiArchParameter, !Ref
 NodeArchitecture ]
      SlurmVersion: !Ref SlurmVersion
```

Outputs

The template outputs cluster identification and management URLs through `ClusterId`, `PcsConsoleUrl`, and `Ec2ConsoleUrl`.

```
Outputs:
  ClusterId:
    Description: The Id of the PCS cluster
    Value: !GetAtt [ PCSCluster, Id ]

  PcsConsoleUrl:
    Description: URL to access the cluster in the PCS console
    Value: !Sub
      - https://${ConsoleDomain}/pcs/home?region=${AWS::Region}#/clusters/${ClusterId}
      - { ConsoleDomain: !If [ GovCloud, 'console.amazonaws-us-gov.com', !If [ China,
 'console.amazonaws.cn', !Sub '${AWS::Region}.console.aws.amazon.com']],
          ClusterId: !GetAtt [ PCSCluster, Id ]
        }
    Export:
      Name: !Sub ${AWS::StackName}-PcsConsoleUrl

  Ec2ConsoleUrl:
    Description: URL to access instance(s) in the login node group via Session Manager
    Value: !Sub
```

```
      - https://${ConsoleDomain}/ec2/home?region=
${AWS::Region}#Instances:instanceState=running;tag:aws:pcs:compute-node-group-id=
${NodeGroupLoginId}
      - { ConsoleDomain: !If [ GovCloud, 'console.amazonaws-us-gov.com', !If [ China,
 'console.amazonaws.cn', !Sub '${AWS::Region}.console.aws.amazon.com']],
          NodeGroupLoginId: !GetAtt [ PCSNodeGroupLogin, Id ]
        }
    Export:
      Name: !Sub ${AWS::StackName}-Ec2ConsoleUrl
```

CloudFormation templates to create a sample AWS PCS cluster

AWS Region name	AWS Region	View source	Launch stack
US East (N. Virginia)	us-east-1	Download YAML	Launch Stack
US East (Ohio)	us-east-2	Download YAML	Launch Stack
US West (Oregon)	us-west-2	Download YAML	Launch Stack
Asia Pacific (Singapore)	ap-southeast-1	Download YAML	Launch Stack
Asia Pacific (Sydney)	ap-southeast-2	Download YAML	Launch Stack
Asia Pacific (Tokyo)	ap-northeast-1	Download YAML	Launch Stack
Europe (Frankfurt)	eu-central-1	Download YAML	Launch Stack
Europe (Ireland)	eu-west-1	Download YAML	Launch Stack
Europe (London)	eu-west-2	Download YAML	Launch Stack
Europe (Stockholm)	eu-north-1	Download YAML	Launch Stack

AWS Region name	AWS Region	View source	Launch stack
AWS GovCloud (US-East)	us-gov-east-1	Download YAML	Launch Stack
AWS GovCloud (US-West)	us-gov-west-1	Download YAML	Launch Stack

AWS PCS clusters

An AWS PCS cluster consists of the following components:

- Managed instances of the HPC system scheduler software, such as the Slurm control daemon (`slurmctld`).
- Components that integrate with the HPC system scheduler to provision and manage Amazon EC2 instances.
- Components that integrate with the HPC system scheduler to transmit logs and metrics to Amazon CloudWatch.

These components run in an account managed by AWS. They work together to manage Amazon EC2 instances in your customer account. AWS PCS provisions elastic network interfaces in your Amazon VPC subnet to provide connectivity from the scheduler software to Amazon EC2 instances (for example, to support scheduling batch jobs on them and enabling users to run scheduler commands to list and manage those jobs).

Topics

Creating a cluster in AWS PCS

This topic provides an overview of available options and describes what to consider when you create a cluster in AWS Parallel Computing Service (AWS PCS). If this is your first time creating an AWS PCS cluster, we recommend you follow Get started with AWS Parallel Computing Service. The tutorial can help you create a working HPC system without expanding into all the available options and system architectures that are possible.

> **Note**
>
> After creating a cluster, you can modify many configuration settings without rebuilding your infrastructure. For more information, see Updating a cluster in AWS PCS.

> **Note**
>
> You can configure custom Slurm settings to implement advanced scheduling policies and resource management. For more information, see Configuring custom Slurm settings in AWS PCS.

Prerequisites

- An existing VPC and subnet that meet AWS PCS Networking requirements. Before you deploy a cluster for production use, we recommend that you have a thorough understanding of the VPC and subnet requirements. To create a VPC and subnet, see Creating a VPC for your AWS PCS cluster.
- An IAM principal with permissions to create and manage AWS PCS resources. For more information, see Identity and Access Management for AWS Parallel Computing Service.

Create an AWS PCS cluster

You can use the AWS Management Console or AWS CLI to create a cluster.

AWS Management Console

To create a cluster

1. Open the AWS PCS console at https://console.aws.amazon.com/pcs/home#/clusters and choose **Create cluster**.
2. In the **Cluster setup** section, enter the following fields:
   - **Cluster name** – A name for your cluster. The name can contain only alphanumeric characters (case-sensitive) and hyphens. It must start with an alphabetic character and

can't be longer than 40 characters. The name must be unique within the AWS Region and AWS account that you're creating the cluster in.

- **Scheduler –** Choose a scheduler and version. For more information, see Slurm versions in AWS PCS.
- **Controller size –** Choose a size for your controller. This determines how many concurrent jobs and compute nodes can be managed by the AWS PCS cluster. You can only set the controller size when the cluster is created. For more information on sizing, see Cluster size in AWS PCS.

3. In the **Networking** section, select values for the following fields:

    - **Network type –** Choose the IP address type for your cluster. Your cluster can use either IPv4 or IPv6, but not both. The VPC and subnets must use the same network address type. The IP address block you use for each subnet must have at least 1 available address. AWS reserves some of the addresses in each subnet. For more information, see Subnet CIDR blocks in the *Amazon VPC User Guide*.
    - **VPC –** Choose an existing VPC that meets AWS PCS requirements. For more information, see AWS PCS VPC and subnet requirements and considerations. After you create the cluster, you can't change its VPC. If no VPCs are listed, you must create one first.
    - **Subnet –** All available subnets in the selected VPC are listed. Choose a subnet that meets the AWS PCS subnet requirements. For more information, see AWS PCS VPC and subnet requirements and considerations. We recommend you select a private subnet to avoid exposing your scheduler endpoints to the public internet.
    - **Security groups –** Specify the security group(s) that you want AWS PCS to associate with the network interfaces it creates for your cluster. You must select at least one security group that allows communication between your cluster and its compute nodes. You can select **Quick create a security group** to have AWS PCS create one with the necessary configuration in your selected VPC, or select an existing security group. For more information, see Security group requirements and considerations.
4. (Optional) In the **Slurm accounting configuration** section, you can enable Slurm accounting and set accounting parameters. For more information, see Slurm accounting in AWS PCS.
5. (Optional) In the **Slurm configuration** section, you can add parameter name and value pairs to configure additional Slurm settings. For a complete list of supported parameters, see Custom Slurm settings for AWS PCS clusters.
6. (Optional) Under **Tags**, add any tags to your AWS PCS cluster.

7. Choose **Create cluster**. The **Status** field shows `Creating` while the AWS PCS creates the cluster. This process can take several minutes.

> ⚠ **Important**
>
> There can only be 1 cluster in a `Creating` state per AWS Region per AWS account. AWS PCS returns an error if there is already a cluster in a `Creating` state when you try to create a cluster.

AWS CLI

To create a cluster

1. Create your cluster with the command that follows. Before running the command, make the following replacements:

   - Replace *`region`* with the ID of the AWS Region that you want to create your cluster in, such as `us-east-1`.
   - Replace *`my-cluster`* with a name for your cluster. The name can contain only alphanumeric characters (case-sensitive) and hyphens. It must start with an alphabetic character and can't be longer than 40 characters. The name must be unique within the AWS Region and AWS account where you're creating the cluster.
   - Replace *`25.05`* with any supported version of Slurm.

     > ⓘ **Note**
     >
     > AWS PCS currently supports Slurm 25.05 and 24.11.

   - Replace *`SMALL`* with any supported cluster size. This determines how many concurrent jobs and compute nodes can be managed by the AWS PCS cluster. It can only be set when the cluster is created. For more information on sizing, see Cluster size in AWS PCS.
   - Replace the value for `subnetIds` with your own. We recommend you select a private subnet to avoid exposing your scheduler endpoints to the public internet.
   - Specify the `securityGroupIds` that you want AWS PCS to associate with the network interfaces it creates for your cluster. The security groups must be in the same VPC as the cluster. You must select at least one security group that allows communication

between your cluster and its compute nodes. For more information, see Security group requirements and considerations.

```
aws pcs create-cluster --region region \
    --cluster-name my-cluster \
    --scheduler type=SLURM,version=25.05 \
    --size SMALL \
    --networking subnetIds=subnet-ExampleId1,securityGroupIds=sg-ExampleId1
```

- to use IPv6, add `networkType=IPV6` to the `--networking` configuration.

```
    --networking networkType=IPV6,subnetIds=subnet-
ExampleId1,securityGroupIds=sg-ExampleId1
```

- Optionally, you can add the `--slurm-configration` option to customize the Slurm behavior and specify Slurm configuration options. The following example sets the scale-down idle time to 60 minutes (3600 seconds), enables Slurm accounting, and specifies `slurm.conf` settings as the value for `slurmCustomSettings`. For more information, see Slurm accounting in AWS PCS.

> **Note**
>
> Accounting is supported for Slurm 24.11 or later.

```
aws pcs create-cluster --region region \
    --cluster-name my-cluster \
    --scheduler type=SLURM,version=25.05 \
    --size SMALL \
    --networking subnetIds=subnet-ExampleId1,securityGroupIds=sg-ExampleId1
    --slurm-configuration
 scaleDownIdleTimeInSeconds=3600,accounting='{mode=STANDARD}',slurmCustomSettings='[{p
```

2. It can take several minutes to provision the cluster. You can query the status of your cluster with the following command. Don’t proceed to creating queues or compute node groups until the cluster’s status field is `ACTIVE`.

```
aws pcs get-cluster --region region --cluster-identifier my-cluster
```

Important

There can only be 1 cluster in a `Creating` state per AWS Region per AWS account. AWS PCS returns an error if there is already a cluster in a `Creating` state when you try to create a cluster.

Recommended next steps for your cluster

- Add compute node groups.
- Add queues.
- Enable logging.

Updating a cluster in AWS PCS

AWS PCS lets you update cluster configurations after creation through the UpdateCluster API or console. You can modify cluster settings without rebuilding your infrastructure, which reduces operational overhead and minimizes interruptions.

Benefits of cluster updates

Updating AWS PCS clusters lets you adapt HPC infrastructure to new requirements without service disruption. Configuration changes take minutes instead of the hour or more needed to rebuild clusters. This capability is important for production environments that require minimal downtime and for teams that need to adjust cluster settings as workload patterns change.

Supported configuration changes

You can modify three main categories of settings:

- **Accounting configuration** - Enable or disable managed accounting and configure retention settings.
- **Scale-down behavior** - Adjust the `scaleDownIdleTime` parameter, which controls how long dynamic instances remain idle before AWS PCS automatically terminates them.

- **Slurm custom settings** - Modify any of the supported Slurm settings that apply at the cluster level, including Prolog, Epilog, and SelectTypeParameters.

Limitations

You cannot modify certain configurations after cluster creation. These include:

- Security group configurations
- VPC subnet selection
- Cluster size
- Slurm version
- Cluster name

These settings are foundational to the cluster's architecture and require creating a new cluster to modify them.

Prerequisites for cluster updates

Before updating a cluster, ensure the following conditions are met:

- Cluster must be in `ACTIVE`, `UPDATE_FAILED`, or `SUSPENDED` state
- All associated resources (Queues, Compute Node Groups) must be in `ACTIVE` state
- You must have appropriate IAM permissions for the UpdateCluster operation
- No other update operations can be in progress

Update process and job impact

During an update operation, compute nodes continue to run existing jobs even when the cluster controller becomes briefly unreachable. However, the system cannot accept new job submissions or make scheduling decisions during this period.

You can monitor cluster updates through both the console and API interfaces. The cluster will transition through the following states during an update:

- `UPDATING` - Update in progress

- `ACTIVE` - Update completed successfully
- `UPDATE_FAILED` - Update encountered an error

Billing during updates

Standard hourly charges for your AWS PCS cluster continue during update operations. When you update a cluster to disable accounting, billing for the accounting feature stops as soon as the cluster enters the `UPDATING` state. When enabling accounting, billing doesn't begin until the cluster successfully completes the update and returns to the `ACTIVE` state.

Topics

- Update an AWS PCS cluster
- Frequently asked questions about updating clusters in AWS PCS
- Troubleshooting AWS PCS cluster updates

Update an AWS PCS cluster

Use these steps to modify scheduler settings, accounting configuration, and Slurm custom settings on your cluster. For more information, see Custom Slurm settings for AWS PCS clusters.

Prerequisites

- Cluster must be in `ACTIVE`, `UPDATE_FAILED`, or `SUSPENDED` state
- All associated resources (Queues, Compute Node Groups) must be in `ACTIVE` state
- No other update operations can be in progress

Procedure

AWS Management Console

1. Open the AWS PCS console at https://console.aws.amazon.com/pcs/.
2. In the navigation pane, choose **Clusters**.
3. Select the cluster to update.
4. Choose **Edit**.
5. On the Edit cluster page, modify the desired settings:

- Under **Scheduler configuration**, update **Scale-down idle time** to control how long dynamic instances remain idle before automatic termination.
- Modify **Prolog**, **Epilog**, and **Select-type parameters** settings as needed.
- Enable, disable, or configure retention time for **managed accounting**.
- Under **Additional scheduler settings**, add, edit, or remove **Slurm custom settings**. For more information about supported parameters, see Custom Slurm settings for AWS PCS clusters.

> **Note**
>
> Fields that cannot be edited display as read-only and show their current values.

6. Choose **Update** to submit the changes.
7. Monitor the cluster status, which shows as "Updating" during the process. The status changes when the update completes successfully.

AWS CLI

1. Open a terminal or command prompt.
2. Verify the cluster status using the following command:

```
aws pcs get-cluster --cluster-identifier my-cluster
```

3. Submit an update request using one of the following examples:

- To enable managed accounting:

```
aws pcs update-cluster --cluster-identifier my-cluster \
--slurm-configuration 'accounting={mode=STANDARD}'
```

- To update a Slurm Prolog setting:

```
aws pcs update-cluster --cluster-identifier my-cluster \
--slurm-configuration \
'SlurmCustomSettings=[{parameterName=Prolog,parameterValue="/path/to/
prolog.sh"}]'
```

- To update scale-down idle time:

  ```
  aws pcs update-cluster --cluster-identifier my-cluster \
  --slurm-configuration 'scaleDownIdleTimeInSeconds=300'
  ```

4. Monitor update progress by checking cluster status:

   ```
   aws pcs get-cluster --cluster-identifier my-cluster
   ```

After a successful update request, the command returns the Cluster object with all changes. The cluster status changes from `UPDATING` to `ACTIVE` when complete.

Frequently asked questions about updating clusters in AWS PCS

Get answers to common questions about updating cluster configurations in AWS PCS.

What settings can I modify?

You can modify accounting configuration (enable/disable managed accounting), scale-down behavior (scaleDownIdleTime parameter), and any of the supported Slurm custom settings that apply at the cluster level. You cannot modify security groups, VPC subnets, cluster size, Slurm version, or cluster name.

Can I queue multiple updates?

No. You must wait for the cluster to return to the `ACTIVE` state before submitting another update. All associated resources (Queues, Compute Node Groups) must also be in `ACTIVE` state.

Can I cancel a cluster update operation?

No, you cannot cancel an ongoing cluster update operation.

Can I submit jobs while my cluster is updating?

We recommend that you avoid submitting jobs during cluster updates. The Slurm controller might be unavailable during the update process.

Will my jobs continue to run during cluster updates?

Yes, running jobs continue to execute on compute nodes even when the cluster controller becomes briefly unreachable during the update process. However, job status might not update until the controller becomes available again.

How is billing affected during updates?

Standard hourly charges continue during update operations. When disabling accounting, billing stops when the cluster enters `UPDATING` state. When enabling accounting, billing begins when the cluster successfully returns to `ACTIVE` state.

Troubleshooting AWS PCS cluster updates

This topic helps you identify and resolve common problems that can occur when updating cluster configurations.

Update fails with accounting configuration error

Common cause

The cluster enters `UPDATE_FAILED` state and the error message indicates an accounting configuration issue. This typically occurs when the accounting configuration is incompatible with the current Slurm version or contains invalid settings.

Resolution

Review your accounting settings for compatibility with your cluster's Slurm version and submit a corrected update request with valid configuration parameters.

Update fails with custom settings error

Common cause

The cluster enters `UPDATE_FAILED` state and the error message indicates a Slurm custom settings issue. This occurs when you provide invalid Slurm parameter values or unsupported parameter combinations.

Resolution

Validate your Slurm custom settings against the supported parameters and submit a corrected update request with valid parameter values and combinations.

Cannot submit update request

Common cause

The update button is disabled in the console or the API returns a 400-level error. This occurs when the cluster is not in an appropriate state, associated resources are not active, or there are validation failures in your configuration.

Resolution

Wait for the cluster and all associated resources to reach `ACTIVE` state, then review your configuration for validation errors before resubmitting the update request.

Validation errors

Common cause

The command returns immediately with a 400-level HTTP error and descriptive message. This occurs due to invalid cluster state, resource state, or configuration parameters.

Resolution

Address the specific validation error mentioned in the response and retry the update operation.

Deleting a cluster in AWS PCS

This topic provides an overview of how to delete an AWS PCS cluster.

Considerations when deleting an AWS PCS cluster

- All queues associated with the cluster must be deleted before the cluster can be deleted. For more information, see Deleting a queue in AWS PCS.
- All compute node groups associated with the cluster must be deleted before the cluster can be deleted. For more information, see Deleting a compute node group in AWS PCS.

Delete the cluster

You can use the AWS Management Console or AWS CLI to delete a cluster.

AWS Management Console

To delete a cluster

1. Open the AWS PCS console.
2. Select the cluster to delete.
3. Choose **Delete**.
4. The cluster **Status** field shows `Deleting`. It can take several minutes to complete.

AWS CLI

To delete a cluster

1. Use the following command to delete a cluster, with these replacements:

   - Replace *`region-code`* with the AWS Region your cluster is in.
   - Replace *`my-cluster`* with the name or ID of your cluster.

   ```
   aws pcs delete-cluster --region region-code --cluster-identifier my-cluster
   ```

2. It can take several minutes to delete the cluster. You can check the status of your cluster with the following command.

   ```
   aws pcs get-cluster --region region-code --cluster-identifier my-cluster
   ```

Cluster size in AWS PCS

AWS PCS provides highly available and secure clusters, while automating key tasks such as patching, node provisioning, and updates.

When you create a cluster, you select a size for it based on two factors:

- The number of compute nodes it will manage
- The number of active and queued jobs that you expect to run on the cluster

⚠ **Important**

You can't change the cluster size after you create the cluster. If you need to change the size, you must create a new cluster.

Slurm cluster size	Number of instances managed	Number of active and queued jobs
Small	Up to 32	Up to 256
Medium	Up to 512	Up to 8192
Large	Up to 2048	Up to 16384

Examples

- If your cluster will have up to 24 managed instances and run up to 100 jobs, choose **Small**.
- If your cluster will have up to 24 managed instances and run up to 1000 jobs, choose **Medium**.
- If your cluster will have up to 1000 managed instances and run up to 100 jobs, choose **Large**.
- If your cluster will have up to 1000 managed instances and run up to 10,000 jobs, choose **Large**.

Working with cluster secrets in AWS PCS

As part of creating a cluster, AWS PCS creates a cluster secret that is required to connect to the job scheduler on the cluster. You also create AWS PCS compute node groups, which define sets of instances to launch in response to scaling events. AWS PCS configures instances launched by those compute node groups with the cluster secret so they can connect to the job scheduler. There are cases where you might want to configure Slurm clients manually. Examples include building a persistent login node or setting up a workflow manager with job management capabilities.

AWS PCS stores the cluster secret as a managed secret with the prefix `pcs!` in AWS Secrets Manager. The cost of the secret is included in the charge for using AWS PCS. You can rotate cluster secrets through AWS Secrets Manager to maintain security compliance and remediate potential security compromises.

Topics

Use AWS Secrets Manager to find the cluster secret

AWS Management Console

1. Navigate to the Secrets Manager console.
2. Choose **Secrets**, then search for the `pcs!` prefix.

> **Note**
>
> A AWS PCS cluster secret has a name in the form `pcs!slurm-secret-`*`cluster-id`* where *`cluster-id`* is the AWS PCS cluster ID.

AWS CLI

Each AWS PCS cluster secret is also tagged with `aws:pcs:`*`cluster-id`*. You can get the secret ID for a cluster with the command that follows. Make these substitutions before running the command:

- Replace *`region`* with the AWS Region to create your cluster in, such as `us-east-1`.
- Replace *`cluster-id`* with the ID of the AWS PCS cluster to find the cluster secret for.

```
aws secretsmanager list-secrets \
    --region region  \
    --filters Key=tag-key,Values=aws:pcs:cluster-id \
            Key=tag-value,Values=cluster-id
```

Use AWS PCS to find the cluster secret

You can use the AWS CLI to find the ARN for an AWS PCS cluster secret. Enter the command that follows, making the following substitutions:

- Replace *region* with the AWS Region to create your cluster in, such as `us-east-1`.
- Replace *my-cluster* with the name or identifier for your cluster.

```
aws pcs get-cluster --region region --cluster-identifier my-cluster
```

The following example output is from the `get-cluster` command. You can use `secretArn` and `secretVersion` together to get the secret.

```
{
    "cluster": {
        "name": "get-started",
        "id": "pcs_123456abcd",
        "arn": "arn:aws:pcs:us-east-1:111122223333:cluster/pcs_123456abcd",
        "status": "ACTIVE",
        "createdAt": "2024-12-17T21:03:52+00:00",
        "modifiedAt": "2024-12-17T21:03:52+00:00",
        "scheduler": {
            "type": "SLURM",
            "version": "25.05"
        },
        "size": "SMALL",
        "slurmConfiguration": {
            "authKey": {
                "secretArn": "arn:aws:secretsmanager:us-east-1:111122223333:secret:pcs!
slurm-secret-pcs_123456abcd-a12ABC",
                "secretVersion": "ef232370-d3e7-434c-9a87-ec35c1987f75"
            }
        },
        "networking": {
            "subnetIds": [
                "subnet-0123456789abcdef0"
            ],
            "securityGroupIds": [
                "sg-0123456789abcdef0"
            ]
        },
        "endpoints": [
            {
                "type": "SLURMCTLD",
                "privateIpAddress": "10.3.149.220",
                "port": "6817"
            }
        ]
    }
}
```

Get the Slurm cluster secret

You can use Secrets Manager to get the current base64-encoded version of a Slurm cluster secret The following example uses the AWS CLI. Make the following substitutions before running the command.

- Replace *`region`* with the AWS Region to create your cluster in, such as `us-east-1`.
- Replace *`secret-arn`* with the `secretArn` from an AWS PCS cluster.

```
aws secretsmanager get-secret-value \
    --region region \
    --secret-id 'secret-arn' \
    --version-stage AWSCURRENT \
    --query 'SecretString' \
    --output text
```

For information about how to use the Slurm cluster secret, see Using standalone instances as AWS PCS login nodes.

Permissions

You use an IAM principal to get the Slurm cluster secret. The IAM principal must have permission to read the secret. For more information, see Roles terms and concepts in the *AWS Identity and Access Management User Guide*.

The following sample IAM policy allows access to an example cluster secret.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSecretValueRetrievalAndVersionListing",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:ListSecretVersionIds"
            ],
            "Resource": "arn:aws:secretsmanager:us-east-1:012345678901:secret:pcs!
slurm-secret-s3431v9rx2-FN7tJF"
        }
    ]
}
```

Rotating cluster secrets in AWS PCS

Use AWS Secrets Manager Managed Rotation to rotate cluster secrets in AWS PCS. Regular secret rotation is a security best practice for maintaining strong security posture in HPC environments. This capability enables you to meet industry compliance standards including HIPAA and FedRAMP, which mandate regular credential rotation.

The cluster secret serves dual purposes: authenticating compute nodes joining the cluster and as the JWT key for Slurm REST API authentication. When rotated, both aspects are affected simultaneously.

How cluster secret rotation works

Prepare manually to maintain cluster stability during secret rotation:

1. **Preparation –** Scale all compute node groups to 0 capacity and ensure no jobs are running
2. **Rotation –** Initiate rotation through Secrets Manager console or API
3. **Monitoring –** Track progress through CloudTrail events
4. **Recovery –** Scale compute node groups back to desired capacity

During rotation, your cluster remains in `ACTIVE` state and billing continues normally. The process typically takes a few minutes.

Requirements and limitations

Before rotating cluster secrets, complete these requirements:

- Cluster must be in `ACTIVE` or `UPDATE_FAILED` state

- IAM role must have `secretsmanager:RotateSecret` permission
- All compute node groups must be scaled to 0 capacity
- Stop all jobs before rotation

Limitations:

- Manual preparation required for each rotation
- Existing JWT tokens become invalid and require reissuance
- BYO login nodes require manual secret update after rotation

Topics

- Rotate a cluster secret in AWS PCS
- Frequently asked questions about cluster secret rotation in AWS PCS
- Troubleshooting cluster secret rotation in AWS PCS

Rotate a cluster secret in AWS PCS

Rotate your cluster secret to comply with security requirements and address potential compromises. This process requires putting your cluster into maintenance mode.

Prerequisites

- IAM role with `secretsmanager:RotateSecret` permission
- Cluster in `ACTIVE` or `UPDATE_FAILED` state

Procedure

1. Notify cluster users of the upcoming maintenance window.
2. Put the cluster into maintenance mode by scaling all compute node groups to 0 capacity.

    a. Use the UpdateComputeNodeGroup API to set both minInstanceCount and maxInstanceCount to 0 for all compute node groups.

    b. Wait until all nodes stop.

    c. Optional: Drain scheduler queues with Slurm commands before you terminate capacity for graceful job handling.

3. Initiate rotation through Secrets Manager.
    - **Console method**:
        - Navigate to Secrets Manager, select your cluster secret, and choose **Rotate secret**.
    - **API method**:
        - Use Secrets Manager `rotate-secret` API.
4. Monitor rotation progress.
    a. Track progress through CloudTrail events.
    b. Check `lastRotatedDate` through either the Secrets Manager console or the `secretsmanager:describeSecret` API.
    c. Wait for `RotationSucceeded` or `RotationFailed` CloudTrail event.
5. After successful rotation, restore cluster capacity.
    a. Use the UpdateComputeNodeGroup API to reset node groups to desired min/max capacity.
    b. For AWS PCS-managed login nodes: No additional action required.
    c. For BYO login nodes:
        i. Connect to login nodes.
        ii. Update `/etc/slurm/slurm.key` with the new secret from Secrets Manager.
        iii. Restart the Slurm Auth and Cred Kiosk Daemon (sackd).

Frequently asked questions about cluster secret rotation in AWS PCS

Find answers to common questions about cluster secret rotation in AWS PCS.

What is a cluster secret?

A cluster secret is a secure credential that enables secure communications between the Slurm controller and AWS PCS compute nodes. It also serves as the JSON Web Token (JWT) key for Slurm REST API authentication.

What's the difference between cluster secret and JWT key?

In AWS PCS, the cluster secret and JWT key are the same resource serving different purposes. The cluster secret authenticates Slurm internal communications, while the JWT key signs tokens for REST API authentication. When rotated, both aspects are affected simultaneously.

How long does rotation take?

The rotation process typically takes a few minutes. Your cluster remains in ACTIVE state and billing continues normally during rotation.

Can I schedule automatic rotations?

You can enable scheduled rotation in Secrets Manager. However, the initial release requires manual preparation (scaling node groups to 0) before each rotation.

Will my existing JWT tokens still work after rotation?

No, existing JWT tokens become invalid after rotation. Issue new tokens for REST API clients.

Where can I find my cluster secret?

You can find your cluster secret in the Secrets Manager console or through the AWS PCS console. For detailed instructions, see Use AWS Secrets Manager to find the cluster secret and Use AWS PCS to find the cluster secret.

Why does rotation require scaling node groups to 0?

Rotation requires no running instances to ensure cluster stability during the secret update process. This prevents authentication conflicts between old and new secrets.

What compliance requirements does this feature support?

This feature enables AWS PCS to meet industry compliance standards including HIPAA and FedRAMP, which mandate regular credential rotation as part of their security controls.

Troubleshooting cluster secret rotation in AWS PCS

Cluster secret rotation fails if the environment isn't properly prepared. The most common cause is active instances in your cluster. To prevent failure:

1. Set all node groups to 0 capacity.
2. Wait for nodes to stop.

3. Verify your cluster isn't in these states: `CREATE_FAILED`, `DELETE_FAILED`, `RESUMING`, `SUSPENDING`, or `SUSPENDED`.

If rotation fails:

- A RotationFailed CloudTrail event appears
- The cluster secret remains unchanged
- Check the RotationFailed event in CloudTrail for details
- Complete all preparation steps for successful rotation

AWS PCS compute node groups

An AWS PCS compute node group is a logical collection of nodes (Amazon EC2 instances). These nodes can be used to run computing jobs, as well as to provide interactive, shell-based access to an HPC system. A compute node group consists of rules for creating nodes, including which Amazon EC2 instances types to use, how many instances to run, whether to use Spot Instances or On-demand Instances, which subnets and security groups to use, and how to configure each instance when it launches. When those rules are updated, AWS PCS updates resources associated with the compute node group to match.

Topics

Creating a compute node group in AWS PCS

This topic provides an overview of available options and describes what to consider when you create a compute node group in AWS Parallel Computing Service (AWS PCS). If this is your first time creating a compute node group in AWS PCS, we recommend you follow the tutorial in Get started with AWS Parallel Computing Service. The tutorial can help you create a working HPC system without expanding into all the available options and system architectures that are possible.

> **Note**
>
> You can configure custom Slurm settings on compute node groups to control resource utilization and node-level behaviors. For more information, see Configuring custom Slurm settings in AWS PCS.

> ⚠ **Important**
>
> AWS PCS currently requires a kernel with IPv4 support for local node communication, even when you use AWS PCS in an IPv6-only network. For more information, see Custom Amazon Machine Images (AMIs) for AWS PCS.

Prerequisites

- Sufficient service quotas to launch the desired number of EC2 instances in your AWS Region. You can use the AWS Management Console to check and request increases to your service quotas.
- An existing VPC and subnet(s) that meet AWS PCS networking requirements. We recommend that you thoroughly understand these requirements before you deploy a cluster for production use. For more information, see AWS PCS VPC and subnet requirements and considerations. You can also use a CloudFormation template to create a VPC and subnets. AWS provides an HPC recipe for the CloudFormation template. For more information, see aws-hpc-recipes on GitHub.
- An IAM instance profile with permissions to call the AWS PCS `RegisterComputeNodeGroupInstance` API action and access to any other AWS resources required for your node group instances. For more information, see IAM instance profiles for AWS Parallel Computing Service.
- A launch template for your node group instances. For more information, see Using Amazon EC2 launch templates with AWS PCS.
- To create a compute node group that uses Amazon EC2 **Spot** instances, you must have the **AWSServiceRoleForEC2Spot** service-linked role in your AWS account. For more information, see Amazon EC2 Spot role for AWS PCS.

Create a compute node group in AWS PCS

You can create a compute node group using the AWS Management Console or the AWS CLI.

AWS Management Console

To create your compute node group using the console

1. Open the AWS PCS console.

2. Select the cluster where you want to create a compute node group. Navigate to **Compute node groups** and choose **Create**.

3. In the **Compute node group setup** section, provide a name for your node group. The name can only contain case-sensitive alphanumeric characters and hyphens. It must start with an alphabetic character and can't be longer than 25 characters. The name must be unique within the cluster.

4. Under **Computing configuration**, enter or select these values:

   a. **EC2 launch template** – Select a custom launch template to use for this node group. Launch templates can be used to customize network settings such as subnet, and security groups, monitoring configuration, and instance-level storage. If you don't have a launch template prepared, see Using Amazon EC2 launch templates with AWS PCS to learn how to create one.

   > ⚠ **Important**
   >
   > AWS PCS creates a managed launch template for each compute node group. These are named `pcs-`*`identifier`*`-do-not-delete`. Don't select these when you create or update a compute node group, or the node group won't function correctly.

   b. **EC2 launch template version** – You must select a version of your custom launch template. If you change the version later, you must update the compute node group to detect changes in the launch template. For more information, see Updating an AWS PCS compute node group.

   c. **AMI ID** – if your launch template doesn't include an AMI ID, or if you want to override the value in the launch template, provide an AMI ID here. Note that the AMI used for the node group must be compatible with AWS PCS. You can also select a sample AMI provided by AWS. For more information on this topic, see Amazon Machine Images (AMIs) for AWS PCS.

   d. **IAM instance profile** – Choose an instance profile for the node group. An instance profile grants the instance permissions to access AWS resources and services securely. If you don't have one prepared, you can select **Create a basic profile** to have AWS PCS create one for you with the minimum policy, or see IAM instance profiles for AWS Parallel Computing Service.

e. **Subnets** – Choose one or more subnets in the VPC where your AWS PCS cluster is deployed. If you select multiple subnets, EFA communications won't be available between nodes, and communication between nodes in different subnets might have increased latency. Make sure the subnets you specify here match any that you define in the EC2 launch template.

f. **Instances** – Choose one or more instance types to fulfill scaling requests in the node group. All instance types must have the same processor architecture (x86_64 or arm64) and number of vCPUs. If the instances have GPUs, all instance types must have the same number of GPUs.

g. **Scaling configuration** – Specify the minimum and maximum number of instances for the node group. You can define either a static configuration, where there is a fixed number of nodes running, or a dynamic configuration, where up to the maximum count of nodes can run. For a static configuration, set minimum and maximum to the same, greater than zero number. For a dynamic configuration, set minimum instances to zero and maximum instances to a number greater than zero. AWS PCS doesn't support compute node groups with a mix of static and dynamic instances.

5. (Optional) Under **Additional settings**, specify the following:

   a. **Purchase option** – select On-Demand Instances, Spot Instances, or an existing Capacity Block. Also choose **On-Demand** if you plan to use an On-Demand Capacity Reservation (ODCR). For more information, see Using ODCRs with AWS PCS. Choose **Capacity Block** to use an existing Amazon EC2 Capacity Blocks for ML reservation. For more information, see Using Amazon EC2 Capacity Blocks for ML with AWS PCS.

   b. **Allocation strategy** – if you have selected the Spot purchase option, you can specify how Spot capacity pools are chosen when launching instances in the node group. For more information, see Allocation strategies for Spot Instances in the *Amazon Elastic Compute Cloud User Guide*. This option has no effect if you have selected the On-demand purchase option.

6. (Optional) In the **Slurm custom settings** section, you can add parameter name and value pairs to configure additional Slurm settings. For a complete list of supported parameters, see Custom Slurm settings for AWS PCS compute node groups.

7. (Optional) Under **Tags**, add any tags to your compute node group.

8. Choose **Create compute node group**. The **Status** field shows `Creating` while AWS PCS provisions the node group. This can take several minutes.

Recommended next step

- Add your node group to a queue in AWS PCS to enable it to process jobs.

AWS CLI

To create your compute node group using AWS CLI

Create your queue with the command that follows. Before running the command, make the following replacements:

1. Replace *`region`*with the ID of the AWS Region to create your cluster in, such as `us-east-1`.
2. Replace *`my-cluster`* with the name or `clusterId`of your cluster.
3. Replace *`my-node-group`*with the name for your compute node group. The name can contain only alphanumeric characters (case-sensitive) and hyphens. It must start with an alphabetic character and can't be longer than 25 characters. The name must be unique within the cluster.
4. Replace *`subnet-ExampleID1`* with one or more subnets IDs from your cluster VPC.
5. Replace *`lt-ExampleID1`* with the ID for your custom launch template. If you don't have one prepared, see Using Amazon EC2 launch templates with AWS PCS to learn how to create one.

   > **⚠ Important**
   >
   > AWS PCS creates a managed launch template for each compute node group. These are named `pcs-`*`identifier`*`-do-not-delete`. Don't select these when you create or update a compute node group, or the node group won't function correctly.

6. Replace *`launch-template-version`* with a specific launch template version. AWS PCS associates your node group with that specific version of the launch template.
7. Replace *`arn:InstanceProfile`*with the ARN of your IAM instance profile. If you don't have one prepared, see Using Amazon EC2 launch templates with AWS PCS for guidance.
8. Replace *`min-instances`* and *`max-instances`* with integer values. You can define either a static configuration, where there is a fixed number of nodes running, or a dynamic configuration, where up to the maximum count of nodes can run. For a static configuration, set minimum and maximum to the same, greater than zero number. For a dynamic

configuration, set minimum instances to zero and maximum instances to a number greater than zero. AWS PCS doesn't support compute node groups with a mix of static and dynamic instances.

9. Replace *t3.large* with another instance type. You can add more instance types by specifying a list of `instanceType` settings. For example, *`--instance-configs instanceType=c6i.16xlarge instanceType=c6a.16xlarge`*. All instance types must have the same processor architecture (x86_64 or arm64) and number of vCPUs. If the instances have GPUs, all instance types must have the same number of GPUs.

```
aws pcs create-compute-node-group --region region \
    --cluster-identifier my-cluster \
    --compute-node-group-name my-node-group \
    --subnet-ids subnet-ExampleID1 \
    --custom-launch-template id=lt-ExampleID1,version='launch-template-version' \
    --iam-instance-profile-arn=arn:InstanceProfile \
    --scaling-config minInstanceCount=min-instances,maxInstanceCount=max-instance \
    --instance-configs instanceType=t3.large
```

Example– Creating a compute node group with custom Slurm settings

```
aws pcs create-compute-node-group --region region \
    --cluster-identifier my-cluster \
    --compute-node-group-name my-node-group \
    --subnet-ids subnet-ExampleID1 \
    --custom-launch-template id=lt-ExampleID1,version='launch-template-version' \
    --iam-instance-profile-arn=arn:InstanceProfile \
    --scaling-config minInstanceCount=min-instances,maxInstanceCount=max-instance \
    --instance-configs instanceType=t3.large \
    --slurm-configuration \
    'slurmCustomSettings=[{parameterName=Features,parameterValue="gpu,nvme"}]'
```

For more information, see Custom Slurm settings for AWS PCS compute node groups.

There are several optional configuration settings you can add to the `create-compute-node-group` command.

- You can specify `--amiId` if your custom launch template doesn't include a reference to an AMI, or if you wish to override that value. Note that the AMI used for the node group must be compatible with AWS PCS. You can also select a sample AMI provided by AWS. For more information on this topic, see Amazon Machine Images (AMIs) for AWS PCS.

- Use `--purchase-option` to choose the way AWS PCS purchases EC2 instances for your compute node group. On-Demand is the default.
  - `ONDEMAND` – Use On-Demand Instances. Also choose this option if you plan to use an On-Demand Capacity Reservation (ODCR). For more information, see Using ODCRs with AWS PCS.
  - `SPOT` – Use Spot Instances. If you choose Spot instances, you can also use `--allocation-strategy` to define how AWS PCS chooses Spot capacity pools when it launches instances in the node group. For more information, see Allocation strategies for Spot Instances in the *Amazon Elastic Compute Cloud User Guide*.
  - `CAPACITY_BLOCK` – Use an existing Amazon EC2 Capacity Blocks for ML reservation. For more information, see Using Amazon EC2 Capacity Blocks for ML with AWS PCS.
- It is possible to provide Slurm configuration options for the nodes in the node group using `--slurm-configuration`. You can set the weight (scheduling priority) and real memory. Nodes with lower weights have higher priority, and the units are arbitrary. For more information, see Weight in the Slurm documentation. Real memory sets the size (in GB) of real memory on nodes in the node group. It is meant to be used in conjunction with the `CR_CPU_Memory` option for the cluster in AWS PCS in your Slurm configuration. For more information, see RealMemory in the Slurm documentation.

> **Important**
>
> It can take several minutes to create the compute node group.

You can query the status of your node group with the following command. You won't be able to associate the node group with a queue until its status reaches `ACTIVE`.

```
aws pcs get-compute-node-group --region region \
    --cluster-identifier my-cluster \
    --compute-node-group-identifier my-node-group
```

Updating an AWS PCS compute node group

This topic provides an overview of available options and describes what to consider when you update an AWS PCS compute node group. For information about Slurm custom settings, see Custom Slurm settings for AWS PCS compute node groups.

Options for updating an AWS PCS compute node group

Updating an AWS PCS compute node group enables you to change the properties of instances launched by AWS PCS, as well as the rules for how those instances are launched. For example, you can replace the AMI for node group instances with another one with different software installed on it. Or, you can update security groups to change inbound or outbound network connectivity. You can also change the scaling configuration and the preferred purchase option.

The following node group settings cannot be altered after creation:

- Name
- Instances

Considerations when updating an AWS PCS compute node group

Compute node groups define EC2 instances that are used to process jobs, provide interactive shell access, and other tasks. They are often associated with one or more AWS PCS queues. As you update your compute node group to change its behavior (or that of its nodes), consider the following:

- Changes to compute node group properties become effective when the compute node group status changes from **Updating** to **Active**. New instances launch with the updated properties.
- Updates that don't impact the configuration of specific nodes don't affect running nodes. For example, adding a subnet and changing the allocation strategy.
- If you update the launch template for a compute node group, you must update the compute node group to use the new version.
- To add or remove a security group from nodes in a compute node group, edit its launch template and update the compute node group. New instances launch with the updated set of security groups.
- If you directly edit a security group used by a compute node group, it takes immediate effect on running and future instances.

- If you add or remove permissions from the IAM instance profile used by a compute node group, it takes immediate effect on running and future instances.
- To change the AMI used by a compute node group's instances, update the compute node group (or its launch template) to use the new AMI and wait for AWS PCS to replace the instances.
- AWS PCS replaces existing instances in the node group after a node group update operation. If there are jobs running on a node, those jobs are allowed to complete before AWS PCS replaces the node. Interactive user processes (such as on login node instances) are terminated. Node group status returns to `Active` when AWS PCS marks the instances for replacement, but the actual replacement occurs when the instances are idle.
- If you decrease the maximum number of instances allowed in a compute node group, AWS PCS removes nodes from Slurm to meet the new maximum. AWS PCS terminates running instances associated with the removed Slurm nodes. The running jobs on the removed nodes fail and return to their queues.
- AWS PCS creates a managed launch template for each compute node group. They are named `pcs-`*`identifier`*`-do-not-delete`. Don't select them when you create or update a compute node group, or the node group will not function correctly.
- If you update a compute node group to use **Spot** for its purchase option, you must have the **AWSServiceRoleForEC2Spot** service-linked role in your account. For more information, see Amazon EC2 Spot role for AWS PCS.

To update an AWS PCS compute node group

You can update a node group using the AWS Management Console or the AWS CLI.

AWS Management Console

To update a compute node group

1. Open the AWS PCS console at `https://console.aws.amazon.com/pcs/home#/clusters`
2. Select the cluster where you wish to update a compute node group.
3. Navigate to **Compute node groups**, go to the node group you wish to update, then select **Edit**.
4. In the **Computing configuration**, **Additional settings**, and **Slurm customization** settings sections, update any values except:

- **Instances –** You can't change the instances in a compute node group.

For more information about Slurm custom settings, see Custom Slurm settings for AWS PCS compute node groups.

5. Choose **Update**. The **Status** field will show *Updating* while changes are being applied.

> ⚠ **Important**
>
> Compute node group updates can take several minutes.

AWS CLI

To update a compute node group

1. Update your compute node group with the command that follows. Before running the command, make the following replacements:

   a. Replace *region-code* with the AWS Region that you want to create your cluster in.

   b. Replace *my-node-group* with the name or `computeNodeGroupId` for your compute node group.

   c. Replace *my-cluster* with the name or `clusterId` of your cluster.

```
aws pcs update-compute-node-group --region region-code \
    --cluster-identifier my-cluster \
    --compute-node-group-identifier my-node-group
```

Example– Updating a compute node group with custom Slurm settings

```
aws pcs update-compute-node-group --region region-code \
    --cluster-identifier my-cluster \
    --compute-node-group-identifier my-node-group \
    --slurm-configuration \
    'slurmCustomSettings=[{parameterName=Features,parameterValue="gpu,nvme"}]'
```

For more information, see Custom Slurm settings for AWS PCS compute node groups.

2. Update any node group parameters except for `--instance-configs`. For example, to set a new AMI ID, pass `--amiId my-custom-ami-id` where *`my-custom-ami-id`* is replaced by your AMI of choice.

> **Important**
>
> It can take several minutes to update the compute node group.

You can query the status of your node group with the following command.

```
aws pcs get-compute-node-group --region region-code \
    --cluster-identifier my-cluster \
    --compute-node-group-identifier my-node-group
```

Deleting a compute node group in AWS PCS

This topic provides an overview of available options and describes what to consider when you delete an compute node group in AWS PCS.

