Once the connection succeeds, a message confirms the connection and the start of the Spark application. You can begin submitting your data processing jobs to your EMR Serverless application.

Stop or delete an EMR Serverless application from the Studio UI

You can stop (transition to the `Stopped` state) or delete (transition to the `Deleted` state) an EMR Serverless application from the list of applications in the Studio UI.

To stop or delete an application, navigate to the list of available EMR Serverless applications.

1. In the Studio UI, navigate to the left-side panel and select the **Data** node in the left navigation menu. Then, scroll and choose the **Amazon EMR applications and clusters** option. This opens up a page that displays the Amazon EMR applications that you can access from within the Studio environment, under the **Serverless applications** tab.
2. Select the name of the application that you want to stop or delete, and then choose the corresponding **Stop** or **Delete** button.
3. A confirmation message informs you that any pending job will be lost permanently.

Data preparation using Amazon EMR

Important

Amazon SageMaker Studio and Amazon SageMaker Studio Classic are two of the machine learning environments that you can use to interact with SageMaker AI.
If your domain was created after November 30, 2023, Studio is your default experience.
If your domain was created before November 30, 2023, Amazon SageMaker Studio Classic is your default experience. To use Studio if Amazon SageMaker Studio Classic is your default experience, see Migration from Amazon SageMaker Studio Classic.
When you migrate from Amazon SageMaker Studio Classic to Amazon SageMaker Studio, there is no loss in feature availability. Studio Classic also exists as an application within Amazon SageMaker Studio to help you run your legacy machine learning workflows.

Amazon SageMaker Studio and Studio Classic come with built-in integration with Amazon EMR. Within JupyterLab and Studio Classic notebooks, data scientists and data engineers can discover and connect to existing Amazon EMR clusters, then interactively explore, visualize, and prepare large-scale data for machine learning using Apache Spark, Apache Hive, or Presto. With a single click, they can access the Spark UI to monitor the status and metrics of their Spark jobs without leaving their notebook.

Administrators can create CloudFormation templates that define Amazon EMR clusters. They can then make those cluster templates available in the AWS Service Catalog for Studio and Studio Classic users to launch. Data scientists can then choose a predefined template to self-provision an Amazon EMR cluster directly from their Studio environment. Administrators can further parameterize the templates to let users choose aspects of the cluster within predefined values. For example, users may want to specify the number of core nodes or select the instance type of a node from a dropdown menu.

Using CloudFormation, administrators can control the organizational, security, and networking setup of Amazon EMR clusters. Data scientists and data engineers can then customize those templates for their workloads to create on-demand Amazon EMR clusters directly from Studio and Studio Classic without setting up complex configurations. Users can terminate Amazon EMR clusters after use.

- **If you are an administrator**:

  Ensure that you have enabled communication between Studio or Studio Classic and Amazon EMR clusters. For instructions, see the Configure network access for your Amazon EMR cluster section. Once this communication is enabled, you can:

  - Configure Amazon EMR CloudFormation templates in the Service Catalog
  - Configure listing Amazon EMR clusters
- **If you are a data scientist or data engineer**, you can:
  - Launch an Amazon EMR cluster from Studio or Studio Classic
  - List Amazon EMR clusters from Studio or Studio Classic
  - Connect to an Amazon EMR cluster from SageMaker Studio or Studio Classic
  - Terminate an Amazon EMR cluster from Studio or Studio Classic
  - Access Spark UI from Studio or Studio Classic

List of topics

- User guide
- Blogs and whitepapers
- Troubleshooting

Quickstart: Create a SageMaker AI sandbox domain to launch Amazon EMR clusters in Studio

This section walks you through the quick set up of a complete test environment in Amazon SageMaker Studio. You will be creating a new Studio domain that lets users launch new Amazon EMR clusters directly from Studio. The steps provide an example notebook that you can connect to an Amazon EMR cluster to start running Spark workloads. Using this notebook, you will build a Retrieval Augmented Generation System (RAG) using Amazon EMR Spark distributed processing and OpenSearch vector database.

Note

To get started, sign in to the AWS Management Console using an AWS Identity and Access Management (IAM) user account with admin permissions. For information on how to sign up for an AWS account and create a user with administrative access, see the section called "Complete Amazon SageMaker AI prerequisites".