Considerations when deleting a compute node group

Compute node groups define EC2 instances that are used to process jobs, provide interactive shell access, and other tasks. They are often associated with one or more AWS PCS queues. Before you delete a compute node group, consider the following:

- Any EC2 instances launched by the compute node group will be terminated. This will cancel jobs that are running on these instances, and terminate running interactive processes.
- You must disassociate the compute node group from all queues before you can delete it. For more information, see Updating an AWS PCS queue.

Delete the compute node group

You can use the AWS Management Console or AWS CLI to delete a compute node group.

AWS Management Console

To delete a compute node group

1. Open the AWS PCS console.
2. Select the cluster of the compute node group.
3. Navigate to **Compute node groups** and select the compute node group to delete.
4. Choose **Delete**.
5. The **Status** field shows `Deleting`. It can take several minutes to complete.

> **Note**
>
> You can use commands native to your scheduler to confirm that the compute node group is deleted. For example, use `sinfo` or `squeue` for Slurm.

AWS CLI

To delete a compute node group

- Use the following command to delete a compute node group, with these replacements:
  - Replace *`region-code`* with the AWS Region your cluster is in.
  - Replace *`my-node-group`* with the name or ID of your compute node group.
  - Replace *`my-cluster`* with the name or ID of your cluster.

  ```
  aws pcs delete-compute-node-group --region region-code \
         --compute-node-group-identifier my-node-group \
         --cluster-identifier my-cluster
  ```

  It can take several minutes to delete the compute node group.

  > **Note**
  >
  > You can use commands native to your scheduler to confirm that the compute node group is deleted. For example, use `sinfo` or `squeue` for Slurm.

Get compute node group details in AWS PCS

You can use the AWS Management Console or AWS CLI to get details about a compute node group, such as its compute node group ID, Amazon Resource Name (ARN), and Amazon Machine Image (AMI) ID. These details are often required values for AWS PCS API actions and configurations.

AWS Management Console

To get compute node group details

1. Open the AWS PCS console.
2. Select the cluster.
3. Choose **Compute node groups**.
4. Choose a compute node group from the list pane.

AWS CLI

To get compute node group details

1. Use the ListClusters API action to find your cluster name or ID.

   ```
   aws pcs list-clusters
   ```

   Example output:

   ```
   {
       "clusters": [
           {
               "name": "get-started-cfn",
               "id": "pcs_abc1234567",
               "arn": "arn:aws:pcs:us-east-1:111122223333:cluster/pcs_abc1234567",
               "createdAt": "2025-04-01T20:11:22+00:00",
               "modifiedAt": "2025-04-01T20:11:22+00:00",
               "status": "ACTIVE"
           }
       ]
   }
   ```

2. Use the ListComputeNodeGroups API action to list the compute node groups in a cluster.

```
aws pcs list-compute-node-groups --cluster-identifier cluster-name-or-id
```

Example call:

```
aws pcs list-compute-node-groups --cluster-identifier get-started-cfn
```

Example output:

```
{
    "computeNodeGroups": [
        {
            "name": "compute-1",
            "id": "pcs_abc123abc1",
            "arn": "arn:aws:pcs:us-east-1:111122223333:cluster/pcs_abc1234567/
computenodegroup/pcs_abc123abc1",
            "clusterId": "pcs_abc1234567",
            "createdAt": "2025-04-01T20:19:25+00:00",
            "modifiedAt": "2025-04-01T20:19:25+00:00",
            "status": "ACTIVE"
        },
        {
            "name": "login",
            "id": "pcs_abc456abc7",
            "arn": "arn:aws:pcs:us-east-1:111122223333:cluster/pcs_abc1234567/
computenodegroup/pcs_abc456abc7",
            "clusterId": "pcs_abc1234567",
            "createdAt": "2025-04-01T20:19:31+00:00",
            "modifiedAt": "2025-04-01T20:19:31+00:00",
            "status": "ACTIVE"
        }
    ]
}
```

3. Use the GetComputeNodeGroup API action to get additional details for a compute node group.

```
aws pcs get-compute-node-group --cluster-identifier cluster-name-or-id --
compute-node-group-identifier compute-node-group-name-or-id
```

Example call:

```
aws pcs get-compute-node-group --cluster-identifier get-started-cfn --compute-
node-group-identifier compute-1
```

Example output:

```
{
    "computeNodeGroup": {
        "name": "compute-1",
        "id": "pcs_abc123abc1",
        "arn": "arn:aws:pcs:us-east-1:111122223333:cluster/pcs_abc1234567/
computenodegroup/pcs_abc123abc1",
        "clusterId": "pcs_abc1234567",
        "createdAt": "2025-04-01T20:19:25+00:00",
        "modifiedAt": "2025-04-01T20:19:25+00:00",
        "status": "ACTIVE",
        "amiId": "ami-0123456789abcdef0",
        "subnetIds": [
            "subnet-abc012345789abc12"
        ],
        "purchaseOption": "ONDEMAND",
        "customLaunchTemplate": {
            "id": "lt-012345abcdef01234",
            "version": "1"
        },
        "iamInstanceProfileArn": "arn:aws:iam::111122223333:instance-profile/
AWSPCS-get-started-cfn-us-east-1",
        "scalingConfiguration": {
            "minInstanceCount": 0,
            "maxInstanceCount": 4
        },
        "instanceConfigs": [
            {
                "instanceType": "c6i.xlarge"
            }
        ]
    }
}
```

Finding compute node group instances in AWS PCS

Each AWS PCS compute node group can launch EC2 instances with shared configurations. You can use EC2 tags to find instances in a compute node group in the AWS Management Console or with the AWS CLI.

AWS Management Console

To find your compute node group instances

1. Open the AWS PCS console.
2. Select the cluster.
3. Choose **Compute node groups**.
4. Find the ID for the login node group you created.
5. Navigate to the EC2 console and choose **Instances**.
6. Search for the instances with the following tag. Replace *node-group-id* with the **ID** (not the name) of your compute node group.

   ```
   aws:pcs:compute-node-group-id=node-group-id
   ```

7. (Optional) You can change the value of **Instance state** in the search field to find instances that are being configured or that were recently terminated.
8. Find the instance ID and IP address for each instance in the list of tagged instances.

AWS CLI

To find your node group instances, use the commands that follow. Before running the commands, make the following replacements:

- Replace *region-code* with the AWS Region of your cluster. Example: `us-east-1`
- Replace *node-group-id* with the **ID** (not the name) of your compute node group. To find the ID of a compute node group, see Get compute node group details in AWS PCS.
- Replace `running` with other instance states such as `pending` or `terminated` to find EC2 instances in other states.

```
aws ec2 describe-instances \
```

```
     --region region-code --filters \
     "Name=tag:aws:pcs:compute-node-group-id,Values=node-group-id" \
     "Name=instance-state-name,Values=running" \
     --query 'Reservations[*].Instances[*].
{InstanceID:InstanceId,State:State.Name,PublicIP:PublicIpAddress,PrivateIP:PrivateIpAddress}
```

The command returns output similar to the following. The value of `PublicIP` is `null` if the instance is in a private subnet.

```
[
    [
        {
            "InstanceID": "i-0123456789abcdefa",
            "State": "running",
            "PublicIP": "18.189.32.188",
            "PrivateIP": "10.0.0.1"
        }
    ]
]
```

Note

If you expect `describe-instances` to return a large number of instances, you must use options for multiple pages. For more information, see DescribeInstances in the *Amazon Elastic Compute Cloud API Reference*.

Using Amazon EC2 launch templates with AWS PCS

In Amazon EC2, a launch template can store a set of preferences so that you don't have to specify them individually when you launch instances. AWS PCS incorporates launch templates as a flexible way to configure compute node groups. When you create a node group, you provide a launch template. AWS PCS creates a derived launch template from it that includes transformations to help ensure it works with the service.

Understanding what the options and considerations are when writing a custom launch template can help you write one for use with AWS PCS. For more information on launch templates, see Launching an Instance from a Launch an instance from a launch template in the *Amazon EC2 User Guide*.

Topics

Overview of launch templates in AWS PCS

There are over 30 parameters available you can include in an EC2 launch template, controlling many aspects of how instances are configured. Most are fully compatible with AWS PCS, but there are some exceptions.

The following parameters of EC2 Launch template will be ignored by AWS PCS as these properties have to be directly managed by the service:

- **Instance type/Specify instance type attributes** (`InstanceRequirements`) – AWS PCS does not support attribute-based instance selection.
- **Instance type** (`InstanceType`) – Specify instance types when you create a node group.
- **Advanced details/IAM instance profile** (`IamInstanceProfile`) – You provide this when you create or update the node group.

- **Advanced details/Disable API termination** (`DisableApiTermination`) – AWS PCS must control the lifecycle of node group instances it launches.
- **Advanced details/Disable API stop** (`DisableApiStop`) – AWS PCS must control the lifecycle of node group instances it launches.
- **Advanced details/Stop – Hibernate behavior** (`HibernationOptions`) – AWS PCS does not support instance hibernation.
- **Advanced details/Elastic GPU** (`ElasticGpuSpecifications`) – Amazon Elastic Graphics reached end of life on January 8, 2024.
- **Advanced details/Elastic inference** (`ElasticInferenceAccelerators`) – Amazon Elastic Inference is no longer available to new customers.
- **AAdvanced details/Specify CPU options/Threads per core** (`ThreadsPerCore`) – AWS PCS sets the number of threads per core to 1.

These parameters have special requirements that support compatibility with AWS PCS:

- **User data**(`UserData`) – This must be multi-part encoded. See Working with Amazon EC2 user data for AWS PCS.
- **Application and OS Images**(`ImageId`) – You can include this. However, if you specify an AMI ID when you create or update the node group, it will override the value in the launch template. The AMI you provide must be compatible with AWS PCS. For more information, see "Amazon Machine Images (AMIs) for AWS PCS.
- **Network settings/Firewall (security groups)**(`SecurityGroups`) – A list of security group names can’t be set in an AWS PCS launch template. You can set a list of security group IDs (`SecurityGroupIds`), unless you define network interfaces in the launch template. Then, you must specify security group IDs for each interface. For more information, see Security groups in AWS PCS.
- **Network settings/Advanced network configuration**(`NetworkInterfaces`) – If you use EC2 instances with a single network card, and don't require any specialized networking configuration, AWS PCS can configure instance networking for you. To configure multiple network cards or to enable Elastic Fabric Adapter on your instances, use `NetworkInterfaces`. Each network interface must have a list of security group IDs under `Groups`. For more information, see Multiple network interfaces in AWS PCS.
- **Advanced details/Capacity reservation**(`CapacityReservationSpecification`) – This can be set, but cannot reference a specific `CapacityReservationId` when working with AWS PCS.

You can, however, reference a capacity reservation group, where that group contains one or more capacity reservations. For more information, see Capacity Reservations in AWS PCS.

Create a basic launch template

You can create a launch template using the AWS Management Console or the AWS CLI.

AWS Management Console

To create a launch template

1. Open the Amazon EC2 console and select **Launch templates**.
2. Choose **Create launch template**.
3. Under **Launch template name and description** enter a unique, distinctive name for **Launch template name**
4. Under Key pair (login) at Key pair name, select the SSH key pair that will be used to log into EC2 instances managed by AWS PCS. This is optional, but recommended.
5. Under **Network settings**, then **Firewall (security groups)**, choose security groups to attach to the network interface. All security groups in the launch template must be from your AWS PCS cluster VPC. At minimum, choose:
    - A security group that allows communication with the AWS PCS cluster
    - A security group that allows communication between EC2 instances launched by AWS PCS
    - (Optional) A security group that allows inbound SSH access to interactive instances
    - (Optional) A security group that allows compute nodes to make outgoing connections to the Internet
    - (Optional) Security group(s) that allow access to networked resources such as shared file systems or a database server.
6. Your new launch template ID will be accessible in the Amazon EC2 console under **Launch templates**. The launch template ID will have the form `lt-0123456789abcdef01`.

Recommended next step

- Use the new launch template to create or update an AWS PCS compute node group.

AWS CLI

To create a launch template

Create your launch template with the command that follows.

- Before running the command, make the following replacements:

  a. Replace *region-code* with the AWS Region where you are working with AWS PCS

  b. Replace *my-launch-template-name* with a name for your template. It must be unique to the AWS account and AWS Region you are using.

  c. Replace *my-ssh-key-name* with name of your preferred SSH key.

  d. Replace *sg-ExampleID1* and *sg-ExampleID2* with security group IDs that allow communication between your EC2 instances and the scheduler and communication between EC2 instances. If you only have one security group that enables all this traffic, you can remove `sg-ExampleID2` and its preceding comma character. You can also add more security group IDs. All security groups you include in the launch template must be from your AWS PCS cluster VPC.

```
aws ec2 create-launch-template --region region-code \
    --launch-template-name my-template-name \
    --launch-template-data '{"KeyName":"my-ssh-key-name","SecurityGroupIds":
 ["sg-ExampleID1","sg-ExampleID2"]}'
```

The AWS CLI will output text resembling the following. The launch template ID is found in `LaunchTemplateId`.

```
{
    "LaunchTemplate": {
        "LatestVersionNumber": 1,
        "LaunchTemplateId": "lt-0123456789abcdef01",
        "LaunchTemplateName": "my-launch-template-name",
        "DefaultVersionNumber": 1,
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "CreateTime": "2019-04-30T18:16:06.000Z"
    }
}
```

Recommended next step

- Use the new launch template to create or update an AWS PCS compute node group.

Working with Amazon EC2 user data for AWS PCS

You can supply EC2 user data in your launch template that `cloud-init` runs when your instances launch. User data blocks with the content type `cloud-config` run before the instance registers with the AWS PCS API, while user data blocks with content type `text/x-shellscript` run after registration completes, but before the Slurm daemon starts. For more information about content types, see the cloud-init documentation.

our user data can perform common configuration scenarios, including but not limited to the following:

- Including users or groups
- Installing packages
- Creating partitions and file systems
- Mounting network file systems

User data in launch templates must be in the MIME multi-part archive format. This is because your user data is merged with other AWS PCS user data that is required to configure nodes in your node group. You can combine multiple user data blocks together into a single MIME multi-part file.

A MIME multi-part file consists of the following components:

- The content type and part boundary declaration: `Content-Type: multipart/mixed; boundary="==BOUNDARY=="`
- The MIME version declaration: `MIME-Version: 1.0`
- One or more user data blocks that contain the following components:
  - The opening boundary that signals the beginning of a user data block: `--==BOUNDARY==`. You must keep the line before this boundary blank.
  - The content type declaration for the block: `Content-Type: text/cloud-config; charset="us-ascii"` or `Content-Type: text/x-shellscript; charset="us-ascii"`. You must keep the line after the content type declaration blank.
  - The content of the user data, such as a list of shell commands or `cloud-config` directives.

- The closing boundary that signals the end of the MIME multi-part file: `--==BOUNDARY==--`. You must keep the line before the closing boundary blank.

> **Note**
>
> If you add user data to a launch template in the Amazon EC2 console, you can paste it in as plain text. Or, you can upload it from a file. If you use the AWS CLI or an AWS SDK, you must first base64 encode the user data and submit that string as the value of the `UserData` parameter when you call CreateLaunchTemplate, as shown in this JSON file.

```
{
    "LaunchTemplateName": "base64-user-data",
    "LaunchTemplateData": {
        "UserData":
 "ewogICAgIkxhdW5jaFRlbXBsYXRlTmFtZSI6ICJpbmNyZWFzZS1jb250YWluZXItdm9sdW..."
    }
}
```

Examples

- Example: Install software from a package repository
- Example: Run scripts from an S3 bucket
- Example: Set global environment variables
- Using network file systems with AWS PCS
- Example: Use an EFS file system as a shared home directory

Example: Install software for AWS PCS from a package repository

Provide this script as the value of `"userData"` in your launch template. For more information, see Working with Amazon EC2 user data for AWS PCS.

This script uses **cloud-config** to install software packages on node group instances at launch. For more information, see the User data formats in the *cloud-init documentation*. This example installs `curl` and `llvm`.

Note

Your instances must be able to connect to their configured package repositories.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

packages:
- python3-devel
- rust
- golang

--==MYBOUNDARY==--
```

Example: Run additional scripts for AWS PCS from an S3 bucket

Provide this script as the value of `"userData"` in your launch template. For more information, see Working with Amazon EC2 user data for AWS PCS.

The following user data script uses **cloud-config** to import a script from an S3 bucket and run it on node group instances at launch. For more information, see the User data formats in the *cloud-init documentation*.

Replace the following values with your own details:

- *`amzn-s3-demo-bucket`* – The name of an S3 bucket your account can read from.
- *`object-key`* – The S3 object key of the script to import. This includes the name of the script and its location in the folder structure of the bucket. For example, `scripts/script.sh`. For more information, see Organizing objects in the Amazon S3 console by using folders in the *Amazon Simple Storage Service User Guide*.
- *`shell`* – The Linux shell to use to run the script, such as `bash`.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="
```

```
--==MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

runcmd:
- aws s3 cp s3://amzn-s3-demo-bucket/object-key /tmp/script.sh
- /usr/bin/shell /tmp/script.sh

--==MYBOUNDARY==--
```

The IAM instance profile for the node group must have access to the bucket. The following IAM policy is an example for the bucket in the user data script above.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

Example: Set global environment variables for AWS PCS

Provide this script as the value of `"userData"` in your launch template. For more information, see Working with Amazon EC2 user data for AWS PCS.

The following example uses `/etc/profile.d` to set global variables on node group instances.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="
```

```
--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
touch /etc/profile.d/awspcs-userdata-vars.sh
echo MY_GLOBAL_VAR1=100 >> /etc/profile.d/awspcs-userdata-vars.sh
echo MY_GLOBAL_VAR2=abc >> /etc/profile.d/awspcs-userdata-vars.sh

--==MYBOUNDARY==--
```

Example: Use an EFS file system as a shared home directory for AWS PCS

Provide this script as the value of `"userData"` in your launch template. For more information, see Working with Amazon EC2 user data for AWS PCS.

This example extends the example EFS mount in Using network file systems with AWS PCS to implement a shared home directory. The contents of /home are backed up before the EFS file system is mounted. The contents are then quickly copied into place on the shared storage after the mount completes.

Replace the following values in this script with your own details:

- *`/mount-point-directory`* – The path on an instance where you want to mount the EFS file system.
- *`filesystem-id`* – The file system ID for the EFS file system.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

packages:
  - amazon-efs-utils

runcmd:
  - mkdir -p /tmp/home
  - rsync -a /home/ /tmp/home
  - echo "filesystem-id:/ /mount-point-directory efs tls,_netdev" >> /etc/fstab
  - mount -a -t efs defaults
  - rsync -a --ignore-existing /tmp/home/ /home
  - rm -rf /tmp/home/

--==MYBOUNDARY==--
```

Example: Enabling passwordless SSH

You can build on the shared home directory example to implement SSH connections between cluster instances using SSH keys. For each user using the shared home file system, run a script that resembles the following:

```
#!/bin/bash

mkdir -p $HOME/.ssh && chmod 700 $HOME/.ssh
touch $HOME/.ssh/authorized_keys
chmod 600 $HOME/.ssh/authorized_keys

if [ ! -f "$HOME/.ssh/id_rsa" ]; then
    ssh-keygen -t rsa -b 4096 -f $HOME/.ssh/id_rsa -N ""
    cat ~/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
fi
```

Note

The instances must use a security group that allows SSH connections between cluster nodes.

Capacity Reservations in AWS PCS

You can reserve Amazon EC2 capacity in a specific Availability Zone and for a specific duration using On-Demand Capacity Reservations or Amazon EC2 Capacity Blocks for ML to make sure that you have the necessary compute capacity available when you need it.

On-Demand Capacity Reservations (ODCRs) let you reserve compute capacity for your Amazon EC2 instances in a specific Availability Zone for any duration. You can create and cancel reservations at any time, with no long-term commitments or upfront payments. ODCRs are ideal when you need flexible capacity reservations that you can modify as your requirements change.

For more information, see On-Demand Capacity Reservations in the *Amazon Elastic Compute Cloud User Guide*.

Amazon EC2 Capacity Blocks for ML let you reserve GPU-based accelerated computing instances for future use, up to 8 weeks in advance. You can reserve blocks of 1-64 instances for durations from 1 day to 6 months. Capacity Blocks are ideal for machine learning workloads that require guaranteed access to GPU capacity at specific times. For more information, see Capacity Blocks for ML in the *Amazon Elastic Compute Cloud User Guide*.

Topics

Using ODCRs with AWS PCS

You can choose how AWS PCS consumes your reserved instances. If you create an **open** ODCR, any matching instances launched by AWS PCS or other processes in your account count against the reservation. With a **targeted** ODCR, only instances launched with the specific reservation ID count against the reservation. For time-sensitive workloads, targeted ODCRs are more common.

You can configure an AWS PCS compute node group to use a targeted ODCR by adding it to a launch template. Here are the steps to do so:

1. Create a targeted On-Demand Capacity Reservation (ODCR) using the Amazon EC2 Create a Capacity Reservation User Guide .
2. Associate the ODCR with a launch template. There are two ways to do that:

   a. **Direct ODCR association:** Reference the ODCR ID directly in the launch template. This approach provides strict capacity control and does not support instance backfilling (If the compute node group requests more instances than available in the ODCR, no additional instances will be launched).

   b. **Capacity Reservation group association:** Add the ODCR to a Capacity Reservation group and reference the group in the launch template. This approach supports instance backfilling, allowing AWS PCS to launch additional On-Demand instances if the reservation capacity is exceeded.
3. Create or update an AWS PCS compute node group to use the launch template. For more information, see AWS PCS Compute Node Groups User Guide.

- Set the `purchaseOption` of the compute node group to `ONDEMAND`.

Example: Reserve and use hpc6a.48xlarge instances with a targeted ODCR

This example command creates a targeted ODCR for 32 hpc6a.48xlarge instances. To launch the reserved instances in a placement group, add `--placement-group-arn` to the command. You can define a stop date with `--end-date` and `--end-date-type`, otherwise the reservation will continue until it is manually terminated.

```
aws ec2 create-capacity-reservation \
    --instance-type hpc6a.48xlarge \
    --instance-platform Linux/UNIX \
    --availability-zone us-east-2a \
    --instance-count 32 \
    --instance-match-criteria targeted
```

The result from this command will be an ARN for the new ODCR. The ODCR ID can be retrieved from the ARN "`arn:aws:ec2:us-east-2:123456789012:capacity-reservation/ODCR-ID`" or by using the Amazon EC2 DescribeCapacityReservations.

Direct ODCR association: Add the ODCR ID to the launch template. Here is an example launch template that references the ODCR ID.

```
{
  "CapacityReservationSpecification": {
    "CapacityReservationTarget": {
      "CapacityReservationId": "cr-1234567890abcdef1"
    }
  }
}
```

Capacity Reservation group association: Create a Capacity Reservation group and add the group to the launch template. The following command creates a Capacity Reservation group named `EXAMPLE-CR-GROUP`.

```
aws resource-groups create-group \
    --name EXAMPLE-CR-GROUP \
    --configuration \
        '{"Type": "AWS::EC2::CapacityReservationPool"}' \
        '{"Type": "AWS::ResourceGroups::Generic", "Parameters": [{"Name": "allowed-
resource-types", "Values": ["AWS::EC2::CapacityReservation"]}]}'
```

The following command adds the ODCR to the Capacity Reservation group.

```
aws resource-groups group-resources --group EXAMPLE-CR-GROUP \
    --resource-arns arn:aws:ec2:us-east-2:123456789012:capacity-reservation/
cr-1234567890abcdef1
```

With the ODCR created and added to a Capacity Reservation group, it can now be connected to an AWS PCS compute node group by adding it to a launch template. Here is an example launch template that references the Capacity Reservation group.

```
{
  "CapacityReservationSpecification": {
    "CapacityReservationResourceGroupArn": "arn:aws:resource-groups:us-
east-2:123456789012:group/EXAMPLE-CR-GROUP"
  }
}
```

Finally, create or update an AWS PCS compute node group to use hpc6a.48xlarge instances and use the launch template that references the ODCR. For a static node group, set minimum and maximum instances to the size of the reservation (32). For a dynamic node group, set the minimum instances to 0 and the maximum to your desired instance size.

This example is a simple implementation of a single ODCR that is provisioned for one compute node group. But, AWS PCS supports many other designs. For example, you can subdivide a large ODCR or Capacity Reservation group among multiple compute node groups. Or, you can use ODCRs that another AWS account has created and shared with yours.

For more information, see On-Demand Capacity Reservations and Capacity Blocks for ML in the *Amazon Elastic Compute Cloud User Guide*.

Using Amazon EC2 Capacity Blocks for ML with AWS PCS

Amazon EC2 Capacity Blocks for ML is an Amazon EC2 purchasing option that enables you to pay in advance to reserve GPU-based accelerated computing instances within a specific date and time range to support short duration workloads. Instances that run inside a Capacity Block are automatically placed close together inside Amazon EC2 UltraClusters, for low-latency, petabitscale, non-blocking networking. For more information, see Capacity Blocks for ML in the *Amazon Elastic Compute Cloud User Guide*.

You can use a launch template to have AWS PCS use a Capacity Block when it launches instances for a compute node group.

Note

AWS PCS introduced support for Capacity Blocks since Slurm version 24.05.

Limitations

- AWS PCS only supports Capacity Blocks with P5en, P5e, P5, and P4d instance families.
- You can only associate a compute node group with 1 Capacity Block at a time.
- You can't associate a compute node group with a capacity reservation group that combines multiple Capacity Blocks.
- Capacity Blocks must be in a `scheduled` or `active` state to use with AWS PCS. You can't use Capacity Blocks in other states, such as `payment-failed`. For more information, see View Capacity Blocks in the *Amazon Elastic Compute Cloud User Guide*.

Capacity Block expiration

Capacity Blocks are limited to a specific date and time range. When a Capacity Block expires:

- The compute node group associated with that Capacity Block continues to exist and remains associated with the same queues.
- All instances in the compute node group are terminated and active jobs might fail, based on your Slurm settings.
- AWS PCS can't launch new instances in the compute node group.
- All queued or newly submitted jobs remain in pending state until another compute node group is attached to the queue or you update the compute node group to use a new launch template that specifies a new Capacity Block.

Configure an AWS PCS compute node group to use a Capacity Block

To associate a Capacity Block with a compute node group

1. Create an Amazon EC2 launch template for AWS PCS that specifies your Capacity Block. For more information about creating a launch template for AWS PCS, see Using Amazon EC2 launch templates with AWS PCS.

   Your launch template must include:

   - The value `MarketType` of `InstanceMarketOptions` must be set to `capacity-block`.
   - A `CapacityReservationSpecification` with a valid `CapacityReservationId`
   - A valid `InstanceType` that matches the instance type of the Capacity Block you purchased.

2. Create a compute node group that uses the launch template. For more information, see Creating a compute node group in AWS PCS. You can also update an existing compute node group to use the launch template. For more information, see Updating an AWS PCS compute node group.

   When you create or update the compute node group:

   - The IAM identity you use to create or update the compute node group must have the following permission:

     ```
     ec2:DescribeCapacityReservations
     ```

     For more information, see Minimum permissions for AWS PCS.
   - The Capacity Block must be in a `scheduled` or `active` state.
   - Set the `purchaseOption` of the compute node group to `CAPACITY_BLOCK`.
   - The `maxInstanceCount` of the compute node group must not exceed the size of the Capacity Block.
   - The availability zone of the compute node group must match 1 of the compute node group's subnet availability zones.

> **Important**
>
> You can't change the instance type of a compute node group when you update it. You can only use a Capacity Block with the same instance type as the compute node group. If you

want to use a Capacity Block with a different instance type, you must create a new compute node group.

Frequently asked questions about using Capacity Blocks with AWS PCS

I just paid for a Capacity Block and immediately attempted to use it with AWS PCS but compute node group creation failed. What happened?

Your Capacity Block might not be in a `scheduled` or `active` state. Try again after the Capacity Block is `scheduled` or `active`.

I am using a Capacity Block in AWS PCS and I purchased an extension before it expired. How do I continue using it in AWS PCS?

You don't have to do anything to continue using the Capacity Block in AWS PCS. The end date of your Capacity Block updates after your extension payment succeeds. As long as your Capacity Block doesn't expire, the compute node group continues to operate. If your extension payment fails, your Capacity Block remains `active` and the compute node group operates until the Capacity Block expires on its original end date.

What happens to my queued and running jobs if my Capacity Block expires?

Queued jobs that didn't start before the Capacity Block expired remain pending until you attach another compute node group to the queue or you update the compute node group with a new Capacity Block. You can still submit jobs to the queue. Your Slurm settings affect active jobs. By default, active jobs are automatically re-queued, but might have errors or fail.

My Capacity Block expired. Should I do something?

You don't have to do anything. You can check the Amazon EC2 console for the status of your EC2 capacity reservations. When a Capacity Block expires, the compute node group associated with that Capacity Block continues to exist and handle the same queues. The compute node group doesn't have any instances to run jobs. You can delete the compute node group or disassociate it from the queues to prevent users from submitting jobs that won't run.

I want to use a new Capacity Block with my AWS PCS compute node group. What should I do?

We recommend you create a new compute node group to use the new Capacity Block. For more information, see Configure an AWS PCS compute node group to use a Capacity Block.

How can I share 1 Capacity Block across clusters and services?

You can split a Capacity Block across multiple clusters and services. For example, to split a Capacity Block with 64 `p5.48xlarge` instances with 20 nodes on PCS-Cluster-1, 16 nodes on PCS-Cluster-2, and the remaining nodes for other services, set both `minInstanceCount` and `maxInstanceCount` to 20 for PCS-Cluster-1 and 16 for PCS-Cluster-2.

Can I use more than 1 Capacity Block or combined capacity with 1 compute node group?

No. Only 1 Capacity Block can be associated with a single compute node group. AWS PCS doesn't support capacity reservation groups that combine multiple Capacity Blocks.

How do I know when my Capacity Blocks start or expire?

Independent from AWS PCS, Amazon EC2 sends a `Capacity Block Reservation Delivered` event through EventBridge when a Capacity Block reservation starts and a `Capacity Block Reservation Expiration Warning` event 40 minutes before the Capacity Block reservation expires. For more information, see Monitor Capacity Blocks using EventBridge in the *Amazon Elastic Compute Cloud User Guide*.

How does Slurm track the state of my Capacity Block?

You can run `sinfo` to understand how AWS PCS uses the Capacity Block. In the following example output, a queue is associated with a compute node group that runs 4 instances from an `active` Capacity Block. The nodes are in the `idle` Slurm state (available for use and not yet allocated to any jobs).

```
$ sinfo
PARTITION AVAIL TIMELIMIT NODES STATE NODELIST
fanout up infinite 4 idle node-fanout-[1-4]
```

If the nodes are instead in `maint` state, you can run `scontrol show res` to see details about the Slurm reservation that controls this state. In the following example output, the Capacity Block is `scheduled` with a future start date.

```
$ scontrol show res

ReservationName=node-fanout-scheduled StartTime=2025-10-14T13:09:17
 EndTime=2025-10-14T13:11:17 Duration=00:02:00
   Nodes=node-fanout-[1-4] NodeCnt=4 CoreCnt=16 Features=(null) PartitionName=(null)
 Flags=MAINT,SPEC_NODES
```

```
   TRES=cpu=16

   Users=root Groups=(null) Accounts=(null) Licenses=(null) State=ACTIVE
 BurstBuffer=(null)
   MaxStartDelay=(null)

   Comment=node-fanout Scheduled
```

How can I tell if the errors I'm getting while launching capacity are because my Capacity Block is shared?

Check **Capacity Reservations** in the Amazon EC2 console to find how many instances from the Capacity Block are actively provisioned. Check the tags of each instance to find which service or cluster uses it. For example, all instances for AWS PCS have AWS PCS tags such as `aws:pcs:cluster-id = pcs_l0mizqyk5o` **|** `aws:pcs:compute-node-group-id = pcs_ic7onkmfqk` that indicate which clusters and compute node groups the instance belongs to. You can then check if the Capacity Block is at maximum capacity.

You use `scontrol show nodes` to check if a Capacity Block node in an AWS PCS cluster is triggering `ReservationCapacityExceeded`:

```
[root@ip-172-16-10-54 ~]# scontrol show nodes test-node-8-gamma-cb-2
NodeName=test-8-gamma-cb-2 CoresPerSocket=1
   CPUAlloc=0 CPUEfctv=8 CPUTot=8 CPULoad=0.00
   AvailableFeatures=test-8-gamma-cb,gpu
   ActiveFeatures=test-8-gamma-cb,gpu
   Gres=gpu:H100:1
   NodeAddr=test-8-gamma-cb-2 NodeHostName=test-8-gamma-cb-2
   RealMemory=249036 AllocMem=0 FreeMem=N/A Sockets=8 Boards=1
   State=IDLE+CLOUD+POWERING_DOWN ThreadsPerCore=1 TmpDisk=0 Weight=1 Owner=N/A
 MCS_label=N/A
   Partitions=my-q
   BootTime=None SlurmdStartTime=None
   LastBusyTime=Unknown ResumeAfterTime=None
   CfgTRES=cpu=8,mem=249036M,billing=8
   AllocTRES=
   CurrentWatts=0 AveWatts=0
   Reason=Failed to launch backing instance (Error Code:
 ReservationCapacityExceeded) [root@2025-08-28T15:15:33]
```

When multiple compute node groups are attached to the same queue, how can I force a job to run on Capacity Block-backed instances?

> You can use Slurm features and constraints to lock a job to a certain set of nodes. We recommend that you don't set Slurm weights for each compute node group because that only works with nodes that aren't in the `maint` state.

Useful launch template parameters

This section describes some launch template parameters that may be broadly useful with AWS PCS.

Turn on detailed CloudWatch monitoring

You can enable collection of CloudWatch metrics at a shorter interval using a launch template parameter.

AWS Management Console

> On the console pages for creating or editing launch templates, this option is found under the **Advanced details** section. Set **Detailed CloudWatch monitoring** to *Enable*.

YAML

```
Monitoring:
    Enabled: True
```

JSON

```
{"Monitoring": {"Enabled": "True"}}
```

For more information, see Enable or turn off detailed monitoring for your instances in the *Amazon Elastic Compute Cloud User Guide for Linux Instances*.

Instance Metadata Service Version 2 (IMDS v2)

Using IMDS v2 with EC2 instances offers significant security enhancements and helps mitigate potential risks associated with accessing instance metadata in AWS environments.

AWS Management Console

On the console pages for creating or editing launch templates, this option is found under the **Advanced details** section. Set **Metadata accessible** to *Enabled*, **Metadata version** to *V2 only (token required)*, and **Metadata response hop limit** to *4*.

YAML

```
MetadataOptions:
  HttpEndpoint: enabled
  HttpTokens: required
  HttpPutResponseHopLimit: 4
```

JSON

```
{
    "MetadataOptions": {
        "HttpEndpoint": "enabled",
        "HttpPutResponseHopLimit": 4,
        "HttpTokens": "required"
    }
}
```

.

AWS PCS queues

An AWS PCS queue is a lightweight abstraction over the scheduler's native implementation of a work queue. In the case of Slurm, an AWS PCS queue is equivalent to a Slurm partition.

Users submit jobs to a queue where they reside until they can be scheduled to run on nodes provided by one or more compute node groups. An AWS PCS cluster can have multiple job queues. For example, you can create a queue that uses Amazon EC2 On-demand Instances for high priority jobs and another queue that uses Amazon EC2 Spot Instances for low-priority jobs.

Topics

Creating a queue in AWS PCS

This topic provides an overview of available options and describes what to consider when you create a queue in AWS PCS.

> **Note**
>
> You can configure custom Slurm settings on queues to implement partition-specific scheduling policies and resource management. For more information, see Configuring custom Slurm settings in AWS PCS.

Prerequisites

- An AWS PCS cluster - queues can only be created in association with a specific AWS PCS cluster.
- One or more AWS PCS compute node groups - a queue must be associated with at least one AWS PCS compute node group.

To create a queue in AWS PCS

You can create a queue using the AWS Management Console or the AWS CLI.

AWS Management Console

To create a queue using the console

1. Open the AWS PCS console.
2. Select the cluster for the queue. Navigate to **Queues** and choose **Create queue**.
3. In the **Queue configuration** section, provide the following values:

   a. **Queue name** – A name for your queue. The name can contain only alphanumeric characters (case-sensitive) and hyphens. It must start with an alphabetic character and can't be longer than 25 characters. The name must be unique within the cluster.

   b. **Compute node groups** – Select 1 or more compute node groups to service this queue. A compute node group can be associated with more than 1 queue.
4. (Optional) In the **Additional scheduler settings** section, you can add parameter name and value pairs to configure additional Slurm settings. For a complete list of supported parameters, see Custom Slurm settings for AWS PCS queues.
5. (Optional) Under **Tags**, add any tags to your AWS PCS queue
6. Choose **Create queue**. The **Status** field will show **Creating** while AWS PCS creates the queue. Queue creation can take several minutes.

Recommended next step

- Submit a job to your new queue.

AWS CLI

To create a queue using AWS CLI

Use the following command to create your queue. Make the following replacements:

1. Replace *`region-code`* with the AWS Region of the cluster. For example, `us-east-1`.
2. Replace *`my-queue`* with the name for your queue. The name can contain only alphanumeric characters (case-sensitive) and hyphens. It must start with an alphabetic character and can't be longer than 25 characters. The name must be unique within the cluster.
3. Replace *`my-cluster`* with the name or ID of your cluster.

4. Replace *compute-node-group-id* with the ID of the compute node group to service the queue. For example, `pcs_abcdef12345`.

> **Note**
>
> When you create a queue, you must provide the ID of the compute node group and not its name.

```
aws pcs create-queue --region region-code \
    --queue-name my-queue \
    --cluster-identifier my-cluster \
    --compute-node-group-configurations \
    computeNodeGroupId=compute-node-group-id
```

Example– Creating a queue with custom Slurm settings

```
aws pcs create-queue --region region-code \
    --queue-name my-queue \
    --cluster-identifier my-cluster \
    --compute-node-group-configurations \
    computeNodeGroupId=compute-node-group-id \
    --slurm-configuration \
    'slurmCustomSettings=[{parameterName=Default,parameterValue=YES}]'
```

For more information, see Custom Slurm settings for AWS PCS queues.

It can take several minutes to create the queue. You can query the status of your queue with the following command. You won't be able to submit jobs to the queue until its status reaches `ACTIVE`.

```
aws pcs get-queue --region region-code \
    --cluster-identifier my-cluster \
    --queue-identifier my-queue
```

Recommended next step

- Submit a job to your new queue

Updating an AWS PCS queue

This topic provides an overview of available options and describes what to consider when you update an AWS PCS queue. For information about Slurm custom settings, see Custom Slurm settings for AWS PCS queues.

Considerations when updating an AWS PCS queue

Queue updates will not impact running jobs but the cluster may not be able to accept new jobs while the queue is being updated.

To update an AWS PCS queue

You can use the AWS Management Console or AWS CLI to update a queue.

AWS Management Console

To update a queue

1. Open the AWS PCS console at `https://console.aws.amazon.com/pcs/home#/clusters`
2. Select the cluster where you wish to update a queue.
3. Navigate to **Queues**, go to the queue wish to update, then select **Edit**.
4. In the queue configuration section, update any of the following values:
   - **Node groups –** Add or remove compute node groups from association with the queue.
   - **Additional scheduler settings –** Add, modify, or remove custom Slurm settings for the queue. For more information, see Custom Slurm settings for AWS PCS queues.
   - **Tags –** Add or remove tags for the queue.
5. Choose **Update**. The **Status** field will show *Updating* while changes are being applied.

> ⚠ **Important**
>
> Queue updates can take several minutes.

AWS CLI

To update a queue

1. Update your queue with the command that follows. Before running the command, make the following replacements:

    a. Replace *region-code* with the AWS Region that you want to create your cluster in.

    b. Replace *my-queue* with the name or `computeNodeGroupId` for your queue.

    c. Replace *my-cluster* with the name or `clusterId` of your cluster.

    d. To change compute node group associations, provide an updated list for `--compute-node-group-configurations`.

    - For example, to add a second compute node group `computeNodeGroupExampleID2`:

    ```
    --compute-node-group-configurations
     computeNodeGroupId=computeNodeGroupExampleID1,computeNodeGroupId=computeNodeGro
    ```

    ```
    aws pcs update-queue --region region-code \
        --queue-identifier my-queue \
        --cluster-identifier my-cluster \
        --compute-node-group-configurations \
        computeNodeGroupId=computeNodeGroupExampleID1
    ```

    Example– Updating a queue with custom Slurm settings

    ```
    aws pcs update-queue --region region-code \
        --queue-identifier my-queue \
        --cluster-identifier my-cluster \
        --slurm-configuration \
        'slurmCustomSettings=[{parameterName=Default,parameterValue=YES}]'
    ```

    For more information, see Custom Slurm settings for AWS PCS queues.

2. It can take several minutes to update the queue. You can query the status of your queue with the following command. You won't be able to submit jobs to the queue until its status reaches `ACTIVE`.

```
aws pcs get-queue --region region-code \
    --cluster-identifier my-cluster \
    --queue-identifier my-queue
```

Recommended next steps

- Submit a job to your updated queue.

Deleting a queue in AWS PCS

This topic provides an overview of how to delete an queue in AWS PCS.

Considerations when deleting a queue

- If there are jobs running in the queue, they will be terminated by the scheduler when the queue is deleted. Pending jobs in the queue will be canceled. Consider waiting for jobs in the queue to finish or manually stop/cancel them using the scheduler's native commands (such as `scancel` for Slurm).

Delete the queue

You can use the AWS Management Console or AWS CLI to delete a queue.

AWS Management Console

To delete a queue

1. Open the AWS PCS console.
2. Select the cluster of the queue.
3. Navigate to **Queues** and select the queue to delete.
4. Choose **Delete**.
5. The **Status** field shows `Deleting`. It can take several minutes to complete.

> **Note**
>
> You can use commands native to your scheduler to confirm that the queue is deleted. For example, use `sinfo` or `squeue` for Slurm.

AWS CLI

To delete a queue

- Use the following command to delete a queue, with these replacements:
  - Replace *`region-code`* with the AWS Region your cluster is in.
  - Replace *`my-queue`* with the name or ID of your queue.
  - Replace *`my-cluster`* with the name or ID of your cluster.

```
aws pcs delete-queue --region region-code \
       --queue-identifier my-queue \
       --cluster-identifier my-cluster
```

  It can take several minutes to delete the queue.

  > **Note**
  >
  > You can use commands native to your scheduler to confirm that the queue is deleted. For example, use `sinfo` or `squeue` for Slurm.

AWS PCS login nodes

An AWS PCS cluster usually needs at least 1 login node to support interactive access and job management. A way to accomplish this is with a static AWS PCS compute node group configured for login node capability. You can also configure a standalone EC2 instance to act as a login node.

Topics

Using an AWS PCS compute node group to provide login nodes

This topic provides an overview of suggested configuration options and describes what to consider when you use an AWS PCS compute node group to provide persistent, interactive access to your cluster.

Creating an AWS PCS compute node group for login nodes

Operationally, this is not much different from creating a regular compute node group. However, there are some key configuration choices make:

- Set a static scaling configuration of at least one EC2 instance in the compute node group.
- Choose on-demand purchase option to avoid having your instance(s) reclaimed.
- Choose an informative name for the compute node group, such as login.
- If you want the login node instance(s) to be accessible outside your VPC, consider using a public subnet.
- If you intend to allow SSH access, the launch template will need have a security group that exposes the SSH port to your choice of IP addresses.
- The IAM instance profile should have only the AWS permissions you want your end users to have. See IAM instance profiles for AWS Parallel Computing Service for details.
- Consider allowing AWS Systems Manager Session Manager to manage your login instances.
- Consider restricting access to the instance AWS credentials to only administrative users

- Select less expensive instance types than for regular compute node groups, since the login node(s) will be running continuously.
- Use the same (or a derivative) AMI as for your other compute node groups to help ensure all instances have the same software installed. For more information about customizing AMIs, see Amazon Machine Images (AMIs) for AWS PCS
- Configure the same network file system (Amazon EFS, Amazon FSx for Lustre, etc.) mounts on your login nodes as on your compute instances. For more information, see Using network file systems with AWS PCS.

Access your login nodes

Once your new compute node group reaches ACTIVE status, you can find the EC2 instance(s) it has created and log into them. For more information, see Finding compute node group instances in AWS PCS.

Updating an AWS PCS compute node group for login nodes

You can update a login node group using UpdateComputeNodeGroup. As part of the node group update process, running instances will be replaced. Note that this will interrupt any active user sessions or processes on the instance. Running or queued Slurm jobs will be unaffected. For more information, see Updating an AWS PCS compute node group.

You can also edit the launch template used by your compute node group. You must use UpdateComputeNodeGroup to apply the updated launch template to the compute node group. New EC2 instances launched in the compute node group use the updated launch template. For more information, see Using Amazon EC2 launch templates with AWS PCS.

Deleting an AWS PCS compute node group for login nodes

You can update a login node group using the **delete compute node group** mechanism in AWS PCS. Running instances will be terminated as part of node group deletion. Please note that this will interrupt any active user sessions or processes on the instance. Running or queued Slurm jobs will be unaffected. For more information, see Deleting a compute node group in AWS PCS.

Using standalone instances as AWS PCS login nodes

You can set up independent EC2 instances to interact with an AWS PCS cluster's Slurm scheduler. This is useful for creating login nodes, workstations, or dedicated workflow management hosts that work with AWS PCS clusters but operate outside of AWS PCS management. To do this, each standalone instance must:

1. Have a compatible Slurm software version installed.
2. Be able to connect to the AWS PCS cluster's Slurmctld endpoint.
3. Have the Slurm Auth and Cred Kiosk Daemon (`sackd`) properly configured with the AWS PCS cluster's endpoint and secret. For more information, see sackd in the Slurm documentation.

This tutorial helps you configure an independent instance that connects to an AWS PCS cluster.

Contents

Step 1 – Retrieve the address and secret for the target AWS PCS cluster

Retrieve details about the target AWS PCS cluster using the AWS CLI with the command that follows. Before running the command, make the following replacements:

- Replace *`region-code`* with the AWS Region where the target cluster is running.
- Replace *`cluster-ident`* with the name or identifier for the target cluster

```
aws pcs get-cluster --region region-code --cluster-identifier cluster-ident
```

The command will return output similar to this example.

```
{
    "cluster": {
        "name": "get-started",
        "id": "pcs_123456abcd",
        "arn": "arn:aws:pcs:us-east-1:111122223333:cluster/pcs_123456abcd",
        "status": "ACTIVE",
        "createdAt": "2024-12-17T21:03:52+00:00",
        "modifiedAt": "2024-12-17T21:03:52+00:00",
        "scheduler": {
            "type": "SLURM",
            "version": "25.05"
        },
        "size": "SMALL",
        "slurmConfiguration": {
            "authKey": {
                "secretArn": "arn:aws:secretsmanager:us-east-1:111122223333:secret:pcs!
slurm-secret-pcs_123456abcd-a12ABC",
                "secretVersion": "ef232370-d3e7-434c-9a87-ec35c1987f75"
            }
        },
        "networking": {
            "subnetIds": [
                "subnet-0123456789abcdef0"
            ],
            "securityGroupIds": [
                "sg-0123456789abcdef0"
            ]
        },
        "endpoints": [
            {
                "type": "SLURMCTLD",
                "privateIpAddress": "10.3.149.220",
                "port": "6817"
            }
        ]
    }
}
```

In this sample, the cluster Slurm controller endpoint has an IP address of `10.3.149.220` and it is running on port `6817`. The `secretArn` will be used in later steps to retrieve the cluster secret. The IP address and port will be used in later steps to configure the `sackd` service.

Step 2 – Launch an EC2 instance

To launch an EC2 instance

1. Open the [Amazon EC2 console](https://console.aws.amazon.com/ec2/).

2. In the navigation pane, choose **Instances**, and then choose **Launch Instances** to open the new launch instance wizard.
3. (Optional) In the **Name and tags** section, provide a name for the instance, such as `PCS-LoginNode`. The name is assigned to the instance as a resource tag (`Name=PCS-LoginNode`).
4. In the **Application and OS Images** section, select an AMI for one of the operating systems supported by AWS PCS. For more information, see Supported operating systems.
5. In the **Instance type** section, select a supported instance type. For more information, see Supported instance types.
6. In the **Key pair** section, select the SSH key pair to use for the instance.
7. In the **Network settings** section:
   - Choose **Edit**.
     i. Select the VPC of your AWS PCS cluster.
     ii. For **Firewall (security groups)**, choose **Select existing security group**.
        A. Select a security group that permits traffic between the instance and the target AWS PCS cluster's Slurm controller. For more information, see Security group requirements and considerations.
        B. (Optional) Select a security group that allows inbound SSH access to your instance.
8. In the **Storage** section, configure storage volumes as needed. Make sure to configure sufficient space to install applications and libraries to enable your use case.
9. Under **Advanced**, choose an IAM role that allows access to the cluster secret. For more information, see Get the Slurm cluster secret.
10. In the **Summary** pane, choose **Launch instance**.

Step 3 – Install Slurm on the instance

When the instance has launched and becomes active, connect to it using your preferred mechanism. Use the Slurm installer provided by AWS to install Slurm onto the instance. For more information, see Slurm installer.

Download the Slurm installer, uncompress it, and use the `installer.sh` script to install Slurm. For more information, see Step 3 – Install Slurm.

Step 4 – Retrieve and store the cluster secret

These instructions require the AWS CLI. For more information, see Install or update to the latest version of the AWS CLI in the *AWS Command Line Interface User Guide for Version 2*.

Store the cluster secret with the following commands.

- Create the configuration directory for Slurm.

```
sudo mkdir -p /etc/slurm
```

- Retrieve, decode, and store the cluster secret. Before running this command, replace *`region-code`* with the Region where the target cluster is running, and replace *`secret-arn`* with the value for `secretArn` retrieved in Step 1.

```
aws secretsmanager get-secret-value \
 --region region-code \
 --secret-id 'secret-arn' \
 --version-stage AWSCURRENT \
 --query 'SecretString' \
 --output text | base64 -d | sudo tee /etc/slurm/slurm.key
```

> **Warning**
>
> In a multiuser environment, any user with access to the instance might be able to fetch the cluster secret if they can access the instance metadata service (IMDS). This, in turn, could allow them to impersonate other users. Consider restricting access to IMDS to root or administrative users only. Alternatively, consider using a different mechanism that doesn't rely on the instance profile to fetch and configure the secret.

- Set ownership and permissions on the Slurm key file.

```
sudo chmod 0600 /etc/slurm/slurm.key
sudo chown slurm:slurm /etc/slurm/slurm.key
```

> **Note**
>
> The Slurm key must be owned by the user and group that the `sackd` service runs as.

Step 5 – Configure the connection to the AWS PCS cluster

To establish a connection to the AWS PCS cluster, launch `sackd` as a system service by following these steps.

> **Note**
>
> If you use Slurm 25.05 or later, you can use a script to set up your login node to connect to multiple clusters instead. For more information, see Connecting a standalone login node to multiple clusters in AWS PCS.

1. Set up the environment file for the `sackd` service with the command that follows. Before running the command, replace *`ip-address`* and *`port`* with the values retrieved from endpoints in Step 1.

```
sudo echo "SACKD_OPTIONS='--conf-server=ip-address:port'" > /etc/sysconfig/sackd
```

2. Create a `systemd` service file for managing the `sackd` process.

```
sudo cat << EOF > /etc/systemd/system/sackd.service
[Unit]
Description=Slurm auth and cred kiosk daemon
After=network-online.target remote-fs.target
Wants=network-online.target
ConditionPathExists=/etc/sysconfig/sackd

[Service]
Type=notify
EnvironmentFile=/etc/sysconfig/sackd
User=slurm
Group=slurm
RuntimeDirectory=slurm
RuntimeDirectoryMode=0755
ExecStart=/opt/aws/pcs/scheduler/slurm-25.05/sbin/sackd --systemd \$SACKD_OPTIONS
ExecReload=/bin/kill -HUP \$MAINPID
KillMode=process
LimitNOFILE=131072
LimitMEMLOCK=infinity
LimitSTACK=infinity
```

```
[Install]
WantedBy=multi-user.target
EOF
```

3. Set ownership of the sackd service file.

   ```
   sudo chown root:root /etc/systemd/system/sackd.service && \
       sudo chmod 0644 /etc/systemd/system/sackd.service
   ```

4. Enable the sackd service.

   ```
   sudo systemctl daemon-reload && sudo systemctl enable sackd
   ```

5. Start the sackd service.

   ```
   sudo systemctl start sackd
   ```

Step 6 – (Optional) Test the connection

Confirm that the sackd service is running. Sample output follows. If there are errors, they will commonly show up here.

```
[root@ip-10-3-27-112 ~]# systemctl status sackd
[x] sackd.service - Slurm auth and cred kiosk daemon
   Loaded: loaded (/etc/systemd/system/sackd.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2024-12-17 16:34:55 UTC; 8s ago
 Main PID: 9985 (sackd)
   CGroup: /system.slice/sackd.service
           ##9985 /opt/aws/pcs/scheduler/slurm-25.05/sbin/sackd --systemd --conf-
server=10.3.149.220:6817

Dec 17 16:34:55 ip-10-3-27-112.ec2.internal systemd[1]: Starting Slurm auth and cred
 kiosk daemon...
Dec 17 16:34:55 ip-10-3-27-112.ec2.internal systemd[1]: Started Slurm auth and cred
 kiosk daemon.
Dec 17 16:34:55 ip-10-3-27-112.ec2.internal sackd[9985]: sackd: running
```

Confirm connections to the cluster are working using Slurm client commands such as sinfo and squeue. Here is example output from sinfo.

```
[root@ip-10-3-27-112 ~]# /opt/aws/pcs/scheduler/slurm-25.05/bin/sinfo
```

```
PARTITION AVAIL TIMELIMIT NODES STATE NODELIST
all up infinite 4 idle~ compute-[1-4]
```

You should also be able to submit jobs. For example, a command similar to this example would launch an interactive job on 1 node in the cluster.

```
/opt/aws/pcs/scheduler/slurm-25.05/bin/srun --nodes=1 -p all --pty bash -i
```

Connecting a standalone login node to multiple clusters in AWS PCS

The `pcs-multi-cluster-login-configure.sh` script provides an automated way to configure multiple Slurm `sackd` daemons on a single standalone login node. It enables the login node to communicate with multiple clusters. The script automates the following operations:

- Uses AWS PCS API actions to get cluster information
- Prompts for the base64-encoded Slurm authentication key
- Creates a Slurm JWKS file with cluster authentication key
- Configures the `sackd` service with cluster endpoints and ports
- Creates a `systemd` service file for a cluster-specific `sackd` daemon
- Generates an activation script for cluster environment setup
- Enables and starts the `sackd` service

> **Note**
>
> This script requires Slurm version 25.05 or later.

Slurm must already be installed on the instance (equivalent to step 3 in the manual process). The instance must be able to reach the target cluster's endpoints. The script performs the equivalent operations of step 4 and step 5 in the manual configuration process. It automatically gets the cluster information, configures the `sackd` service, creates the necessary `systemd` service files, and creates an activation script that users can use to configure their shell environment for cluster interaction.

Topics

- Prerequisites for the AWS PCS multi-cluster login node configuration script
- AWS PCS multi-cluster login node configuration script code
- Using the AWS PCS multi-cluster login node configuration script

Prerequisites for the AWS PCS multi-cluster login node configuration script

System requirements

- Linux OS with `systemd` support
- Root privileges for system configuration

Required commands and packages

- `bash` – Shell interpreter (version 4.0+)
- `curl` – For AWS IMDS v2 metadata retrieval
- `jq` – JSON processor for parsing AWS API responses
- `aws` – AWS CLI v2 to run AWS PCS API actions and for Secrets Manager access
- `systemctl` – `systemd` service management
- `find` – File system search utility
- `grep` – Text pattern matching
- `sed` – Stream editor for text manipulation
- `sort` – Text sorting utility
- `tail` – Displays the last lines of a file
- `mkdir` – Directory creation
- `chmod` – Changes file permissions
- `chown` – Changes file ownership
- `ldconfig` – Dynamic linker configuration

AWS requirements

- An AWS PCS cluster that runs Slurm version 25.05 or later
- AWS credentials configured (through an IAM role, credentials file, or environment variables)
- Permissions for:
  - `pcs:GetCluster`
  - `secretsmanager:GetSecretValue` (if you use an alternate secret)

System users and groups

- The `slurm` user and group must exist on the system

Slurm installation

- Slurm must be installed in the same location as AWS PCS Slurm installer packages:

  ```
  /opt/aws/pcs/scheduler/slurm-version
  ```

AWS PCS multi-cluster login node configuration script code

Save the following source code to a file with the following name:

```
pcs-multi-cluster-login-configure.sh
```

Script source code

```
#!/bin/bash
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

# AWS PCS Multi-Cluster Standalone Login Node Configuration Script
#
# This script configures AWS Parallel Computing Service (PCS) multi-cluster stand alone
 login nodes
# by setting up the Slurm authentication and credential kiosk daemon (sackd)
# for connecting to remote PCS clusters.
#
# Prerequisites:
```

```
# - AWS CLI configured with appropriate permissions
# - Slurm version 25.05 or later
# - Root privileges for system configuration
# - Network connectivity to AWS PCS endpoints


set -eo pipefail

# Function to display usage
usage() {
    echo "Usage: $0 --cluster-identifier <cluster-identifier> [--endpoint-url
 <endpoint-url>]"
    echo "       $0 -h|--help"
}

# Function to display help
help() {
    echo "AWS PCS Multi-Cluster Standalone Login Node Configuration Script"
    echo "==============================================="
    echo
    echo "This script configures multi-cluster standalone login node for AWS Parallel
 Computing Service (PCS)"
    echo "by setting up the Slurm authentication and credential kiosk daemon (sackd)."
    echo
    usage
    echo
    echo "Options:"
    echo "  --cluster-identifier <id>    AWS PCS cluster identifier (required)"
    echo "  --endpoint-url <url>         Custom PCS endpoint URL (optional)"
    echo "  -h, --help                   Show this help message"
    echo
    echo "Examples:"
    echo "  $0 --cluster-identifier my-pcs-cluster"
    echo
    echo "Note: This script requires root privileges and Slurm version 25.05 or later."
}

# Function to retrieve authentication key
get_auth_key() {
    if [ "$ALTERNATE_SECRET_RETRIEVAL" = "true" ]; then
        echo "Retrieving authentication key from AWS Secrets Manager..." >&2
        local auth_key_arn=$(echo "$CLUSTER_INFO" | jq -r
 '.cluster.slurmConfiguration.authKey.secretArn')
```

```
        local auth_key_version=$(echo "$CLUSTER_INFO" | jq -r
 '.cluster.slurmConfiguration.authKey.secretVersion')

        if [ "$auth_key_arn" = "null" ] || [ "$auth_key_version" = "null" ]; then
            echo "Error: Auth key information not found in cluster configuration" >&2
            exit 1
        fi

        if ! aws secretsmanager get-secret-value --secret-id "$auth_key_arn" --version-
id "$auth_key_version" --query SecretString --output text --region "$REGION" 2>/dev/
null; then
            echo "Error: Failed to retrieve auth key from Secrets Manager" >&2
            exit 1
        fi
    else
        echo "Please enter the base64-encoded Slurm authentication key:" >&2
        echo -n "Base64 of the Slurm secret key: " >&2
        local key
        read -rs key
        echo >&2
        echo "$key"
    fi
}

# Function to get next available SACKD port
get_next_sackd_port() {
    local exclude_file="$1"
    local port=6918
    local used_ports=()

    # Get all currently used SACKD ports into an array
    while IFS= read -r line; do
        used_ports+=("$line")
    done < <(find /etc/sysconfig -name "sackd-pcs-*" ! -path "$exclude_file" \
             -exec grep SACKD_PORT= '{}' ';' 2>/dev/null | \
             sed 's/.*SACKD_PORT=//' | sort -n)

    # Loop through used ports to find first available port
    for used_port in "${used_ports[@]}"; do
        if [ "$port" -lt "$used_port" ]; then
            break
        elif [ "$port" -eq "$used_port" ]; then
            ((port++))
        fi
    done

    echo "$port"
}

# Function to configure cluster
configure_cluster() {
    mkdir -p /etc/slurm
    SLURM_JWKS_FILE="/etc/slurm/slurm-${CLUSTER_NAME}.jwks"
    echo '{"keys":
[{"alg":"HS256","kty":"oct","kid":"key-'"${CLUSTER_ID}"'","k":"'"${BASE64_SLURM_KEY}"'"}]}'
 | jq -c '.' > "${SLURM_JWKS_FILE}"

    chmod 0600 "$SLURM_JWKS_FILE"
    chown slurm:slurm "$SLURM_JWKS_FILE"

    SLURM_INSTALL_PATH="/opt/aws/pcs/scheduler/slurm-${SLURM_VERSION}"

    SACKD_RUNTIME_DIRECTORY="/run/slurm-${CLUSTER_NAME}"
    mkdir -p "${SACKD_RUNTIME_DIRECTORY}"
    chown slurm:slurm "${SACKD_RUNTIME_DIRECTORY}"

    mkdir -p /etc/sysconfig
    SACKD_SERVICE_NAME="sackd-pcs-${CLUSTER_NAME}"
    SACKD_SERVICE_ENV="/etc/sysconfig/${SACKD_SERVICE_NAME}"
    SACKD_PORT=$(get_next_sackd_port "$SACKD_SERVICE_ENV")
    cat > "${SACKD_SERVICE_ENV}" << EOF
SACKD_OPTIONS='--conf-server=$ENDPOINTS'
SLURM_SACK_JWKS='$SLURM_JWKS_FILE'
RUNTIME_DIRECTORY='$SACKD_RUNTIME_DIRECTORY'
SACKD_PORT=$SACKD_PORT
EOF

    SACKD_SERVICE_PATH="/etc/systemd/system/${SACKD_SERVICE_NAME}.service"

    cat << EOF > "$SACKD_SERVICE_PATH"
[Unit]
Description=Slurm auth and cred kiosk daemon
After=network-online.target remote-fs.target
Wants=network-online.target
ConditionPathExists=${SACKD_SERVICE_ENV}

[Service]
Type=notify
```

```
EnvironmentFile=${SACKD_SERVICE_ENV}
User=slurm
Group=slurm
RuntimeDirectory=slurm-${CLUSTER_NAME}
RuntimeDirectoryMode=0755
ExecStart=${SLURM_INSTALL_PATH}/sbin/sackd --systemd \$SACKD_OPTIONS
ExecReload=/bin/kill -HUP \$MAINPID
KillMode=process
LimitNOFILE=131072
LimitMEMLOCK=infinity
LimitSTACK=infinity

[Install]
WantedBy=multi-user.target
EOF

    chown root:root "$SACKD_SERVICE_PATH"
    chmod 0644 "$SACKD_SERVICE_PATH"
    systemctl daemon-reload && systemctl enable "$SACKD_SERVICE_NAME"
    systemctl restart "$SACKD_SERVICE_NAME"

    ACTIVATE_SCRIPT="activate-pcs-${CLUSTER_NAME}"
    cat > "$ACTIVATE_SCRIPT" << EOF
# Activate script for Slurm cluster ${CLUSTER_NAME}

# Add Slurm paths
export PATH="${SLURM_INSTALL_PATH}/bin:\$PATH"
export MANPATH="${SLURM_INSTALL_PATH}/share/man:\$MANPATH"
export LD_LIBRARY_PATH="${SLURM_INSTALL_PATH}/lib:\$LD_LIBRARY_PATH"
ldconfig

# Set Slurm configuration
export SLURM_CONF="/run/slurm-${CLUSTER_NAME}/conf/slurm.conf"
export PCS_CLUSTER_NAME="${CLUSTER_NAME}"
export PCS_CLUSTER_IDENTIFIER="${CLUSTER_IDENTIFIER}"
export PCS_CLUSTER_ID="${CLUSTER_ID}"

echo "Activated PCS cluster environment: ${CLUSTER_NAME}"

# Deactivate function
function deactivate-pcs-${CLUSTER_NAME}() {
    export PATH="\$(echo "\$PATH" | sed -e "s|${SLURM_INSTALL_PATH}/bin:||g" -e "s|:${SLURM_INSTALL_PATH}/bin||g" -e "s|^${SLURM_INSTALL_PATH}/bin\$||")"
    export MANPATH="\$(echo "\$MANPATH" | sed -e "s|${SLURM_INSTALL_PATH}/share/man:||
g" -e "s|:${SLURM_INSTALL_PATH}/share/man||g" -e "s|^${SLURM_INSTALL_PATH}/share/man\
$||")"
    export LD_LIBRARY_PATH="\$(echo "\$LD_LIBRARY_PATH" | sed -e "s|
${SLURM_INSTALL_PATH}/lib:||g" -e "s|:${SLURM_INSTALL_PATH}/lib||g" -e "s|^
${SLURM_INSTALL_PATH}/lib\$||")"
    unset SLURM_CONF
    unset PCS_CLUSTER_NAME
    unset PCS_CLUSTER_IDENTIFIER
    unset PCS_CLUSTER_ID
    unset -f deactivate-pcs-${CLUSTER_NAME}
    ldconfig
    echo "Deactivated PCS cluster environment: ${CLUSTER_NAME}"
}

export -f deactivate-pcs-${CLUSTER_NAME}

EOF
}

# Main function
main() {
    # Parse arguments
    CLUSTER_IDENTIFIER=""
    PCS_ENDPOINT_URL=""

    while [ "$1" != "" ]; do
        case $1 in
            --cluster-identifier)
                shift
                CLUSTER_IDENTIFIER="$1"
                ;;
            --endpoint-url)
                shift
                PCS_ENDPOINT_URL="--endpoint-url $1"
                ;;
            -h|--help)
                help
                exit 0
                ;;
            *)
                echo "Invalid argument: $1" >&2
                usage >&2
                exit 1
                ;;
        esac
        shift
    done

    # Validate required arguments
    if [ -z "$CLUSTER_IDENTIFIER" ]; then
        echo "Error: --cluster-identifier is required" >&2
        usage >&2
        exit 1
    fi

    # Validate running as root
    if [ "$EUID" -ne 0 ]; then
        echo "Error: This script must be run as root" >&2
        exit 1
    fi

    # Validate required commands are available
    for cmd in aws jq curl; do
        if ! command -v "$cmd" &> /dev/null; then
            echo "Error: Required command '$cmd' not found" >&2
            exit 1
        fi
    done

    # Get the region name from IMDS v2 with error handling (try IPv6 first, fallback to
 IPv4)
    echo "Retrieving AWS region from instance metadata..."
    # Try IPv6 IMDS endpoint first (fd00:ec2::254) with fast timeout (1s connect, 2s
 total)
    # If IPv6 fails, fallback to IPv4 IMDS endpoint (169.254.169.254)
    IMDS_ENDPOINT="http://[fd00:ec2::254]"
    if ! TOKEN=$(curl -s -X PUT "${IMDS_ENDPOINT}/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600" --connect-timeout 1 --max-time 2 2>/dev/null); then
        IMDS_ENDPOINT="http://169.254.169.254"
        if ! TOKEN=$(curl -s -X PUT "${IMDS_ENDPOINT}/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600" --max-time 5); then
            echo "Error: Failed to retrieve IMDS token. Ensure this script is running
 on an EC2 instance." >&2
            exit 1
        fi
    fi
    if ! REGION=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" "${IMDS_ENDPOINT}/
latest/dynamic/instance-identity/document" --max-time 5 | jq -r '.region'); then
        echo "Error: Failed to retrieve AWS region from instance metadata" >&2
        exit 1
    fi

    echo "Detected AWS region: $REGION"

    # Retrieve cluster information from AWS PCS
    echo "Retrieving cluster information for: $CLUSTER_IDENTIFIER"
    # shellcheck disable=SC2086
    if ! CLUSTER_INFO=$(aws pcs get-cluster --region "$REGION" --cluster-identifier
 "$CLUSTER_IDENTIFIER" $PCS_ENDPOINT_URL 2>/dev/null); then
        echo "Error: Failed to retrieve cluster information. Check cluster identifier
 and AWS permissions." >&2
        exit 1
    fi

    CLUSTER_ID=$(echo "$CLUSTER_INFO" | jq -r '.cluster.id')
    CLUSTER_NAME="$(echo "$CLUSTER_INFO" | jq -r '.cluster.name')"
    SLURM_VERSION=$(echo "$CLUSTER_INFO" | jq -r '.cluster.scheduler.version')
    SLURM_VERSION=${SLURM_VERSION#Slurm_}

    # Check if Slurm version is >= 25.05
    # shellcheck disable=SC2072
    if [[ "$SLURM_VERSION" < "25.05" ]]; then
        echo "Error: This script requires Slurm version 25.05 or later. Found version:
 $SLURM_VERSION" >&2
        exit 1
    fi

    ENDPOINTS=$(echo "$CLUSTER_INFO" | jq -r '.cluster.endpoints[] | select(.type
 == "SLURMCTLD") | (if .privateIpAddress != "" then .privateIpAddress else "["
 + .ipv6Address + "]" end) + ":" + .port' | tr '\n' ',' | sed 's/,$//')

    # Get BASE64_SLURM_KEY
    BASE64_SLURM_KEY=$(get_auth_key)

    if [ -z "$BASE64_SLURM_KEY" ]; then
        echo "Error: base64 Slurm key cannot be empty" >&2
        exit 1
    fi

    configure_cluster
    # Final configuration summary
    echo "========================================"
    echo "Configuration completed successfully!"
    echo "========================================"
    echo "Cluster Name: $CLUSTER_NAME"
    echo "Cluster ID: $CLUSTER_ID"
    echo "Slurm Version: $SLURM_VERSION"
    echo "Service Name: $SACKD_SERVICE_NAME"
    echo "SACKD Port: $SACKD_PORT"
    echo
    echo "To activate this cluster environment, run:"
    echo "  source ./$ACTIVATE_SCRIPT"
    echo
    echo "To deactivate this cluster environment, run:"
    echo "  deactivate-pcs-${CLUSTER_NAME}"
    echo
    echo "To check service status:"
    echo "  systemctl status $SACKD_SERVICE_NAME"
    echo
    echo "To view service logs:"
    echo "  journalctl -u $SACKD_SERVICE_NAME -f"
}

# Exit if being sourced for testing
[[ "${BASH_SOURCE[0]}" != "${0}" ]] && return

# Execute main function
main "$@"
```

Using the AWS PCS multi-cluster login node configuration script

Running the script

To run the configuration script

1. Save the content of the script in a file named:

   ```
   pcs-multi-cluster-login-configure.sh
   ```

2. Make it executable:

```
chmod +x pcs-multi-cluster-login-configure.sh
```

3. Run the script:

```
./pcs-multi-cluster-login-configure.sh --cluster-identifier cluster-name
```

Cluster interaction environments

After successful configuration, the script generates a cluster-specific activation script in the current directory. The script has the name `activate-pcs-cluster-name`. The activation script configures the necessary environment variables and paths to interact with the target cluster.

To activate a cluster environment

- Use the `source` command to run the activation script

```
source ./activate-pcs-cluster-name
```

Example

```
# Activate cluster environment for cluster 'my-cluster'
source ./activate-pcs-my-cluster

# Now you can use Slurm commands
sinfo
squeue
sbatch my-job.sh
```

What the activation script does

- Sets the `SLURM_CONF` environment variable to point to the cluster's configuration.
- Updates the `PATH` to include the cluster's Slurm binaries.
- Configures other necessary Slurm environment variables (`MANPATH`, `LD_LIBRARY_PATH`).
- Sets AWS PCS cluster identification variables.
- Enables seamless interaction with the target AWS PCS cluster.

To deactivate a cluster environment

- Run the deactivation command.

  ```
  deactivate-pcs-cluster-name
  ```

  Example

  ```
  # After activating a cluster
  source ./activate-pcs-my-cluster

  # Work with the cluster
  sinfo

  # Deactivate when done
  deactivate-pcs-my-cluster
  ```

What the deactivate command does

- Restores the original PATH environment variable.
- Unsets cluster-specific Slurm environment variables.
- Returns the shell environment to its pre-activation state.

> **Note**
>
> The activation is session-specific and must be sourced in the shell session where you want to interact with the cluster.

AWS PCS Networking

Your AWS PCS cluster is created in an Amazon VPC. This chapter includes the following topics about networking for your cluster's scheduler and nodes.

Except for choosing a subnet to launch instances in, you must use EC2 launch templates to configure networking for AWS PCS compute node groups. For more information about launch templates, see Using Amazon EC2 launch templates with AWS PCS.

Topics

AWS PCS VPC and subnet requirements and considerations

When you create an AWS PCS cluster, you specify a VPC a subnet in that VPC. This topic provides an overview of AWS PCS specific requirements and considerations for the VPC and subnet(s) that you use with your cluster. If you don't have a VPC to use with AWS PCS, you can create one using an AWS-provided CloudFormation template. For more information about VPCs, see Virtual private clouds (VPC) in the *Amazon VPC User Guide*.

VPC requirements and considerations

When you create a cluster, the VPC that you specify must meet the following requirements and considerations:

- The VPC must have a sufficient number of IP addresses available for the cluster, any nodes, and other cluster resources that you want to create. For more information, see IP addressing for your VPCs and subnets in the *Amazon VPC User Guide*.
- If your cluster uses IPv6:
  - Associate an IPv6 CIDR block with your VPC. For more information, see Create a VPC in the *Amazon VPC User Guide*.

> ⚠ **Important**
>
> Although you can configure your VPC with both IPv4 and IPv6, you can only choose 1 network type for your cluster.

  - Enable **auto-assign IPv6 address** for your subnets.
  - For more information, see:
    - IPv6 on AWS
    - Understanding IPv6 addressing on AWS and designing a scalable addressing plan
- The VPC must have a DNS hostname and DNS resolution support. Otherwise, nodes can't register the customer cluster. For more information, see DNS attributes for your VPC in the *Amazon VPC User Guide*.
- The VPC might require VPC endpoints using AWS PrivateLink to be able to contact the AWS PCS API. For more information, see Connect your VPC to services using AWS PrivateLink in the *Amazon VPC User Guide*.

> ⚠ **Important**
>
> AWS PCS doesn't support a VPC with dedicated instance tenancy. The VPC you use for AWS PCS must use `default` instance tenancy. You can change the instance tenancy for an existing VPC. For more information, see Change the instance tenancy of a VPC in the *Amazon Elastic Compute Cloud User Guide*.

Subnet requirements and considerations

When you create a Slurm cluster, AWS PCS creates an Elastic Network Interface(ENI) in the subnet you specified. This network interface enables communication between the scheduler controller and the customer VPC. The network interface also enables Slurm to communicate with the components deployed in your account. You can only specify the subnet for a cluster at creation time.

Subnet requirements for clusters

The subnet that you specify when you create a cluster must meet the following requirements:

- The subnet must have at least 1 IP address for use by AWS PCS.

- If your cluster uses IPv6, all of the subnets in your cluster must use IPv6.

> ⚠ **Important**
>
> Compute node groups configured with AWS PCS sample AMIs and multiple network interfaces won't work currently if the subnets are only configured to use IPv6. Use dual-stack subnets (IPv4 and IPv6) or IPv4-only subnets instead. For more information, see Using sample Amazon Machine Images (AMIs) with AWS PCS.

- The subnet can't reside in AWS Outposts, AWS Wavelength, or an AWS Local Zone.
- The subnet can be a public or private. We recommend that you specify a private subnet, if possible. A public subnet is a subnet with a route table that includes a route to an internet gateway; a private subnet is a subnet with a route table that doesn't include a route to an internet gateway.

Subnet requirements for nodes

You can deploy nodes and other cluster resources to the subnet you specify when you create your AWS PCS cluster, and to other subnets in the same VPC.

Any subnet that you deploy nodes and cluster resources to must meet the following requirements:

- You must ensure that the subnet has enough available IP addresses to deploy all the nodes and cluster resources.
- If your cluster uses IPv4 and you plan to deploy nodes to a public subnet, that subnet must auto-assign IPv4 public addresses.

> ⓘ **Note**
>
> Instances in a public subnet must use a security group with inbound rules that permit traffic from public IP addresses. Unless you have specific source address restrictions, this means an IPv4 source address of 0.0.0.0/0 or an IPv6 source address of ::/0.

- If the subnet where you deploy nodes to is a private subnet and its route table doesn't include a route to a network address translation (NAT) device (IPv4), add VPC endpoints using AWS PrivateLink to the customer VPC. VPC endpoints are needed for all the AWS services that the nodes contact. The only required endpoint is for AWS PCS to allow the node to call the `RegisterComputeNodeGroupInstance` API action. For more information, see RegisterComputeNodeGroupInstance in the *AWS PCS API Reference*.

- Public or private subnet status doesn't impact AWS PCS; the required endpoints must be reachable.

Creating a VPC for your AWS PCS cluster

You can create an Amazon Virtual Private Cloud (Amazon VPC) for your clusters within AWS Parallel Computing Service (AWS PCS).

Use Amazon VPC to launch VPC resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you might operate in your own data center. However, it comes with the benefits of using the scalable infrastructure of Amazon Web Services. We recommend that you have a thorough understanding of the Amazon VPC service before deploying production VPC clusters. For more information, see What is Amazon VPC? in the author visual mode.*Amazon VPC User Guide*.

An PCS cluster, nodes, and supporting resources (such as file systems and directory services) are deployed within your Amazon VPC. If you want to use an existing Amazon VPC with PCS, it must meet the requirements described in AWS PCS VPC and subnet requirements and considerations . This topic describes how to create a VPC that meets PCS requirements using an AWS–provided CloudFormation template. Once you've deployed a template, you can view the resources created by the template to know exactly what resources it created, and the configuration of those resources.

Prerequisites

To create an Amazon VPC for PCS, you must have the necessary IAM permissions to create Amazon VPC resources. These resources are VPCs, subnets, security groups, route tables and routes, and internet and NAT gateways. For more information, see Create a VPC with a public subnet in the *Amazon VPC User Guide*. To review the full list for Amazon EC2, see Actions, resources, and condition keys for Amazon EC2 in the *Service Authorization Reference*.

Create an Amazon VPC

Create a VPC by copy and pasting the appropriate URL for the AWS Region where you will use PCS. You may also download the CloudFormation template and upload it yourself to the CloudFormation console.

- **US East (N. Virginia) (us-east-1)**

  ```
  https://console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/
  create/review?stackName=hpc-networking&templateURL=https://aws-hpc-recipes.s3.us-
  east-1.amazonaws.com/main/recipes/net/hpc_large_scale/assets/main.yaml
  ```

- **US East (Ohio) (us-east-2)**

  ```
  https://console.aws.amazon.com/cloudformation/home?region=us-east-2#/stacks/
  create/review?stackName=hpc-networking&templateURL=https://aws-hpc-recipes.s3.us-
  east-1.amazonaws.com/main/recipes/net/hpc_large_scale/assets/main.yaml
  ```

- **US West (Oregon) (us-west-2)**

  ```
  https://console.aws.amazon.com/cloudformation/home?region=us-west-2#/stacks/
  create/review?stackName=hpc-networking&templateURL=https://aws-hpc-recipes.s3.us-
  east-1.amazonaws.com/main/recipes/net/hpc_large_scale/assets/main.yaml
  ```

- **Template only**

  ```
  https://aws-hpc-recipes.s3.us-east-1.amazonaws.com/main/recipes/net/hpc_large_scale/
  assets/main.yaml
  ```

To create an Amazon VPC for PCS

1. Open the template in the CloudFormation console.

   > **Note**
   >
   > These are pre-populated in the template so that you can simply leave them as the default values.

2. Under **Provide a stack name**, then **Stack name**, enter `hpc-networking`.
3. Under **parameters**, enter the following details:

   a. Under **VPC**, then **CidrBlock**, enter `10.3.0.0/16`

   b. Under **Subnets A**:

      i. Then **CidrPublicSubnetA**, enter `10.3.0.0/20`

   ii. Then **CidrPrivateSubnetA**, enter `10.3.128.0/20`

c. Under **Subnets B**:

   i. Then **CidrPublicSubnetB**, enter `10.3.16.0/20`

   ii. Then **CidrPrivateSubnetA**, enter `10.3.144.0/20`

d. Under **Subnets C**:

   i. For **ProvisionSubnetsC**, select `True`.

   > **Note**
   >
   > If you are creating a VPC in a Region that has less than three Availability Zones, this option will be ignored if set to `True`.

   ii. Then **CidrPublicSubnetB**, enter `10.3.32.0/20`

   iii. Then **CidrPrivateSubnetA**, enter `10.3.160.0/20`

4. Under **Capabilities**, check the box for **I acknowledge that AWS CloudFormation might create IAM resources**.

Monitor the status of the CloudFormation stack. When it reaches `CREATE_COMPLETE`, the VPC resource are ready for you to use.

> **Note**
>
> To see all the resources the CloudFormation template created, open the CloudFormation console. Choose the `hpc-networking` stack and then choose the **Resources** tab.

Security groups in AWS PCS

Security groups in Amazon EC2 act as virtual firewalls to control inbound and outbound traffic to instances. Use a launch template for an AWS PCS compute node group to add or remove security groups to its instances. If your launch template doesn't contain any network interfaces, use `SecurityGroupIds` to provide a list of security groups. If your launch template defines network interfaces, you must use the `Groups` parameter to assign security groups to each network interface. For more information about launch templates, see Using Amazon EC2 launch templates with AWS PCS.

Note

Changes to the security group configuration in the launch template only affects new instances launched after the compute node group is updated.

Security group requirements and considerations

AWS PCS creates a cross-account Elastic Network Interface (ENI) in the subnet you specify when creating a cluster. This provides the HPC scheduler, which is running in an account managed by AWS, a path to communicate with EC2 instances launched by AWS PCS. You must provide a security group for that ENI that allows 2-way communication between the scheduler ENI and your cluster EC2 instances.

A straightforward way to accomplish this is to create a permissive self-referencing security group that permits TCP/IP traffic on all ports between all members of the group. You can attach this to both the cluster and to node group EC2 instances.

Example permissive security group configuration

IPv4

Rule type	Protocols	Ports	Source	Destination
Inbound	All	All	Self	
Outbound	All	All		0.0.0.0/0
Outbound	All	All		Self

IPv6

Rule type	Protocols	Ports	Source	Destination
Inbound	All	All	Self	
Outbound	All	All		::/0

Rule type	Protocols	Ports	Source	Destination
Outbound	All	All		Self

These rules allow all traffic to flow freely between the Slurm controller and nodes, allows all outbound traffic to any destination, and enables EFA traffic.

Example restrictive security group configuration

You can also limit the open ports between the cluster and its compute nodes. For the Slurm scheduler, the security group attached to your cluster must allow the following ports:

- 6817 – enable inbound connections to `slurmctld` from EC2 instances
- 6818 – enable outbound connections from `slurmctld` to `slurmd` running on EC2 instances

The security group attached to your compute nodes must allow the following ports:

- 6817 – enable outbound connections to `slurmctld` from EC2 instances.
- 6818 – enable inbound and outbound connections to `slurmd` from `slurmctld` and from `slurmd` on node group instances
- 60001–63000 – inbound and outbound connections between node group instances to support `srun`
- EFA traffic between node group instances. For more information, see Prepare an EFA-enabled security group in the *User Guide for Linux Instances*
- Any other inter-node traffic required by your workload

Multiple network interfaces in AWS PCS

Some EC2 instances have multiple network cards. This allows them to provide higher network performance, including bandwidth capabilities above 100 Gbps and improved packet handling. For more information about instances with multiple network cards, see Elastic network interfaces in the *Amazon Elastic Compute Cloud User Guide*.

Configure additional network cards for instances in an AWS PCS compute node group by adding network interfaces to its EC2 launch template. Below is an example launch template that enables two network cards, such as can be found on an `hpc7a.96xlarge` instance. Note the following details:

- The subnet for each network interface must be the same as you choose when configuring the AWS PCS compute node group that will use the launch template.
- The primary network device, where routine network communication such as SSH and HTTPS traffic will occur, is established by setting a `DeviceIndex` of `0`. Other network interfaces have a `DeviceIndex` of `1`. There can only be one primary network interface—all other interfaces are secondary.
- All network interfaces must have a unique `NetworkCardIndex`. A recommended practice is to number them sequentially as they are defined in the launch template.
- Security groups for each network interface are set using `Groups`. In this example, an inbound SSH security group (`sg-SshSecurityGroupId`) is added to the primary network interface, as well as the security group enabling within-cluster communications (`sg-ClusterSecurityGroupId`). Finally, a security group allowing outbound connections to the internet (`sg-InternetOutboundSecurityGroupId`) is added to both primary and secondary interfaces.

```
{
    "NetworkInterfaces": [
        {
            "DeviceIndex": 0,
            "NetworkCardIndex": 0,
            "SubnetId": "subnet-SubnetId",
            "Groups": [
               "sg-SshSecurityGroupId",
               "sg-ClusterSecurityGroupId",
               "sg-InternetOutboundSecurityGroupId"
            ]
        },
        {
            "DeviceIndex": 1,
            "NetworkCardIndex": 1,
            "SubnetId": "subnet-SubnetId",
            "Groups": ["sg-InternetOutboundSecurityGroupId"]
        }
    ]
}
```

Placement groups for EC2 instances in AWS PCS

You can use a **placement group** to influence the placement of EC2 instances to suit the needs of the workload that runs on them.

Placement group types

- **Cluster** – Packs instances close together in an Availability Zone to optimize for low-latency communication.
- **Partition** – Spreads instances across logical partitions to help maximize resilience.
- **Spread** – Strictly enforces that a small number of instances launch on distinct hardware, which can also help with resiliency.

For more information, see Placement groups for your Amazon EC2 instances in the *Amazon Elastic Compute Cloud User Guide*.

We recommended you include a **cluster** placement group when you configure an AWS PCS compute node group to use Elastic Fabric Adapter (EFA).

To create a cluster placement group that works with EFA

1. Create a placement group with the type **cluster** for the compute node group.
   - Use the following AWS CLI command:

     ```
     aws ec2 create-placement-group --strategy cluster --group-name PLACEMENT-GROUP-NAME
     ```

   - You can also use a CloudFormation template to create a placement group. For more information, see Working with CloudFormation templates in the *AWS CloudFormation User Guide*. Download the template from the following URL and upload it into the CloudFormation console.

     ```
     https://aws-hpc-recipes.s3.amazonaws.com/main/recipes/pcs/enable_efa/assets/efa-placement-group.yaml
     ```

2. Include the placement group in the EC2 launch template for the AWS PCS compute node group.

Using Elastic Fabric Adapter (EFA) with AWS PCS

Elastic Fabric Adapter (EFA) is a high performance advanced networking interconnect from AWS that you can attach to your EC2 instance to accelerate High Performance Computing (HPC) and machine learning applications. Enabling your applications running on an AWS PCS cluster with EFA involves configuring the AWS PCS compute node group instances to use EFA as follows.

> **Note**
>
> **Install EFA on an AWS PCS-compatible AMI** – The AMI used in the AWS PCS compute node group must have the EFA driver installed and loaded. For information on how to build a custom AMI with EFA software installed, see Custom Amazon Machine Images (AMIs) for AWS PCS.

Contents

Identify EFA-enabled EC2 instances

To use EFA, all instance types that are allowed for an AWS PCS compute group must support EFA, and must have the same number of vCPUs (and GPUs if appropriate). For a list of EFA-enabled instances, see Elastic Fabric Adapter for HPC and ML workloads on Amazon EC2 in the *Amazon Elastic Compute Cloud User Guide*. You can also use the AWS CLI to view a list of instance types that support EFA. Replace *region-code* with the AWS Region where you use AWS PCS, such as `us-east-1`.

```
aws ec2 describe-instance-types \
   --region region-code \
   --filters Name=network-info.efa-supported,Values=true \
   --query "InstanceTypes[*].[InstanceType]" \
   --output text | sort
```

> **Note**
>
> **Determine how many network interfaces are available** – Some EC2 instances have multiple network cards. This allows them to have multiple EFAs. For more information, see Multiple network interfaces in AWS PCS.

Create a security group to support EFA communications

AWS CLI

You can use the following AWS CLI command to create a security group that supports EFA. The command outputs a security group ID. Make the following replacements:

- *region-code* – Specify the AWS Region where you use AWS PCS, such as `us-east-1`.
- *vpc-id* – Specify the ID of the VPC that you use for AWS PCS.
- *efa-group-name* – Provide your chosen name for the security group.

```
aws ec2 create-security-group \
    --group-name efa-group-name \
    --description "Security group to enable EFA traffic" \
    --vpc-id vpc-id \
    --region region-code
```

Use the following commands to attach inbound and outbound security group rules. Make the following replacement:

- *efa-secgroup-id* – Provide the ID of the EFA security group you just created.

```
aws ec2 authorize-security-group-ingress \
    --group-id efa-secgroup-id \
    --protocol -1 \
    --source-group efa-secgroup-id
```

```
aws ec2 authorize-security-group-egress \
    --group-id efa-secgroup-id \
    --protocol -1 \
    --source-group efa-secgroup-id
```

CloudFormation template

You can use a CloudFormation template to create a security group that supports EFA. Download the template from the following URL, then upload it into the AWS CloudFormation console.

```
https://aws-hpc-recipes.s3.amazonaws.com/main/recipes/pcs/enable_efa/assets/efa-
sg.yaml
```

With the template open in the AWS CloudFormation console, enter the following options.

- Under **Provide a stack name**
  - Under **Stack name**, enter a name such as `efa-sg-stack`.
- Under **Parameters**
  - Under **SecurityGroupName**, enter a name such as `efa-sg`.
  - Under **VPC**, select the VPC where you will use AWS PCS.

Finish creating the CloudFormation stack and monitor its status. When it reaches `CREATE_COMPLETE` the EFA security group is ready for use.

(Optional) Create a placement group

We recommended you launch all instances that use EFA in a cluster placement group to minimize the physical distance between them. Create a placement group for each compute node group where you plan to use EFA. See Placement groups for EC2 instances in AWS PCS to create a placement group for your compute node group.

Create or update an EC2 launch template

EFA network interfaces are set up in the EC2 launch template for an AWS PCS compute node group. If there are multiple network cards, multiple EFAs can be configured. The EFA security group and the optional placement group are included in the launch template as well.

Here is an example launch template for instances with two network cards, such as **hpc7a.96xlarge**. The instances will be launched in `subnet-SubnetID1` in cluster placement group `pg-PlacementGroupId1`.