To set up your Studio test environment and start running Spark jobs:

- Step 1: Create a SageMaker AI domain for launching Amazon EMR clusters in Studio
- Step 2: Launch a new Amazon EMR cluster from Studio UI
- Step 3: Connect a JupyterLab notebook to the Amazon EMR cluster
- Step 4: Clean up your CloudFormation stack

Step 1: Create a SageMaker AI domain for launching Amazon EMR clusters in Studio

In the following steps, you apply a CloudFormation stack to automatically create a new SageMaker AI domain. The stack also creates a user profile and configures the needed environment and permissions. The SageMaker AI domain is configured to let you directly launch Amazon EMR clusters from Studio. For this example, the Amazon EMR clusters are created in the same AWS account as SageMaker AI without authentication. You can find additional CloudFormation stacks supporting various authentication methods like Kerberos in the getting_started GitHub repository.

Note

SageMaker AI allows 5 Studio domains per AWS account and AWS Region by default. Ensure your account has no more than 4 domains in your region before you create your stack.

Follow these steps to set up a SageMaker AI domain for launching Amazon EMR clusters from Studio.

1. Download the raw file of this CloudFormation template from the `sagemaker-studio-emr` GitHub repository.
2. Go to the CloudFormation console: https://console.aws.amazon.com/cloudformation
3. Choose **Create stack** and select **With new resources (standard)** from the drop down menu.
4. In **Step 1**:
    a. In the **Prepare template** section, select **Choose an existing template**.
    b. In the **Specify template** section, choose **Upload a template file**.
    c. Upload the downloaded CloudFormation template and choose **Next**.
5. In **Step 2**, enter a **Stack name** and a **SageMakerDomainName** then choose **Next**.
6. In **Step 3**, keep all default values and choose **Next**.
7. In **Step 4**, check the box to acknowledge resource creation and choose **Create stack**. This creates a Studio domain in your account and region.

Step 2: Launch a new Amazon EMR cluster from Studio UI

In the following steps, you create a new Amazon EMR cluster from the Studio UI.

1. Go to the SageMaker AI console at https://console.aws.amazon.com/sagemaker/ and choose **Domains** in the left menu.
2. Click on your domain name **GenerativeAIDomain** to open the **Domain details** page.
3. Launch Studio from the user profile `genai-user`.
4. In the left navigation pane, go to **Data** then **Amazon EMR Clusters**.
5. On the Amazon EMR clusters page, choose **Create**. Select the template **SageMaker Studio Domain No Auth EMR** created by the CloudFormation stack and then choose **Next**.

6. Enter a name for the new Amazon EMR cluster. Optionally update other parameters such as the instance type of core and master nodes, idle timeout, or number of core nodes.
7. Choose **Create resource** to launch the new Amazon EMR cluster.

   After creating the Amazon EMR cluster, follow the status on the **EMR Clusters** page. When the status changes to `Running/Waiting`, your Amazon EMR cluster is ready to use in Studio.

Step 3: Connect a JupyterLab notebook to the Amazon EMR cluster

In the following steps, you connect a notebook in JupyterLab to your running Amazon EMR cluster. For this example, you import a notebook allowing you to build a Retrieval Augmented Generation (RAG) system using Amazon EMR Spark distributed processing and OpenSearch vector database.

1. **Launch JupyterLab**

   From Studio, launch the JupyterLab application.
2. **Create a private space**

   If you have not created a space for your JupyterLab application, choose **Create a JupyterLab space**. Enter a name for the space, and keep the space as **Private**. Leave all other settings at their default values, and then choose **Create space**.

   Otherwise, run your JupyterLab space to launch a JupyterLab application.
3. **Deploy your LLM and embedding models for inference**

   - From the top menu, choose **File**, **New**, and then **Terminal**.
   - In the terminal, run the following command.

   ```
   wget --no-check-certificate https://raw.githubusercontent.com/
   aws-samples/sagemaker-studio-foundation-models/main/lab-00-setup/
   Lab_0_Warm_Up_Deploy_EmbeddingModel_Llama2_on_Nvidia.ipynb
   mkdir AWSGuides
   cd AWSGuides
   wget --no-check-certificate https://raw.githubusercontent.com/aws-
   samples/sagemaker-studio-foundation-models/main/lab-03-rag/AWSGuides/
   AmazonSageMakerDeveloperGuide.pdf
   wget --no-check-certificate https://raw.githubusercontent.com/aws-
   samples/sagemaker-studio-foundation-models/main/lab-03-rag/AWSGuides/
   EC2DeveloperGuide.pdf
   ```

```
wget --no-check-certificate https://raw.githubusercontent.com/aws-samples/
sagemaker-studio-foundation-models/main/lab-03-rag/AWSGuides/S3DeveloperGuide.pdf
```

This retrieves the the `Lab_0_Warm_Up_Deploy_EmbeddingModel_Llama2_on_Nvidia.ipynb` notebook to your local directory and downloads three PDF files into a local `AWSGuides` folder.

- Open `lab-00-setup/Lab_0_Warm_Up_Deploy_EmbeddingModel_Llama2_on_Nvidia.ipynb`, keep the `Python 3 (ipykernel)` kernel, and run each cell.

> **Warning**
>
> In the **Llama 2 License Agreement** section, ensure to accept the Llama2 EULA before you continue.
> The notebook deploys two models, `Llama 2` and `all-MiniLM-L6-v2 Models`, on `ml.g5.2xlarge` for inference.

The deployment of the models and the creation of the endpoints may take some time.

4. **Open your main notebook**

   In JupyterLab, open your terminal and run the following command.

   ```
   cd ..
   wget --no-check-certificate https://raw.githubusercontent.com/
   aws-samples/sagemaker-studio-foundation-models/main/lab-03-rag/
   Lab_3_RAG_on_SageMaker_Studio_using_EMR.ipynb
   ```

   You should see the additional `Lab_3_RAG_on_SageMaker_Studio_using_EMR.ipynb` notebook in the left panel of JupyterLab.

5. **Choose a PySpark kernel**

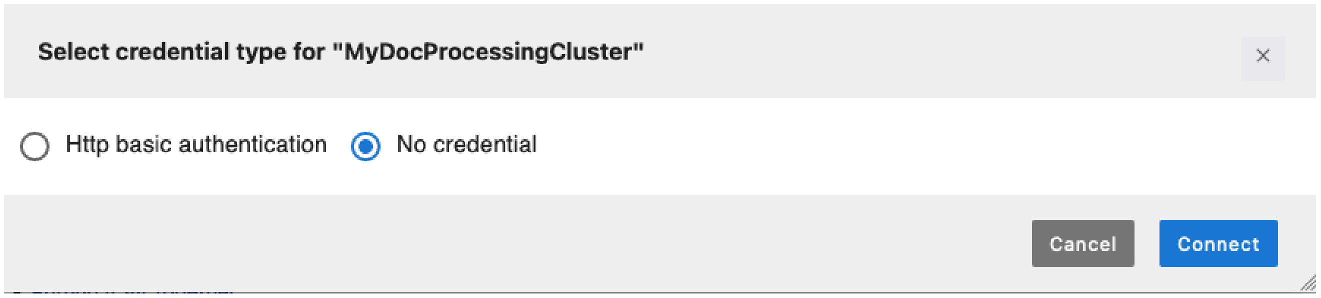
   Open your `Lab_3_RAG_on_SageMaker_Studio_using_EMR.ipynb` notebook and ensure that you are using the `SparkMagic PySpark` kernel. You can switch kernel at the top right of your notebook. Choose the current kernel name to open up a kernel selection modal, and then choose `SparkMagic PySpark`.

6. **Connect your notebook to the cluster**

    a. At the top right of your notebook, choose **Cluster**. This action opens a modal window that lists all of the running clusters that you have permission to access.

    b. Select your cluster then choose **Connect**. A new credential type selection modal window opens up.

    c. Choose **No credential** and then **Connect**.


Select credential type for "MyDocProcessingCluster"
×
Http basic authentication
No credential
Cancel
Connect

    d. A notebook cell automatically populates and runs. The notebook cell loads the `sagemaker_studio_analytics_extension.magics` extension, which provides functionality to connect to the Amazon EMR cluster. It then uses the `%sm_analytics` magic command to initiate the connection to your Amazon EMR cluster and the Spark application.