Security groups must be added specifically to each EFA interface. Every EFA needs the security group that enables EFA traffic (`sg-EfaSecGroupId`). Other security groups, especially ones that handle regular traffic like SSH or HTTPS, only need to be attached to the primary network interface (designated by a `DeviceIndex` of `0`). Launch templates where network interfaces are defined do not support setting security groups using the `SecurityGroupIds` parameter—you must set a value for `Groups` in each network interface that you configure.

```
{
    "Placement": {
        "GroupId": "pg-PlacementGroupId1"
    },
    "NetworkInterfaces": [
        {
            "DeviceIndex": 0,
            "InterfaceType": "efa",
            "NetworkCardIndex": 0,
            "SubnetId": "subnet-SubnetId1",
            "Groups": [
                "sg-SecurityGroupId1",
                "sg-EfaSecGroupId"
            ]
        },
        {
            "DeviceIndex": 1,
            "InterfaceType": "efa",
            "NetworkCardIndex": 1,
            "SubnetId": "subnet-SubnetId1"
            "Groups": ["sg-EfaSecGroupId"]
        }
    ]
}
```

Create or update compute node groups for EFA

Your AWS PCS compute node groups must contain instances that have the same number of vCPUs, processor architecture, and EFA support. Configure the compute node group to use the AMI with the EFA software installed on it, and to use the launch template that configures EFA-enabled network interfaces.

(Optional) Test EFA

You can demonstrate EFA-enabled communication between two nodes in a compute node group by running the `fi_pingpong` program, which is included in the EFA software installation. If this test is successful, it is likely that EFA is configured properly.

To start, you need two running instances in the compute node group. If your compute node group uses static capacity, there should be already be instances available. For a compute node group that uses dynamic capacity, you can launch two nodes using the `salloc` command. Here is an example from a cluster with a dynamic node group named `hpc7g` associated with a queue named `all`.

```
% salloc --nodes 2 -p all
salloc: Granted job allocation 6
salloc: Waiting for resource configuration
... a few minutes pass ...
salloc: Nodes hpc7g-[1-2] are ready for job
```

Find out the IP address for the two allocated nodes using `scontrol`. In the example that follows, the addresses are `10.3.140.69` for `hpc7g-1` and `10.3.132.211` for `hpc7g-2`.

```
% scontrol show nodes hpc7g-[1-2]
NodeName=hpc7g-1 Arch=aarch64 CoresPerSocket=1
   CPUAlloc=0 CPUEfctv=64 CPUTot=64 CPULoad=0.00
   AvailableFeatures=hpc7g
   ActiveFeatures=hpc7g
   Gres=(null)
   NodeAddr=10.3.140.69 NodeHostName=ip-10-3-140-69 Version=25.05.4
   OS=Linux 5.10.218-208.862.amzn2.aarch64 #1 SMP Tue Jun 4 16:52:10 UTC 2024
   RealMemory=124518 AllocMem=0 FreeMem=110763 Sockets=64 Boards=1
   State=IDLE+CLOUD ThreadsPerCore=1 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
   Partitions=efa
   BootTime=2024-07-02T19:00:09 SlurmdStartTime=2024-07-08T19:33:25
   LastBusyTime=2024-07-08T19:33:25 ResumeAfterTime=None
   CfgTRES=cpu=64,mem=124518M,billing=64
   AllocTRES=
   CapWatts=n/a
   CurrentWatts=0 AveWatts=0
   ExtSensorsJoules=n/a ExtSensorsWatts=0 ExtSensorsTemp=n/a
   Reason=Maintain Minimum Number Of Instances [root@2024-07-02T18:59:00]
   InstanceId=i-04927897a9ce3c143 InstanceType=hpc7g.16xlarge

NodeName=hpc7g-2 Arch=aarch64 CoresPerSocket=1
   CPUAlloc=0 CPUEfctv=64 CPUTot=64 CPULoad=0.00
   AvailableFeatures=hpc7g
   ActiveFeatures=hpc7g
   Gres=(null)
   NodeAddr=10.3.132.211 NodeHostName=ip-10-3-132-211 Version=25.05.4
   OS=Linux 5.10.218-208.862.amzn2.aarch64 #1 SMP Tue Jun 4 16:52:10 UTC 2024
   RealMemory=124518 AllocMem=0 FreeMem=110759 Sockets=64 Boards=1
   State=IDLE+CLOUD ThreadsPerCore=1 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
   Partitions=efa
   BootTime=2024-07-02T19:00:09 SlurmdStartTime=2024-07-08T19:33:25
   LastBusyTime=2024-07-08T19:33:25 ResumeAfterTime=None
   CfgTRES=cpu=64,mem=124518M,billing=64
   AllocTRES=
   CapWatts=n/a
   CurrentWatts=0 AveWatts=0
   ExtSensorsJoules=n/a ExtSensorsWatts=0 ExtSensorsTemp=n/a
   Reason=Maintain Minimum Number Of Instances [root@2024-07-02T18:59:00]
   InstanceId=i-0a2c82623cb1393a7 InstanceType=hpc7g.16xlarge
```

Connect to one of the nodes (in this example case, `hpc7g-1`) using SSH (or SSM). Note that this is an internal IP address, so you may need to connect from one of your login nodes if you use SSH. Also be aware that the instance needs to be configured with an SSH key by way of the compute node group launch template.

```
% ssh ec2-user@10.3.140.69
```

Now, launch `fi_pingpong` in server mode.

```
/opt/amazon/efa/bin/fi_pingpong -p efa
```

Connect to the second instance (`hpc7g-2`).

```
% ssh ec2-user@10.3.132.211
```

Run `fi_pingpong` in client mode, connecting to the server on `hpc7g-1`. You should see output that resembles the example below.

```
% /opt/amazon/efa/bin/fi_pingpong -p efa 10.3.140.69
```

```
bytes   #sent   #ack     total       time     MB/sec    usec/xfer   Mxfers/sec
64      10      =10      1.2k        0.00s      3.08      20.75       0.05
256     10      =10      5k          0.00s     21.24      12.05       0.08
1k      10      =10      20k         0.00s     82.91      12.35       0.08
4k      10      =10      80k         0.00s    311.48      13.15       0.08
[error] util/pingpong.c:1876: fi_close (-22) fid 0
```

(Optional) Use a CloudFormation template to create an EFA-enabled launch template

Because there are several dependencies to setting up EFA, a CloudFormation template has been provided that you can use to configure a compute node group. It supports instances with up to four network cards. To learn more about instances with multiple network cards, see Elastic network interfaces in the *Amazon Elastic Compute Cloud User Guide*.

Download the CloudFormation template from the following URL, then upload it to the CloudFormation console in the AWS Region where you use AWS PCS.

```
https://aws-hpc-recipes.s3.amazonaws.com/main/recipes/pcs/enable_efa/assets/pcs-lt-
efa.yaml
```

With the template open in the CloudFormation console, enter the following values. Note that the template will provide some default parameter values—you can leave them as their default values.

- Under **Provide a stack name**
  - Under **Stack name**, enter a descriptive name. We recommend incorporating the name you will choose for your AWS PCS compute node group, such as `*NODEGROUPNAME*-efa-lt`.
- Under **Parameters**
  - Under **NumberOfNetworkCards**, choose the number of network cards in the instances that will be in your node group.
  - Under **VpcId**, choose the VPC where your AWS PCS cluster is deployed.
  - Under **NodeGroupSubnetId**, choose the subnet in your cluster VPC where EFA-enabled instances will be launched.
  - Under **PlacementGroupName**, leave the field blank to create a new cluster placement group for the node group. If you have an existing placement group you want to use, enter its name here.

  - Under **ClusterSecurityGroupId**, choose the security group you are using to allow access to other instances in the cluster and to the AWS PCS API. Many customers choose the default security group from their cluster VPC.
  - Under **SshSecurityGroupId**, provide the ID for a security group you are using to allow inbound SSH access to nodes in your cluster.
  - For **SshKeyName**, select the SSH keypair for access to nodes in your cluster.
  - For **LaunchTemplateName**, enter a descriptive name for the launch template such as `*NODEGROUPNAME*-efa-lt`. The name must be unique to your AWS account in the AWS Region where you will use AWS PCS.
- Under **Capabilities**
  - Check the box for **I acknowledge that AWS CloudFormation might create IAM resources**.

Monitor the status of the CloudFormation stack. When it reaches `CREATE_COMPLETE` the launch template is ready to be used. Use it with an AWS PCS compute node group, as described above in Create or update compute node groups for EFA.

Using network file systems with AWS PCS

You can attach network file systems to nodes launched in an AWS Parallel Computing Service (AWS PCS) compute node group to provide a persistent location where data and files can be written and accessed. You can use file systems provided by AWS services, including Amazon Elastic File System (Amazon EFS), Amazon FSx for Lustre, Amazon FSx for NetApp ONTAP, Amazon FSx for OpenZFS, and Amazon File Cache. You can also use self-managed file systems, such as NFS servers.

This topic covers considerations for and examples of using network file systems with AWS PCS.

Considerations for using network file systems

The implementation details for various file systems are different, but there are some common considerations.

- The relevant file system software must be installed on the instance. For example, to use Amazon FSx for Lustre, the appropriate Lustre package should be present. This can be accomplished by including it in the compute node group AMI or using a script that runs at instance boot.
- There must be a network route between the shared network file system and the compute node group instances.
- The security group rules for both the shared network file system and the compute node group instances must allow connections to the relevant ports.
- You must maintain a consistent POSIX user and group namespace across resources that access the file systems. Otherwise, jobs and interactive processes that run on your PCS cluster may encounter permissions errors.
- File system mounts are done using EC2 launch templates. Errors or timeouts in mounting a network file system may prevent instances from becoming available to run jobs. This, in turn, may lead to unexpected costs. For more information about debugging launch templates, see Using Amazon EC2 launch templates with AWS PCS.

Example network mounts

You can create file systems using Amazon EFS, Amazon FSx for Lustre, Amazon FSx for NetApp ONTAP, Amazon FSx for OpenZFS, and Amazon File Cache. Expand the relevant section below to see an example of each network mount.

Amazon EFS

File system setup

Create an Amazon EFS file system. Make sure it has a mount target in each Availability Zone where you will launch PCS compute node group instances. Also ensure each mount target is associated with a security group that allows inbound and outbound access from the PCS compute node group instances. For more information, see Mount targets and security groups in the *Amazon Elastic File System User Guide*.

Launch template

Add the security group(s) from your file system setup to the launch template you will use for the compute node group.

Include user data that uses `cloud-config` mechanism to mount the Amazon EFS file system. Replace the following values in this script with your own details:

- *`mount-point-directory`* – The path on a each instance where you will mount Amazon EFS
- *`filesystem-id`* – The file system ID for the EFS file system

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

packages:
  - amazon-efs-utils

runcmd:
  - mkdir -p /mount-point-directory
  - echo "filesystem-id:/ /mount-point-directory efs tls,_netdev" >> /etc/fstab
  - mount -a -t efs defaults

--==MYBOUNDARY==--
```

Amazon FSx for Lustre

File system setup

Create an FSx for Lustre file system in the VPC where you will use AWS PCS. To minimize inter-zone transfers, deploy in a subnet in the same Availability Zone where you will launch the majority of your PCS compute node group instances. Ensure the file system is associated with a security group that allows inbound and outbound access from the PCS compute node group instances. For more information on security groups, see File system access control with Amazon VPC in the *Amazon FSx for Lustre User Guide*.

Launch template

Include user data that uses `cloud-config` to mount the FSx for Lustre file system. Replace the following values in this script with your own details:

- *`mount-point-directory`* – The path on an instance where you want to mount FSx for Lustre
- *`filesystem-id`* – The file system ID for the FSx for Lustre file system
- *`mount-name`* – The mount name for the FSx for Lustre file system
- *`region-code`* – The AWS Region where the FSx for Lustre file system is deployed (must be the same as your AWS PCS system)
- (Optional) *`latest`* – Any version of Lustre supported by FSx for Lustre

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

runcmd:
- amazon-linux-extras install -y lustre=latest
- mkdir -p /mount-point-directory
- mount -t lustre filesystem-id.fsx.region-code.amazonaws.com@tcp:/mount-name /mount-point-directory

--==MYBOUNDARY==
```

Amazon FSx for NetApp ONTAP

File system setup

Create an Amazon FSx for NetApp ONTAP file system in the VPC where you will use AWS PCS. To minimize inter-zone transfers, deploy in a subnet in the same Availability Zone where you will launch the majority of your AWS PCS compute node group instances. Make sure the file system is associated with a security group that allows inbound and outbound access from the AWS PCS compute node group instances. For more information on security groups, see File System Access Control with Amazon VPC in the *FSx for ONTAP User Guide*.

Launch template

Include user data that uses `cloud-config` to mount the root volume for an FSx for ONTAP file system. Replace the following values in this script with your own details:

- *`mount-point-directory`* – The path on an instance where you want to mount your FSx for ONTAP volume
- *`svm-id`* – The SVM ID for the FSx for ONTAP file system
- *`filesystem-id`* – The file system ID for the FSx for ONTAP file system
- *`region-code`* – The AWS Region where the FSx for ONTAP file system is deployed (must be the same as your AWS PCS system)
- *`volume-name`* – The FSx for ONTAP volume name

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

runcmd:
- mkdir -p /mount-point-directory
- mount -t nfs svm-id.filesystem-id.fsx.region-code.amazonaws.com:/volume-name /mount-point-directory

--==MYBOUNDARY==
```

Amazon FSx for OpenZFS

File system setup

Create an FSx for OpenZFS file system in the VPC where you will use AWS PCS. To minimize inter-zone transfers, deploy in a subnet in the same Availability Zone where you will launch the majority of your AWS PCS compute node group instances. Make sure the file system is associated with a security group that allows inbound and outbound access from the AWS PCS compute node group instances. For more information on security groups, see Managing file system access with Amazon VPC in the *FSx for OpenZFS User Guide*.

Launch template

Include user data that uses `cloud-config` to mount the root volume for an FSx for OpenZFS file system. Replace the following values in this script with your own details:

- *`mount-point-directory`* – The path on an instance where you want to mount your FSx for OpenZFS share
- *`filesystem-id`* – The file system ID for the FSx for OpenZFS file system
- *`region-code`* – The AWS Region where the FSx for OpenZFS file system is deployed (must be the same as your AWS PCS system)

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

runcmd:
- mkdir -p /mount-point-directory
- mount -t nfs -o noatime,nfsvers=4.2,sync,rsize=1048576,wsize=1048576 filesystem-id.fsx.region-code.amazonaws.com:/fsx/ /mount-point-directory

--==MYBOUNDARY==
```

Amazon File Cache

File system setup

Create an Amazon File Cache in the VPC where you will use AWS PCS. To minimize inter-zone transfers, choose a subnet in the same Availability Zone where you will launch the majority of your PCS compute node group instances. Ensure the File Cache is associated with a security group that allows inbound and outbound traffic on port 988 between your PCS instances and the File Cache. For more information on security groups, see Cache access control with Amazon VPC in the *Amazon File Cache User Guide*.

Launch template

Add the security group(s) from your file system setup to the launch template you will use for the compute node group.

Include user data that uses `cloud-config` to mount the Amazon File Cache. Replace the following values in this script with your own details:

- *`mount-point-directory`* – The path on an instance where you want to mount FSx for Lustre
- *`cache-dns-name`* – The Domain Name System (DNS) name for the File Cache
- *`mount-name`* – The mount name for the File Cache

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

runcmd:
- amazon-linux-extras install -y lustre=2.12
- mkdir -p /mount-point-directory
- mount -t lustre -o relatime,flock cache-dns-name@tcp:/mount-name /mount-point-directory

--==MYBOUNDARY==
```

Amazon Machine Images (AMIs) for AWS PCS

AWS PCS works with AMIs that you provide, affording great flexibility in the software and configuration found on nodes in your cluster. If you are trying out AWS PCS, you can use a sample AMI provided by and maintained by AWS. If you are using AWS PCS in production, we recommend you build your own AMIs. This topic covers how to discover and use the sample AMIs, as well as how to build and use your own customized AMIs.

Topics

Using sample Amazon Machine Images (AMIs) with AWS PCS

AWS provides sample AMIs that you can use as a starting point for working with AWS PCS.

⚠ **Important**

Sample AMIs are for demonstration purposes and are not recommended for production workloads.

⚠ **Important**

Compute node groups configured with AWS PCS sample AMIs and multiple network interfaces won't work currently if the subnets are only configured to use IPv6. Use dual-stack subnets (IPv4 and IPv6) or IPv4-only subnets instead.

Find current AWS PCS sample AMIs

AWS Management Console

AWS PCS sample AMIs have the following naming convention:

```
aws-pcs-sample_ami-OS-architecture-scheduler-scheduler-major-version
```

Accepted values

- *OS* – amzn2
- *architecture* – x86_64 or arm64
- *scheduler* – slurm
- *scheduler-major-version* – 25.05

To find AWS PCS sample AMIs

1. Open the Amazon EC2 console.
2. Navigate to **AMIs**.
3. Choose **Public images**.
4. In **Find AMI by attribute or tag**, search for an AMI using the templated name.

   Examples

   - Sample AMI for Slurm 25.05 on Arm64 instances

     ```
     aws-pcs-sample_ami-amzn2-arm64-slurm-25.05
     ```

   - Sample AMI for Slurm 25.05 on x86 instances

     ```
     aws-pcs-sample_ami-amzn2-x86_64-slurm-25.05
     ```

   > **Note**
   >
   > If there are multiple AMIs, use the AMI with the most recent time stamp.

5. Use the AMI ID when you create or update a compute node group.

AWS CLI

You can find the latest AWS PCS sample AMI with the commands that follow. Replace *region-code* with the AWS Region where you use AWS PCS, such as us-east-1.

- **x86_64**

  ```
  aws ec2 describe-images --region region-code --owners amazon \
  --filters 'Name=name,Values=aws-pcs-sample_ami-amzn2-x86_64-slurm-25.05*' \
              'Name=state,Values=available' \
  --query 'sort_by(Images, &CreationDate)[-1].[Name,ImageId]' --output text
  ```

- **Arm64**

  ```
  aws ec2 describe-images --region region-code --owners amazon \
  --filters 'Name=name,Values=aws-pcs-sample_ami-amzn2-arm64-slurm-25.05*' \
              'Name=state,Values=available' \
  --query 'sort_by(Images, &CreationDate)[-1].[Name,ImageId]' --output text
  ```

Use the AMI ID when you create or update a compute node group.

Learn more about AWS PCS sample AMIs

To view the contents, configuration details for current and previous releases of the AWS PCS sample AMIs, see Release notes for AWS PCS sample AMIs.

Build your own AMIs compatible with AWS PCS

To learn how to build your own AMIs that work with AWS PCS, see Custom Amazon Machine Images (AMIs) for AWS PCS.

Custom Amazon Machine Images (AMIs) for AWS PCS

AWS PCS is designed to work with Amazon Machine Images (AMI) that you bring to the service. These AMIs can have arbitrary software and configurations installed on them, so long as they have the AWS PCS agent and a compatible version of Slurm installed and configured correctly. You must use AWS-provided installers to install the AWS PCS software on your custom AMI. We recommend you use AWS-provided installers to install Slurm on your custom AMI but you can install Slurm on your own if you prefer (not recommended).

Note

If you want to try AWS PCS without building a custom AMI, you can use a sample AMI provided by AWS. For more information, see Using sample Amazon Machine Images (AMIs) with AWS PCS.

Important

AWS PCS currently requires a kernel with IPv4 support for local node communication, even when you use AWS PCS in an IPv6-only network.

This tutorial helps you create an AMI that can be used with PCS compute node groups to power your HPC and AI/ML workloads.

Topics

Step 1 – Launch a temporary instance

Launch a temporary instance that you can use to install and configure the AWS PCS software and Slurm scheduler. You use this instance to create an AMI compatible with AWS PCS.

To launch a temporary instance

1. Open the Amazon EC2 console.
2. In the navigation pane, choose **Instances**, then choose **Launch instances** to open the new launch instance wizard.

3. (Optional) In the **Name and tags** section, provide a name for the instance, such as `PCS-AMI-instance`. The name is assigned to the instance as a resource tag (`Name=PCS-AMI-instance`).
4. In the **Application and OS Images** section, select an AMI for one of the supported operating systems.
5. In the **Instance type** section, select a supported instance type.
6. In the **Key pair** section, select the key pair to use for the instance.
7. In the **Network settings** section:
   - For **Firewall (security groups)**, choose **Select existing security group**, then select a security group that allows inbound SSH access to your instance.
8. In the **Storage** section, configure the volumes as needed. Make sure to configure sufficient space to install your own applications and libraries.
9. In the **Summary** panel, choose **Launch instance**.

Step 2 – Install the AWS PCS agent

Install the agent that configures the instances launched by AWS PCS for use with Slurm. For more information about the AWS PCS agent, see AWS PCS agent versions.

To install the AWS PCS agent

1. Connect to the instance you launched. For more information, see Connect to your Linux instance.
2. (Optional) To ensure that all of your software packages are up to date, perform a quick software update on your instance. This process may take a few minutes.
   - **Amazon Linux 2, Amazon Linux 2023, RHEL 9, RHEL 8, Rocky Linux 9, and Rocky Linux 8**

     ```
     sudo yum update -y
     ```
   - **Ubuntu 22.04 and Ubuntu 24.04**

     ```
     sudo apt-get update && sudo apt-get upgrade -y
     ```
3. Reboot the instance and reconnect to it.

4. Download the AWS PCS agent installation files. The installation files are packaged into a compressed tarball (`.tar.gz`) file. To download the latest *stable* version, use the following command. Substitute *`region`* with the AWS Region where you launched your temporary instance, such as `us-east-1`.

   ```
   curl https://aws-pcs-repo-region.s3.region.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.3.2-1.tar.gz -o aws-pcs-agent-v1.3.2-1.tar.gz
   ```

   You can also get the latest version by replacing the version number with `latest` in the preceding command (for example: `aws-pcs-agent-v1-latest.tar.gz`).

   Note

   This might change in future releases of the AWS PCS agent software.

5. (Optional) Verify the authenticity and integrity of the AWS PCS software tarball. We recommend that you do this to verify the identity of the software publisher and to check that the file has not been altered or corrupted since it was published.

   a. Download the public GPG key for AWS PCS and import it into your keyring. Substitute *`region`* with the AWS Region where you launched your temporary instance. The command should return a key value. Record the key value; you use it in the next step.

      ```
      wget https://aws-pcs-repo-public-keys-region.s3.region.amazonaws.com/aws-pcs-public-key.pub && \
          gpg --import aws-pcs-public-key.pub
      ```

   b. Run the following command to verify the GPG key's fingerprint.

      ```
      gpg --fingerprint 7EEF030EDDF5C21C
      ```

      The command should return a fingerprint that is identical to the following:

      ```
      1C24 32C1 862F 64D1 F90A  239A 7EEF 030E DDF5 C21C
      ```

> **Important**
>
> Don't run the AWS PCS agent installation script if the fingerprint doesn't match. Contact AWS Support.

c. Download the signature file and verify the signature of the AWS PCS software tarball file. Replace *region* with the AWS Region where you launched your temporary instance, such as `us-east-1`.

```
wget https://aws-pcs-repo-region.s3.region.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.3.2-1.tar.gz.sig && \
    gpg --verify ./aws-pcs-agent-v1.3.2-1.tar.gz.sig
```

The output should be similar to the following:

```
gpg: assuming signed data in './aws-pcs-agent-v1.3.2-1.tar.gz'
gpg: Signature made Thu 06 Nov 2025 11:10:36 AM CET using RSA key ID ECC0AE5C
gpg: Good signature from "AWS PCS Packages (AWS PCS Packages)"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 1C24 32C1 862F 64D1 F90A  239A 7EEF 030E DDF5 C21C
   Subkey fingerprint: B7E1 8788 3517 6A74 C3D5  EAF5 6088 136D ECC0 AE5C
```

If the result includes `Good signature` and the fingerprint matches the fingerprint returned in the previous step, proceed to the next step.

> **Important**
>
> Don't run the AWS PCS software installation script if the fingerprint doesn't match. Contact AWS Support.

6. Extract the files from the compressed `.tar.gz` file and navigate to the extracted directory.

```
tar -xf aws-pcs-agent-v1.3.2-1.tar.gz && \
    cd aws-pcs-agent
```

7. Install the AWS PCS software.

```
sudo ./installer.sh
```

8. Check the AWS PCS software version file to confirm a successful installation.

   ```
   cat /opt/aws/pcs/version
   ```

   The output should be similar to the following:

   ```
   AGENT_INSTALL_DATE='Fri Dec 13 12:28:43 UTC 2024'
   AGENT_VERSION='1.3.2'
   AGENT_RELEASE='1'
   ```

Step 3 – Install Slurm

Install a version of Slurm that is compatible with AWS PCS. For more information, see Slurm versions in AWS PCS.

> **Note**
>
> If you have an AMI with a previous version of the Slurm software installed on it, you must perform the following steps to install the new version of Slurm. The AWS PCS agent enables the correct version of the Slurm binaries at runtime, according to the Slurm version configured at cluster creation time.

To install Slurm

1. Connect to the same temporary instance where you installed the AWS PCS software.
2. Download the Slurm installer software. The Slurm installer is packaged into a compressed tarball (`.tar.gz`) file. To download the latest *stable* version, use the following command. Substitute *`region`* with the AWS Region of your temporary instance, such as `us-east-1`.

   ```
   curl https://aws-pcs-repo-region.s3.region.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-25.05-installer-25.05.4-1.tar.gz \
        -o aws-pcs-slurm-25.05-installer-25.05.4-1.tar.gz
   ```

You can also get the latest version by replacing the version number with `latest` in the preceding command (for example: `aws-pcs-slurm-25.05-installer-latest.tar.gz`). For a complete list of available versions with checksums, see Slurm versions in AWS PCS.

> **Note**
>
> This might change in future releases of the Slurm installer software.

3. (Optional) Verify the authenticity and integrity of the Slurm installer tarball. We recommend that you do this to verify the identity of the software publisher and to check that the file has not been altered or corrupted since it was published.

   a. Download the public GPG key for AWS PCS and import it into your keyring. Substitute *`region`* with the AWS Region where you launched your temporary instance. The command should return a key value. Record the key value; you use it in the next step.

   ```
   wget https://aws-pcs-repo-public-keys-region.s3.region.amazonaws.com/aws-pcs-public-key.pub && \
       gpg --import aws-pcs-public-key.pub
   ```

   b. Run the following command to verify the GPG key's fingerprint.

   ```
   gpg --fingerprint 7EEF030EDDF5C21C
   ```

   The command should return a fingerprint that is identical to the following:

   ```
   1C24 32C1 862F 64D1 F90A  239A 7EEF 030E DDF5 C21C
   ```

   > **Important**
   >
   > Don't run the Slurm installation script if the fingerprint doesn't match. Contact AWS Support.

   c. Download the signature file and verify the signature of the Slurm installer tarball file. Replace *`region`* with the AWS Region where you launched your temporary instance, such as `us-east-1`.

```
wget https://aws-pcs-repo-region.s3.region.amazonaws.com/aws-pcs-slurm/aws-pcs-
slurm-25.05-installer-25.05.4-1.tar.gz.sig && \
     gpg --verify ./aws-pcs-slurm-25.05-installer-25.05.4-1.tar.gz.sig
```

The output should be similar to the following:

```
gpg: assuming signed data in './aws-pcs-slurm-25.05-installer-25.05.4-1.tar.gz'
gpg: Signature made Fri 24 Oct 2025 05:05:11 PM UTC using RSA key ID ECC0AE5C
gpg: Good signature from "AWS PCS Packages (AWS PCS Packages)"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 1C24 32C1 862F 64D1 F90A  239A 7EEF 030E DDF5 C21C
     Subkey fingerprint: B7E1 8788 3517 6A74 C3D5  EAF5 6088 136D ECC0 AE5C
```

If the result includes `Good signature` and the fingerprint matches the fingerprint returned in the previous step, proceed to the next step.

> ⚠ **Important**
>
> Don't run the Slurm installation script if the fingerprint doesn't match. Contact AWS Support.

4. Extract the files from the compressed `.tar.gz` file and navigate into the extracted directory.

```
tar -xf aws-pcs-slurm-25.05-installer-25.05.4-1.tar.gz && \
    cd aws-pcs-slurm-25.05-installer
```

5. Install Slurm. The installer downloads, compiles, and installs Slurm and its dependencies. It takes several minutes, depending on the specifications of the temporary instance you selected.

```
sudo ./installer.sh -y
```

6. Check the scheduler version file to confirm the installation.

```
cat /opt/aws/pcs/scheduler/slurm-25.05/version
```

The output should be similar to the following:

```
SLURM_INSTALL_DATE='Mon Nov 3 14:23:38 UTC 2025'
```

```
SLURM_VERSION='25.05.4'
PCS_SLURM_RELEASE='1'
```

Step 4 – (Optional) Install additional drivers, libraries, and application software

Install additional drivers, libraries, and application software on the temporary instance. The installation procedures will vary depending on the specific applications and libraries. If you have not built a custom AMI for AWS PCS before, we recommend you first build and test an AMI with just the AWS PCS software and Slurm installed, then incrementally add your own software and configurations once you have confirmed initial success.

Examples

- Elastic Fabric Adapter (EFA) software. For more information, see Get started with EFA and MPI for HPC workloads on Amazon EC2 in the *Amazon Elastic Compute Cloud User Guide*.
- Amazon Elastic File System (Amazon EFS) client. For more information, see Manually installing the Amazon EFS client in the *Amazon Elastic File System User Guide*.
- Lustre client, to use Amazon FSx for Lustre and Amazon File Cache. For more information, see Installing the Lustre client in the *FSx for Lustre User Guide*.
- Amazon CloudWatch agent, to use CloudWatch Logs and Metrics. For more information, see Install the CloudWatch agent in the *Amazon CloudWatch User Guide*.
- AWS Neuron, to use **trn*** and **inf*** instance types. For more information, see the AWS Neuron documentation.
- NVIDIA Driver, CUDA, and DCGM, to use **p*** or **g*** instance types.

Step 5 – Create an AMI compatible with AWS PCS

After you have installed the required software components, you create an AMI that you can reuse to launch instances in AWS PCS compute node groups.

⚠ **Important**

AWS PCS currently requires a kernel with IPv4 support for local node communication, even when you use AWS PCS in an IPv6-only network.

To create an AMI from your temporary instance

1. Open the Amazon EC2 console.
2. In the navigation pane, choose **Instances**.
3. Select the temporary instance that you created. Choose **Actions**, **Image**, **Create image**.
4. For **Create image**, do the following:
   a. For **Image name**, enter a descriptive name for the AMI.
   b. (Optional) For **Image description**, enter a brief description of the purpose of the AMI.
   c. Choose **Create image**.
5. In the navigation pane, choose **AMIs**.
6. Locate the AMI that you created in the list. Wait for its status to change from **Pending** to **Available**, then use it with a AWS PCS compute node group.

Step 6 – Use the custom AMI with an AWS PCS compute node group

You can use your custom AMI with a new or existing AWS PCS compute node group.

> ⚠ **Important**
>
> AWS PCS currently requires a kernel with IPv4 support for local node communication, even when you use AWS PCS in an IPv6-only network.

New compute node group

To use the custom AMI

1. Open the AWS PCS console.
2. In the navigation pane, choose **Clusters**.
3. Choose the cluster where you will use the custom AMI, then select **Compute node groups**.
4. Create a new compute node group. For more information, see Creating a compute node group in AWS PCS. Under **AMI ID**, search for the name or ID of the custom AMI you want to use. Finish configuring the compute node group, then choose **Create compute node group**.

5. (Optional) Confirm the AMI supports instance launches. Launch an instance in the compute node group. You can do this by configuring the compute node group to have a single static instance, or you can submit a job to a queue that uses the compute node group.

    a. Check the Amazon EC2 console until an instance appears tagged with the new compute node group ID. For more information on this, see Finding compute node group instances in AWS PCS..

    b. When you see an instance launch and complete its bootstrap process, confirm it is using the expected AMI. To do this, select the instance, then inspect **AMI ID** under **Details**. It should match the AMI you configured in the compute node group settings.

    c. (Optional) Update the compute node group scaling configuration to your preferred values.

Existing compute node group

To use the custom AMI

1. Open the AWS PCS console.
2. In the navigation pane, choose **Clusters**.
3. Choose the cluster where you will use the custom AMI, then select **Compute node groups**.
4. Select the node group you wish to configure and choose **Edit**. Under **AMI ID**, search for the name or ID of the custom AMI you want to use. Finish configuring the compute node group, then choose **Update**. New instances launched in the compute node group will use the updated AMI ID. Existing instances will continue to use the old AMI until AWS PCS replaces them. For more information, see Updating an AWS PCS compute node group.
5. (Optional) Confirm the AMI supports instance launches. Launch an instance in the compute node group. You can do this by configuring the compute node group to have a single static instance, or you can submit a job to a queue that uses the compute node group.

    a. Check the Amazon EC2 console until an instance appears tagged with the new compute node group ID. For more information on this, see Finding compute node group instances in AWS PCS..

    b. When you see an instance launch and complete its bootstrap process, confirm it is using the expected AMI. To do this, select the instance, then inspect **AMI ID** under **Details**. It should match the AMI you configured in the compute node group settings.

c. (Optional) Update the compute node group scaling configuration to your preferred values.

Step 7 – Terminate the temporary instance

After you have confirmed that your AMI works as intended with AWS PCS, you can terminate the temporary instance to stop incurring charges for it.

To terminate the temporary instance

1. Open the Amazon EC2 console.
2. In the navigation pane, choose **Instances**.
3. Select the temporary instance that you created and choose **Actions**, **Instance state**, **Terminate instance**.
4. When prompted to confirm, choose **Terminate**.

Software installers to build custom AMIs for AWS PCS

AWS provides a downloadable file that can install the AWS PCS software on an instance. AWS also provides software that can download, compile, and install relevant versions of Slurm and its dependencies. You can use these instructions to build custom AMIs for use with AWS PCS or you can use your own methods.

Contents

AWS PCS agent software installer

The AWS PCS agent software installer configures an instance to work with AWS PCS during the instance bootstrap process. You must use AWS-provided installers to install the AWS PCS agent on your custom AMI.

For more information about the AWS PCS agent software, see AWS PCS agent versions.

Slurm installer

The Slurm installer downloads, compiles, and installs relevant versions of Slurm and its dependencies. You can use the Slurm installer to build custom AMIs for AWS PCS. You can also use your own mechanisms if they are consistent with the software configuration that the Slurm installer provides. For more information about AWS PCS support for Slurm, see Slurm versions in AWS PCS.

The AWS-provided software installs the following:

- Slurm at the requested major and maintenance version (currently version 25.05.x) - License GPL 2
  - Slurm is built with `--sysconfdir` set to `/etc/slurm`
  - Slurm is built with the option `--enable-pam` and `--without-munge`
  - Slurm is built with the option `--sharedstatedir=/run/slurm/`
  - Slurm is built with PMIX and JWT support
  - Slurm is installed at `/opt/aws/pcs/schedulers/slurm-25.05`
- OpenPMIX (version 4.2.6) – License
  - OpenPMIX is installed as a subdirectory of `/opt/aws/pcs/scheduler/`
- libjwt (version 1.17.0) – License MPL-2.0
  - libjwt is installed as a subdirectory of `/opt/aws/pcs/scheduler/`

The AWS-provided software changes the system configuration as follows:

- The Slurm `systemd` file created by the build is copied to `/etc/systemd/system/` with file name `slurmd-25.05.service`.
- If they don't exist, a Slurm user and group (`slurm:slurm`) are created with UID/GID of `401`.

- The folder /etc/aws/pcs/scheduler/slurm-25.05/plugstack.conf.d/ is created, to store your Extend Slurm functionality on AWS PCS with SPANK plugins configuration.
- On Amazon Linux 2 and Rocky Linux 9 the installation adds the EPEL repository to install the required software to build Slurm or its dependencies.
- On RHEL9 the installation will enable codeready-builder-for-rhel-9-rhui-rpms and epel-release-latest-9 from fedoraproject to install the required software to build Slurm or its dependencies.

Supported operating systems

See Supported operating systems in AWS PCS.

Note

AWS Deep Learning AMIs (DLAMI) versions based on Amazon Linux 2 and Ubuntu 22.04 should be compatible with the AWS PCS software and Slurm installers. For more information, see Choosing Your DLAMI in the *AWS Deep Learning AMIs Developer Guide*.

Supported instance types

AWS PCS software and Slurm installers support any x86_64 or arm64 instance type than can run one of the supported operating systems.

Supported Slurm versions

See Slurm versions in AWS PCS.

Verify installers using a checksum

You can use SHA256 checksums to verify the installer tarball (.tar.gz) files. We recommend that you do this to verify the identity of the software publisher and to check that the application has not been altered or corrupted since it was published.

To verify a tarball

Use the **sha256sum** utility for the SHA256 checksum and specify the tarball filename. You must run the command from the directory where you saved the tarball file.

- SHA256

  ```
  $ sha256sum tarball_filename.tar.gz
  ```

The command should return a checksum value in the following format.

```
checksum_value tarball_filename.tar.gz
```

Compare the checksum value returned by the command with the checksum value provided in the following table. If the checksums match, then it's safe to run the installation script.

> **Important**
>
> If the checksums don't match, don't run the installation script. Contact Support.

For example, the following command generates the SHA256 checksum for the Slurm 25.05.4-1 tarball.

```
$ sha256sum aws-pcs-slurm-25.05-installer-25.05.4-1.tar.gz
```

Example output:

```
3b0f93bce441d4f4f6935175f2c1e81cd961cb923adb416fa6689f5592047a7d aws-pcs-slurm-25.05-installer-25.05.4-1.tar.gz
```

The following tables list the checksums for recent versions of the installers. Replace *us-east-1* with the AWS Region where you use AWS PCS.

AWS PCS agent

Installer	Download URL	SHA256 checksum
AWS PCS agent 1.3.2-1	`https://aws-pcs-repo-us-east-1.s3.us-east-1.amazonaws.com/aws-pcs-agent/aws-pcs`	`06b32a952a1c849e3442e35c28ac2e4d6962b09286cad748f3c83d561b52ec6f`

Installer	Download URL	SHA256 checksum
	-agent-v1.3.2-1.tar.gz	
AWS PCS agent 1.3.1-1	https://aws-pcs-repo-*us-east-1*.s3.*us-east-1*.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.3.1-1.tar.gz	5b7f1eb7b3a86bd2d331b5cb0138d868dc9452da34b480becd86af892c7e8d19
AWS PCS agent 1.3.0-1	https://aws-pcs-repo-*us-east-1*.s3.*us-east-1*.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.3.0-1.tar.gz	eadc9b65c3db248bdde2a6c41814dfb1b97239f24ad55e03d8526dd9ab4a8d16
AWS PCS agent 1.2.2-1	https://aws-pcs-repo-*us-east-1*.s3.*us-east-1*.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.2.2-1.tar.gz	fd7b6ea5442db75d723fc4971781ce6ae511baa21b87c4286fc1df8127b282b8
AWS PCS agent 1.2.1-1	https://aws-pcs-repo-*us-east-1*.s3.*us-east-1*.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.2.1-1.tar.gz	2b784643ca01ccca1baa64fbfb34bb41efe8bdca69470998b74ce3962bc271d4

Installer	Download URL	SHA256 checksum
AWS PCS agent 1.2.0-1	https://aws-pcs-repo-*us-east-1*.s3.*us-east-1*.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.2.0-1.tar.gz	470db8c4fc9e50277b6317f98584b6b547e73523043e34f018eecae767846805
AWS PCS agent 1.1.1-1	https://aws-pcs-repo-*us-east-1*.s3.*us-east-1*.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.1.1-1.tar.gz	bef078bf60a6d8ecde2e6c49cd34d088703f02550279e3bf483d57a235334dc6
AWS PCS agent 1.1.0-1	https://aws-pcs-repo-*us-east-1*.s3.*us-east-1*.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.1.0-1.tar.gz	594c32194c71bccc5d66e5213213ae38dd2c6d2f9a950bb01accea0bbab0873a
AWS PCS agent 1.0.1-1	https://aws-pcs-repo-*us-east-1*.s3.*us-east-1*.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.0.1-1.tar.gz	04e22264019837e3f42d8346daf5886eaacecd21571742eb505ea8911786bcb2
AWS PCS agent 1.0.0-1	https://aws-pcs-repo-*us-east-1*.s3.*us-east-1*.amazonaws.com/aws-pcs-agent/aws-pcs-agent-v1.0.0-1.tar.gz	d2d3d68d00c685435c38af471d7e2492dde5ce9eb222d7b6ef0042144b134ce0

Slurm installer

Installer	Download URL	SHA256 checksum
Slurm 25.05.4-1	`https://aws-pcs-repo-us-east-1.s3.us-east-1.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-25.05-installer-25.05.4-1.tar.gz`	`3b0f93bce441d4f4f6935175f2c1e81cd961cb923adb416fa6689f5592047a7d`
Slurm 25.05.3-1	`https://aws-pcs-repo-us-east-1.s3.us-east-1.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-25.05-installer-25.05.3-1.tar.gz`	`851bb5815b6700ceb30cc4a3fda204ca8ce362c14528c339908983255a936cf0`
Slurm 24.11.6-2	`https://aws-pcs-repo-us-east-1.s3.us-east-1.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.11-installer-24.11.6-2.tar.gz`	`f17cd78e0bc6b9c818b794d9d2685cceabdc73f4fbb12f7566ae5b86a5abc32b`
Slurm 24.11.6-1	`https://aws-pcs-repo-us-east-1.s3.us-east-1.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.11-installer-24.11.6-1.tar.gz`	`225de9fc18206f5f65f412effe1fd457614ac97ee9822b3ff804a452b0fae522`
Slurm 24.11.5-1	`https://aws-pcs-repo-us-east-1.s3.us-east-1.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.11-installer-24.11.5-1.tar.gz`	`593efe4d66bef2f3e46d5a382fb5a32f7a3ca2510bcf1b3c85739f4f951810d5`

Installer	Download URL	SHA256 checksum
Slurm 24.05.8-2	https://aws-pcs-repo-*us-east-1*.s3.*us-east-1*.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.05-installer-24.05.8-2.tar.gz	c494b0b55c319a4c2f3faf668c759d46c32c4c7aa94ae97d94128328fe95364b
Slurm 24.05.8-1	https://aws-pcs-repo-*us-east-1*.s3.*us-east-1*.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.05-installer-24.05.8-1.tar.gz	210a43b376af082bbad640b2032655885790c5dab0e6489cc327c7310a375849
Slurm 24.05.7-1	https://aws-pcs-repo-*us-east-1*.s3.*us-east-1*.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.05-installer-24.05.7-1.tar.gz	0b5ed7c81195de2628c78f37c79e63fc4ae99132ca6b019b53a0d68792ee82c5
Slurm 24.05.5-2	https://aws-pcs-repo-*us-east-1*.s3.*us-east-1*.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-24.05-installer-24.05.5-2.tar.gz	7cc8d8294f2fbff95fe0602cf9e21e02003b5d96c0730e0a18c6aa04c7a4967b
Slurm 23.11.10-4 (deprecated)	https://aws-pcs-repo-*us-east-1*.s3.*us-east-1*.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-23.11-installer-23.11.10-4.tar.gz	bb2d8c919c69dba38d14358f49c7f0427564c5dd4af85a1c9eca2c57ceeae29a

Installer	Download URL	SHA256 checksum
Slurm 23.11.10-3 (deprecated)	`https://aws-pcs-repo-`*`us-east-1`*`.s3.`*`us-east-1`*`.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-23.11-installer-23.11.10-3.tar.gz`	`488a10ee0fbd57ec0e0ff7ea708a9e3038fafdc025c6bb391c75c2e2a7852a00`
Slurm 23.11.10-2 (deprecated)	`https://aws-pcs-repo-`*`us-east-1`*`.s3.`*`us-east-1`*`.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-23.11-installer-23.11.10-2.tar.gz`	`0bbe85423305c05987931168caf98da08a34c25f9eec0690e8e74de0b7bc8752`
Slurm 23.11.10-1 (deprecated)	`https://aws-pcs-repo-`*`us-east-1`*`.s3.`*`us-east-1`*`.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-23.11-installer-23.11.10-1.tar.gz`	`27e8faa9980e92cdfd8cfdc71f937777f0934552ce61e33dac4ecf5a20321e44`
Slurm 23.11.9-1 (deprecated)	`https://aws-pcs-repo-`*`us-east-1`*`.s3.`*`us-east-1`*`.amazonaws.com/aws-pcs-slurm/aws-pcs-slurm-23.11-installer-23.11.9-1.tar.gz`	`1de7d919c8632fe8e2806611bed4fde1005a4fadc795412456e935c7bba2a9b8`

Release notes for AWS PCS sample AMIs

AMIs for the latest supported major versions of the scheduler receive security updates and critical bug fixes. These incremental security patches aren't included in official release notes.

⚠ **Important**

Sample AMIs related to old scheduler versions aren't supported and don't receive updates.

⚠ **Important**

Sample AMIs are for demonstration purposes and are not recommended for production workloads.

Contents

AWS PCS sample AMIs for x86_64 (Amazon Linux 2)

Slurm 25.05

AMI name

- `aws-pcs-sample_ami-amzn2-x86_64-slurm-25.05`

Supported EC2 instances

- All instances with an 64-bit x86 processor. To find compatible instances, navigate to the Amazon EC2 console. Choose Instance Types, then search for Architectures=x86_64.

AMI contents

- Supported AWS Service: AWS PCS
- Operating System: Amazon Linux 2
- Compute Architecture: x86_64
- EBS volume type: gp2
- EFA Installer: 1.43.1
- GDRCopy: 2.5.1

- NVIDIA Driver: 550.127.08
- NVIDIA CUDA: 12.4.1_550.54.15

Slurm 24.11

> **Note**
>
> AWS PCS supports accounting for Slurm 24.11 and later. For more information, see Slurm accounting in AWS PCS.

AMI name

- `aws-pcs-sample_ami-amzn2-x86_64-slurm-24.11`

Supported EC2 instances

- All instances with an 64-bit x86 processor. To find compatible instances, navigate to the Amazon EC2 console. Choose **Instance Types**, then search for `Architectures=x86_64`.

AMI contents

- Supported AWS Service: AWS PCS
- Operating System: Amazon Linux 2
- Compute Architecture: x86_64
- EBS volume type: gp2
- EFA Installer: 1.33.0
- GDRCopy: 2.4
- NVIDIA Driver: 550.127.08
- NVIDIA CUDA: 12.4.1_550.54.15

Slurm 24.05

AMI name

- `aws-pcs-sample_ami-amzn2-x86_64-slurm-24.05`

Supported EC2 instances

- All instances with an 64-bit x86 processor. To find compatible instances, navigate to the Amazon EC2 console. Choose **Instance Types**, then search for `Architectures=x86_64`.

AMI contents

- Supported AWS Service: AWS PCS
- Operating System: Amazon Linux 2
- Compute Architecture: x86_64
- EBS volume type: gp2
- EFA Installer: 1.33.0
- GDRCopy: 2.4
- NVIDIA Driver: 550.127.08
- NVIDIA CUDA: 12.4.1_550.54.15

Slurm 23.11

AMI name

- `aws-pcs-sample_ami-amzn2-x86_64-slurm-23.11`

Supported EC2 instances

- All instances with an 64-bit x86 processor. To find compatible instances, navigate to the Amazon EC2 console. Choose **Instance Types**, then search for `Architectures=x86_64`.

AMI contents

- Supported AWS Service: AWS PCS
- Operating System: Amazon Linux 2
- Compute Architecture: x86_64
- EBS volume type: gp2
- EFA Installer: 1.33.0

- GDRCopy: 2.4
- NVIDIA Driver: 550.127.08
- NVIDIA CUDA: 12.4.1_550.54.15

AWS PCS sample AMIs for Arm64 (Amazon Linux 2)

Slurm 25.05

AMI name

- `aws-pcs-sample_ami-amzn2-arm64-slurm-25.05`

Supported EC2 instances

- All instances with an 64-bit Arm processor. To find compatible instances, navigate to the Amazon EC2 console. Choose Instance Types, then search for Architectures=arm64.

AMI contents

- Supported AWS Service: AWS PCS
- Operating System: Amazon Linux 2
- Compute Architecture: arm64
- EBS volume type: gp2
- EFA Installer: 1.43.1
- GDRCopy: 2.5.1
- NVIDIA Driver: 550.127.08
- NVIDIA CUDA: 12.4.1_550.54.15

Slurm 24.11

Note

AWS PCS supports accounting for Slurm 24.11 and later. For more information, see Slurm accounting in AWS PCS.

AMI name

- `aws-pcs-sample_ami-amzn2-arm64-slurm-24.11`

Supported EC2 instances

- All instances with an 64-bit Arm processor. To find compatible instances, navigate to the Amazon EC2 console. Choose **Instance Types**, then search for `Architectures=arm64`.

AMI contents

- Supported AWS Service: AWS PCS
- Operating System: Amazon Linux 2
- Compute Architecture: arm64
- EBS volume type: gp2
- EFA Installer: 1.33.0
- GDRCopy: 2.4
- NVIDIA Driver: 550.127.08
- NVIDIA CUDA: 12.4.1_550.54.15

Slurm 24.05

AMI name

- `aws-pcs-sample_ami-amzn2-arm64-slurm-24.05`

Supported EC2 instances

- All instances with an 64-bit Arm processor. To find compatible instances, navigate to the Amazon EC2 console. Choose **Instance Types**, then search for `Architectures=arm64`.

AMI contents

- Supported AWS Service: AWS PCS
- Operating System: Amazon Linux 2

- Compute Architecture: arm64
- EBS volume type: gp2
- EFA Installer: 1.33.0
- GDRCopy: 2.4
- NVIDIA Driver: 550.127.08
- NVIDIA CUDA: 12.4.1_550.54.15

Slurm 23.11

AMI name

- `aws-pcs-sample_ami-amzn2-arm64-slurm-23.11`

Supported EC2 instances

- All instances with an 64-bit Arm processor. To find compatible instances, navigate to the Amazon EC2 console. Choose **Instance Types**, then search for `Architectures=arm64`.

AMI contents

- Supported AWS Service: AWS PCS
- Operating System: Amazon Linux 2
- Compute Architecture: arm64
- EBS volume type: gp2
- EFA Installer: 1.33.0
- GDRCopy: 2.4
- NVIDIA Driver: 550.127.08
- NVIDIA CUDA: 12.4.1_550.54.15

Supported operating systems in AWS PCS

AWS PCS uses the Amazon Machine Image (AMI) configured for a compute node group to launch EC2 instances in that compute node group. The AMI determines the operating system that the EC2 instances use. You can't change the operating system in AWS PCS sample AMIs. You must create a custom AMI if you want to use a different operating system. For more information, see Amazon Machine Images (AMIs) for AWS PCS.

Supported operating systems

- **Amazon Linux 2**

  This is the operating system in the AWS PCS sample AMIs.

  > ⚠ **Important**
  >
  > Sample AMIs are for demonstration purposes and are not recommended for production workloads. You should create and use a custom AMI for production workloads, even if you intend to use Amazon Linux 2.

- **Amazon Linux 2023**
- **RedHat Enterprise Linux 9 (RHEL 9)**

  The on-demand cost for RHEL any instance type is higher than for other supported operation systems. For more information about pricing, see On-Demand Pricing and How is Red Hat Enterprise Linux on Amazon Elastic Compute Cloud offered and priced?.

- **RedHat Enterprise Linux 8 (RHEL 8)**
- **Rocky Linux 9**

  You can use the official Rocky Linux 9 AMIs as a base for a custom AMI. Your custom AMI build might fail if the base AMI doesn't have the latest kernel.

  To upgrade the kernel

  1. Launch an instance using a rocky9 AMI id from here: https://rockylinux.org/cloud-images/
  2. ssh into the instance and run the following command:

     ```
     sudo yum -y update
     ```

3. Create an image from the instance. You specify this image as the `ParentImage` for your custom AMI.

- **Rocky Linux 8**
- **Ubuntu 22.04**

  Ubuntu 22.04 requires more secure keys for SSH and doesn't support RSA keys by default. We recommend you generate and use an ED25519 key instead.
- **Ubuntu 24.04**

AWS PCS agent versions

The AWS PCS agent software configures the EC2 instances AWS PCS launches for use with Slurm. You include the agent in an Amazon Machine Images (AMI) that you specify when you create compute node groups for your cluster. The EC2 instances launched in those compute node groups use the specified AMI and its included AWS PCS agent software. The AWS PCS agent enables an EC2 instance to register itself as part of the cluster. To use the latest AWS PCS agent software, you must update your custom AMIs. For more information, see Step 2 – Install the AWS PCS agent in Custom Amazon Machine Images (AMIs) for AWS PCS.

AWS PCS agent version	Release date	Release notes
v1.3.2-1	March 10, 2026	• Fixed an issue where compute nodes running RHEL 8.10 or Rocky Linux 8.10 failed to bootstrap due to a faulty `curl` SigV4 backport in those operating systems.
v1.3.1-1	November 7, 2025	• Improved disablement of hyperthreading by using the \`smt/control\` sysfs parameter when available. • Fixed a potential race condition when the CPU is locked during boot while the PCS Agent attempts to disable hyperthreading. • Fixed issue that caused the `InstanceId` and `InstanceType` fields of the Slurm compute nodes to be populated with a timestamp and a hyphen, respectively.

AWS PCS agent version	Release date	Release notes
v1.3.0-1	November 3, 2025	• Added support for new operating systems: Amazon Linux 2023, Ubuntu 24, RHEL 8, Rocky 8.
v1.2.2-1	October 16, 2025	• Allowed instance metadata queries to an IPv6 endpoint if an IPv4 endpoint isn't available. • Fixed an issue that prevented hyperthreading from being disabled if the kernel returned sibling threads as CPU ID ranges. • Fixed an issue that produced false failure messages in logs when hyperthreading was successfully disabled.
v1.2.1-1	June 19, 2025	• The AWS PCS agent now tries to start slurmd for up to 30 minutes if the controller isn't available. • Fixed an issue that produced an incorrect slurmd configuration if the response to RegisterComputeNodeGroupInstance contained a SLURMDBD endpoint.
v1.2.0-1	March 7, 2025	• Enabled support for IPv6 in `slurmd.conf` .

AWS PCS agent version	Release date	Release notes
v1.1.1-1	December 13, 2024	- Fixed an issue where an incorrect Slurm version was reported in the call to RegisterComputeNodeGroupInstance. - Fixed an issue where instance metadata wasn't fetched correctly if a custom script in `/opt/aws/pcs/etc/bootstrap_hooks/` was executed.
v1.1.0-1	December 6, 2024	- Enabled custom scripts in `/opt/aws/pcs/etc/bootstrap_hooks/` to run before bootstrap steps.
v1.0.1-1	October 22, 2024	- Fixed an issue where NVIDIA devices didn't work when `slurmd` started on GPU-enabled instances.
v1.0.0-1	August 28, 2024	- Initial release.

Slurm scheduler in AWS PCS

Slurm is an open-source workload manager designed for Linux clusters that provides job scheduling, resource allocation, and job monitoring capabilities for HPC workloads. AWS PCS supports the Slurm scheduler to manage your cluster workloads.

Topics

Slurm versions in AWS PCS

SchedMD continually enhances Slurm with new capabilities, optimizations, and security patches. SchedMD releases a new major version at regular intervals and plans to support up to 3 versions at any given time. AWS PCS is designed to automatically update the Slurm controller with patch versions.

When SchedMD ends support for a particular major version, AWS PCS designates that version as End of Life (EOL). After EOL, no new clusters can be created with that version, though existing clusters can continue running for up to 12 months without guaranteed support. AWS PCS sends advance notice if a Slurm major version is close to EOL, to help customers know when to upgrade their clusters to a newer supported version.

We recommend you use the latest supported Slurm version to deploy your cluster, to access the most recent advancements and improvements.

Supported Slurm versions in AWS PCS

The following table shows the supported Slurm versions and important dates and information for each version.

Slurm version	SchedMD release date	AWS PCS release date	AWS PCS EOL date	Minimum compatible AWS PCS agent version	Supported AWS PCS sample AMIs
25.05	5/29/2025	10/16/2025	5/31/2027	1.0.0-1	• `aws-pcs-sample_ami-amzn2-x86_64-slurm-25.05` • `aws-pcs-sample_ami-amzn2-arm64-slurm-25.05`
24.11	11/29/2024	5/14/2025	5/31/2026	1.0.0-1	• `aws-pcs-sample_ami-amzn2-x86_64-slurm-24.11` • `aws-pcs-sample_ami-amzn2-arm64-slurm-24.11`

Unsupported Slurm versions in AWS PCS

The following table shows Slurm versions that aren't supported in AWS PCS.

Slurm version	SchedMD release date	AWS PCS release date	AWS PCS EOL date
24.05	5/30/2024	12/18/2024	11/30/2025
23.11	11/21/2023	8/28/2024	5/31/2025

Release notes for Slurm versions in AWS PCS

This topic describes important changes for each Slurm version currently supported in AWS PCS. We recommend you review the changes between the old and new versions when you upgrade your cluster.

Slurm 25.05

Changes implemented in AWS PCS

- The Slurm requeue_on_resume_failure SchedulerParameter is now Enabled by default.
- "stderr" was removed as an option for LogTimeFormat, as it was disabled in Slurm 25.05.
- AWS PCS supports Multi-cluster sackd configuration: login node can access multiple clusters.

For more information about Slurm 25.05, see the following publications:

- SchedMD release announcement: https://www.schedmd.com/slurm-version-25-05-0-is-now-available/
- SchedMD release notes: https://github.com/SchedMD/slurm/blob/slurm-25-05-0-1/RELEASE_NOTES.md

Slurm 24.11

Changes implemented in AWS PCS

- AWS PCS supports Slurm accounting. For more information, see Slurm accounting in AWS PCS.

For more information about Slurm 24.11, see the following publications:

- SchedMD release announcement
- SchedMD release notes

Slurm 24.05

Changes implemented in AWS PCS

- The new Slurm Step Manager module is now enabled by default in AWS PCS. This module provides significant benefits by offloading step management from the central controller to compute nodes, substantially improving system concurrency in environments with heavy step usage. To support this configuration and better isolate `Prolog` and `Epilog` process execution, new prolog flags (`Contain`, `Alloc`) are enabled.
- Hierarchical communication from controller to compute nodes is enabled to optimize Slurm intra-node communication, which improves scalability and performance. Additionally, the routing configuration now uses partition node lists for communications from the controller, instead of the plugin's default routing algorithm, enhancing system resiliency.
- A new hash plugin `HashPlugin=hash/sha3` replaces the previous `hash/k12 plugin`. This is now enabled by default in AWS PCS clusters.
- Slurm controller logs now include enhanced auditing capabilities for all inbound remote procedure calls (RPC) to `slurmctld`. The logs include the source address, authenticated user, and RPC type before connection processing.

For more information about Slurm 24.05, see the following publications:

- SchedMD release announcement
- SchedMD release notes

Slurm 23.11

Slurm settings you can change in AWS PCS

- The `SuspendTime` defaults to `60`. Use the AWS PCS `scaleDownIdleTimeInSeconds` configuration parameter to set it. For more information, see the `scaleDownIdleTimeInSeconds` parameter of the `ClusterSlurmConfiguration` data type in the *AWS PCS API Reference*.

- The `MaxJobCount` and `MaxArraySize` is based on the size you choose for the cluster. For more information, see the [`size`] parameter of the `CreateCluster` API action in the *AWS PCS API Reference*.
- The `SelectTypeParameters` Slurm setting defaults to `CR_CPU`. You can provide it as a value for `slurmCustomSettings` to set it when you create a cluster. For more information, see the `slurmCustomSettings` parameter of the `CreateCluster` API action and SlurmCustomSetting in the *AWS PCS API Reference*.
- You can set `Prolog` and `Epilog` at the cluster level. You can provide it as a value for `slurmCustomSettings` to set it when you create a cluster. For more information, see `CreateCluster` and SlurmCustomSetting in the *AWS PCS API Reference*.
- You can set `Weight` and `RealMemory` at the compute node group level. You can provide it as a value for `slurmCustomSettings` to set it when you create a compute node group. For more information, see `CreateComputeNodeGroup` and SlurmCustomSetting in the *AWS PCS API Reference*.

Frequently asked questions about Slurm versions in AWS PCS

AWS PCS maintains support for multiple Slurm versions. When a new Slurm version is introduced, AWS PCS provides technical support and security patches until that version reaches its end of support (EOS) from SchedMD. AWS PCS refers to the EOS date for a Slurm version as end of life (EOL) to be consistent with AWS terminology.

How long does AWS PCS support a Slurm version?

AWS PCS support for Slurm versions aligns with SchedMD’s support cycles for major versions. AWS PCS supports the current version and the 2 most recent previous major versions. When SchedMD releases a new major version, AWS PCS ends support for the oldest supported version. AWS PCS releases new major versions of Slurm as soon as possible but there might be a delay between SchedMD's release and its availability in AWS PCS.

How do my clusters get new Slurm patch version releases?

To address bugs and security fixes, AWS PCS is designed to automatically apply patches to cluster controllers that run in internal service-owned accounts. To install patches on EC2 instances in your AWS account, update the Amazon Machine Image (AMI) for your compute node groups and update the compute node groups to use the updated AMI. For more information, see Custom Amazon Machine Images (AMIs) for AWS PCS.

Note

Slurm controllers are unavailable while we update them. Running jobs aren't affected. Jobs submitted before the cluster's controller became unavailable are held until the controller is available.

How am I informed about an upcoming Slurm version EOL event?

We send you an email message 6 months before the EOL date. We send you an email message each month before the EOL, with a final email message 1 week before the EOL date. After the EOL date, we send monthly email messages for 12 months to customers running AWS PCS clusters with EOL Slurm versions. We might suspend a cluster with an EOL Slurm version if security vulnerabilities are identified for that version.

How can I determine if the Slurm version used by my cluster is running an EOL Slurm version?

We send you an email message to notify you that you have a running cluster with an EOL Slurm version. We post an alert to the AWS Health Dashboard alerts that contains the details of your clusters with EOL Slurm versions. You can also use the AWS PCS console to identify the clusters with EOL Slurm versions.

What do I have to do if my Slurm version is near or beyond EOL?

Create a new cluster with a newer supported version of Slurm and update the Slurm version in your compute node group AMIs. The Slurm version in your AMIs and running EC2 instances can’t be more than 2 versions behind the cluster’s Slurm version. For more information, see Custom Amazon Machine Images (AMIs) for AWS PCS.

What will happen if I don’t switch to a newer version of Slurm by the EOL date?

You can’t create new clusters with an EOL Slurm version. Existing clusters can operate for up to 12 months without AWS support, and no immediate action is required to maintain their operation. After the EOL date, support, security updates, and availability are not guaranteed. We might suspend a cluster for security reasons. We strongly recommend you use a supported Slurm version to maintain security and support for your AWS PCS clusters.

What are the risks of operating a cluster with EOL Slurm versions?

Clusters with EOL Slurm versions present significant security and operational risks. Without SchedMD's active monitoring, security vulnerabilities might remain undetected or unaddressed. If critical vulnerabilities are discovered, we might suspend your clusters immediately.

What happens to my jobs, cluster compute, storage and networking resources when my cluster is suspended?

All resources managed by AWS PCS are terminated. This includes the Slurm controller, compute node groups, and EC2 instances. Any jobs running on compute instances are immediately terminated, and the cluster enters a suspended state. Customer-managed resources, such as external file systems, remain intact. You can use the AWS PCS console and API actions to access the cluster's configuration.

Can I restart a suspended cluster to resume its remaining jobs?

No, you can’t restart a suspended cluster. You can use your suspended cluster’s configuration to create a new cluster with a supported Slurm version. You can run your remaining jobs if you saved them in an external file system.

Can I request an extension beyond the 12-month grace period?

No, you can’t request an extension to run your cluster beyond the 12-month grace period. We provide the extended time to help you switch to a supported Slurm version. To avoid disruption to your cluster operations, we recommend you switch before your Slurm version reaches EOL.

Slurm accounting in AWS PCS

You can enable accounting on your new AWS PCS clusters to monitor cluster usage, enforce resource limits, and manage fine-grained access control to specific queues or compute node groups. AWS PCS creates and manages the accounting database for your cluster, eliminating the need for you to create and manage your own separate accounting database. AWS PCS uses the accounting feature in Slurm. For more information about the accounting feature in Slurm, see the Slurm documentation at SchedMD.

To use accounting, enable it when you create a new cluster and optionally set accounting parameters. After your cluster status is `Active` and has compute node groups, you can connect to the Linux shell of a login node to perform accounting functions, such as viewing job data with the Slurm `sacct` command.

Note

Accounting is supported for Slurm 24.11 or later.

AWS PCS console

On the **Create cluster** page, You must select a valid version of Slurm (version 24.11 or later). Under **Scheduler settings**, enable **Accounting**.

AWS PCS API

Provide the `accounting` configuration in your call to the `CreateCluster` API action. In the `accounting` object, set the `mode` to `STANDARD`. For more information, see [CreateCluster]() and [Accounting]() in the *AWS PCS API Reference*.

The following example uses the AWS CLI to call the `CreateCluster` API action. The parameter value substring `accounting='{mode=STANDARD}'` enables accounting.

```
aws pcs create-cluster --cluster-name cluster-name \
                       --scheduler type=SLURM,version=24.11 \
                       --size SMALL \
                       --networking subnetIds=cluster-subnet-
id,securityGroupIds=cluster-security-group-id \
                       --slurm-configuration
 scaleDownIdleTimeInSeconds=180,accounting='{mode=STANDARD}',slurmCustomSettings='[{paramete
```

Important

You get additional billing charges if you enable accounting. For more information, see the [AWS PCS pricing page]().

Modifying accounting settings

You can enable or disable accounting on existing clusters without rebuilding your infrastructure. For more information, see [Updating a cluster in AWS PCS]().

When you disable accounting, billing for the accounting feature stops as soon as the cluster enters the `UPDATING` state. When you enable accounting, billing begins when the cluster successfully returns to the `ACTIVE` state.

Key concepts for Slurm accounting in AWS PCS

The following concepts are specific to AWS PCS and control how AWS PCS implements Slurm accounting.

Accounting database

AWS PCS stores your accounting data in a database created in an AWS account that AWS owns. You don't have access to the `slurmdbd.conf.`

Default purge time

This AWS PCS setting specifies the retention period (in days) for all accounting record types (jobs, events, reservations, steps, suspensions, transactions, usage data). For example, if the value is 30, AWS PCS retains accounting records for 30 days. You provide this value when you create the cluster. If you don't provide a value, AWS PCS retains accounting records in the database indefinitely.

AWS PCS console

> You specify default purge time as part of the steps to create a cluster. On the **Create cluster** page, You must select a valid version of Slurm (version 24.11 or later) and enable accounting. Under **Scheduler settings**, provide an integer value for **Default purge time (days)**.

AWS PCS API

> Specify the `defaultPurgeTimeInDays` as part of the `accounting` information you provide in your call to the `CreateCluster` API action. For more information, see CreateCluster and Accounting in the *AWS PCS API Reference*.
>
> **Note**
>
> When you use the AWS PCS API to create a cluster, the default value for `defaultPurgeTimeInDays` is `-1` and `0` isn't a valid value.

Accounting policy enforcement

This setting determines how strictly Slurm enforces job submission rules, resource limits, and accounting policies for your cluster. This setting corresponds to the `AccountingStorageEnforce` parameter in your cluster's `slurm.conf` file. You can select any combination of enforcement options. If you don’t select any options, there are no accounting constraints applied to jobs on the cluster. AWS PCS supports the following options:

- **associations** — job-to-account mapping
- **limits** — resource constraints
- **QoS** — quality of service requirements
- **safe mode** — guaranteed completion within limits
- **nosteps** — disable step accounting
- **nojobs** — disable job accounting

For more information about these options, see the Slurm documentation at SchedMD.

AWS PCS console

> You set the options as part of the steps to create a cluster. On the **Create cluster** page, You must select a valid version of Slurm (version 24.11 or later) and enable accounting. Select the options you want from the **Accounting policy enforcement** dropdown list under **Scheduler settings**.

AWS PCS API

> In Slurm, these options are set in a cluster's `slurm.conf` file. You don't have direct access to the `slurm.conf` for your AWS PCS cluster. Instead, you provide `SlurmCustomSettings` to the `CreateCluster` API action when you create a cluster. For more information, see CreateCluster in the *AWS PCS API Reference*.

Get the accounting configuration for an existing AWS PCS cluster

The Slurm accounting configuration is included in the Slurm configuration for your cluster.

AWS PCS console

1. Choose **Clusters** from the navigation pane.

2. Choose the cluster name from the list.
3. On the **Configuration** tab, find the accounting configuration under **Slurm configuration**

AWS PCS API

Use the `GetCluster` API action to get the cluster configuration. You can find the accounting configuration in the `slurmConfiguration`. The setting for `mode` and the value of `defaultPurgeTimeInDays` are under `accounting`. The selected accounting policy enforcement options are under `slurmCustomSettings`. For more information, see GetCluster in the *AWS PCS API Reference*.

Slurm REST API in AWS PCS

AWS PCS provides managed support for Slurm's native REST API through `slurmrestd`, delivering an HTTP interface for programmatic cluster interaction. You can submit jobs, monitor cluster status, and manage resources through standard HTTP requests without requiring direct shell access to your cluster.

Common use cases

The Slurm REST API supports various integration scenarios:

- **Web Application Integration**: Build custom frontends and web applications that submit and manage jobs directly.
- **Jupyter Notebook Integration**: Allows researchers to submit jobs from notebook environments without leaving their development workflow.
- **Partner Solution Integration**: Connect third-party HPC tools and workflow managers to your AWS PCS clusters.
- **Programmatic Cluster Management**: Automate job submission, monitoring, and resource management workflows.
- **Research Computing Workflows**: Support academic and enterprise research enviornments that require API-driven job management.

Requirements and limitations

Before using the Slurm REST API, review these details:

- Your cluster must use Slurm version 25.05 or higher.
- The API endpoint will only be accessible via private IP address within your cluster's VPC.
- Your cluster security group must allow HTTP traffic on port 6820.
- Authentication requires JWT tokens with specific user identity claims.

Current limitations include:

- Tokens generated by `scontrol token` are not supported.
- `X-SLURM-USER-NAME` header impersonation is not available.
- Some functionality requires Slurm accounting to be enabled.
- Not compatible with the Slurm CLI filter plugin mechanism.
- Connections to the REST API endpoint are not encrypted with TLS.

Topics

Enabling Slurm REST API in AWS PCS

Enable the Slurm REST API to access your cluster's HTTP interface for programmatic job management and monitoring. You can enable this feature during cluster creation or update an existing cluster that meets the requirements.

Prerequisites

Before enabling the Slurm REST API, ensure you have:

- **Cluster version**: Slurm version 25.05 or higher.
- **Security group**: Rules allowing HTTP traffic on port 6820 from your desired sources.

Procedure

To enable Slurm REST API on a new cluster

AWS Management Console

1. Open the AWS PCS console at https://console.aws.amazon.com/pcs/.
2. Choose **Create cluster**.
3. Under **Cluster details**, choose Slurm version 25.05 or higher.
4. Configure the other cluster settings as needed.
5. In the **Scheduler configuration** section, set **REST API** to **Enabled**.
6. Configure your cluster security group to allow HTTP traffic on port 6820 from your desired sources.
7. Complete the cluster creation process.

AWS CLI

1. Add a Slurm REST configuration when creating your cluster.

```
aws pcs create-cluster --region region \
    --cluster-name my-cluster \
    --scheduler type=SLURM, version=25.05 \
    --size SMALL \
    --networking subnetIds=subnet-ExampleId1,securityGroupIds=sg-ExampleId1 \
    --slurm-configuration slurmRest='{mode=STANDARD}'
```

2. Configure your cluster security group to allow HTTP traffic on port 6820 from your desired sources.

To enable Slurm REST API on an existing cluster

AWS Management Console

1. Open the AWS PCS console at https://console.aws.amazon.com/pcs/.
2. Choose your cluster from the list.
3. Verify that your cluster uses Slurm version 25.05 or higher in the cluster details.
4. Choose **Edit cluster**.
5. In the **Scheduler configuration** section, set **REST API** to **Enabled**.
6. Choose **Update cluster** to apply the changes.

7. Configure your cluster security group to allow HTTP traffic on port 6820 from your desired sources.

AWS CLI

1. Update your cluster with a Slurm REST configuration, as in this example.

   ```
   aws pcs update-cluster --cluster-identifier my-cluster \
       --slurm-configuration 'slurmRest={mode=STANDARD}'
   ```

2. Configure your cluster security group to allow HTTP traffic on port 6820 from your desired sources.

What happens after enabling

When you enable the REST API, AWS PCS automatically:

- Generates a JWT signing key and stores it in AWS Secrets Manager.
- Exposes the API endpoint at `https://<clusterPrivateIpAddress>:6820` within your VPC.
- Updates your cluster configuration to show the REST API endpoint details.

You can now authenticate and use the REST API for job management and cluster operations.

Authenticating with Slurm REST API in AWS PCS

The Slurm REST API in AWS PCS uses JSON Web Token (JWT) authentication to ensure secure access to your cluster resources. AWS PCS provides a managed signing key stored in AWS Secrets Manager, which you use to generate JWT tokens containing specific user identity claims.

Prerequisites

Before authenticating with the Slurm REST API, ensure you have:

- **Cluster configuration**: AWS PCS cluster with Slurm 25.05+ and REST API enabled.
- **AWS permissions**: Access to AWS Secrets Manager for the JWT signing key.
- **User information**: Username, POSIX user ID, and one or more POSIX group IDs for your cluster account.
- **Network access**: Connectivity within your cluster's VPC with security group allowing port 6820.

Procedure

To retrieve the Slurm REST API endpoint address

AWS Management Console

1. Open the AWS PCS console at https://console.aws.amazon.com/pcs/.
2. Choose your cluster from the list.
3. In the cluster configuration details, locate the **Endpoints** section.
4. Note the private IP address and port for **Slurm REST API (slurmrestd)**.
5. You can make API calls by sending properly formatted HTTP requests to this address.

AWS CLI

1. Query your cluster status with `aws pcs get-cluster`. Look for the `SLURMRESTD` endpoint in the `endpoints` field in the response. Here is an example:

   ```
   "endpoints": [
         {
             "type": "SLURMCTLD",
             "privateIpAddress": "192.0.2.1",
             "port": "6817"
         },
         {
             "type": "SLURMRESTD",
             "privateIpAddress": "192.0.2.1",
             "port": "6820"
         }
     ]
   ```

2. You can make API calls by sending properly formatted HTTP requests to `http://<privateIpAddress>:<port>/`

To retrieve the JWT signing key

1. Open the AWS PCS console at https://console.aws.amazon.com/pcs/.
2. Choose your cluster from the list.
3. In the cluster configuration details, locate the **Scheduler Authentication** section.

4. Note the **JSON Web Token (JWT) key** ARN and version.

5. Use the AWS CLI to retrieve the signing key from Secrets Manager:

```
aws secretsmanager get-secret-value --secret-
id arn:aws:secretsmanager:region:account:secret:name --version-id version
```

To generate a JWT token

1. Create a JWT with the following required claims:

   - `exp` – Expiration time in seconds since 1970 for the JWT
   - `iat` – Current time in seconds since 1970
   - `sun` – The username for authentication
   - `uid` – The POSIX user ID
   - `gid` – The POSIX group ID
   - `id` – Additional POSIX identity properties
     - `gecos` – User comment field, often used to store a human-readable name
     - `dir` – User's home directory
     - `shell` – User's default shell
     - `gids` – List of additional POSIX group IDs the user is in

2. Sign the JWT using the signing key retrieved from Secrets Manager.

3. Set an appropriate expiration time for the token.

Note

As an alternative to the `sun` claim, you can provide any of the following:

- `username`
- A custom field name that you define via the `userclaimfield` in the `AuthAltParameters Slurm custom settings`
- A name field within the `id` claim

To authenticate API requests

1. Include the JWT token in your HTTP requests using one of these methods:

   - **Bearer token** – Add `Authorization: Bearer <jwt>` header
   - **Slurm header** – Add `X-SLURM-USER-TOKEN: <jwt>` header

2. Make HTTP requests to the REST API endpoint:

   Here is an example of accessing the `/ping` API using curl and the `Authorized: Bearer` header.

   ```
   curl -X GET -H "Authorization: Bearer <jwt>" \
         http://<privateIpAddress>:6820/slurm/v0.0.43/ping
   ```

Example JWT generation

Fetch the AWS PCS cluster JWT signing key and store it as a local file. Replace values for **aws-region**, **secret-arn**, and **secret version** with values appropriate for your cluster.

```
#!/bin/bash
SECRET_KEY=$(aws secretsmanager get-secret-value \
  --region aws-region \
  --secret-id secret-arn \
  --version-stage secret-version \
  --query 'SecretString' \
  --output text)
echo "$SECRET_KEY" | base64 --decode > jwt.key
```

This Python example illustrates how to use the signing key to generate a JWT token:

```
#!/usr/bin/env python3

import sys
import os
import pprint
import json
import time
from datetime import datetime, timedelta, timezone
from jwt import JWT
from jwt.jwa import HS256
```

```
from jwt.jwk import jwk_from_dict
from jwt.utils import b64decode,b64encode
if len(sys.argv) != 3:
    sys.exit("Usage: gen_jwt.py [jwt_key_file] [expiration_time_seconds]")
SIGNING_KEY = sys.argv[1]
EXPIRATION_TIME = int(sys.argv[2])
with open(SIGNING_KEY, "rb") as f:
    priv_key = f.read()
signing_key = jwk_from_dict({
    'kty': 'oct',
    'k': b64encode(priv_key)
})
message = {
    "exp": int(time.time() + EXPIRATION_TIME),
    "iat": int(time.time()),
    "sun": "ec2-user",
    "uid": 1000,
    "gid": 1000,
    "id": {
        "gecos": "EC2 User",
        "dir": "/home/ec2-user",
        "gids": [1000],
        "shell": "/bin/bash"
    }
}
a = JWT()
compact_jws = a.encode(message, signing_key, alg='HS256')
print(compact_jws)
```

The script will print a JWT to the screen.

```
abcdefgtjwttoken...
```

Using Slurm REST API for job management in AWS PCS

Slurm REST API overview

The Slurm REST API provides programmatic access to cluster management functions through HTTP requests. Understanding these key characteristics will help you effectively use the API with AWS PCS:

- **Access Protocol**: The API uses HTTP (not HTTPS) for communication within your cluster's private network.
- **Connection Details**: Access the API using your cluster's private IP address and the `slurmrestd` port (typically 6820). The full base URL format is `http://<privateIpAddress>:6820`.
- **API Versioning**: The API version corresponds to your Slurm installation. For Slurm 25.05, use version **v0.0.43**. The version number changes with each Slurm release. You can find the currently supported API versions in the Slurm release notes.
- **URL Structure**: The URL structure for the Slurm REST API is `http://<privateIpAddress>:<port>/<api-version>/<endpoint>`. Detailed usage information for REST API endpoints can be found in the Slurm documentation.

Prerequisites

Before using the Slurm REST API, ensure you have:

- **Cluster configuration**: AWS PCS cluster with Slurm 25.05+ and REST API enabled.
- **Authentication**: Valid JWT token with proper user identity claims.
- **Network access**: Connectivity within your cluster's VPC with a security group allowing port 6820.

Procedure

To submit a job using the REST API

1. Create a job submission request with the required parameters:

```
{
  "job": {
    "name": "my-job",
    "partition": "compute",
    "nodes": 1,
    "tasks": 1,
    "script": "#!/bin/bash\necho 'Hello from Slurm REST API'"
  }
}
```

2. Submit the job using an HTTP POST request:

```
curl -X POST \
  -H "Authorization: Bearer <jwt>" \
  -H "Content-Type: application/json" \
  -d '<job-json>' \
  https://<privateIpAddress>:6820/slurm/v0.0.43/job/submit
```

3. Note the job ID returned in the response for monitoring purposes.

To monitor job status

1. Get information about a specific job:

```
curl -X GET -H "Authorization: Bearer <jwt>" \
    https://<privateIpAddress>:6820/slurm/v0.0.43/job/<job-id>
```

2. List all jobs for the authenticated user:

```
curl -X GET -H "Authorization: Bearer <jwt>" \
    https://<privateIpAddress>:6820/slurm/v0.0.43/jobs
```

To cancel a job

- Send a DELETE request to cancel a specific job:

```
curl -X DELETE -H "Authorization: Bearer <jwt>" \
    https://<privateIpAddress>:6820/slurm/v0.0.43/job/<job-id>
```

Slurm REST API frequently asked questions in AWS PCS

This section answers frequently asked questions about the Slurm REST API in AWS PCS.

What is the Slurm REST API?

The Slurm REST API is an HTTP interface that allows you to interact with the Slurm workload manager programmatically. You can use standard HTTP methods like GET, POST, and DELETE to submit jobs, monitor cluster status, and manage resources without requiring command-line access to the cluster.

Can I use tokens generated by `scontrol token`?

No, standard `scontrol token` output is not compatible with AWS PCS. The PCS Slurm REST API requires enriched JWT tokens containing specific identity claims that include username(`sun`), POSIX user ID(`uid`), and group IDs(`gids`). Standard Slurm tokens lack these required claims and will be rejected by the API.

Can I access the API from outside my VPC?

No, the REST API endpoint is only accessible from within your VPC using the Slurm controller's private IP address. To enable external access, implement AWS services such as Application Load Balancer with VPC Link, API Gateway, or establish VPC peering or VPN connections for secure connectivity.

Why does the API use HTTP instead of HTTPS?

The Slurm REST API is intended to be an internal endpoint within your cluster's private network. For production deployments requiring encryption, you can implement SSL/TLS termination at a higher level in your architecture, such as through an API gateway, load balancer, or reverse proxy.

How do I control access to the REST API?

Configure your cluster's security group rules to restrict access to port 6820 on the Slurm controller. Set inbound rules to allow connections only from trusted IP ranges or specific sources within your VPC, blocking unauthorized access to the API endpoint.

How do I rotate the JWT signing key?

Put your cluster in maintenance mode with no active instances, then initiate key rotation through AWS Secrets Manager. After rotation completes, re-enable the queues. All existing JWT tokens will become invalid and must be regenerated using the new signing key from Secrets Manager.

Do I need Slurm accounting enabled to use the REST API?

No, Slurm accounting is not required for basic REST API operations like job submission and monitoring. However, the entire `/slurmdb` endpoint requires accounting to be active.

What third-party tools work with the AWS PCS REST API?

Many existing Slurm REST API clients should work with AWS PCS, including Slurm Exporter for Prometheus, SlurmWeb, and custom applications that follow the standard Slurm REST API format. However, tools that rely on `scontrol token` for authentication will need modification to work with AWS PCS JWT requirements.

Are there any additional costs for using the REST API?

No, there are no additional charges for enabling or using the Slurm REST API feature. You only pay for the underlying cluster resources as usual.

How can I troubleshoot the REST API?

- **Network connectivity issues**

  If you cannot reach the API endpoint, you'll see connection timeouts or "connection refused" errors when making HTTP requests to the cluster controller.

  What to do: Verify your client is in the same VPC or has proper network routing, and confirm your security group allows HTTP traffic on port 6820 from your source IP or subnet.

- **Slurm REST authentication issues**

  If your JWT token is invalid, expired, or improperly signed, API requests will return "Protocol authentication error" in the errors field of the response.

  Example error message:

  ```
  {
  "errors": [
      {
      "description": "Batch job submission failed",
      "error_number": 1007,
      "error": "Protocol authentication error",
      "source": "slurm_submit_batch_job()"
      }
    ]
  }
  ```

  What to do: Check that your JWT token is properly formatted, not expired, and signed with the correct key from Secrets Manager. Verify that the token is properly formed and includes the required claims and that you're using the correct authentication header format.

- **Job failing to run after submission**

  If your JWT token is valid but contains incorrect internal structure or content, jobs may have entered a paused (PD) state with reason code `JobAdminHead`. Use `scontrol show job`

<job-id> to inspect the job – you'll see `JobState=PENDING, Reason=JobHeldAdmin`, and `SystemComment=slurm_cred_create failure, holding job`.

What to do: The root cause may be mistaken values in JWT. Verify that the token is properly structured and includes the required claims as per the PCS documentation.

- **Working directory permission issues**

  If the user identity specified in your JWT lacks write permissions to the job's working directory, the job will fail with permission errors, similar to using `sbatch --chdir` with an inaccessible directory.

  What to do: Ensure the user specified in your JWT token has appropriate permissions for the job's working directory.

Rebooting compute nodes with Slurm in AWS PCS

AWS PCS supports Slurm's native `scontrol reboot` command. Use this command to reboot compute nodes without EC2 instance replacement. Other reboot methods (Amazon EC2 console, AWS CLI, automated patches, or system maintenance) cause AWS PCS to consider the EC2 instance unhealthy and replace it.

Benefits of Slurm reboot

Slurm reboot provides several advantages for cluster maintenance:

- **Preserve capacity** – Avoid losing capacity-constrained EC2 instances to other customers.
- **Reduce costs** – Eliminate unnecessary instance replacement cycles and continued billing for idle nodes.
- **Faster recovery** – No provisioning delays compared to instance replacement.
- **Operational flexibility** – Clear memory leaks, remove temporary files, and recover nodes from degraded states.

When to use Slurm reboot

Use Slurm reboot for common operational maintenance scenarios:

- **Troubleshooting –** Resolve performance issues or unresponsive processes, especially for GPU nodes.
- **Resource cleanup –** Clear memory leaks, temporary files in `/tmp`, or stuck processes that affect job performance.
- **Recovery –** Recover nodes from hung or degraded states before requiring full node replacement.

Limitations

- Only Slurm Admin users (root users) can execute reboot commands.
- Reboot support is limited to `scontrol reboot` only.
- RebootProgram configuration isn't supported.
- No console interface – command-line only.

Topics

- Reboot a compute node using Slurm in AWS PCS
- Cancel a pending reboot in AWS PCS
- Slurm reboot frequently asked questions in AWS PCS
- Troubleshooting Slurm reboot issues in AWS PCS

Reboot a compute node using Slurm in AWS PCS

Use Slurm's native reboot command to resolve performance issues, clear resource problems, or recover from degraded states without loss of EC2 instance capacity.

Prerequisites

- Slurm Admin privileges (root user access)
- Access to a login node in the AWS PCS cluster

Procedure

1. Connect to a login node through the EC2 console.

    a. In the EC2 console, choose **Instances**.

   b. Select your login node instance.

   c. Choose **Connect**.

2. Identify the target compute node name using `sinfo` or `scontrol show node`.

```
sinfo
# or
scontrol show node
```

3. Execute the reboot command using one of these options:

> ⚠ **Warning**
>
> Don't use `nextstate=DOWN` with the `scontrol reboot` command. This parameter marks the node as unhealthy and triggers instance replacement.

- Basic reboot (waits for node to become idle):

```
scontrol reboot nodename
```

- Immediate reboot (drains node and reboots when jobs complete):

```
scontrol reboot ASAP nodename
```

- Reboot with reason:

```
scontrol reboot ASAP reason="troubleshooting" nodename
```

- Reboot with resume state:

```
scontrol reboot ASAP nextstate=RESUME nodename
```

4. Monitor reboot progress using `scontrol show node`.