Note

Ensure that the connection string to your Amazon EMR cluster has an authentication type set to None. This is illustrated by the value `--auth-type None` in the following example. You can modify the field if necessary.

```
%load_ext sagemaker_studio_analytics_extension.magics
%sm_analytics emr connect --verify-certificate False --cluster-id your-cluster-id --auth-type None --language python
```

    e. Once you successfully establish the connection, your connection cell output message should display your `SparkSession` details including your cluster ID, YARN application ID, and a link to the Spark UI to monitor your Spark jobs.

You are ready to use the `Lab_3_RAG_on_SageMaker_Studio_using_EMR.ipynb` notebook. This example notebook runs distributed PySpark workloads for building a RAG system using LangChain and OpenSearch.

Step 4: Clean up your CloudFormation stack

After you are finished, make sure to terminate your two endpoints and delete your CloudFormation stack to prevent continued charges. Deleting the stack cleans up all the resources that were provisioned by the stack.

To delete your CloudFormation stack when you are done with it

1. Go to the CloudFormation console: https://console.aws.amazon.com/cloudformation
2. Select the stack you want to delete. You can search for it by name or find it in the list of stacks.
3. Click the **Delete** button to finalize deleting the stack and then **Delete** again to acknowledge that this will delete all resources created by the stack.

   Wait for the stack deletion to complete. This can take a few minutes. CloudFormation automatically cleans up all resources defined in the stack template.
4. Verify that all resources created by the stack have been deleted. For example, check for any leftover Amazon EMR cluster.

To remove the API endpoints for a model

1. Go to the SageMaker AI console: https://console.aws.amazon.com/sagemaker/.
2. In the left navigation pane, choose **Inference** and then **Endpoints**.
3. Select the endpoint `hf-allminil6v2-embedding-ep` and then choose **Delete** in the **Actions** drop down list. Repeat the step for the endpoint `meta-llama2-7b-chat-tg-ep`.

Admin guide

This section provides prerequisites, networking instructions for allowing the communication between Studio or Studio Classic and Amazon EMR clusters. It covers different deployment scenarios - when Studio and Amazon EMR are provisioned within private Amazon VPCs without public internet access, as well as when they need to communicate over the internet.

It walks through how administrators can use the AWS Service Catalog to make CloudFormation templates available to Studio, allowing data scientists to discover and self-provision Amazon EMR clusters directly from within Studio. This involves creating a Service Catalog portfolio, granting requisite permissions, referencing the Amazon EMR templates, and parameterizing them to enable customizations during cluster creation.

Last, it provides guidance on configuring discoverability of existing running Amazon EMR clusters from Studio, and Studio Classic, covering single account and cross-account access scenarios along with the necessary IAM permissions.

Topics

Configure Amazon EMR CloudFormation templates in the Service Catalog

This topic assumes administrators are familiar with CloudFormation, portfolios and products in AWS Service Catalog, as well as Amazon EMR.

To simplify the creation of Amazon EMR clusters from Studio, administrators can register an Amazon EMR CloudFormation template as a product in an AWS Service Catalog portfolio. To make the template available to data scientists, they must associate the portfolio with the SageMaker AI execution role used in Studio or Studio Classic. Finally, to allow users to discover templates, provision clusters, and connect to Amazon EMR clusters from Studio or Studio Classic, administrators need to set appropriate access permissions.

The Amazon EMR CloudFormation templates can allow end-users to customize various cluster aspects. For example, administrators can define an approved list of instance types that users can choose from when creating a cluster.

The following instructions use end-to-end CloudFormation stacks to setup a Studio or Studio Classic domain, a user profile, a Service Catalog portfolio, and populate an Amazon EMR launch template. The following steps highlight the specific settings that administrators must apply in their end-to-end stack to enable Studio or Studio Classic to access Service Catalog products and provision Amazon EMR clusters.

Note

The GitHub repository aws-samples/sagemaker-studio-emr contains example end-to-end CloudFormation stacks that deploy the necessary IAM roles, networking, SageMaker domain, user profile, Service Catalog portfolio, and add an Amazon EMR launch CloudFormation template. The templates provide different authentication options between

Studio or Studio Classic and the Amazon EMR cluster. In these example templates, the parent CloudFormation stack passes SageMaker AI VPC, security group, and subnet parameters to the Amazon EMR cluster template.
The sagemaker-studio-emr/cloudformation/emr_servicecatalog_templates repository contains various sample Amazon EMR CloudFormation launch templates, including options for single account and cross-account deployments.
Refer to Connect to an Amazon EMR cluster from SageMaker Studio or Studio Classic for details on the authentication methods you can use to connect to an Amazon EMR cluster.

To let data scientists discover Amazon EMR CloudFormation templates and provision clusters from Studio or Studio Classic, follow these steps.

Step 0: Check your networking and prepare your CloudFormation stack

Before you start:

- Ensure that you have reviewed the networking and security requirements in Configure network access for your Amazon EMR cluster.
- You must have an existing end-to-end CloudFormation stack that supports the authentication method of your choice. You can find examples of such CloudFormation templates in the aws-samples/sagemaker-studio-emr GitHub repository. The following steps highlight the specific configurations in your end-to-end stack to enable the use of Amazon EMR templates within Studio or Studio Classic.

Step 1: Associate your Service Catalog portfolio with SageMaker AI

In your Service Catalog portfolio, associate your portfolio ID with the SageMaker AI execution role accessing your cluster.

To do so, add the following section (here in YAML format) to your stack. This grants the SageMaker AI execution role access to the specified Service Catalog portfolio containing products like Amazon EMR templates. It allows roles assumed by SageMaker AI to launch those products.

Replace *SageMakerExecutionRole.Arn* and *SageMakerStudioEMRProductPortfolio.ID* with their actual values.

```
SageMakerStudioEMRProductPortfolioPrincipalAssociation:
    Type: AWS::ServiceCatalog::PortfolioPrincipalAssociation
    Properties:
      PrincipalARN: SageMakerExecutionRole.Arn
      PortfolioId: SageMakerStudioEMRProductPortfolio.ID
      PrincipalType: IAM
```

For details on the required set of IAM permissions, see the permissions section.

Step 2: Reference an Amazon EMR template in a Service Catalog product

In a Service Catalog product of your portfolio, reference an Amazon EMR template resource and ensure its visibility in Studio or Studio Classic.

To do so, reference the Amazon EMR template resource in the Service Catalog product definition, and then add the following tag key `"sagemaker:studio-visibility:emr"` set to the value `"true"` (see the example in YAML format).

In the Service Catalog product definition, the CloudFormation template of the cluster is referenced via URL. The additional tag set to true ensures the visibility of the Amazon EMR templates in Studio or Studio Classic.

Note

The Amazon EMR template referenced by the provided URL in the example does not enforce any authentication requirements when launched. This option is meant for demonstration and learning purposes. It is not recommended in a production environment.

```
SMStudioEMRNoAuthProduct:
    Type: AWS::ServiceCatalog::CloudFormationProduct
    Properties:
      Owner: AWS
      Name: SageMaker Studio Domain No Auth EMR
      ProvisioningArtifactParameters:
        - Name: SageMaker Studio Domain No Auth EMR
          Description: Provisions a SageMaker domain and No Auth EMR Cluster
          Info:
            LoadTemplateFromURL: Link to your CloudFormation template. For example,
 https://aws-blogs-artifacts-public.s3.amazonaws.com/artifacts/astra-m4-sagemaker/end-
to-end/CFN-EMR-NoStudioNoAuthTemplate-v3.yaml
      Tags:
        - Key: "sagemaker:studio-visibility:emr"
```

```
          Value: "true"
```