```
scontrol show node nodename
```

5. Verify the node returns to service after reboot completion.

Cancel a pending reboot in AWS PCS

Cancel a pending reboot to avoid unnecessary downtime when the issue has been resolved or when reboot is no longer needed.

Prerequisites

- Slurm Admin privileges
- Node must have a pending reboot (showing "reboot issued" status)
- Access to login node for command execution

Procedure

1. Connect to the login node.
2. Verify the node has a pending reboot using `scontrol show node`.

   ```
   scontrol show node nodename
   ```

   Look for "reboot issued" in the node status.
3. Execute the cancel command.

   ```
   scontrol cancel_reboot nodename
   ```
4. Verify reboot cancellation and node status return to normal.

   ```
   scontrol show node nodename
   ```

Slurm reboot frequently asked questions in AWS PCS

Find answers to common questions about using Slurm reboot in AWS PCS.

What is Slurm reboot support?

Support for the native Slurm `scontrol reboot` command. Use this command to reboot compute nodes without automatic instance replacement, which preserves EC2 instance capacity and reduces operational costs.

Who can use Slurm reboot commands?

Only Slurm Admin users (root users) can execute reboot commands. Regular users attempting to use `scontrol reboot` will receive a permission denied error from Slurm without affecting the node.

What happens to running jobs during a reboot?

By default, jobs complete normally before reboot occurs. With the ASAP option, the node is drained to prevent new jobs, and reboot happens after current jobs finish. Jobs can be cancelled or requeued for immediate reboots.

How is this different from EC2 console reboot?

Slurm reboot preserves the EC2 instance and avoids replacement, while EC2 console reboots trigger PCS to replace the instance due to failed health checks during the reboot process.

Can I configure custom reboot scripts?

No, RebootProgram configuration is not supported in the initial release. The feature uses standard Slurm reboot behavior without custom script support.

How long does a Slurm reboot take?

Reboot time varies based on instance type, customer boot processes, AMI configuration, and whether jobs need to complete first. The process includes waiting for jobs to complete, physical reboot, health checks, and slurmd daemon registration.

Can I see a history of reboots?

Reboot events are recorded in Slurm logs (slurmctld and slurmd) which can be monitored through CloudWatch. The reason field in node status shows the reboot reason during the process.

What if a node gets stuck during reboot?

If a node doesn't complete the reboot process within ResumeTimeout, it will be marked as DOWN. Check CloudWatch logs for errors, verify network connectivity, and examine slurmd logs. Contact AWS Support if issues persist.

Can I reboot multiple nodes at once?

Yes, you can specify multiple nodes in the reboot command:

```
scontrol reboot ASAP node1,node2,node3
```

How can I reboot a node without waiting for jobs to complete?

For immediate node reboots when facing issues like problematic nodes affecting multi-node jobs, significant performance degradation, or unstable GPU behavior, you have two options:

- **Cancel and Reboot** – First, cancel affected jobs using `scancel <job_id>`, then initiate an immediate reboot using `scontrol reboot ASAP <nodename>`. Running jobs will be terminated and need to be resubmitted after the node recovers.
- **Drain and Requeue (less impactful)** – Start by initiating a drain and reboot with `scontrol reboot ASAP <nodename>`, then requeue affected jobs using `scontrol requeue <job_id>`. This puts jobs back into pending state instead of cancelling them.

What happens if I specify nextstate=DOWN?

If you specify `nextstate=DOWN`, the node will be marked as unhealthy after reboot and trigger instance replacement. To avoid instance replacement, don't specify nextstate or use `nextstate=RESUME`.

Additional resources

- For basic reboot procedures, see Reboot a compute node using Slurm in AWS PCS.
- For troubleshooting reboot issues, see Troubleshooting Slurm reboot issues in AWS PCS.
- For Slurm reboot documentation, see Slurm scontrol documentation.

Troubleshooting Slurm reboot issues in AWS PCS

When you encounter node reboot problems, first check the node status using `scontrol show node` *`nodename`*. Then examine CloudWatch logs for both Slurm (slurmctld and slurmd) and system logs to identify potential errors.

For basic troubleshooting, verify network connectivity, check security group settings, and ensure all required services are running after the reboot. If problems persist after basic troubleshooting steps, contact AWS Support. When reaching out to support, provide relevant log excerpts, node status information, and a timeline of the reboot attempt to help expedite the resolution process.

Additional resources

- For monitoring AWS PCS instances using CloudWatch, see Monitoring AWS PCS instances using Amazon CloudWatch.

- For general troubleshooting, see Troubleshooting problems in AWS Parallel Computing Service.
- For Slurm documentation, see Slurm Troubleshooting Guide.

Configuring custom Slurm settings in AWS PCS

Use custom Slurm settings to configure additional Slurm parameters across Cluster, Queue, and Compute Node Group resources. This release adds support for Slurm settings on Queue resources, providing granular control over partition-specific behaviors.

Benefits of custom Slurm settings

Custom Slurm settings provide sophisticated control over your AWS PCS-based HPC environment. You can implement detailed accounting, enforce access controls, and optimize workload execution through quality-of-service configurations and preemption policies. These capabilities ensure critical jobs receive necessary resources while maintaining efficient cluster utilization. Whether you manage GPU-accelerated workloads, implement fair-share scheduling, or control job lifecycles, custom settings help align your HPC infrastructure with operational requirements and research objectives.

Configuring custom settings

Custom Slurm settings can be configured through the AWS Console, CLI, or SDKs during resource creation or modified later through update operations.

AWS Management Console

> Navigate to **Additional scheduler settings** in the create or edit page for any resource type (cluster, queue, or compute node group).
>
> **To add a new setting**
>
> 1. Choose **Add new setting**.
> 2. Select a **Parameter** name from the dropdown (which includes brief parameter descriptions).
> 3. Provide the corresponding value.

To unset a custom setting

1. Choose **Remove** next to the relevant parameter/value pair.
2. Create or update the resource.

AWS CLI

For programmatic management of custom settings, use the `SlurmCustomSettings` field in create or update operations.

Example– Updating the Prolog parameter on a cluster

```
aws pcs update-cluster --cluster-identifier my-cluster \
--slurm-configuration \
'SlurmCustomSettings=[{parameterName=Prolog,parameterValue="/path/to/prolog.sh"}]'
```

Example– Setting a queue to be the Default on a cluster

```
aws pcs update-queue \
    --cluster-identifier my-cluster \
    --queue-identifier my-queue \
    --slurm-configuration
 'SlurmCustomSettings=[{parameterName=Default,parameterValue=YES}]'
```

Example– Setting custom Features on a compute node group

```
aws pcs update-compute-node-group \
    --cluster-identifier my-cluster \
    --compute-node-group-identifier my-cng-1 \
    --slurm-configuration \
    'SlurmCustomSettings=[{parameterName=Features,parameterValue="gpu,nvme"}]'
```

Validation and error handling

AWS PCS implements a multi-layered validation process for custom Slurm settings. During both create and update operations, we perform synchronous validations that include:

- Field-level checks: We validate individual settings for correct data types, allowed values, and format requirements. For example, we ensure time values are in the correct Slurm format and boolean values use accepted Slurm boolean representations.
- Context-aware validations: Some settings are checked against the broader configuration context. For instance, certain parameters are only valid when Slurm accounting is enabled.
- Inter-setting consistency: We verify that mutually exclusive options aren't set together and that interdependent settings are configured correctly.

If validation fails, you'll receive a `ValidationException` with a specific error code (e.g., InvalidInput), a clear error message describing the issue, and a list of the invalid fields and their respective error details.

While many issues are caught during this initial validation, some complex interactions between settings may only become apparent when applying the configuration. In such cases, the operation will fail with an informative error message, and any partial changes will be rolled back.

Limitations

AWS PCS implements an allow-list approach to protect service security and operational stability. Settings that could compromise service account security or interfere with managed service capabilities are restricted. However, we continuously evaluate customer needs and can add support for additional settings based on customer feedback.

Topics

Custom Slurm settings for AWS PCS clusters

The following custom Slurm settings are supported at the cluster level:

- AccountingStorageEnforce

⚠ **Important**

AWS PCS supports a subset of the options for `AccountingStorageEnforce`. For more information, see Slurm accounting in AWS PCS.

- AccountingStorageTRES
- AccountingStoreFlags
- DefMemPerCPU
- Epilog
- EnforcePartLimits
- FairShareDampeningFactor
- HealthCheckInterval
- HealthCheckNodeState
- HealthCheckProgram
- JobRequeue
- LaunchParameters
- Licenses
- MinJobAge

ⓘ **Note**

AWS PCS supports a minimum value of 5 seconds for `MinJobAge`.

- OverTimeLimit
- PreemptExemptTime
- PreemptMode
- PreemptParameters
- PreemptType
- PriorityCalcPeriod
- PriorityDecayHalfLife
- PriorityFavorSmall

- PriorityFlags
- PriorityMaxAge
- PriorityUsageResetPeriod
- PriorityWeightAge
- PriorityWeightAssoc
- PriorityWeightFairshare
- PriorityWeightJobSize
- PriorityWeightPartition
- PriorityWeightQOS
- PriorityWeightTRES
- PrivateData
- Prolog
- PrologFlags
- PropagatePrioProcess
- PropagateResourceLimits
- PropagateResourceLimitsExcept
- RequeueExit
- RequeueExitHold
- SchedulerParameters
- SelectTypeParameters
- SrunPortRange
- TaskEpilog
- TaskPluginParam
- TaskProlog
- UnkillableStepProgram
- UnkillableStepTimeout

Custom Slurm settings for AWS PCS compute node groups

The following custom Slurm settings are supported at the compute node group level:

- CpuSpecList
- Features
- MemSpecLimit
- RealMemory
- Weight

Custom Slurm settings for AWS PCS queues

The following custom Slurm settings are supported at the queue level:

- AllowAccounts
- AllowQoS
- Default
- DefaultTime
- DenyAccounts
- DenyQoS
- ExclusiveUser
- GraceTime
- MaxTime
- OverSubscribe
- OverTimeLimit
- PreemptMode
- PriorityJobFactor
- PriorityTier
- QOS
- TRESBillingWeights

Troubleshooting custom Slurm settings in AWS PCS

If you encounter errors when creating or updating AWS PCS resources with Slurm custom settings, you can use logging to diagnose and resolve the issues.

Troubleshooting incompatible Slurm custom settings

Problem: You receive an error message similar to the following when performing cluster, compute node group, or queue operations:

```
{OPERATION} failed. The Slurm custom settings of the cluster might be incompatible.
 Check the settings and try again.
```

This error can occur with the following operations:

- CreateCluster
- CreateComputeNodeGroup
- UpdateComputeNodeGroup
- CreateQueue
- UpdateQueue

Solution: Enable logging to understand the specific issue and troubleshoot the incompatible settings.

To troubleshoot incompatible Slurm custom settings

1. Create the cluster if it doesn't exist yet, or ensure your existing cluster is in a state where logging can be enabled.
2. Enable logging for your cluster. For detailed instructions, see Logging and monitoring for AWS PCS.

   > **Note**
   >
   > Logging can be enabled once the cluster is in creation.

3. Review the logs to identify the specific Slurm configuration issue causing the incompatibility.
4. Correct the incompatible custom settings based on the log information and retry the operation.

For information about supported Slurm custom settings, see:

- Custom Slurm settings for AWS PCS clusters

- Custom Slurm settings for AWS PCS compute node groups
- Custom Slurm settings for AWS PCS queues

Extend Slurm functionality on AWS PCS with SPANK plugins

Use SPANK (Slurm Plug-in Architecture for Node and job Kontrol) plugins to extend and modify Slurm's behavior during job launch and execution on AWS PCS clusters. SPANK plugins provide a generic interface to intercept and modify job launch stages.

Install SPANK plugins on your compute node AMI and configure them to customize your Slurm cluster's behavior for your workload requirements. For more information about SPANK, see the SPANK documentation on the SchedMD website.

Contents

Install SPANK plugins on AWS PCS

Follow the plugin's documentation to install SPANK plugins on your AMI.

Compile SPANK plugins for the specific Slurm version on your cluster. The Slurm installer provided by AWS PCS stores Slurm in `/opt/aws/pcs/scheduler/slurm-`*`version`*. When you compile the plugin, specify the Slurm version.

The following example shows how to specify the Slurm version for some plugins:

```
export CFLAGS="-I/opt/aws/pcs/scheduler/slurm-version/include"
```

If you have multiple Slurm versions in the AMI, compile the plugin for each version. Store the compiled plugins in versioned folders.

The following example shows how to specify the destination folder for some plugins:

```
export DESTDIR="your-preferred-versioned-path"
```

⚠ **Important**

Plugins might require different variables. See the official documentation for the plugin that you're installing.

Configure SPANK plugins on AWS PCS

By default, store configuration files in `/etc/aws/pcs/scheduler/slurm-`*`version`*`/plugstack.conf.d/`.

To store your SPANK configuration in a different location, add your locations to a configuration file in the default directory.

The following example shows how to include configuration files from other directories:

```
# content of /etc/aws/pcs/scheduler/slurm-version/any-filename.conf
include path-to-your-configuration-folder/*.conf
include path-to-a-second-configuration-folder/*.conf
```

Store each configuration in a dedicated file, or in a common file. You can use multiple configuration files.

The following examples show sample configuration files:

```
# content of path-to-your-or-default-config-folder/filename-1.conf
required path-to-plugin-1 arguments
optional path-to-plugin-2 arguments
```

```
# content of path-to-your-or-default-config-folder/filename-2.conf
required path-to-plugin-3 arguments
```

For additional information about how to configure your plugins, see the SPANK configuration documentation on the SchedMD website.

⚠ **Important**

Set folder permissions to prevent unauthorized changes to your plugin configuration.

Note

AWS PCS doesn't manage your SPANK plugins. If you get errors related to plugins, check the error logs on your compute nodes.

Note

Slurm incorrectly logs an error similar to the following when it loads your SPANK configuration:

```
error: "Include" failed in file /etc/slurm/plugstack.conf line 3
```

You can ignore this error. It doesn't affect how SPANK plugins work.

Frequently asked questions about SPANK plugins on AWS PCS

This section addresses common questions about installing and configuring SPANK plugins on AWS PCS clusters.

Do I need to install SPANK plugins on both login nodes and compute nodes?

Some SPANK plugins don't require installation on all nodes; but for better compatibility, we recommend you install all SPANK plugins on every node.

What additional configuration is needed for production use of SPANK plugins?

Beyond the basic installation and configuration shown in the examples, production deployments typically require additional setup. Container-based plugins such as Pyxis might require you to set environment variables for Enroot, enable PMI (Process Management Interface), and configure permissions for the container runtime. See the specific plugin's documentation for detailed production deployment requirements.

How do I troubleshoot SPANK plugin issues?

AWS PCS doesn't manage SPANK plugins. Examine error logs on your compute nodes to troubleshoot issues.

Use Slurm CLI Filter Plugins to customize job submission in AWS PCS

AWS PCS supports Slurm CLI Filter Plugins to run custom Lua scripts that validate and modify job submission parameters on login and compute nodes. For detailed information about CLI Filter Plugins, see the cli_filter Plugin API documentation on the SchedMD website.

Requirements

CLI Filter Plugins require Slurm version 24.11 or later and a Lua script deployed to all login and compute nodes.

> ⚠ **Important**
>
> For Slurm versions 24.11 and 25.05, CLI Filter Plugins require installing Slurm using AWS PCS Slurm installer (version 24.11.6-2+ or 25.05.4-1+). For more information about installing Slurm, see Step 3 – Install Slurm.

Limitations and security considerations

- **Security enforcement –** CLI Filter Plugins can be easily bypassed by any user and must not be used for security-critical policies. Users can disable CLI Filter Plugins by providing a custom configuration that has `CLIFilterPlugins` disabled while submitting jobs.
- **Lua implementation only –** Lua script implementation is supported. C implementation is not supported.

Topics

Configure Slurm CLI Filter Plugins on an AWS PCS cluster

Configure CLI Filter Plugins when you create a new AWS PCS cluster. You can enable or disable CLI Filter Plugins on existing clusters using the Update API or console without recreating the cluster.

Prerequisites

Before you configure CLI Filter Plugins, complete these tasks:

- Write and test a Lua script that implements CLI Filter Plugin API
- Name your Lua script exactly `cli_filter.lua`
- Choose a method to deploy your script to all cluster instances (AMI, S3, or file system)
- Verify you are using Slurm version 24.11 or later

Enable CLI Filter Plugins on a new cluster

AWS PCS console

1. Open the AWS PCS console at https://console.aws.amazon.com/pcs/.
2. In the navigation pane, choose **Clusters**.
3. Choose **Create cluster**.
4. Select a valid version of Slurm (version 24.11 or later).
5. Under **Scheduler settings**, expand **Additional scheduler settings**.
6. Add a new Slurm custom setting with **Parameter name** set to `CliFilterPlugins` and **Parameter value** set to `cli_filter/lua`.
7. Complete the remaining cluster configuration and choose **Create cluster**.

AWS PCS API

Provide the `slurmCustomSettings` configuration in your call to the `CreateCluster` API action. Set the `parameterName` to `CliFilterPlugins` and `parameterValue` to `cli_filter/lua`. For more information, see [CreateCluster]() in the *AWS PCS API Reference*.

The following example uses the AWS CLI to call the `CreateCluster` API action. The custom setting `CliFilterPlugins=cli_filter/lua` enables CLI Filter Plugins.

```
aws pcs create-cluster --cluster-name cluster-name \
--scheduler type=SLURM,version=24.11 \
--size SMALL \
--networking subnetIds=cluster-subnet-id,securityGroupIds=cluster-security-group-id
 \
--slurm-configuration \
'slurmCustomSettings=[{parameterName=CliFilterPlugins,parameterValue="cli_filter/
lua"}]'
```

Deploy CLI Filter Plugin scripts

To deploy CLI Filter Plugin scripts to your cluster

1. Ensure all AMIs used in compute node groups have Slurm installed via the AWS PCS Slurm installer.

   > **Note**
   >
   > If you use the AWS PCS Sample AMI for all compute node groups, skip this step. Slurm is already installed.

2. Deploy your `cli_filter.lua` script to `/etc/aws/pcs/scheduler/slurm-<version>/cli_filter.lua` on all instances in the cluster.

   For example, for Slurm version 24.11:

   ```
   /etc/aws/pcs/scheduler/slurm-24.11/cli_filter.lua
   ```

3. Launch all login and compute nodes using your prepared AMIs.
4. Test job submission to verify CLI Filter Plugin executes correctly.

Enable or disable CLI Filter Plugins on existing clusters

You can enable or disable CLI Filter Plugins on existing clusters without rebuilding your infrastructure. For more information, see Updating a cluster in AWS PCS.

AWS PCS console

1. Open the AWS PCS console at https://console.aws.amazon.com/pcs/.
2. In the navigation pane, choose **Clusters**.

3. Select the cluster to update.
4. Choose **Edit** action.
5. On the Edit cluster page, under **Additional scheduler settings**:
    - To enable CLI Filter Plugins: Add a new Slurm custom setting with **Parameter name** set to `CliFilterPlugins` and **Parameter value** set to `cli_filter/lua`.
    - To disable CLI Filter Plugins: Remove the existing `CliFilterPlugins` setting.
6. Choose **Update cluster** to submit the changes.
7. Monitor the cluster status, which shows as "Updating" during the process and "Active" when the update is complete.

AWS PCS API

Use the `UpdateCluster` API action to enable or disable CLI Filter Plugins. For more information, see UpdateCluster in the *AWS PCS API Reference*.

To enable CLI Filter Plugins on an existing cluster:

```
aws pcs update-cluster --cluster-identifier my-cluster \
--slurm-configuration \
'slurmCustomSettings=[{parameterName=CliFilterPlugins,parameterValue="cli_filter/
lua"}]'
```

To disable CLI Filter Plugins on an existing cluster:

```
aws pcs update-cluster --cluster-identifier my-cluster \
--slurm-configuration \
'slurmCustomSettings=[]'
```

Expected results

After you complete the configuration:

- Your cluster is created with CLI Filter Plugin turned on
- Job submissions trigger your custom validation logic before reaching the Slurm controller
- Non-compliant jobs are rejected with your custom error messages

- Compliant jobs proceed normally through the Slurm scheduler

Troubleshooting

CLI Filter Plugin script missing on any node

Symptoms: Job submission fails immediately with plugin loading error.

Likely cause: Script not deployed to all instances or incorrect file path or name.

Resolution: Verify script exists at correct path on all login and compute nodes with exact filename `cli_filter.lua`.

Invalid CLI Filter Plugin configuration

Symptoms: Cluster creation fails with validation error.

Likely cause: `CliFilterPlugins` parameter not set to `cli_filter/lua` format.

Resolution: Use exact parameter value `cli_filter/lua` in `slurmCustomSettings`.

Use Amazon S3 to deploy a CLI Filter Plugin script in AWS PCS

Use S3 to deploy your CLI Filter Plugin script when you want to update job submission logic on a live cluster without rebuilding AMIs. This approach downloads the script from S3 during instance launch using user data.

Prerequisites

Before you deploy your script using S3, complete these tasks:

- Create an S3 bucket with your CLI Filter Plugin Lua script
- Configure IAM instance profile with read access to the S3 bucket
- Set up S3 VPC Gateway endpoint for direct access without internet
- Prepare user-data script to download from S3

To deploy CLI Filter Plugin script using S3

1. Upload your `cli_filter.lua` script to your S3 bucket.
2. Configure your IAM instance profile with S3 read permissions for the bucket.

3. Add shell code to your launch template user-data to download the script:

```
aws s3 cp s3://my-bucket/cli_filter.lua /etc/aws/pcs/scheduler/slurm-24.11/
cli_filter.lua
chmod 644 /etc/aws/pcs/scheduler/slurm-24.11/cli_filter.lua
```

4. Deploy compute node groups with your updated launch templates.
5. Test job submission to verify script functionality.

Expected results

After you complete the S3 deployment:

- CLI Filter Plugin script is automatically downloaded to all instances during launch
- Script updates in S3 are reflected on newly launched instances
- Job submission policies are enforced consistently across the cluster

Troubleshooting

S3 access denied

Symptoms: Instance launch fails or script not downloaded.

Likely cause: Missing IAM permissions or S3 VPC endpoint.

Resolution: Verify IAM instance profile has `s3:GetObject` permission and S3 VPC endpoint is configured.

Translate a Slurm Job Submit plugin script to use CLI Filter Plugin in AWS PCS

Translate your existing Job Submit Plugin Lua script to CLI Filter Plugin when you migrate from other Slurm environments. The translation process involves updating function names and field access patterns to work with the CLI Filter Plugin API.

Prerequisites

Before you translate your script, complete these tasks:

- Review your existing Job Submit Plugin Lua script
- Understand differences between Job Submit and CLI Filter Plugin APIs
- Access Slurm CLI Filter Plugin documentation

To translate Job Submit Plugin script to CLI Filter Plugin

1. Review your existing Job Submit Plugin script functions (`slurm_job_submit`, `slurm_job_modify`).
2. Identify equivalent CLI Filter Plugin functions:
   - `slurm_job_submit` becomes `slurm_cli_pre_submit`
   - Add `slurm_cli_setup_defaults` for default parameter setting
   - Add `slurm_cli_post_submit` for post-submission actions
3. Translate job validation logic from `job_desc` fields to `options` array access:
   - `job_desc.account` becomes `options["account"]`
   - `job_desc.partition` becomes `options["partition"]`
   - `job_desc.features` becomes `options["constraint"]`
4. Update logging calls from `slurm.log_user()` to `slurm.log_error()`.
5. Test your translated script on a development cluster.
6. Deploy to your production cluster following the standard CLI Filter Plugin deployment process.

Expected results

After you complete the translation:

- Your translated script provides equivalent job submission validation
- Users see similar error messages and prompts as with your original Job Submit Plugin
- Job submission policies are maintained during migration to AWS PCS

Troubleshooting

Script translation errors

Symptoms: Job submissions fail with Lua execution errors.

Likely cause: Incorrect field access or function calls in translated script.

Resolution: Review CLI Filter Plugin API documentation and compare field mappings between Job Submit and CLI Filter interfaces.

Frequently asked questions about Slurm CLI Filter Plugins in AWS PCS

Review these frequently asked questions about CLI Filter Plugins.

What is the difference between CLI Filter Plugin and Job Submit Plugin?

CLI Filter Plugin runs client-side on login and compute nodes before job submission reaches the controller, while Job Submit Plugin runs server-side on the controller after job submission. CLI Filter Plugin can be bypassed by users but doesn't hold controller locks, while Job Submit is secure but can impact cluster performance during execution.

Does AWS PCS support Slurm Job Submit Plugin?

No, Job Submit Plugin is not supported in AWS PCS. Use CLI Filter Plugin instead for job submission validation and modification.

Can I use CLI Filter Plugin for security enforcement?

No, CLI Filter Plugin can be bypassed by determined users and should not be relied upon for security enforcement. Use it for user experience improvements, default parameter setting, and policy guidance rather than security-critical policies.

Why must the script be on all compute nodes, not just login nodes?

Slurm commands like `srun` can be executed within job scripts on compute nodes, which also triggers CLI Filter Plugin execution. The script must be available wherever Slurm commands are executed.

Can I modify the CLI Filter Plugin script on a live cluster?

Yes, if you use the S3 or file system deployment approach. New instances will get the updated script, but existing instances need the script updated manually or through your chosen deployment method.

Can I use different CLI Filter Plugin scripts on different compute node groups?

Yes, but this is not recommended. You can provide scripts with different logic to different compute node groups, but you are responsible for managing interdependencies and preventing overlapping logic. Most customers provide one set of logic across an entire cluster.

Can I use CLI Filter Plugin with C implementation instead of Lua?

C implementation is not supported. Only Lua script implementation is supported in AWS PCS. SchedMD recommends customers use Lua over C for ease of use when implementing CLI Filter Plugins.

Can I turn CLI Filter Plugin on or off on an existing cluster?

Yes, you can enable or disable CLI Filter Plugin on existing clusters using the Update API without recreating the cluster.

Troubleshooting Slurm CLI Filter Plugin issues in AWS PCS

Use this troubleshooting information to resolve common CLI Filter Plugin issues.

Job submission fails immediately with plugin loading error

Symptoms: Users receive error messages about missing or failed CLI Filter Plugin when submitting jobs.

Possible causes:

- CLI Filter Plugin script missing from one or more nodes
- Incorrect script filename (must be exactly `cli_filter.lua`)
- Script deployed to wrong directory path
- Script has incorrect file permissions

Resolution:

- Verify script exists at `/etc/aws/pcs/scheduler/slurm-<version>/cli_filter.lua` on all login and compute nodes
- Check script filename is exactly `cli_filter.lua`
- Ensure script has readable permissions (644 or similar)
- Test script deployment on a single login node before deploying to full cluster

Cluster creation fails with CLI Filter Plugin validation error

Symptoms: Cluster creation fails with error about invalid `CliFilterPlugins` parameter.

Possible causes:

- Incorrect parameter value format in `slurmCustomSettings`

- Typo in parameter name or value

Resolution:

- Use exact parameter name: `CliFilterPlugins`
- Use exact parameter value: `cli_filter/lua`
- Verify JSON syntax in `slurmCustomSettings` array

CLI Filter Plugin script executes but job validation doesn't work as expected

Symptoms: Jobs submit successfully but custom validation logic doesn't trigger or produces unexpected results.

Possible causes:

- Lua script syntax errors
- Incorrect field access patterns (using Job Submit Plugin syntax instead of CLI Filter Plugin)
- Logic errors in validation conditions

Resolution:

- Review Lua script for syntax errors
- Verify field access uses `options["field_name"]` format instead of `job_desc.field_name`
- Add logging statements to debug script execution flow
- Test script logic with simple validation cases first

S3 script deployment fails

Symptoms: Instances launch but CLI Filter Plugin script is not downloaded from S3.

Possible causes:

- IAM instance profile lacks S3 read permissions
- S3 VPC endpoint not configured
- Incorrect S3 bucket or object path in user data

Resolution:

- Verify IAM instance profile has `s3:GetObject` permission for your bucket
- Configure S3 VPC Gateway endpoint for direct access
- Check S3 bucket name and object path in user data script

- Review instance user data logs for S3 download errors

Security in AWS Parallel Computing Service

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 Parallel Computing Service, 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 AWS PCS. The following topics show you how to configure AWS PCS to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS PCS resources.

Topics

Data protection in AWS Parallel Computing Service

The AWS shared responsibility model applies to data protection in AWS Parallel Computing Service. 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. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with AWS PCS 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.

Encryption at rest

Encryption is enabled by default for data at rest when you create an AWS Parallel Computing Service (AWS PCS) cluster with the AWS Management Console, AWS CLI, AWS PCS API, or AWS SDKs. AWS PCS uses an **AWS owned KMS key** to encrypt data at rest. For more information, see Customer keys and AWS keys in the *AWS KMS Developer Guide*. You can also use a customer managed key. For more information, see Required KMS key policy for use with encrypted EBS volumes in AWS PCS.

The **cluster secret** is stored in AWS Secrets Manager and is encrypted with the Secrets Manager managed KMS key. For more information, see Working with cluster secrets in AWS PCS.

In an AWS PCS cluster, the following data is *at rest*:

- **Scheduler state** – It includes data on running jobs and provisioned nodes in the cluster. This is the data that Slurm persists in the `StateSaveLocation` defined in your `slurm.conf`. For more information, see the description of StateSaveLocation in the Slurm documentation. AWS PCS deletes job data after a job completes.
- **Scheduler auth secret** – AWS PCS uses it to authenticate all scheduler communications in the cluster.

For scheduler state information, AWS PCS automatically encrypts data and metadata before it writes them to the file system. The encrypted file system uses industry standard AES-256 encryption algorithm for data at rest.

Encryption in transit

Your connections to the AWS PCS API use TLS encryption with the Signature Version 4 signing process, regardless of whether you use the AWS Command Line Interface (AWS CLI) or AWS SDKs. For more information, see Signing AWS API requests in the *AWS Identity and Access Management User Guide*. AWS manages access control through the API with the IAM policies for the security credentials you use to connect.

AWS PCS uses TLS to connect to other AWS services.

Within a Slurm cluster, the scheduler is configured with the `auth/slurm` authentication plug-in that provides authentication for all scheduler communications. Slurm doesn't provide encryption at the application level for its communications, all data flowing across cluster instances stays local to the EC2 VPC and therefore is subject to VPC encryption if those instances support encryption in transit. For more information, see Encryption in transit in the *Amazon Elastic Compute Cloud User Guide*. Communication is encrypted between the controller (provisioned in a service account) the cluster nodes in your account.

Key management

AWS PCS uses an **AWS owned KMS key** to encrypt data. For more information, see Customer keys and AWS keys in the *AWS KMS Developer Guide*. You can also use a customer managed key. For more information, see Required KMS key policy for use with encrypted EBS volumes in AWS PCS.

The **cluster secret** is stored in AWS Secrets Manager and is encrypted with the Secrets Manager managed KMS key. For more information, see Working with cluster secrets in AWS PCS.

Inter-network traffic privacy

AWS PCS compute resources for a cluster reside within 1 VPC in the customer's account. Therefore, all internal AWS PCS service traffic within a cluster stays within the AWS network and doesn't travel across the internet. Communication between the user and AWS PCS nodes can travel across the internet and we recommend using SSH or Systems Manager to connect to the nodes. For more information, see What is AWS Systems Manager? in the *AWS Systems Manager User Guide*.

You can also use the following offerings to connect your on-premises network to AWS:

- AWS Site-to-Site VPN. For more information, see What is AWS Site-to-Site VPN? in the *AWS Site-to-Site VPN User Guide*.
- An AWS Direct Connect. For more information, see What is AWS Direct Connect? in the *AWS Direct Connect User Guide*.

You access the AWS PCS API to perform administrative tasks for the service. You and your users access the Slurm endpoint ports to interact with the scheduler directly.

Encrypting API traffic

To access the AWS PCS API, clients must support Transport Layer Security (TLS) 1.2 or later. We require TLS 1.2 and recommend TLS 1.3. Clients must also support cipher suites with Perfect Forward Secrecy (PFS), such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Diffie-Hellman Ephemeral (ECDHE). 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. You can also use AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Encrypting data traffic

Encryption of data in transit is enabled from supported EC2 instances accessing the scheduler endpoint and between ComputeNodeGroup instances from within the AWS Cloud. For more information, see Encryption in transit.

Required KMS key policy for use with encrypted EBS volumes in AWS PCS

AWS PCS uses service-linked roles to delegate permissions to other AWS services. The AWS PCS service-linked role is predefined and includes permissions that AWS PCS requires to call other AWS services on your behalf. The predefined permissions also include access to your AWS managed keys but not to your customer managed keys.

This topic describes how to set up the key policy required to launch instances when you specify a customer managed key for Amazon EBS encryption.

> **Note**
>
> AWS PCS doesn't require additional authorization to use the default AWS managed key to protect the encrypted volumes in your account.

Contents

Overview

You can use the following AWS KMS keys for Amazon EBS encryption when AWS PCS launches instances:

- AWS managed key – An encryption key in your account that Amazon EBS creates, owns, and manages. This is the default encryption key for a new account. Amazon EBS uses the AWS managed key for encryption unless you specify a customer managed key.
- Customer managed key – A custom encryption key that you create, own, and manage. For more information, see Create a KMS key in the *AWS Key Management Service Developer Guide*.

> **Note**
>
> The key must be symmetric. Amazon EBS doesn't support asymmetric customer managed keys.

You configure customer managed keys when you create encrypted snapshots or a launch template that specifies encrypted volumes, or when you choose to enable encryption by default.

Configure key policies

Your KMS keys must have a key policy that allows AWS PCS to launch instances with Amazon EBS volumes encrypted with a customer managed key.

Use the examples on this page to configure a key policy to give AWS PCS access to your customer managed key. You can modify the customer managed key's key policy when you create the key or at a later time.

The key policy must have the following statements:

- A statement that allows the IAM identity specified in the `Principal` element to use the customer managed key directly. It includes permissions to perform the AWS KMS `Encrypt`, `Decrypt`, `ReEncrypt*`, `GenerateDataKey*`, and `DescribeKey` operations on the key.
- A statement that allows the IAM identity specified in the `Principal` element to use the `CreateGrant` operation to generate grants that delegate a subset of its own permissions to AWS services that are integrated with AWS KMS or another principal. This allows them to use the key to create encrypted resources on your behalf.

Don't change any existing statements in the policy when you add the new policy statements to your key policy.

For more information, see:

- create-key in the *AWS CLI Command Reference*
- put-key-policy in the *AWS CLI Command Reference*
- Find the key ID and key ARN in the *AWS Key Management Service Developer Guide*
- Service-linked roles for AWS PCS
- Amazon EBS encryption in the *Amazon EBS User Guide*
- AWS Key Management Service in the *AWS Key Management Service Developer Guide*

Example 1: Key policy sections that allow access to the customer managed key

Add the following policy statements to the key policy of the customer managed key. Replace the example ARN with the ARN of the your `AWSServiceRoleForPCS` service-linked role. This example policy gives the AWS PCS service-linked role (`AWSServiceRoleForPCS`) permissions to use the customer managed key.

```
{
   "Sid": "Allow service-linked role use of the customer managed key",
   "Effect": "Allow",
   "Principal": {
       "AWS": [
           "arn:aws:iam::account-id:role/aws-service-role/pcs.amazonaws.com/
AWSServiceRoleForPCS"
       ]
   },
   "Action": [
       "kms:Encrypt",
       "kms:Decrypt",
       "kms:ReEncrypt*",
       "kms:GenerateDataKey*",
       "kms:DescribeKey"
   ],
   "Resource": "*"
}
```

```
{
   "Sid": "Allow attachment of persistent resources",
   "Effect": "Allow",
   "Principal": {
       "AWS": [
           "arn:aws:iam::account-id:role/aws-service-role/pcs.amazonaws.com/
AWSServiceRoleForPCS"
       ]
   },
   "Action": [
       "kms:CreateGrant"
   ],
   "Resource": "*",
   "Condition": {
       "Bool": {
           "kms:GrantIsForAWSResource": true
       }
    }
}
```

Example 2: Key policy sections that allow cross-account access to the customer managed key

If you create a customer managed key in a different account than your AWS PCS cluster, you must use a **grant** in combination with the key policy to allow cross-account access to the key.

To grant access to the key

1. Add the following policy statements to the customer managed key's key policy. Replace the example ARN with the ARN of the other account. Replace *111122223333* with the actual account ID of the AWS account that you want to create the AWS PCS cluster in. This allows you to give an IAM user or role in the specified account permission to create a grant for the key using the CLI command that follows. By default, users don't have access to the key.

    ```
    {.
       "Sid": "Allow external account 111122223333 use of the customer managed key",
       "Effect": "Allow",
       "Principal": {
           "AWS": [
               "arn:aws:iam::111122223333:root"
           ]
       },
       "Action": [
           "kms:Encrypt",
           "kms:Decrypt",
           "kms:ReEncrypt*",
           "kms:GenerateDataKey*",
    ```

```
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

```
{
    "Sid": "Allow attachment of persistent resources in external
 account 111122223333",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::111122223333:root"
        ]
    },
    "Action": [
        "kms:CreateGrant"
    ],
    "Resource": "*"
}
```

2. From the account that you want to create the AWS PCS cluster in, create a grant that delegates the relevant permissions to the AWS PCS service-linked role. The value of `grantee-principal` is the ARN of the service-linked role. The value of `key-id` is the ARN of the key.

   The following example create-grant CLI command gives the service-linked role named `AWSServiceRoleForPCS` in account *111122223333* permissions to use the customer managed key in account *444455556666*.

```
aws kms create-grant \
  --region us-west-2 \
  --key-id arn:aws:kms:us-
west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d \
  --grantee-principal arn:aws:iam::111122223333:role/aws-service-role/
pcs.amazonaws.com/AWSServiceRoleForPCS \
  --operations "Encrypt" "Decrypt" "ReEncryptFrom" "ReEncryptTo" "GenerateDataKey"
 "GenerateDataKeyWithoutPlaintext" "DescribeKey" "CreateGrant"
```

> **Note**
>
> The user making the request must have permissions to use the `kms:CreateGrant` action.

The following example IAM policy allows an IAM identity (user or role) in account *111122223333* to create a grant for the customer managed key in account *444455556666*.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreationOfGrantForTheKMSKeyinExternalAccount444455556666",
      "Effect": "Allow",
      "Action": "kms:CreateGrant",
      "Resource": "arn:aws:kms:us-
west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
    }
  ]
}
```

For more information about creating a grant for a KMS key in a different AWS account, see Grants in AWS KMS in the *AWS Key Management Service Developer Guide*.

> **Important**
>
> The service-linked role name specified as the grantee principal must be the name of an existing role. After creating the grant, to ensure that the grant allows AWS PCS to use the specified KMS key, do not delete and recreate the service-linked role.

Edit key policies in the AWS KMS console

The examples in the previous sections show only how to add statements to a key policy, which is just one way of changing a key policy. The easiest way to change a key policy is to use the AWS KMS console's default view for key policies and make an IAM identity (user or role) one of the *key users* for the appropriate key policy. For more information, see Using the AWS Management Console default view in the *AWS Key Management Service Developer Guide*.

> ⚠ **Warning**
>
> The console's default view policy statements include permissions to perform AWS KMS `Revoke` operations on the customer managed key. If you revoke a grant that gave an AWS account access to a customer managed key in your account, users in that AWS account lose access to the encrypted data and the key.

Access AWS Parallel Computing Service using an interface endpoint (AWS PrivateLink)

You can use AWS PrivateLink to create a private connection between your VPC and AWS Parallel Computing Service (AWS PCS). You can access AWS PCS as if it were in your VPC, without the use of an internet gateway, NAT device, VPN connection, or Direct Connect connection. Instances in your VPC don't need public IP addresses to access AWS PCS.

You establish this private connection by creating an *interface endpoint*, powered by AWS PrivateLink. We create an endpoint network interface in each subnet that you enable for the interface endpoint. These are requester-managed network interfaces that serve as the entry point for traffic destined for AWS PCS.

For more information, see Access AWS services through AWS PrivateLink in the *AWS PrivateLink Guide*.

Considerations for AWS PCS

Before you set up an interface endpoint for AWS PCS, review Access an AWS service using an interface VPC endpoint in the *AWS PrivateLink Guide*.

AWS PCS supports making calls to all of its API actions through the interface endpoint.

If your VPC doesn't have direct internet access, you must configure a VPC endpoint to enable your compute node group instances to call the AWS PCS `RegisterComputeNodeGroupInstance` API action.

Create an interface endpoint for AWS PCS

You can create an interface endpoint for AWS PCS using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Create an interface endpoint in the *AWS PrivateLink Guide*.

Create an interface endpoint for AWS PCS using the following service name:

```
com.amazonaws.region.pcs
```

Replace *`region`* with the ID of the AWS Region to create the endpoint in, such as `us-east-1`.

If you enable private DNS for the interface endpoint, you can make API requests to AWS PCS using its default Regional DNS name. For example, `pcs.us-east-1.amazonaws.com`.

Create an endpoint policy for your interface endpoint

An endpoint policy is an IAM resource that you can attach to an interface endpoint. The default endpoint policy allows full access to AWS PCS through the interface endpoint. To control the access allowed to AWS PCS from your VPC, attach a custom endpoint policy to the interface endpoint.

An endpoint policy specifies the following information:

- The principals that can perform actions (AWS accounts, IAM users, and IAM roles).
- The actions that can be performed.
- The resources on which the actions can be performed.

For more information, see Control access to services using endpoint policies in the *AWS PrivateLink Guide*.

Example: VPC endpoint policy for AWS PCS actions

The following is an example of a custom endpoint policy. When you attach this policy to your interface endpoint, it grants access to the listed AWS PCS actions for all principals to the cluster with the specified *`cluster-id`*. Replace *`region`* with the ID of the AWS Region of the cluster, such as `us-east-1`. Replace *`account-id`* with the AWS account number of the cluster.

```
{
    "Statement": [
            {
                "Action": [
                "pcs:CreateCluster",
                "pcs:ListClusters",
                "pcs:DeleteCluster",
                "pcs:GetCluster",
                ],
                "Effect": "Allow",
                "Principal": "*",
                "Resource": [
                    "arn:aws:pcs:region:account-id:cluster/cluster-id*"
                ]
            }
        ]
}
```

Identity and Access Management for AWS Parallel Computing Service

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 AWS PCS resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Audience
- Authenticating with identities
- Managing access using policies
- How AWS Parallel Computing Service works with IAM
- Identity-based policy examples for AWS Parallel Computing Service
- AWS managed policies for AWS Parallel Computing Service
- Service-linked roles for AWS PCS
- Amazon EC2 Spot role for AWS PCS
- Minimum permissions for AWS PCS

- IAM instance profiles for AWS Parallel Computing Service
- Troubleshooting AWS Parallel Computing Service identity and access

Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting AWS Parallel Computing Service identity and access)
- **Service administrator** - determine user access and submit permission requests (see How AWS Parallel Computing Service works with IAM)
- **IAM administrator** - write policies to manage access (see Identity-based policy examples for AWS Parallel Computing Service)

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

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

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. You must specify a principal in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS Parallel Computing Service works with IAM

Before you use IAM to manage access to AWS PCS, learn what IAM features are available to use with AWS PCS.

IAM features you can use with AWS Parallel Computing Service

IAM feature	AWS PCS support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Yes
Temporary credentials	Yes
Principal permissions	Yes
Service roles	No
Service-linked roles	Yes

To get a high-level view of how AWS PCS 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 AWS PCS

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for AWS PCS

To view examples of AWS PCS identity-based policies, see Identity-based policy examples for AWS Parallel Computing Service.

Resource-based policies within AWS PCS

Supports resource-based policies: No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for AWS PCS

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The Action element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Include actions in a policy to grant permissions to perform the associated operation.

To see a list of AWS PCS actions, see Actions Defined by AWS Parallel Computing Service in the *Service Authorization Reference*.

Policy actions in AWS PCS use the following prefix before the action:

```
pcs
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "pcs:action1",
      "pcs:action2"
         ]
```

Policy resources for AWS PCS

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The Resource JSON policy element specifies the object or objects to which the action applies. As a best practice, specify a resource using its Amazon Resource Name (ARN). For actions that don't support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of AWS PCS resource types and their ARNs, see Resources Defined by AWS Parallel Computing Service in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions Defined by AWS Parallel Computing Service.

To view examples of AWS PCS identity-based policies, see Identity-based policy examples for AWS Parallel Computing Service.

Policy condition keys for AWS PCS

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element specifies when statements execute based on defined criteria. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of AWS PCS condition keys, see Condition Keys for AWS Parallel Computing Service in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions Defined by AWS Parallel Computing Service .

To view examples of AWS PCS identity-based policies, see Identity-based policy examples for AWS Parallel Computing Service.

ACLs in AWS PCS

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 AWS PCS

Supports ABAC (tags in policies): Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes called tags. You can attach tags to IAM entities and AWS resources, then design ABAC policies to allow operations when the principal's tag matches the tag on the resource.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with AWS PCS

Supports temporary credentials: Yes

Temporary credentials provide short-term access to AWS resources and are automatically created when you use federation or switch roles. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM and AWS services that work with IAM in the *IAM User Guide*.

Cross-service principal permissions for AWS PCS

Supports forward access sessions (FAS): Yes

Forward access sessions (FAS) use the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. For policy details when making FAS requests, see Forward access sessions.

Service roles for AWS PCS

Supports service roles: No

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> ⚠ **Warning**
>
> Changing the permissions for a service role might break AWS PCS functionality. Edit service roles only when AWS PCS provides guidance to do so.

Service-linked roles for AWS PCS

Supports service-linked roles: Yes

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 AWS PCS service-linked roles, see Service-linked roles for AWS PCS.

Identity-based policy examples for AWS Parallel Computing Service

By default, users and roles don't have permission to create or modify AWS PCS resources. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by AWS PCS, including the format of the ARNs for each of the resource types, see Actions, Resources, and Condition Keys for AWS Parallel Computing Service in the *Service Authorization Reference*.

Topics

- Policy best practices
- Using the AWS PCS console
- Allow users to view their own permissions

Policy best practices

Identity-based policies determine whether someone can create, access, or delete AWS PCS resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on

specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.

- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the AWS PCS console

To access the AWS Parallel Computing Service console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the AWS PCS 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.

For more information about minimum permissions required to use the AWS PCS console, see Minimum permissions for AWS PCS.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS managed policies for AWS Parallel Computing Service

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: AWSPCSComputeNodePolicy

You can attach AWSPCSComputeNodePolicy to your IAM entities. You can attach this policy to an AWS PCS compute node IAM role that you specify to allow nodes that use that role to connect to an AWS PCS cluster.

AWS PCS attaches this policy to a compute node group role when you use the console to create a compute node group.

Permissions details

This policy includes the following permissions.

- `pcs:RegisterComputeNodeGroupInstance` – Allow an AWS PCS compute node (EC2 instance) to register with an AWS PCS cluster.

To view the permissions for this policy, see AWSPCSComputeNodePolicy in the *AWS Managed Policy Reference*.

AWS managed policy: AWSPCSServiceRolePolicy

You can't attach AWSPCSServiceRolePolicy to your IAM entities. This policy is attached to a service-linked role that allows AWS PCS to perform actions on your behalf. For more information, see Service-linked roles for AWS PCS.

Permissions details

This policy includes the following permissions.

- `ec2` – Allows AWS PCS to create and manage Amazon EC2 resources.
- `iam` – Allows AWS PCS to create a service-linked role for the Amazon EC2 fleet and to pass the role to Amazon EC2.
- `cloudwatch` – Allows AWS PCS to publish service metrics to Amazon CloudWatch.
- `secretsmanager` – Allows AWS PCS to manage secrets for AWS PCS cluster resources.

To view the permissions for this policy, see AWSPCSServiceRolePolicy in the *AWS Managed Policy Reference*.

AWS PCS updates to AWS managed policies

View details about updates to AWS managed policies for AWS PCS since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the AWS PCS Document history page.

Change	Description	Date
AWSPCSServiceRolePolicy – Update to an existing policy	AWS PCS added new permissions to support Capacity Blocks for predictable compute capacity. Added `ec2:DescribeCapacityReservations` permission to enable AWS PCS to discover and use Capacity Block reservations for compute node groups.	September 11, 2025

Change	Description	Date
AWSPCSComputeNodePolicy – New policy	AWS PCS added a new policy to grant permission to AWS PCS compute nodes to connect to AWS PCS clusters. AWS PCS attaches this policy to an IAM role when you create a compute node group in the AWS PCS console.	June 23, 2025
Updated the JSON in this document	Corrected the JSON in this document to include `"arn:aws:ec2:*:*:spot-instances-request/*"` .	September 5, 2024
AWS PCS started tracking changes	AWS PCS started tracking changes for its AWS managed policies.	August 28, 2024

Service-linked roles for AWS PCS

AWS Parallel Computing Service uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to AWS PCS. Service-linked roles are predefined by AWS PCS and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up AWS PCS easier because you don't have to manually add the necessary permissions. AWS PCS defines the permissions of its service-linked roles, and unless defined otherwise, only AWS PCS can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting its related resources. This protects your AWS PCS resources because you can't accidentally remove permission to access the resources.

For information about other services that support service-linked roles, see AWS services that work with IAM and look for the services that have **Yes** in the **Service-linked roles** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for AWS PCS

AWS PCS uses the service-linked role named **AWSServiceRoleForPCS** – Grants permission to AWS PCS to manage Amazon EC2 resources.

The AWSServiceRoleForPCS service-linked role trusts the following services to assume the role:

- `pcs.amazonaws.com`

The role permissions policy named AWSPCSServiceRolePolicy allows AWS PCS to complete actions on specific resources.

You must configure permissions to allow your users, groups, or roles to create, edit, or delete a service-linked role. For more information, see Service-linked role permissions in the *IAM User Guide*.

Creating a service-linked role for AWS PCS

You don't need to manually create a service-linked role. AWS PCS creates a service-linked role for you when you create a cluster.

Editing a service-linked role for AWS PCS

AWS PCS does not allow you to edit the AWSServiceRoleForPCS service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

Deleting a service-linked role for AWS PCS

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don’t have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

> **Note**
>
> If the AWS PCS service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To remove AWS PCS resources used by the AWSServiceRoleForPCS

You must delete all your clusters to delete the AWSServiceRoleForPCS service-linked role. For more information, see Delete a cluster.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the AWSServiceRoleForPCS service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Supported Regions for AWS PCS service-linked roles

AWS PCS supports using service-linked roles in all of the Regions where the service is available. For more information, see AWS Regions and endpoints.

Amazon EC2 Spot role for AWS PCS

If you want to create an AWS PCS compute node group that uses **Spot** as its purchase option, you must also have the **AWSServiceRoleForEC2Spot** service-linked role in your AWS account. You can use the following AWS CLI command to create the role. For more information, see Create a service-linked role and Create a role to delegate permissions to an AWS service in the *AWS Identity and Access Management User Guide*.

```
aws iam create-service-linked-role --aws-service-name spot.amazonaws.com
```

> **Note**
>
> You receive the following error if your AWS account already has an `AWSServiceRoleForEC2Spot` IAM role.
>
> ```
>  An error occurred (InvalidInput) when calling the CreateServiceLinkedRole
> operation: Service role name AWSServiceRoleForEC2Spot has been taken in this
> account, please try a different suffix.
> ```

Minimum permissions for AWS PCS

This section describes the minimum IAM permissions required for an IAM identity (user, group, or role) to use the service.

Contents

Minimum permissions to use API actions

API action	Minimum permissions	Additional permissions for the console

CreateCluster	`ec2:CreateNetworkInterface,`
	`ec2:DescribeVpcs,`
	`ec2:DescribeSubnets,`
	`ec2:DescribeSecurityGroups,`
	`ec2:GetSecurityGroupsForVpc,`
	`iam:CreateServiceLinkedRole,`
	`secretsmanager:CreateSecret,`
	`secretsmanager:TagResource,`
	`secretsmanager:RotateSecret,`
	`pcs:CreateCluster`	

API action	Minimum permissions	Additional permissions for the console
ListClusters	`pcs:ListClusters`	
GetCluster	`pcs:GetCluster`	`ec2:DescribeSubnets`
DeleteCluster	`pcs:DeleteCluster`	

CreateComputeNodeGroup	`ec2:DescribeVpcs,`
	`ec2:DescribeSubnets,`
	`ec2:DescribeSecurityGroups,`
	`ec2:DescribeLaunchTemplates,`
	`ec2:DescribeLaunchTemplateVersions,`
	`ec2:DescribeInstanceTypes,`
	`ec2:DescribeInstanceTypeOfferings,`
	`ec2:RunInstances,`
	`ec2:CreateFleet,`
	`ec2:CreateTags,`
	`iam:PassRole,`
	`iam:GetInstanceProfile,`
	`pcs:CreateComputeNodeGroup`
		`iam:ListInstanceProfiles,`
		`ec2:DescribeImages,`
		`pcs:GetCluster`

ListComputerNodeGroups	`pcs:ListComputeNodeGroups`	`pcs:GetCluster`
GetComputeNodeGroup	`pcs:GetComputeNodeGroup`	`ec2:DescribeSubnets`

API action	Minimum permissions	Additional permissions for the console
UpdateComputeNodeGroup	`ec2:DescribeVpcs, ec2:DescribeSubnets, ec2:DescribeSecurityGroups, ec2:DescribeLaunchTemplates, ec2:DescribeLaunchTemplateVersions, ec2:DescribeInstanceTypes, ec2:DescribeInstanceTypeOfferings, ec2:RunInstances, ec2:CreateFleet, ec2:CreateTags, iam:PassRole, iam:GetInstanceProfile, pcs:UpdateComputeNodeGroup`	`pcs:GetComputeNodeGroup, iam:ListInstanceProfiles, ec2:DescribeImages, pcs:GetCluster`
DeleteComputeNodeGroup	`pcs:DeleteComputeNodeGroup`	
CreateQueue	`pcs:CreateQueue`	`pcs:ListComputeNodeGroups, pcs:GetCluster`
ListQueues	`pcs:ListQueues`	`pcs:GetCluster`
GetQueue	`pcs:GetQueue`	

API action	Minimum permissions	Additional permissions for the console

UpdateQueue	`pcs:UpdateQueue`	`pcs:ListComputeNodeGroups,`
		`pcs:GetQueue`

DeleteQueue	`pcs:DeleteQueue`	

Minimum permissions to use tags

The following permissions are required to use tags with your resources in AWS PCS.

```
pcs:ListTagsForResource,
pcs:TagResource,
pcs:UntagResource
```

Minimum permissions to support logs

AWS PCS sends log data to Amazon CloudWatch Logs (CloudWatch Logs). You must make sure that your identity has the minimum permissions to use CloudWatch Logs. For more information, see Overview of managing access permissions to your CloudWatch Logs resources in the *Amazon CloudWatch Logs User Guide*.

For information about permissions required for a service to send logs to CloudWatch Logs, see Enabling logging from AWS services in the *Amazon CloudWatch Logs User Guide*.

Minimum permissions to use Capacity Blocks

Amazon EC2 Capacity Blocks for ML is an Amazon EC2 purchasing option that enables you to pay in advance to reserve GPU-based accelerated computing instances within a specific date and time range to support short duration workloads. For more information, see Using Amazon EC2 Capacity Blocks for ML with AWS PCS.

You choose to use Capacity Blocks when you create or update a compute node group. The IAM identity you use to create or update the compute node group must have the following permission:

```
ec2:DescribeCapacityReservations
```

Minimum permissions for a service administrator

The following IAM policy specifies the minimum permissions required for an IAM identity (user, group, or role) to configure and manage the AWS PCS service.

> **Note**
>
> Users who don't configure and manage the service don't require these permissions. Users who only run jobs use secure shell (SSH) to connect to the cluster. AWS Identity and Access Management (IAM) doesn't handle authentication or authorization for SSH.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PCSAccess",
      "Effect": "Allow",
      "Action": [
        "pcs:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "EC2Access",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DescribeImages",
        "ec2:GetSecurityGroupsForVpc",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVpcs",
        "ec2:DescribeLaunchTemplates",
        "ec2:DescribeLaunchTemplateVersions",
        "ec2:DescribeInstanceTypes",
        "ec2:DescribeInstanceTypeOfferings",
        "ec2:RunInstances",
        "ec2:CreateFleet",
        "ec2:CreateTags",
        "ec2:DescribeCapacityReservations"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IamInstanceProfile",
      "Effect": "Allow",
      "Action": [
        "iam:GetInstanceProfile"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IamPassRole",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/*/AWSPCS*",
        "arn:aws:iam::*:role/AWSPCS*",
        "arn:aws:iam::*:role/aws-pcs/*",
        "arn:aws:iam::*:role/*/aws-pcs/*"
      ],
      "Condition": {
        "StringEquals": {
           "iam:PassedToService": [
             "ec2.amazonaws.com"
           ]
        }
      }
    },
    {
      "Sid": "SLRAccess",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/aws-service-role/pcs.amazonaws.com/AWSServiceRoleFor*",
        "arn:aws:iam::*:role/aws-service-role/spot.amazonaws.com/AWSServiceRoleFor*"
      ],
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": [
            "pcs.amazonaws.com",
            "spot.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "AccessKMSKey",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:Encrypt",
        "kms:GenerateDataKey",
        "kms:CreateGrant",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
      "Sid": "SecretManagementAccess",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:CreateSecret",
        "secretsmanager:TagResource",
        "secretsmanager:UpdateSecret",
        "secretsmanager:RotateSecret"
      ],
      "Resource": "*"
    },
    {
       "Sid": "ServiceLogsDelivery",
       "Effect": "Allow",
       "Action": [
         "pcs:AllowVendedLogDeliveryForResource",
         "logs:PutDeliverySource",
         "logs:PutDeliveryDestination",
         "logs:CreateDelivery"
       ],
       "Resource": "*"
    }
  ]
}
```

IAM instance profiles for AWS Parallel Computing Service

Applications that run on an EC2 instance must include AWS credentials in any AWS API requests they make. We recommend you use an IAM role to manage temporary credentials on the EC2 instance. You can define an instance profile to do this, and attach it to your instances. For more information, see IAM roles for Amazon EC2 in the *Amazon Elastic Compute Cloud User Guide*.

> **Note**
>
> When you use the AWS Management Console to create an IAM role for Amazon EC2, the console creates an instance profile automatically and gives it the same name as the IAM role. If you use the AWS CLI, AWS API actions, or an AWS SDK to create the IAM role, you create the instance profile as a separate action. For more information, see Instance profiles in the *Amazon Elastic Compute Cloud User Guide*.

You must specify the Amazon Resource Name (ARN) of an instance profile when you create a compute node groups. You can choose different instance profiles for some or all compute node groups.

Requirements

IAM role of the instance profile

The IAM role associated with the instance profile must have `/aws-pcs/` in its path, or its name must start with `AWSPCS`.

Example IAM role ARNs

- `arn:aws:iam::*:role/AWSPCS-example-role-1`
- `arn:aws:iam::*:role/aws-pcs/example-role-2`

Permissions

The IAM role associated with the instance profile for AWS PCS must include the following policy.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "pcs:RegisterComputeNodeGroupInstance"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

Additional policies

Consider adding managed policies to the instance profile. For example:

- AmazonS3ReadOnlyAccess provides read-only access to all S3 buckets.
- AmazonSSMManagedInstanceCore enables AWS Systems Manager service core functionality, such as remote access directly from the Amazon Management Console.
- CloudWatchAgentServerPolicy contains permissions required to use AmazonCloudWatchAgent on servers.

You can also include your own IAM policies that support your specific use case.

Create an instance profile for AWS PCS

AWS PCS console

Select **Create a basic profile** when you create a compute node group to have AWS PCS create one for you with the minimum required policy.

Amazon EC2 console

You can create an instance profile directly from the Amazon EC2 console. For more information, see Using instance profiles in the *AWS Identity and Access Management User Guide*.

> ⚠ **Important**
>
> Make sure to use the required prefix AWSPCS in the IAM role name.

AWS CLI

Setting up Basic instance profile using AWS CLI

> **Note**
>
> Replace *example-role* in the following examples with the name of your IAM role.

1. Create IAM role with `/aws-pcs/` as the path attribute or a name that starts with `AWSPCS`.

   a. Copy and paste the following content to a new text file named `trust_policy.json`.

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
                "Principal": {
                   "Service": [
                       "ec2.amazonaws.com"
                   ]
               },
               "Action": [
                   "sts:AssumeRole"
               ]
           }
       ]
   }
   ```

   b. Use 1 of the following commands to create the IAM role.

   ```
   aws iam create-role --path /aws-pcs/ --role-name example-role --assume-role-policy-document file://trust_policy.json
   ```

   or

   ```
   aws iam create-role --role-name AWSPCS-example-role --assume-role-policy-document file://trust_policy.json
   ```

2. **Attach permissions.**

a. Copy and paste the following content to a new text file named `policy_document.json`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "pcs:RegisterComputeNodeGroupInstance"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

b. Attach the policy document to the role. This command attaches the policy as an inline policy.

```
aws iam put-role-policy \
    --role-name example-role \
    --policy-name pcsRegisterInstancePolicy \
    --policy-document file://policy_document.json
```

3. **Create an instance profile. Replace *`example-profile`* with the name of your instance profile.**

```
aws iam create-instance-profile --instance-profile-name example-profile
```

4. **Associate the IAM role with the instance profile.**

```
aws iam add-role-to-instance-profile \
   --instance-profile-name example-profile \
   --role-name example-role
```

Find instance profiles used with AWS PCS

1. If you don't know the exact names of your IAM roles for AWS PCS, use the following AWS CLI command to list the IAM roles that meet the AWS PCS name requirements.

```
aws iam list-roles --query "Roles[?starts_with(RoleName, 'AWSPCS') ||
 contains(Path, '/aws-pcs/')].[RoleName]" --output text
```

2. Use the following AWS CLI command to list the instance profiles associated with a specific IAM role. Replace *role-name* with the name of an IAM role that meets AWS PCS name requirements.

```
aws iam list-instance-profiles-for-role --role-name role-name
```

Troubleshooting AWS Parallel Computing Service identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS PCS and IAM.

Topics

- I am not authorized to perform an action in AWS PCS
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my AWS PCS resources

I am not authorized to perform an action in AWS PCS

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 `pcs:`*GetWidget* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 pcs: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 `pcs:`*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 AWS PCS.

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 AWS PCS. 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 AWS PCS 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 AWS PCS supports these features, see How AWS Parallel Computing Service works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.

- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Compliance validation for AWS Parallel Computing Service

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. For more information about your compliance responsibility when using AWS services, see AWS Security Documentation.

Resilience in AWS Parallel Computing Service

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.

Infrastructure Security in AWS Parallel Computing Service

As a managed service, AWS Parallel Computing Service 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 AWS PCS 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.

When AWS PCS creates a cluster, the service launches the Slurm controller in a service-owned account, separate from the compute nodes in your account. To bridge communication between the controller and the compute nodes, AWS PCS creates a cross-account Elastic Network Interface (ENI) in your VPC. The Slurm controller uses the ENI to manage and communicate with the compute nodes across different AWS accounts, maintaining the security and isolation of resources while facilitating efficient HPC and AI/ML operations.

Vulnerability analysis and management in AWS Parallel Computing Service

Configuration and IT controls are a shared responsibility between AWS and you. For more information, see the AWS shared responsibility model. AWS handles basic security tasks for the underlying infrastructure in the service account, such as patching the operating system on controller instances, firewall configuration, and AWS infrastructure disaster recovery. These procedures have been reviewed and certified by the appropriate third parties. For more details, see Best Practices for Security, Identity, and Compliance.

> **Note**
>
> Slurm controllers are unavailable while we update them. Running jobs aren't affected. Jobs submitted when the cluster's controller is unavailable are held until the controller is available.

You are responsible for the security of the underlying infrastructure in your AWS account:

- Maintain your code, including updates and security patches.
- Patch and update the operating system in the Amazon Machine Image (AMI) for your compute node groups and update your compute node groups to use the updated AMI.
- Update the scheduler to keep it within supported versions. Update the AMI for your compute node groups and update your compute node group to use the updated AMI.

- Authenticate and encrypt communication between user clients and the nodes they connect to.

For more information about updating the AMI for your compute node groups, see Amazon Machine Images (AMIs) for AWS PCS.

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 Parallel Computing Service (AWS PCS) 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 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:servicename:*: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 value of `aws:SourceArn` must be a cluster ARN.

The following example shows how you can use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in AWS PCS to prevent the confused deputy problem.

```
{
"Version": "2012-10-17",
  "Statement": {
```

```
"Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "pcs.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": [
          "arn:aws:pcs:us-east-1:123456789012:cluster/*"
        ]
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

IAM role for Amazon EC2 instances provisioned as part of a compute node group

AWS PCS automatically orchestrates Amazon EC2 capacity for each of the configured compute node groups in a cluster. When creating a compute node group, users must provide an IAM instance profile through the `iamInstanceProfileArn` field. The instance profile specifies the permissions associated with the provisioned EC2 instances. AWS PCS accepts any role that has `AWSPCS` as role name prefix or `/aws-pcs/` as part of the role path. The `iam:PassRole` permission is required on the IAM identity (user or role) that creates or updates a compute node group. When a user calls the `CreateComputeNodeGroup` or `UpdateComputeNodeGroup` API actions, AWS PCS checks to see if the user is allowed to perform the `iam:PassRole` action.

The following example policy grants permissions to pass only IAM roles whose name begins with `AWSPCS`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/AWSPCS*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Security best practices for AWS Parallel Computing Service

This section describes security best practices that are specific to AWS Parallel Computing Service (AWS PCS). To learn more about security best practices in AWS, see Best Practices for Security, Identity, and Compliance.

AMI-related security

- Don’t use AWS PCS sample AMIs for production workloads. The sample AMIs are unsupported and only intended for testing.
- Regularly update the operating system and software in the AMI for your compute node groups to mitigate vulnerabilities.
- Only use authenticated official AWS PCS packages downloaded from official AWS sources.
- Regularly update AWS PCS packages in the AMI for compute node groups and update the compute nodes to use the updated AMI. Consider automating this process to minimize vulnerabilities.

For more information, see Custom Amazon Machine Images (AMIs) for AWS PCS.

Slurm Workload Manager security

- Implement access controls and network restrictions to secure Slurm control and compute nodes. Only allow trusted users and systems to submit jobs and access Slurm management commands.
- Use Slurm's built-in security features, such as Slurm authentication, to ensure that job submissions and communications are authenticated.
- Update Slurm versions to maintain smooth operations and cluster support.

> ⚠ **Important**
>
> Any cluster that uses a version of Slurm that has reached *end of support life* (EOSL) is stopped immediately. Use the link at the top of the user guide pages to subscribe to the AWS PCS documentation RSS feed to receive notification when a Slurm version approaches EOSL.

For more information, see Slurm versions in AWS PCS.

- Regularly rotate cluster secrets to maintain security compliance and remediate potential security compromises. This is required for HIPAA and FedRAMP compliance.

  For more information, see Rotating cluster secrets in AWS PCS.

Monitoring and logging

- Use Amazon CloudWatch Logs and AWS CloudTrail to monitor and record actions in your clusters and AWS account. Use the data for troubleshooting and auditing.

Network security

- Deploy your AWS PCS clusters in a separate VPC to isolate your HPC environment from other network traffic.
- Use security groups and network access control lists (ACLs) to control inbound and outbound traffic to AWS PCS instances and subnets.
- Use AWS PrivateLink or VPC endpoints to keep network traffic to between your clusters and other AWS services inside the AWS network. For more information, see Access AWS Parallel Computing Service using an interface endpoint (AWS PrivateLink).

Logging and monitoring for AWS PCS

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS PCS and your other AWS resources. AWS provides the following monitoring tools to watch AWS PCS, report when something is wrong, and take automatic actions when appropriate:

- *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.
- *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.
- *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.

Job completion logs in AWS PCS

Job completion logs give you key details about your AWS Parallel Computing Service (AWS PCS) jobs when they complete, at no additional cost. You can use other AWS services to access and process your log data, such as Amazon CloudWatch Logs, Amazon Simple Storage Service (Amazon S3), and Amazon Data Firehose; AWS PCS records metadata about your jobs, such as the following.

- Job ID and name
- User and group information
- Job state (such as `COMPLETED`, `FAILED`, `CANCELLED`)
- Partition used
- Time limits
- Start, end, submit, and eligible times
- Node list and count

- Processor count
- Working directory
- Resource usage (CPU, memory)
- Exit codes
- Node details (names, instance IDs, instance types)

Contents

Prerequisites

The IAM principal that manages the AWS PCS cluster must allow the `pcs:AllowVendedLogDeliveryForResource` action.

The following example IAM policy grants the required permissions.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "PcsAllowVendedLogsDelivery",
         "Effect": "Allow",
         "Action": ["pcs:AllowVendedLogDeliveryForResource"],
         "Resource": [
            "arn:aws:pcs:*::cluster/*"
         ]
      }
   ]
}
```

Set up job completion logs

You can set up job completion logs for your AWS PCS cluster with the AWS Management Console or AWS CLI.

AWS Management Console

To set up job completion logs with the console

1. Open the [AWS PCS console](https://console.aws.amazon.com/pcs/home#/clusters).
2. In the navigation pane, choose **Clusters**.
3. Choose the cluster where you want to add job completion logs.
4. On the cluster details page, choose the **Logs** tab.
5. Under **Job Completion Logs**, choose **Add** to add up to 3 log delivery destinations from among CloudWatch Logs, Amazon S3, and Firehose.
6. Choose **Update log deliveries**.

AWS CLI

To set up job completion logs with the AWS CLI

1. Create a log delivery destination:

   ```
   aws logs put-delivery-destination --region region \
     --name pcs-logs-destination \
     --delivery-destination-configuration \
     destinationResourceArn=resource-arn
   ```

   Replace:

   - *`region`* — The AWS Region where you want to create the destination, such as `us-east-1`
   - *`pcs-logs-destination`* — A name for the destination
   - *`resource-arn`* — The Amazon Resource Name (ARN) of a CloudWatch Logs log group, S3 bucket, or Firehose delivery stream.

For more information, see PutDeliveryDestination in the *Amazon CloudWatch Logs API Reference*.

2. Set the PCS cluster as a log delivery source:

   ```
   aws logs put-delivery-source --region region \
     --name cluster-logs-source-name \
     --resource-arn cluster-arn \
     --log-type PCS_JOBCOMP_LOGS
   ```

   Replace:

   - *region* — The AWS Region of your cluster, such as `us-east-1`
   - *cluster-logs-source-name* — A name for the source
   - *cluster-arn* — the ARN of your AWS PCS cluster

   For more information, see PutDeliverySource in the *Amazon CloudWatch Logs API Reference*.

3. Connect the delivery source to the delivery destination:

   ```
   aws logs create-delivery --region region \
     --delivery-source-name cluster-logs-source \
     --delivery-destination-arn destination-arn
   ```

   Replace:

   - *region* — The AWS Region, such as `us-east-1`
   - *cluster-logs-source* — The name of your delivery source
   - *destination-arn* — The ARN of your delivery destination

   For more information, see CreateDelivery in the *Amazon CloudWatch Logs API Reference*.

How to find job completion logs

You can configure log destinations in CloudWatch Logs and Amazon S3. AWS PCS uses the following structured path names and file names.

CloudWatch Logs

AWS PCS uses the following name format for the CloudWatch Logs stream:

```
AWSLogs/PCS/cluster-id/jobcomp.log
```

For example: `AWSLogs/PCS/pcs_abc123de45/jobcomp.log`

Amazon S3

AWS PCS uses the following name format for the S3 path:

```
AWSLogs/account-id/PCS/region/cluster-id/jobcomp/year/month/day/hour/
```

For example: `AWSLogs/111122223333/PCS/us-east-1/pcs_abc123de45/jobcomp/2025/06/19/11/`

AWS PCS uses the following name format for the log files:

```
PCS_jobcomp_year-month-day-hour_cluster-id_random-id.log.gz
```

For example: `PCS_jobcomp_2025-06-19-11_pcs_abc123de45_04be080b.log.gz`

Job completion log fields

AWS PCS writes job completion log data as JSON objects. The JSON container `jobcomp` holds job details. The following table describes the fields inside the `jobcomp` container. Some fields are only present in specific circumstances, such as for array jobs or heterogeneous jobs.

Job completion log fields

Name	Example value	Required	Notes
`job_id`	`11`	yes	Always present with value
`user`	`"root"`	yes	Always present with value
`user_id`	`0`	yes	Always present with value
`group`	`"root"`	yes	Always present with value

Name	Example value	Required	Notes
`group_id`	`0`	yes	Always present with value
`name`	`"wrap"`	yes	Always present with value
`job_state`	`"COMPLETED"`	yes	Always present with value
`partition`	`"Hydra-MpiQueue-abcdef01-7"`	yes	Always present with value
`time_limit`	`"UNLIMITED"`	yes	Always present, but might be `"UNLIMITED"`
`start_time`	`"2025-06-19T10:58:57"`	yes	Always present, but might be `"Unknown"`
`end_time`	`"2025-06-19T10:58:57"`	yes	Always present, but might be `"Unknown"`
`node_list`	`"Hydra-MpiNG-abcdef01-2345-1"`	yes	Always present with value
`node_cnt`	`1`	yes	Always present with value
`proc_cnt`	`1`	yes	Always present with value
`work_dir`	`"/root"`	yes	Always present, but might be `"Unknown"`
`reservation_name`	`"weekly_maintenance"`	yes	Always present, but might be an empty string `""`

Name	Example value	Required	Notes
tres.cpu	1	yes	Always present with value
tres.mem.val	600	yes	Always present with value
tres.mem.unit	"M"	yes	Can be "M" or "bb"
tres.node	1	yes	Always present with value
tres.billing	1	yes	Always present with value
account	"finance"	yes	Always present, but might be an empty string ""
qos	"normal"	yes	Always present, but might be an empty string ""
wc_key	"project_1"	yes	Always present, but might be an empty string ""
cluster	"unknown"	yes	Always present, but might be "unknown"
submit_time	"2025-06-19T10:55:46"	yes	Always present, but might be "Unknown"
eligible_time	"2025-06-19T10:55:46"	yes	Always present, but might be "Unknown"
array_job_id	12	no	Only present if the job is an array job

Name	Example value	Required	Notes
`array_task_id`	`1`	no	Only present if the job is an array job
`het_job_id`	`10`	no	Only present if the job is a heterogeneous job
`het_job_offset`	`0`	no	Only present if the job is a heterogeneous job
`derived_exit_code_status`	`0`	yes	Always present with value
`derived_exit_code_signal`	`0`	yes	Always present with value
`exit_code_status`	`0`	yes	Always present with value
`exit_code_signal`	`0`	yes	Always present with value
`node_details[0].name`	`"Hydra-MpiNG-abcdef01-2345-1"`	no	Always present, but `node_details` might be `"[]"`
`node_details[0].instance_id`	`"i-0abcdef01234567a"`	no	Always present, but `node_details` might be `"[]"`
`node_details[0].instance_type`	`"t4g.micro"`	no	Always present, but `node_details` might be `"[]"`

Example job completion logs

The following examples show job completion logs for various job types and states:

```
{ "jobcomp": { "job_id": 1, "user": "root", "user_id": 0, "group": "root", "group_id":
 0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-
abcdef01-7", "time_limit": "UNLIMITED", "start_time": "2025-06-19T16:32:57",
 "end_time": "2025-06-19T16:33:03", "node_list": "Hydra-MpiNG-abcdef01-2345-
[1-2]", "node_cnt": 2, "proc_cnt": 2, "work_dir": "/usr/bin", "reservation_name":
 "", "tres": { "cpu": 2, "mem": { "val": 1944, "unit": "M" }, "node": 2,
 "billing": 2 }, "account": "", "qos": "", "wc_key": "", "cluster": "unknown",
 "submit_time": "2025-06-19T16:29:40", "eligible_time": "2025-06-19T16:29:41",
 "derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
 0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-1",
 "instance_id": "i-0abc123def45678", "instance_type": "t4g.micro" }, { "name":
 "Hydra-MpiNG-abcdef01-2345-2", "instance_id": "i-0def456abc78901", "instance_type":
 "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 2, "user": "root", "user_id": 0, "group": "root", "group_id":
 0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-
abcdef01-7", "time_limit": "UNLIMITED", "start_time": "2025-06-19T16:33:13",
 "end_time": "2025-06-19T16:33:14", "node_list": "Hydra-MpiNG-abcdef01-2345-
[1-2]", "node_cnt": 2, "proc_cnt": 2, "work_dir": "/usr/bin", "reservation_name":
 "", "tres": { "cpu": 2, "mem": { "val": 1944, "unit": "M" }, "node": 2,
 "billing": 2 }, "account": "", "qos": "", "wc_key": "", "cluster": "unknown",
 "submit_time": "2025-06-19T16:33:13", "eligible_time": "2025-06-19T16:33:13",
 "derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
 0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-1",
 "instance_id": "i-0abc123def45678", "instance_type": "t4g.micro" }, { "name":
 "Hydra-MpiNG-abcdef01-2345-2", "instance_id": "i-0def456abc78901", "instance_type":
 "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 3, "user": "root", "user_id": 0, "group": "root", "group_id":
 0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-abcdef01-7",
 "time_limit": "UNLIMITED", "start_time": "2025-06-19T22:58:57", "end_time":
 "2025-06-19T22:58:57", "node_list": "Hydra-MpiNG-abcdef01-2345-1", "node_cnt":
 1, "proc_cnt": 1, "work_dir": "/root", "reservation_name": "", "tres": { "cpu":
 1, "mem": { "val": 972, "unit": "M" }, "node": 1, "billing": 1 }, "account": "",
 "qos": "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T22:55:46",
 "eligible_time": "2025-06-19T22:55:46", "derived_exit_code_status": 0,
 "derived_exit_code_signal": 0, "exit_code_status": 0, "exit_code_signal":
 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-1", "instance_id":
 "i-0abc234def56789", "instance_type": "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 4, "user": "root", "user_id": 0, "group": "root",
 "group_id": 0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-
```

```
MpiQueue-abcdef01-7", "time_limit": "525600", "start_time": "2025-06-19T23:04:27",
 "end_time": "2025-06-19T23:04:27", "node_list": "Hydra-MpiNG-abcdef01-2345-
[1-2]", "node_cnt": 2, "proc_cnt": 2, "work_dir": "/root", "reservation_name":
 "", "tres": { "cpu": 2, "mem": { "val": 1944, "unit": "M" }, "node": 2,
 "billing": 2 }, "account": "", "qos": "", "wc_key": "", "cluster": "unknown",
 "submit_time": "2025-06-19T23:01:38", "eligible_time": "2025-06-19T23:01:38",
 "derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
 0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-1",
 "instance_id": "i-0abc234def56789", "instance_type": "t4g.micro" }, { "name":
 "Hydra-MpiNG-abcdef01-2345-2", "instance_id": "i-0def345abc67890", "instance_type":
 "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 5, "user": "root", "user_id": 0, "group": "root", "group_id":
 0, "name": "wrap", "job_state": "FAILED", "partition": "Hydra-MpiQueue-abcdef01-7",
 "time_limit": "UNLIMITED", "start_time": "2025-06-19T23:09:00", "end_time":
 "2025-06-19T23:09:00", "node_list": "(null)", "node_cnt": 0, "proc_cnt": 0,
 "work_dir": "/root", "reservation_name": "", "tres": { "cpu": 1, "mem": { "val":
 1, "unit": "G" }, "node": 1, "billing": 1 }, "account": "", "qos": "", "wc_key":
 "", "cluster": "unknown", "submit_time": "2025-06-19T23:09:00", "eligible_time":
 "2025-06-19T23:09:00", "derived_exit_code_status": 0, "derived_exit_code_signal": 0,
 "exit_code_status": 0, "exit_code_signal": 1, "node_details": [] } }
{ "jobcomp": { "job_id": 6, "user": "root", "user_id": 0, "group": "root", "group_id":
 0, "name": "wrap", "job_state": "CANCELLED", "partition": "Hydra-MpiQueue-
abcdef01-7", "time_limit": "UNLIMITED", "start_time": "2025-06-19T23:09:36",
 "end_time": "2025-06-19T23:09:36", "node_list": "(null)", "node_cnt": 0, "proc_cnt":
 0, "work_dir": "/root", "reservation_name": "", "tres": { "cpu": 1, "mem":
 { "val": 400, "unit": "M" }, "node": 1, "billing": 1 }, "account": "", "qos":
 "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T23:09:35",
 "eligible_time": "2025-06-19T23:09:36", "het_job_id": 6, "het_job_offset": 0,
 "derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status": 0,
 "exit_code_signal": 1, "node_details": [] } }
{ "jobcomp": { "job_id": 7, "user": "root", "user_id": 0, "group": "root", "group_id":
 0, "name": "wrap", "job_state": "CANCELLED", "partition": "Hydra-MpiQueue-
abcdef01-7", "time_limit": "UNLIMITED", "start_time": "2025-06-19T23:10:03",
 "end_time": "2025-06-19T23:10:03", "node_list": "(null)", "node_cnt": 0, "proc_cnt":
 0, "work_dir": "/root", "reservation_name": "", "tres": { "cpu": 1, "mem":
 { "val": 400, "unit": "M" }, "node": 1, "billing": 1 }, "account": "", "qos":
 "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T23:10:03",
 "eligible_time": "2025-06-19T23:10:03", "het_job_id": 7, "het_job_offset": 0,
 "derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status": 0,
 "exit_code_signal": 1, "node_details": [] } }
{ "jobcomp": { "job_id": 8, "user": "root", "user_id": 0, "group": "root", "group_id":
 0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-abcdef01-7",
 "time_limit": "UNLIMITED", "start_time": "2025-06-19T23:11:24", "end_time":
 "2025-06-19T23:11:24", "node_list": "Hydra-MpiNG-abcdef01-2345-1", "node_cnt":
```

```
 1, "proc_cnt": 1, "work_dir": "/root", "reservation_name": "", "tres": { "cpu":
 1, "mem": { "val": 400, "unit": "M" }, "node": 1, "billing": 1 }, "account": "",
 "qos": "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T23:11:23",
 "eligible_time": "2025-06-19T23:11:23", "het_job_id": 8, "het_job_offset": 0,
 "derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
 0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-1",
 "instance_id": "i-0abc234def56789", "instance_type": "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 9, "user": "root", "user_id": 0, "group": "root", "group_id":
 0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-abcdef01-7",
 "time_limit": "UNLIMITED", "start_time": "2025-06-19T23:11:24", "end_time":
 "2025-06-19T23:11:24", "node_list": "Hydra-MpiNG-abcdef01-2345-2", "node_cnt":
 1, "proc_cnt": 1, "work_dir": "/root", "reservation_name": "", "tres": { "cpu":
 1, "mem": { "val": 400, "unit": "M" }, "node": 1, "billing": 1 }, "account": "",
 "qos": "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T23:11:23",
 "eligible_time": "2025-06-19T23:11:23", "het_job_id": 8, "het_job_offset": 1,
 "derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
 0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-2",
 "instance_id": "i-0def345abc67890", "instance_type": "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 10, "user": "root", "user_id": 0, "group": "root", "group_id":
 0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-abcdef01-7",
 "time_limit": "UNLIMITED", "start_time": "2025-06-19T23:12:24", "end_time":
 "2025-06-19T23:12:24", "node_list":"Hydra-MpiNG-abcdef01-2345-1", "node_cnt":
 1, "proc_cnt": 1, "work_dir": "/root", "reservation_name": "", "tres": { "cpu":
 1, "mem": { "val": 400, "unit": "M" }, "node": 1, "billing": 1 }, "account": "",
 "qos": "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T23:12:14",
 "eligible_time": "2025-06-19T23:12:14", "het_job_id": 10, "het_job_offset": 0,
 "derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
 0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-1",
 "instance_id": "i-0abc234def56789", "instance_type": "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 11, "user": "root", "user_id": 0, "group": "root", "group_id":
 0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-abcdef01-7",
 "time_limit": "UNLIMITED", "start_time": "2025-06-19T23:12:24", "end_time":
 "2025-06-19T23:12:24", "node_list":"Hydra-MpiNG-abcdef01-2345-2", "node_cnt":
 1, "proc_cnt": 1, "work_dir": "/root", "reservation_name": "", "tres": { "cpu":
 1, "mem": { "val": 600, "unit": "M" }, "node": 1, "billing": 1 }, "account": "",
 "qos": "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T23:12:14",
 "eligible_time": "2025-06-19T23:12:14", "het_job_id": 10, "het_job_offset": 1,
 "derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
 0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-2",
 "instance_id": "i-0def345abc67890", "instance_type": "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 13, "user": "root", "user_id": 0, "group": "root", "group_id":
 0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-abcdef01-7",
 "time_limit": "UNLIMITED", "start_time": "2025-06-19T23:47:57", "end_time":
 "2025-06-19T23:47:58", "node_list":"Hydra-MpiNG-abcdef01-2345-1", "node_cnt":
```

```
 1, "proc_cnt": 1, "work_dir": "/root", "reservation_name": "", "tres": { "cpu":
 1, "mem": { "val": 972, "unit": "M" }, "node": 1, "billing": 1 }, "account": "",
 "qos": "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T23:43:56",
 "eligible_time": "2025-06-19T23:43:56" , "array_job_id": 12, "array_task_id": 1,
 "derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
 0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-1",
 "instance_id": "i-0abc345def67890", "instance_type": "t4g.micro" } ] } }
{ "jobcomp": { "job_id": 12, "user": "root", "user_id": 0, "group": "root", "group_id":
 0, "name": "wrap", "job_state": "COMPLETED", "partition": "Hydra-MpiQueue-abcdef01-7",
 "time_limit": "UNLIMITED", "start_time": "2025-06-19T23:47:58", "end_time":
 "2025-06-19T23:47:58", "node_list":"Hydra-MpiNG-abcdef01-2345-1", "node_cnt":
 1, "proc_cnt": 1, "work_dir": "/root", "reservation_name": "", "tres": { "cpu":
 1, "mem": { "val": 972, "unit": "M" }, "node": 1, "billing": 1 }, "account": "",
 "qos": "", "wc_key": "", "cluster": "unknown", "submit_time": "2025-06-19T23:43:56",
 "eligible_time": "2025-06-19T23:43:56" , "array_job_id": 12, "array_task_id": 2,
 "derived_exit_code_status": 0, "derived_exit_code_signal": 0, "exit_code_status":
 0, "exit_code_signal": 0, "node_details": [ { "name": "Hydra-MpiNG-abcdef01-2345-1",
 "instance_id": "i-0abc345def67890", "instance_type": "t4g.micro" } ] } }
```

Scheduler logs in AWS PCS

You can configure AWS PCS to send detailed logging data from your cluster scheduler to Amazon CloudWatch Logs, Amazon Simple Storage Service (Amazon S3), and Amazon Data Firehose. This can assist with monitoring and troubleshooting.

Contents

Prerequisites

The IAM principal that manages the AWS PCS cluster must allow the `pcs:AllowVendedLogDeliveryForResource` action.

The following example IAM policy grants the required permissions.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "PcsAllowVendedLogsDelivery",
         "Effect": "Allow",
         "Action": ["pcs:AllowVendedLogDeliveryForResource"],
         "Resource": [
            "arn:aws:pcs:*::cluster/*"
         ]
      }
   ]
}
```

Set up scheduler logs

You can set up scheduler logs for your AWS PCS cluster with the AWS Management Console or AWS CLI.

AWS Management Console

To set up scheduler logs with the console

1. Open the AWS PCS console.
2. In the navigation pane, choose **Clusters**.
3. Choose the cluster where you want to add scheduler logs.
4. On the cluster details page, choose the **Logs** tab.
5. Under **Scheduler Logs**, choose **Add** to add up to 3 log delivery destinations from among CloudWatch Logs, Amazon S3, and Firehose.
6. Choose **Update log deliveries**.

AWS CLI

To set up scheduler logs with the AWS CLI

1. Create a log delivery destination:

```
aws logs put-delivery-destination --region region \
  --name pcs-logs-destination \
  --delivery-destination-configuration \
  destinationResourceArn=resource-arn
```

Replace:

- `region` — The AWS Region where you want to create the destination, such as `us-east-1`
- `pcs-logs-destination` — A name for the destination
- `resource-arn` — The Amazon Resource Name (ARN) of a CloudWatch Logs log group, S3 bucket, or Firehose delivery stream.

For more information, see PutDeliveryDestination in the *Amazon CloudWatch Logs API Reference*.

2. Set the PCS cluster as a log delivery source:

```
aws logs put-delivery-source --region region \
  --name cluster-logs-source-name \
  --resource-arn cluster-arn \
  --log-type PCS_SCHEDULER_LOGS
```

Replace:

- `region` — The AWS Region of your cluster, such as `us-east-1`
- `cluster-logs-source-name` — A name for the source
- `cluster-arn` — the ARN of your AWS PCS cluster

For more information, see PutDeliverySource in the *Amazon CloudWatch Logs API Reference*.

3. Connect the delivery source to the delivery destination:

```
aws logs create-delivery --region region \
  --delivery-source-name cluster-logs-source \
  --delivery-destination-arn destination-arn
```

Replace:

- *region* — The AWS Region, such as `us-east-1`
- *cluster-logs-source* — The name of your delivery source
- *destination-arn* — The ARN of your delivery destination

For more information, see CreateDelivery in the *Amazon CloudWatch Logs API Reference*.

Scheduler log stream paths and names

The path and name for AWS PCS scheduler logs depend on the destination type.

- **CloudWatch Logs**
  - A CloudWatch Logs stream follows this naming convention.

```
AWSLogs/PCS/${cluster_id}/${log_name}_${scheduler_major_version}.log
```

Example

```
AWSLogs/PCS/abcdef0123/slurmctld_24.05.log
```

- **S3 bucket**
  - An S3 bucket output path follows this naming convention:

```
AWSLogs/${account-id}/PCS/${region}/${cluster_id}/${log_name}/
${scheduler_major_version}/yyyy/MM/dd/HH/
```

Example

```
AWSLogs/111111111111/PCS/us-east-2/abcdef0123/slurmctld/24.05/2024/09/01/00.
```

  - An S3 object name follows this convention:

```
PCS_${log_name}_${scheduler_major_version}_#{expr date 'event_timestamp', format:
 "yyyy-MM-dd-HH"}_${cluster_id}_${hash}.log
```

Example

```
PCS_slurmctld_24.05_2024-09-01-00_abcdef0123_0123abcdef.log
```

Example scheduler log record

AWS PCS scheduler logs are structured. They include fields such as the cluster identifier, scheduler type, major and patch versions, in addition to the log message emitted from the Slurm controller process. Here is an example.

```
{
    "resource_id": "s3431v9rx2",
    "resource_type": "PCS_CLUSTER",
    "event_timestamp": 1721230979,
    "log_level": "info",
    "log_name": "slurmctld",
    "scheduler_type": "slurm",
    "scheduler_major_version": "25.05",
    "scheduler_patch_version": "3",
    "node_type": "controller_primary",
    "message": "[2024-07-17T15:42:58.614+00:00] Running as primary controller\n"
}
```

Monitoring AWS Parallel Computing Service with Amazon CloudWatch

Amazon CloudWatch provides monitoring of your AWS Parallel Computing Service (AWS PCS) cluster health and performance by collecting metrics from the cluster at intervals. These metrics are retained, allowing you to access historical data and gain insights into your cluster's performance over time.

CloudWatch also enables you to monitor the EC2 instances launched by AWS PCS to meet your scaling requirements. While you can inspect logs on running instances, CloudWatch metrics and logging data are typically deleted once instances are terminated. However, you can configure the CloudWatch agent on instances using an EC2 launch template to persist metrics and logs even after instance termination, enabling long-term monitoring and analysis.

Explore the topics in this section to learn more about monitoring AWS PCS using CloudWatch.

Topics

- Monitoring AWS PCS metrics using CloudWatch
- Monitoring AWS PCS instances using Amazon CloudWatch

Monitoring AWS PCS metrics using CloudWatch

You can monitor AWS PCS cluster health using Amazon CloudWatch, which collects data from your cluster and turns it into near real-time metrics. These statistics are retained for a period of 15 months, so that you can access historical information and gain a better perspective on how your cluster is performing. Cluster metrics are sent to CloudWatch at 1-minute periods. For more information about CloudWatch, see What Is Amazon CloudWatch? in the *Amazon CloudWatch User Guide*.

AWS PCS publishes the following metrics into the **AWS/PCS** namespace in CloudWatch. They have a single dimension, `ClusterId`.

Name	Description	Units
ActualCapacity	IdleCapacity + UtilizedCapacity	Count
CapacityUtilization	UtilizedCapacity / ActualCapacity	Count
DesiredCapacity	ActualCapacity + PendingCapacity	Count
IdleCapacity	Count of instances that are running but not allocated to jobs	Count
UtilizedCapacity	Count of instances that are running and allocated to jobs	Count

Monitoring AWS PCS instances using Amazon CloudWatch

AWS PCS launches Amazon EC2 instances as needed to meet the scaling requirements defined in your PCS compute node groups. You can monitor these instances while they are running using Amazon CloudWatch. You can inspect the logs of running instances by logging into them and using interactive command line tools. However, by default, CloudWatch metrics data is only retained for a limited period once an instance is terminated, and instance logs are usually deleted along with the EBS volumes that back the instance. To retain metrics or logging data from the instances launched by PCS after they are terminated, you can configure the CloudWatch agent on your instances with an EC2 launch template. This topic provides an overview of monitoring running instances and provides examples of how to configure persistent instance metrics and logs.

Monitoring running instances

Finding AWS PCS instances

To monitor instances launched by PCS, find the running instances associated with a cluster or compute node group. Then, in the EC2 console for a given instance, inspect the **Status and alarms** and **Monitoring** sections. If login access is configured for those instances, you can connect to them and inspect various log files on the instances. For more information on identifying which instances are managed by PCS, see Finding compute node group instances in AWS PCS.

Enabling detailed metrics

By default, instance metrics are collected at 5-minute intervals. To collect metrics at one minute intervals, enable detailed CloudWatch monitoring in your compute node group launch template. For more information, see Turn on detailed CloudWatch monitoring.

Configuring persistent instance metrics and logs

You can retain the metrics and logs from your instances by installing and configuring the Amazon CloudWatch agent on them. This consists of three main steps:

1. Create a CloudWatch agent configuration.
2. Store the configuration where it can be retrieved by PCS instances.
3. Write an EC2 launch template that installs the CloudWatch agent software, fetches your configuration, and starts the CloudWatch agent using the configuration.

For more information, see Collect metrics, logs, and traces with the CloudWatch agent in the *Amazon CloudWatch User Guide*, and Using Amazon EC2 launch templates with AWS PCS.

Create a CloudWatch Agent configuration

Before deploying the CloudWatch agent on your instances, you must generate a JSON configuration file that specifies the metrics, logs, and traces to collect. Configuration files can be created using a wizard or manually, using a text editor. The configuration file will be created manually for this demonstration.

On a computer where you have the AWS CLI installed, create a CloudWatch configuration file named **config.json** with the contents that follow. You can also use the following URL to download a copy of the file.

```
https://aws-hpc-recipes.s3.amazonaws.com/main/recipes/pcs/cloudwatch/assets/config.json
```

Notes

- The log paths in the sample file are for Amazon Linux 2. If your instances will use a different base operating system, change the paths as appropriate.
- To capture other logs, add additional entries under `collect_list`.
- Values in `{brackets}` are templated variables. For the complete list of supported variables, see Manually create or edit the CloudWatch agent configuration file in the *Amazon CloudWatch User Guide*.
- You can choose to omit `logs` or `metrics` if you don't want to collect these information types.

```
{
    "agent": {
        "metrics_collection_interval": 60
    },
    "logs": {
        "logs_collected": {
            "files": {
                "collect_list": [
                    {
                        "file_path": "/var/log/cloud-init.log",
                        "log_group_class": "STANDARD",
                        "log_group_name": "/PCSLogs/instances",
                        "log_stream_name": "{instance_id}.cloud-init.log",
                        "retention_in_days": 30
                    },
                    {
                        "file_path": "/var/log/cloud-init-output.log",
                        "log_group_class": "STANDARD",
                        "log_stream_name": "{instance_id}.cloud-init-output.log",
                        "log_group_name": "/PCSLogs/instances",
                        "retention_in_days": 30
                    },
                    {
                        "file_path": "/var/log/amazon/pcs/bootstrap.log",
                        "log_group_class": "STANDARD",
                        "log_stream_name": "{instance_id}.bootstrap.log",
                        "log_group_name": "/PCSLogs/instances",
                        "retention_in_days": 30
                    },
                    {
                        "file_path": "/var/log/slurmd.log",
                        "log_group_class": "STANDARD",
                        "log_stream_name": "{instance_id}.slurmd.log",
                        "log_group_name": "/PCSLogs/instances",
                        "retention_in_days": 30
                    },
                    {
                        "file_path": "/var/log/messages",
                        "log_group_class": "STANDARD",
                        "log_stream_name": "{instance_id}.messages",
                        "log_group_name": "/PCSLogs/instances",
                        "retention_in_days": 30
                    },
                    {
                        "file_path": "/var/log/secure",
                        "log_group_class": "STANDARD",
                        "log_stream_name": "{instance_id}.secure",
                        "log_group_name": "/PCSLogs/instances",
                        "retention_in_days": 30
                    }
                ]
            }
        }
    },
    "metrics": {
        "aggregation_dimensions": [
            [
                "InstanceId"
            ]
        ],
        "append_dimensions": {
            "AutoScalingGroupName": "${aws:AutoScalingGroupName}",
            "ImageId": "${aws:ImageId}",
            "InstanceId": "${aws:InstanceId}",
            "InstanceType": "${aws:InstanceType}"
        },
        "metrics_collected": {
            "cpu": {
                "measurement": [
                    "cpu_usage_idle",
                    "cpu_usage_iowait",
                    "cpu_usage_user",
                    "cpu_usage_system"
                ],
                "metrics_collection_interval": 60,
                "resources": [
                    "*"
                ],
                "totalcpu": false
            },
            "disk": {
                "measurement": [
                    "used_percent",
                    "inodes_free"
                ],
                "metrics_collection_interval": 60,
                "resources": [
                    "*"
                ]
            },
            "diskio": {
                "measurement": [
                    "io_time"
                ],
                "metrics_collection_interval": 60,
                "resources": [
                    "*"
                ]
            },
            "mem": {
                "measurement": [
                    "mem_used_percent"
                ],
                "metrics_collection_interval": 60
            },
            "swap": {
                "measurement": [
                    "swap_used_percent"
                ],
                "metrics_collection_interval": 60
            }
        }
    }
}
```

This file instructs the CloudWatch agent to monitor several files that can be helpful in diagnosing errors in instance bootstrapping, authentication and login, and other troubleshooting domains. These include:

- `/var/log/cloud-init.log` – Output from the initial stage of instance configuration
- `/var/log/cloud-init-output.log` – Output from commands that run during instance configuration
- `/var/log/amazon/pcs/bootstrap.log` – Output from PCS-specific operations that run during instance configuration
- `/var/log/slurmd.log` – Output from the Slurm workload manager's daemon slurmd
- `/var/log/messages` – System messages from the kernel, system services, and applications
- `/var/log/secure` – Logs related to authentication attempts, such as SSH, sudo, and other security events

The log files are sent to a CloudWatch log group named `/PCSLogs/instances`. The log streams are a combination of the instance ID and the base name of the log file. The log group has a retention time of 30 days.

In addition, the file instructs CloudWatch agent to collect several common metrics, aggregating them by instance ID.

Store the configuration

The CloudWatch agent configuration file has to be stored where it can be accessed by PCS compute node instances. There are two common ways to do this. You can upload it to an Amazon S3 bucket that your compute node group instances will have access to via their instance profile, Alternatively, you can store it as an SSM parameter in Amazon Systems Manager Parameter Store.

Upload to an S3 bucket

To store your file in S3, use the AWS CLI commands that follow. Before running the command, make these replacements:

- Replace *amzn-s3-demo-bucket* with your own S3 bucket name

First, (this is optional if you have an existing bucket), create a bucket to hold your configuration file(s).

```
aws s3 mb s3://amzn-s3-demo-bucket
```

Next, upload the file to the bucket.

```
aws s3 cp ./config.json s3://amzn-s3-demo-bucket/
```

Store as an SSM parameter

To store your file as an SSM parameter, use the command that follows. Before running the command, make these replacements:

- Replace *region-code* with the AWS Region where you are working with AWS PCS.
- (Optional) Replace *AmazonCloudWatch-PCS* with your own name for the parameter. Note that if you change the prefix of the name from `AmazonCloudWatch-` you will need to specifically add read access to the SSM parameter in your node group instance profile.

```
aws ssm put-parameter \
   --region region-code \
   --name "AmazonCloudWatch-PCS" \
   --type String \
   --value file://config.json
```

Write an EC2 launch template

The specific details for the launch template depend on whether your configuration file is stored in S3 or SSM.

Use a configuration stored in S3

This script installs CloudWatch agent, imports a configuration file from an S3 bucket, and launches the CloudWatch agent with it. Replace the following values in this script with your own details:

- *`amzn-s3-demo-bucket`* – The name of an S3 bucket your account can read from
- *`/config.json`* – Path relative to the S3 bucket root where the configuration is stored

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

packages:
- amazon-cloudwatch-agent

runcmd:
- aws s3 cp s3://amzn-s3-demo-bucket/config.json /etc/s3-cw-config.json
- /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m
 ec2 -s -c file://etc/s3-cw-config.json

--==MYBOUNDARY==--
```

The IAM instance profile for the node group must have access to the bucket. Here is an example IAM policy for the bucket in the user data script above.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

Also note that the instances must allow outbound traffic to the S3 and CloudWatch endpoints. This can be accomplished using security groups or VPC endpoints, depending on your cluster architecture.

Use a configuration stored in SSM

This script installs CloudWatch agent, imports a configuration file from an SSM parameter, and launches the CloudWatch agent with it. Replace the following values in this script with your own details:

- (Optional) Replace *`AmazonCloudWatch-PCS`* with your own name for the parameter.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

packages:
- amazon-cloudwatch-agent

runcmd:
- /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m
 ec2 -s -c ssm:AmazonCloudWatch-PCS

--==MYBOUNDARY==--
```

The IAM instance policy for the node group must have the **CloudWatchAgentServerPolicy** attached to it.

If your parameter name does not start with `AmazonCloudWatch-` you will need to specifically add read access to the SSM parameter in your node group instance profile. Here is an example IAM policy that illustrates this for prefix *`DOC-EXAMPLE-PREFIX`*.

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Sid" : "CustomCwSsmMParamReadOnly",
      "Effect" : "Allow",
      "Action" : [
        "ssm:GetParameter"
      ],
      "Resource" : "arn:aws:ssm:*:*:parameter/DOC-EXAMPLE-PREFIX*"
    }
  ]
}
```

Also note that the instances must allow outbound traffic to the SSM and CloudWatch endpoints. This can be accomplished using security groups or VPC endpoints, depending on your cluster architecture.

Logging AWS Parallel Computing Service API calls using AWS CloudTrail

AWS PCS is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS PCS. CloudTrail captures all API calls for AWS PCS as events. The calls captured include calls from the AWS PCS console and code calls to the AWS PCS API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS PCS. 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 AWS PCS, 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.

AWS PCS information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS PCS, 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.

For an ongoing record of events in your AWS account, including events for AWS PCS, 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 and Receiving CloudTrail log files from multiple accounts

All AWS PCS actions are logged by CloudTrail and are documented in the AWS Parallel Computing Service API Reference. For example, calls to the `CreateComputeNodeGroup`, `UpdateQueue`, and `DeleteCluster` actions generate entries in the CloudTrail log files.

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 AWS service.

For more information, see the CloudTrail userIdentity element.

Understanding CloudTrail log file entries from AWS PCS

A trail is a configuration that enables delivery of events as log files to an 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.

The following example shows a CloudTrail log entry for a `CreateQueue` action.

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:admin",
        "arn": "arn:aws:sts::012345678910:assumed-role/Admin/admin",
        "accountId": "012345678910",
        "accessKeyId": "ASIAY36PTPIEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAY36PTPIEEXAMPLE",
                "arn": "arn:aws:iam::012345678910:role/Admin",
                "accountId": "012345678910",
                "userName": "Admin"
            },
            "attributes": {
                "creationDate": "2024-07-16T17:05:51Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2024-07-16T17:13:09Z",
    "eventSource": "pcs.amazonaws.com",
    "eventName": "CreateQueue",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36",
    "requestParameters": {
        "clientToken": "c13b7baf-2894-42e8-acec-example",
        "clusterIdentifier": "abcdef0123",
        "computeNodeGroupConfigurations": [
            {
                "computeNodeGroupId": "abcdef0123"
            }
        ],
        "queueName": "all"
    },
    "responseElements": {
        "queue": {
            "arn": "arn:aws:pcs:us-east-1:609783872011:cluster/abcdef0123/queue/
abcdef0123",
            "clusterId": "abcdef0123",
            "computeNodeGroupConfigurations": [
                {
                    "computeNodeGroupId": "abcdef0123"
                }
            ],
            "createdAt": "2024-07-16T17:13:09.276069393Z",
            "id": "abcdef0123",
            "modifiedAt": "2024-07-16T17:13:09.276069393Z",
            "name": "all",
            "status": "CREATING"
        }
    },
    "requestID": "a9df46d7-3f6d-43a0-9e3f-example",
    "eventID": "7ab18f88-0040-47f5-8388-example",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "012345678910",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "pcs.us-east-1.amazonaws.com"
    },
    "sessionCredentialFromConsole": "true"
}
```

Endpoints and service quotas for AWS PCS

The following sections describe the endpoints and service quotas for AWS Parallel Computing Service (AWS PCS). Service quotas, formerly referred to as *limits*, are the maximum number of service resources or operations for your AWS account.

Your AWS account has default quotas for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some quotas, and other quotas cannot be increased.

For more information, see AWS service quotas in the *AWS General Reference*.

Contents

Service endpoints

Region name	Region	Endpoint	Protocol
US East (Ohio)	us-east-2	pcs.us-east-2.amazonaws.com pcs-fips.us-east-2.amazonaws.com pcs-fips.us-east-2.api.aws pcs.us-east-2.api.aws	HTTPS
US East (N. Virginia)	us-east-1	pcs.us-east-1.amazonaws.com pcs-fips.us-east-1.amazonaws.com	HTTPS

Region name	Region	Endpoint	Protocol
		pcs-fips.us-east-1.api.aws pcs.us-east-1.api.aws	
US West (Oregon)	us-west-2	pcs.us-west-2.amazonaws.com pcs-fips.us-west-2.amazonaws.com pcs-fips.us-west-2.api.aws pcs.us-west-2.api.aws	HTTPS
Asia Pacific (Singapore)	ap-southeast-1	pcs.ap-southeast-1.amazonaws.com pcs.ap-southeast-1.api.aws	HTTPS
Asia Pacific (Sydney)	ap-southeast-2	pcs.ap-southeast-2.amazonaws.com pcs.ap-southeast-2.api.aws	HTTPS
Asia Pacific (Tokyo)	ap-northeast-1	pcs.ap-northeast-1.amazonaws.com pcs.ap-northeast-1.api.aws	HTTPS
Europe (Frankfurt)	eu-central-1	pcs.eu-central-1.amazonaws.com pcs.eu-central-1.api.aws	HTTPS

Region name	Region	Endpoint	Protocol
Europe (Ireland)	eu-west-1	pcs.eu-west-1.amazonaws.com pcs.eu-west-1.api.aws	HTTPS
Europe (London)	eu-west-2	pcs.eu-west-2.amazonaws.com pcs.eu-west-2.api.aws	HTTPS
Europe (Stockholm)	eu-north-1	pcs.eu-north-1.amazonaws.com pcs.eu-north-1.api.aws	HTTPS
AWS GovCloud (US-East)	us-gov-east-1	pcs.us-gov-east-1.amazonaws.com pcs-fips.us-gov-east-1.amazonaws.com pcs-fips.us-gov-east-1.api.aws pcs.us-gov-east-1.api.aws	HTTPS

Region name	Region	Endpoint	Protocol
AWS GovCloud (US-West)	us-gov-west-1	pcs.us-gov-west-1.amazonaws.com pcs-fips.us-gov-west-1.amazonaws.com pcs-fips.us-gov-west-1.api.aws pcs.us-gov-west-1.api.aws	HTTPS

Service quotas

Name	Default	Adjustable	Description
Clusters	5	Yes	The maximum number of clusters per AWS Region.

Note

The default values are the initial quotas set by AWS. These default values are separate from the actual applied quota values and maximum possible service quotas. For more information, see Terminology in Service Quotas in the *Service Quotas User Guide*.

These service quotas are listed under **AWS Parallel Computing Service (PCS)** in the AWS Management Console. To request a quota increase for values that are shown as adjustable, see Requesting a Quota Increase in the *Service Quotas User Guide*.

Important

Remember to check the current AWS Region setting in the AWS Management Console.

Internal quotas

The following quotas are internal and non-adjustable.

Name	Default	Adjustable	Description
Concurrent cluster creation	1	No	The maximum number of clusters in the `Creating` state per AWS Region.
Compute node groups per cluster	10	No	The maximum number of compute node groups per cluster.
Queues per cluster	10	No	The maximum number of queues per cluster.

Relevant quotas for other AWS services

AWS PCS uses other AWS services. Your service quotas for those services impact your use of AWS PCS.

Amazon EC2 service quotas that impact AWS PCS

- Spot instance requests
- Running on-demand instances
- Launch templates
- Launch template versions
- Amazon EC2 API requests

For more information, see Amazon EC2 service quotas in the *Amazon Elastic Compute Cloud User Guide*.

Troubleshooting problems in AWS Parallel Computing Service

The following topics provide guidance to troubleshoot some problems you might encounter in AWS PCS.

- Cluster updates
- Compute node bootstrap problems
- Custom Slurm settings
- EC2 instances terminated after reboot
- Identity and access
- Slurm reboot issues

An EC2 instance in AWS PCS is terminated and replaced after reboot

Problem overview

After an EC2 instance in a compute node group is rebooted, AWS PCS automatically terminates and replaces the instance.

Why this happens

AWS PCS doesn't support instance reboots. If an EC2 instance is rebooted, AWS PCS considers the instance unhealthy and replaces it. If AWS PCS continuously terminates and replaces your instances, it might be because something reboots your instances after they launch. Some examples include reboots by automation on the EC2 instance (such as an automatic reboot after patching), automation external to the EC2 instance (such as a network management application), another AWS service (such as AWS Systems Manager), or a manual reboot by a person.

What to do

You can check your `slurmctld` or `slurmd` logs to see if your instance was rebooted. For more information, see Scheduler logs in AWS PCS and Monitoring AWS PCS instances using Amazon CloudWatch. The following example `slurmctld` log entry indicates that the instance rebooted:

Example

```
[2024-09-12T06:42:50.393+00:00] validate_node_specs: Node Login-1 unexpectedly rebooted
 boot_time=1726123354 last response=1726123285
```

Rebooting because of patching

A reboot is often required after you apply patches. Don't apply patches directly to an EC2 instance that is part of a AWS PCS compute node group. If you must patch your EC2 instances, you should apply your patches to an updated Amazon Machine Image (AMI) and update your compute node groups to use the updated AMI. New EC2 instances that AWS PCS launches for those compute node groups will use the updated (patched) AMI. For more information, see Custom Amazon Machine Images (AMIs) for AWS PCS.

Troubleshoot compute node bootstrap and registration problems in AWS PCS

When compute nodes fail to bootstrap or register properly with your AWS PCS cluster, you might experience the following symptoms:

- Jobs don't start
- You can't connect to instances in AWS Systems Manager
- Instances shut down unexpectedly
- Instances are continuously replaced

These failures can be caused by problems during EC2 instance launch or during the AWS PCS compute node bootstrap process. This topic describes procedures to help you troubleshoot problems during the AWS PCS node bootstrap process. For more information about troubleshooting EC2 instance launch, see Troubleshoot Amazon EC2 instance launch problems in the *Amazon Elastic Compute Cloud User Guide*.

Bootstrap failures occur when an EC2 instance launches successfully but fails during the process of joining the AWS PCS cluster. The bootstrap process includes two main phases:

- **Node registration –** The EC2 instance calls the RegisterComputeNodeGroupInstance AWS PCS API action to register with the AWS PCS service. Failures can occur due to problems in the following:

  - Permissions
    - Wrong instance profile
  - Networking
    - Can't connect to AWS PCS endpoints
    - Misconfigured AWS PCS endpoint
    - Instance in a public subnet without public IP
    - Multi-NIC instance in a public subnet
  - Cluster secret
    - Cluster secret has been deleted or marked for deletion
- **Slurm integration –** The instance runs `slurmd` and joins the Slurm cluster. Failures can occur due to problems in the following:
  - Permissions
    - Security group configuration
    - Slurmctld unable to ping compute node
  - Custom AMI setup
    - Missing NVIDIA drivers
    - ResumeTimeout reached

How Slurm works on AWS PCS

It might help you to compare the standard way Slurm works to the way Slurm works on AWS PCS.

Standard Slurm job processing

The following steps occur in standard Slurm job processing:

1. When you submit a job, `slurmctld` validates and queues the job.
2. When resources become available, `slurmctld` allocates existing nodes.
3. `slurmd` daemons run jobs on allocated nodes.

Slurm job processing on AWS PCS

The following steps occur in AWS PCS job processing:

1. When you submit a job, slurmctld validates and queues the job.
2. **When additional capacity is needed, AWS PCS uses the launch template for the compute node group to launch new EC2 instances.**
3. **New instances bootstrap into the cluster:**
   a. **Instances register with AWS PCS.**
   b. **Instances join the Slurm cluster.**
4. When resources are ready, slurmctld allocates nodes (including newly bootstrapped ones).
5. slurmd daemons run jobs on allocated nodes.

Retrieve instance logs

The first step in troubleshooting compute node bootstrap problems is to retrieve the instance logs. You can use one of the following methods:

AWS CLI

Retrieve the console output from the compute node using the following command:

```
aws ec2 get-console-output --region us-east-1 --instance-id i-1234567890abcdef0 --
output text
```

Replace *us-east-1* with your AWS Region and *i-1234567890abcdef0* with your instance ID.

AWS Systems Manager

If you can connect to the instance using Systems Manager, you can view the bootstrap log file directly:

1. Connect to the instance using Systems Manager. For more information, see Starting a session in the *Systems Manager User Guide*.
2. View the bootstrap log file:

```
sudo cat /var/log/amazon/pcs/bootstrap.log
```

> **Note**
>
> If there's an issue during the initialization phase, you might need to wait approximately 20 minutes before you can connect to the instance. Systems Manager and SSH services start only after initialization is completed or when bootstrap execution reaches a timeout in case of failure.

Retrieve VPC/Subnet/Security Groups from an instance ID

To troubleshoot problems with your compute nodes, you might need to retrieve information about the VPC, subnet, and security groups associated with your instances. If you don't know your instance IDs, see Finding compute node group instances in AWS PCS.

AWS Management Console

To get VPC, subnet, and security groups

1. Open the Amazon EC2 console.
2. Choose **Instances**.
3. In the **Instances** table, choose the instance ID.
4. Find the **VPC ID** and **Subnet ID** in the displayed instance summary for the instance.
5. In the instance summary, choose the **Security** tab.
6. Find the **Security groups** in the **Security** tab.

AWS CLI

Use the following command to retrieve VPC, subnet, and security group information for your instance:

```
aws ec2 describe-instances --instance-ids i-1234567890abcdef0 --query
 'Reservations[*].Instances[*].
{InstanceId:InstanceId,VpcId:VpcId,SubnetId:SubnetId,SecurityGroups:SecurityGroups[*].GroupI
 --output table
```

Node registration problems

Node registration is the first action executed by a compute node during bootstrap. The node calls the AWS PCS API endpoint to register itself with AWS PCS. Registration failures usually show error messages similar to the following:

```
<13>Nov  5 08:10:27 user-data: Recipe: aws-pcs-environment::node_registration
<13>Nov  5 08:10:27 user-data:   * ruby_block[Register NodeGroup Instance] action
 run[2024-11-05T08:10:27+00:00] INFO: Processing ruby_block[Register NodeGroup
 Instance] action run (aws-pcs-environment::node_registration line 19)
<13>Nov  5 08:15:46 user-data: 
<13>Nov  5 08:15:46 user-data: 
<13>Nov  5 08:15:46 user-data: 
 ================================================================================
<13>Nov  5 08:15:46 user-data:     Error executing action `run` on resource
 'ruby_block[Register NodeGroup Instance]'
<13>Nov  5 08:15:46 user-data: 
 ================================================================================
<13>Nov  5 08:15:46 user-data: 
<13>Nov  5 08:15:46 user-data:     EOFError
```

Wrong instance profile

If the instance is unable to register, verify that the instance profile associated with the compute node has the `pcs:RegisterComputeNodeGroupInstance` permission.

For more information about how to create a valid instance profile, see Create an instance profile for AWS PCS.

Can't connect to AWS PCS endpoints

If your compute nodes are in a private subnet, make sure that you have configured VPC endpoints for AWS PCS or that your subnet has a route to a NAT gateway for internet access. For more information, see the following:

- Access an AWS service using an interface VPC endpoint in the *Amazon Virtual Private Cloud AWS PrivateLink* guide.
- Endpoints and service quotas for AWS PCS.
- Connect your VPC to other networks in the *Amazon Virtual Private Cloud User Guide*

- AWS PCS Networking

Misconfigured AWS PCS endpoint

If you see an error message similar to the following, verify the policy associated with your AWS PCS VPC endpoint:

```
com.amazon.coral.security.AccessDeniedException: User: arn:aws:sts::xxx:assumed-
role/rolename/i-instanceid is not authorized to perform:
 pcs:RegisterComputeNodeGroupInstance on resource: arn:aws:pcs:us-west-2:xxx:cluster/
cluster-id as either the resource does not exist, some policy explicitly denies access,
 or no policy grants access
```

For more information about how to configure VPC interface endpoints for AWS PCS, see Access AWS Parallel Computing Service using an interface endpoint (AWS PrivateLink).

Instance in a public subnet without public IP

If your subnet doesn't have **auto-assign public IP** enabled and your route configuration uses an internet gateway, instances can't communicate with the AWS PCS API.

Instances in a subnet with an internet gateway must have a public IP address. To resolve this issue, choose one of the following options:

- Add a VPC endpoint for AWS PCS to your cluster VPC. This enables instances to communicate with AWS PCS without the need for a public IP address to pass through the internet gateway.
- Use a private subnet with a NAT gateway, so that a public IP address is not required.
- Enable automatic public IP address assignment through your subnet or launch template so that instances can contact the API through the internet gateway. Note that this option is not valid for multi-network interface instances.

Multi-NIC instance in a public subnet

You must use a private subnet if you use an instance type that has multiple network interfaces (NICs).

AWS public IP addresses can only be assigned to instances launched with a single network interface. For more information about IP addresses, see Assign a public IPv4 address during instance launch in the *Amazon EC2 User Guide for Linux Instances*.

Multi-NIC instance types require a NAT gateway or an internal proxy in the subnet to access the AWS PCS endpoint. Alternatively, you can add a VPC endpoint for AWS PCS to your cluster VPC.

Cluster secret has been deleted or marked for deletion

If the Slurm shared secret in AWS Secrets Manager has been deleted or marked for deletion, compute nodes will fail to register and your cluster will become impaired.

AWS PCS automatically creates a Slurm shared secret in AWS Secrets Manager (with name format: `pcs!slurm-secret-<cluster-id>`) when you create a cluster. This secret is required for secure communications in the cluster. For more information, see Working with cluster secrets in AWS PCS.

If this secret is deleted or marked for deletion, new nodes will not be able to join the cluster and the controller or other cluster daemons (such as `slurmd` and `slurmdbd`) might not be able to rejoin the cluster if restarted.

To resolve this issue, you can restore the deleted secret if it's still within the recovery window. For detailed instructions, see Restore an AWS Secrets Manager secret.

If the recovery window expires, the secret can't be restored and the affected AWS PCS cluster can't be restored. You need to create a new cluster with the same configuration. AWS PCS automatically creates a new scheduler secret.

Slurm cluster join problems

After successful node registration, the compute node attempts to join the Slurm cluster. The `slurmd` daemon on the node contacts the Slurm controller to register with the cluster. Slurm join failures usually show error messages similar to the following:

```
<13>Nov  5 17:20:29 user-data: [2024-11-05T17:20:28+00:00] FATAL:
 Mixlib::ShellOut::ShellCommandFailed: service[slurmd] (aws-pcs-slurm::finalize_slurm
 line 18) had an error: Mixlib::ShellOut::ShellCommandFailed: Expected process to exit
 with [0], but received '1'
<13>Nov  5 17:20:29 user-data: ---- Begin output of ["/usr/bin/systemctl", "--system",
 "start", "slurmd"] ----
<13>Nov  5 17:20:29 user-data: STDOUT:
<13>Nov  5 17:20:29 user-data: STDERR: Job for slurmd.service failed because the
 control process exited with error code. See "systemctl status slurmd.service" and
 "journalctl -xe" for details.
<13>Nov  5 17:20:29 user-data: ---- End output of ["/usr/bin/systemctl", "--system",
 "start", "slurmd"] ----
```

Security group configuration

Verify that your security groups are configured correctly to allow communication between compute nodes and the Slurm controller. The security groups must allow the following traffic:

- Port 6817 for `slurmd` to communicate with `slurmctld`
- Port 6818 for `slurmctld` to ping `slurmd`

For more information about security group requirements, see the following topics:

- Create security groups for AWS PCS
- Create launch templates for AWS PCS
- Security group requirements and considerations

> **Important**
>
> The cluster security group that you associated with your cluster during cluster creation must also be configured in your compute node group security groups to allow compute nodes to communicate with the controller.

Missing NVIDIA drivers

If the instance bootstraps correctly but jobs don't start, and you see error messages similar to the following in your instance logs, you might be missing NVIDIA drivers:

```
<13>Dec  2 13:52:00 user-data: [2024-12-02T13:52:00.094+00:00] - /opt/aws/pcs/bin/
pcs_bootstrap_config_always.sh: INFO: nvidia-smi not found!
...
<13>Dec  2 13:54:10 user-data: Job for slurmd.service failed because the control
 process exited with error code. See "systemctl status slurmd.service" and "journalctl
 -xe" for details.
<13>Dec  2 13:54:12 user-data: [2024-12-02T13:54:12.718+00:00] - /opt/aws/pcs/bin/
pcs_bootstrap_finalize.sh: INFO: systemctl could not start slurmd!
```

If you connect to the instance and check the `slurmd` daemon status, you might see an error similar to the following:

```
$ systemctl status slurmd
...
fatal: can't stat gres.conf file /dev/nvidia0: No such file or directory
```

To resolve this issue, install NVIDIA drivers on your custom AMI. For more information, see Step 4 – (Optional) Install additional drivers, libraries, and application software.

ResumeTimeout reached

If a compute node and its EC2 instance are terminated because the node is unhealthy, AWS PCS might not support the AMI or there might be network problems. The EC2 instance runs for approximately 30 minutes until Slurm's ResumeTimeout is reached and marks the node as `DOWN`.

If the instance doesn't bootstrap correctly and isn't registered with AWS PCS (no `RegisterComputeNodeGroupInstance` call for the EC2 instance), check your instance logs for error messages similar to the following:

```
/opt/aws/pcs/bin/pcs_bootstrap_init.sh: No such file or directory
```

This error indicates that the AWS PCS bootstrap software is not part of the AMI. To resolve this issue, ensure that your custom AMI includes the AWS PCS bootstrap software. For more information, see Custom Amazon Machine Images (AMIs) for AWS PCS.

Slurmctld unable to ping compute node

If the instance correctly executes the bootstrap procedure and is registered with AWS PCS, but `slurmctld` is unable to see it and submit jobs to it, the instance is set to `DOWN` after some time and then terminated.

This might be caused by misconfigured security groups. For example, if port 6817 is enabled to allow `slurmd` to communicate with `slurmctld`, but port 6818 is missing to allow `slurmctld` to ping `slurmd`.

Verify that your security groups include all required rules as documented in Security group requirements and considerations.

Document history for the AWS PCS User Guide

The following table describes the important changes to the documentation for AWS PCS.

Date	Change	Documentation updates	API versions updated
March 10, 2026	Updated PCS agent	Updated the AMI topic for AWS PCS agent 1.3.2-1. Fixed an issue affecting RHEL 8.10 and Rocky Linux 8.10 compute node bootstrap . For more information, see Software installers to build custom AMIs for AWS PCS and AWS PCS agent versions.	N/A
February 11, 2026	AWS PCS released in Asia Pacific (Mumbai) and Europe (Paris)	AWS PCS is now available in Asia Pacific (Mumbai) (ap-south-1) and Europe (Paris) (eu-west-3). CloudFormation templates are available to get started in the Asia Pacific (Mumbai) AWS Region and Europe (Paris) AWS Region. For more information, see Use CloudFormation to create a sample AWS PCS cluster and CloudFormation templates to create a sample AWS PCS cluster.	N/A

Date	Change	Documentation updates	API versions updated
November 18, 2025	New feature: Slurm REST API	Slurm REST API is now supported for Slurm 25.05 or later. For more information, see Slurm REST API in AWS PCS.	AWS SDK: 2025-11-18
November 10, 2025	New feature: Slurm CLI filter plugin support	AWS PCS now supports Slurm CLI filter plugins to run custom Lua scripts that validate and modify job submissio n parameters before they reach the Slurm controller. Use CLI filters to enforce custom policies, set default parameters, and provide user guidance during job submission. This feature requires Slurm version 25.05 or later. For more information, see Use Slurm CLI Filter Plugins to customize job submission in AWS PCS.	N/A
November 7, 2025	Updated PCS agent	Updated the AMI topic for AWS PCS agent 1.3.1-1. For more information, see Software installers to build custom AMIs for AWS PCS and AWS PCS agent versions.	N/A

Date	Change	Documentation updates	API versions updated
November 3, 2025	Updated PCS agent and Slurm installers	Updated the AMI topic for AWS PCS agent 1.3.0-1 and Slurm installers 24.11.6-2, 24.05.8-2, and 23.11.10-4. Updated list of supported operating systems. For more information, see Software installers to build custom AMIs for AWS PCS and AWS PCS agent versions.	N/A
October 23, 2025	Updated content: pcs-multi-cluster-login-configure.sh	Fixed some errors in the multi-cluster login node configuration script. For more information, see AWS PCS multi-cluster login node configuration script code.	N/A

Date	Change	Documentation updates	API versions updated
October 21, 2025	New feature: Cluster secret rotation	AWS PCS now supports cluster secret rotation to enhance security. For more information, see Rotating cluster secrets in AWS PCS. Updated minimum administrator permissions to support cluster secret rotation. For more information, see Minimum permissions for AWS PCS.	N/A
October 17, 2025	New topic: multi-cluster login node configuration script	Added a new topic that provides a script to configure a standalone login node to connect to multiple AWS PCS clusters. The script automates the configuration of multiple Slurm `sackd` daemons and creates activation scripts for cluster interaction. For more information, see Connecting a standalone login node to multiple clusters in AWS PCS.	N/A

Date	Change	Documentation updates	API versions updated
October 16, 2025	Updated for Slurm 25.05	Updated the user guide for Slurm 25.05 support. Slurm 25.05 is now the default version. For more information, see the following: • Slurm versions in AWS PCS • Software installers to build custom AMIs for AWS PCS • Release notes for AWS PCS sample AMIs	N/A
October 16, 2025	Updated PCS agent	Updated the AMI topic for AWS PCS agent 1.2.2-1. For more information, see Software installers to build custom AMIs for AWS PCS and AWS PCS agent versions.	N/A

Date	Change	Documentation updates	API versions updated
October 2, 2025	New features: Slurm node reboot, cluster updates, and custom Slurm settings	AWS PCS adds support for multiple new features: • **Slurm node reboot** – Use Slurm's native `scontrol reboot` command to reboot compute nodes without instance replacement. For more information, see Rebooting compute nodes with Slurm in AWS PCS. • **Cluster updates** – Modify cluster configurations post-creation without rebuilds. For more information, see Updating a cluster in AWS PCS. • **Slurm custom settings** – Configure advanced Slurm parameters across Cluster, Queue, and Compute Node Group resources. For more information, see Configuring custom Slurm settings in AWS PCS.	2025-10-01

Date	Change	Documentation updates	API versions updated
September 23, 2025	New troubleshooting topic: Compute node bootstrap problems	Added troubleshooting guidance for diagnosing and resolving compute node bootstrap issues. For more information, see Troubleshoot compute node bootstrap and registration problems in AWS PCS.	N/A
September 17, 2025	New feature: Capacity Blocks for ML	AWS PCS now supports Amazon EC2 Capacity Blocks for ML, which enable you to reserve GPU-based accelerated computing instances for your clusters. For more information, see Using Amazon EC2 Capacity Blocks for ML with AWS PCS. Minimum permissions to support Capacity Blocks are now part of the minimum permissions for a service administrator. For more information, see Minimum permissions for AWS PCS.	2025-09-17

Date	Change	Documentation updates	API versions updated
September 11, 2025	AWS managed policy update	AWS PCS updated the AWSPCSServiceRolePolicy to support Capacity Blocks. For more information, see AWS managed policies for AWS Parallel Computing Service.	N/A
August 14, 2025	Updated instance profile documentation	Enhanced the instance profile documentation with comprehensive CLI instructions for creating IAM roles and instance profiles. Added step-by-step procedures for setting up instance profiles using the AWS CLI and improved guidance for finding instance profiles used with AWS PCS. For more information, see IAM instance profiles for AWS Parallel Computing Service.	2025-08-14

Date	Change	Documentation updates	API versions updated
August 1, 2025	New topic: SPANK plugins	Added documentation for SPANK (Slurm Plug-in Architecture for Node and job Kontrol) plugins that you can use to extend and modify Slurm's behavior during job launch and execution on AWS PCS clusters. For more information, see Extend Slurm functionality on AWS PCS with SPANK plugins .	N/A
August 1, 2025	IPv6 networking support	Added support for IPv6 networking when creating AWS PCS clusters. You can now choose IPv6 as the network type for your cluster, with corresponding updates to VPC requirements, subnet configuration, security group settings, and cluster creation procedures. For more information, see AWS PCS VPC and subnet requirements and considerations and Creating a cluster in AWS PCS.	2025-08-01

Date	Change	Documentation updates	API versions updated
July 3, 2025	AWS PCS released in Europe (London)	AWS PCS is now available in Europe (London) (eu-west-2). CloudFormation templates are available to get started in the Europe (London) AWS Region. For more information, see Use CloudFormation to create a sample AWS PCS cluster and CloudFormation templates to create a sample AWS PCS cluster.	N/A
July 1, 2025	Updated console instructions	You can now have AWS PCS create a basic instance profile and security group for you when you create a cluster and compute node group in the console. For more information, see: • Creating a cluster in AWS PCS • Creating a compute node group in AWS PCS • IAM instance profiles for AWS Parallel Computing Service	N/A

Date	Change	Documentation updates	API versions updated
June 23, 2025	New managed policy: AWSPCSComputeNodePolicy	Added a new managed policy that grants permission to AWS PCS compute nodes to connect to AWS PCS clusters. For more information, see AWS managed policy: AWSPCSComputeNodePolicy .	N/A
June 19, 2025	New topic: job completion logs	Use job completion logs to record details about jobs when they complete, at no additional cost. For more information, see Job completion logs in AWS PCS.	N/A

Date	Change	Documentation updates	API versions updated
June 18, 2025	AWS PCS release in AWS GovCloud (US)	AWS PCS is now available in AWS GovCloud (US-East) (us-gov-east-1) and AWS GovCloud (US-West) (us-gov-west-1). CloudFormation templates are available to get started in the AWS GovCloud (US) Regions. For more information, see Use CloudFormation to create a sample AWS PCS cluster and CloudFormation templates to create a sample AWS PCS cluster. For more information about the AWS PCS service endpoints in AWS GovCloud (US) Regions, see Endpoints and service quotas for AWS PCS. For more information about differences in AWS GovCloud (US) Regions, see AWS PCS in AWS GovCloud (US) in the *AWS GovCloud (US) User Guide* .	N/A

Date	Change	Documentation updates	API versions updated
June 18, 2025	Updated PCS agent	Updated the AMI topic for AWS PCS agent 1.2.1-1. For more information, see Software installers to build custom AMIs for AWS PCS.	N/A
May 15, 2025	New feature: accounting	Slurm accounting is now supported for Slurm 24.11 or later. For more information, see Slurm accounting in AWS PCS.	AWS SDK: 2025-05-15
May 15, 2025	Updated for Slurm 24.11	Updated the user guide for Slurm 24.11.5 support. For more information, see the following: • Slurm versions in AWS PCS • Software installers to build custom AMIs for AWS PCS • Release notes for AWS PCS sample AMIs	N/A

Date	Change	Documentation updates	API versions updated
May 5, 2025	Updated Slurm versions FAQ	Updated the Slurm versions frequently asked questions (FAQ) about Slurm versions nearing or beyond end of life (EOL). For more information, see Frequently asked questions about Slurm versions in AWS PCS.	N/A
April 17, 2025	New topic: how to get compute node group details	Learn how to get details for an AWS PCS compute node group, such as its ID, ARN, and AMI ID. For more information, see Get compute node group details in AWS PCS.	N/A
April 2, 2025	Updated Slurm installer	Updated the AMI topic for Slurm installer 24.05.7-1. For more information, see Software installers to build custom AMIs for AWS PCS.	N/A
March 28, 2025	Added quotas for maximum number of compute node groups and queues	Added internal non-adjustable quotas for the maximum number of compute node groups per cluster and the maximum number of queues per cluster. For more information, see Internal quotas.	N/A

Date	Change	Documentation updates	API versions updated
March 14, 2025	Changed a property key in the CloudFormation template	`Id` is now `TemplateId` for the `CustomLaunchTemplate` property in the CloudFormation template. For more information, see Resources in Parts of a CloudFormation template for AWS PCS.	N/A
March 13, 2025	Added version information for the AWS PCS agent and Slurm	Added a new topic that describes the changes for each version of the AWS PCS agent. For more information, see AWS PCS agent versions. Added more information to the Slurm versions topic that describes important support dates and detailed release notes for AWS PCS support for Slurm. For more information, see Slurm versions in AWS PCS.	N/A

Date	Change	Documentation updates	API versions updated
March 7, 2025	Updated PCS agent	Updated the AMI topic for AWS PCS agent 1.2.0-1. For more information, see Software installers to build custom AMIs for AWS PCS.	N/A
February 3, 2025	Added a topic about using AWS CloudFormation with AWS PCS	Added a topic to the user guide that provides an example of how to use CloudFormation with AWS PCS. The topic provides a procedure to use a sample CloudFormation template to create the sample AWS PCS cluster, and briefly describes the sections of that template. For more information, see Get started with CloudFormation and AWS PCS.	N/A
December 18, 2024	Updated for Slurm 24.05	Updated the user guide for Slurm 24.05 support. For more information, see Software installers to build custom AMIs for AWS PCS and Release notes for AWS PCS sample AMIs.	N/A

Date	Change	Documentation updates	API versions updated
December 18, 2024	Updated NVIDIA versions for Slurm 23.11 sample AMIs	Updated NVIDIA driver and CUDA versions in the Slurm 23.11 sample AMIs. For more information, see Release notes for AWS PCS sample AMIs.	N/A
December 17, 2024	Updated Slurm installer	Updated the AMI topic for Slurm installer 23.11.10-3. For more information, see Software installers to build custom AMIs for AWS PCS.	N/A
December 13, 2024	Updated PCS agent	Updated the AMI topic for AWS PCS agent 1.1.1-1. For more information, see Software installers to build custom AMIs for AWS PCS.	N/A
December 6, 2024	Updated PCS agent and Slurm installer	Updated the AMI topic for AWS PCS agent 1.1.0-1 and Slurm installer 23.11.10-2. For more information, see Software installers to build custom AMIs for AWS PCS.	N/A
December 6, 2024	Added a topic about OS support	For more information, see Supported operating systems in AWS PCS.	N/A

Date	Change	Documentation updates	API versions updated
November 8, 2024	Reorganized user guide	We reorganized the user guide to bring topics to the top level, moved some topics to their own pages, and grouped similar topics together.	N/A
November 7, 2024	Updated AMI topics	Updated the AMI topic for Slurm 23.11.10 and libjwt 17.0. For more information, see Software installers to build custom AMIs for AWS PCS and Step 3 – Install Slurm. Simplified and corrected the release notes for AMIs. For more informati on, see Release notes for AWS PCS sample AMIs.	N/A
November 7, 2024	Added a new topic about using encrypted EBS volumes with AWS PCS	Added a topic that describes the KMS key policy required for encrypted EBS volumes in AWS PCS. For more information, see Required KMS key policy for use with encrypted EBS volumes in AWS PCS.	N/A

Date	Change	Documentation updates	API versions updated
October 18, 2024	AWS PCS agent 1.0.1-1 released	Updated AMI-related documentation to refer to AWS PCS agent version 1.0.1-1. For more information, see Software installers to build custom AMIs for AWS PCS and Step 2 – Install the AWS PCS agent.	N/A
October 10, 2024	Added a troubleshooting chapter	Added a troubleshooting chapter with a topic about EC2 instances being automatically replaced after a reboot. For more information, see Troubleshooting problems in AWS Parallel Computing Service.	N/A
September 23, 2024	Updated the minimum permissions to use API actions and for a service administrator	The `ec2:DescribeInstanceTypeOfferings` permission is now required for the `CreateComputeNodeGroup` and `UpdateComputeNodeGroup` API actions. For more information, see Minimum permissions for AWS PCS.	N/A

Date	Change	Documentation updates	API versions updated
September 5, 2024	Updated the example IAM policy for the minimum permissions for a service administrator	For more information, see Minimum permissions for a service administrator.	N/A
September 5, 2024	Added a missing permission to the JSON in the managed policies page	This was a correction to the documentation only. The actual managed policy wasn't changed. For more information, see AWS managed policies for AWS Parallel Computing Service.	N/A
August 28, 2024	Managed policies page added	For more information, see AWS managed policies for AWS Parallel Computing Service.	N/A
August 28, 2024	AWS PCS release	Initial release of the AWS PCS user guide.	AWS SDK: 2024-08-28

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.