Step 3: Parameterize the Amazon EMR CloudFormation template

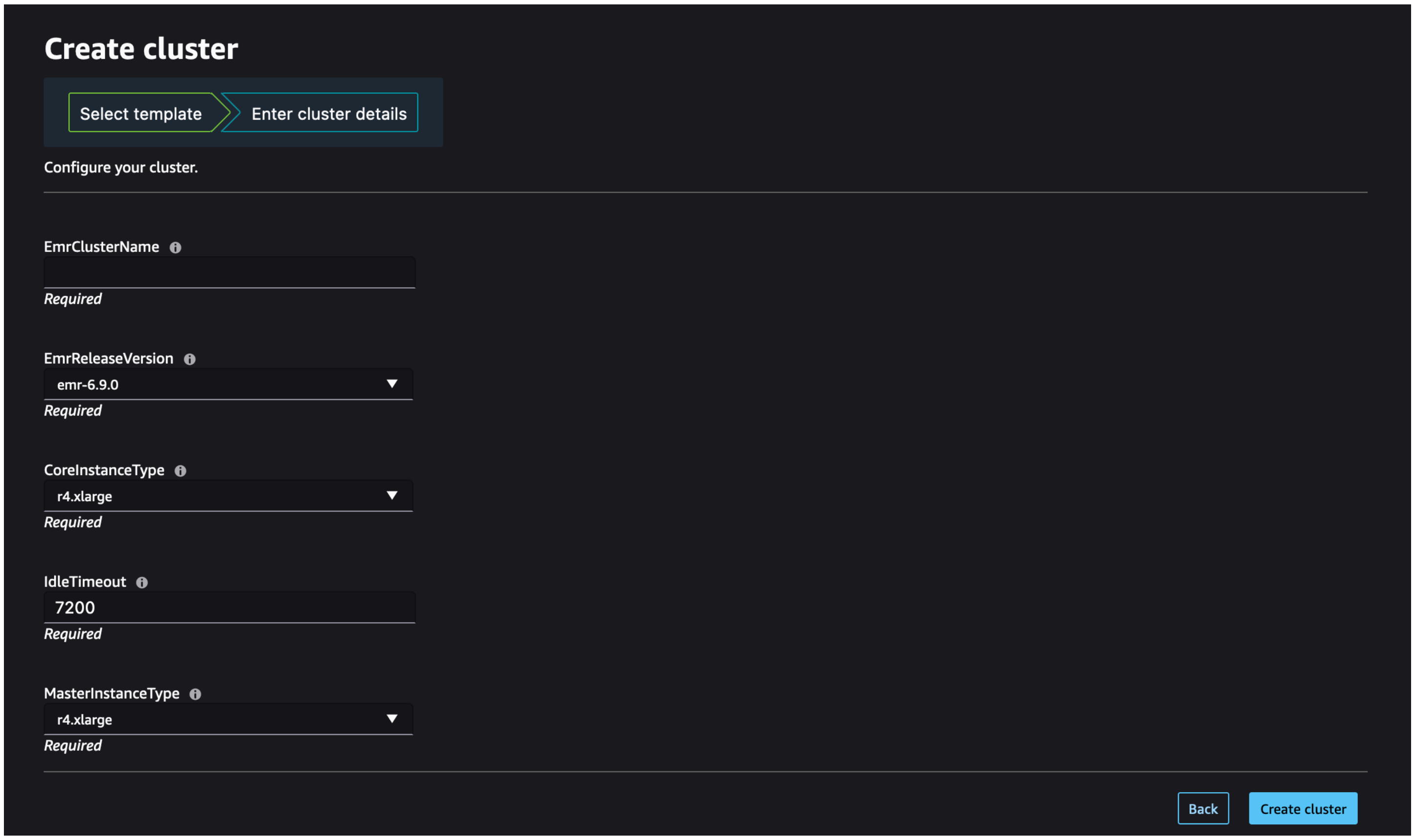
The CloudFormation template used to define the Amazon EMR cluster within the Service Catalog product allows administrators to specify configurable parameters. Administrators can define `Default` values and `AllowedValues` ranges for these parameters within the template's `Parameters` section. During the cluster launch process, data scientists can provide custom inputs or make selections from those predefined options to customize certain aspects of their Amazon EMR cluster.

The following example illustrates additional input parameters that administrators can set when creating an Amazon EMR template.

```
"Parameters": {
    "EmrClusterName": {
      "Type": "String",
      "Description": "EMR cluster Name."
    },
    "MasterInstanceType": {
      "Type": "String",
      "Description": "Instance type of the EMR master node.",
      "Default": "m5.xlarge",
      "AllowedValues": [
        "m5.xlarge",
        "m5.2xlarge",
        "m5.4xlarge"
      ]
    },
    "CoreInstanceType": {
      "Type": "String",
      "Description": "Instance type of the EMR core nodes.",
      "Default": "m5.xlarge",
      "AllowedValues": [
        "m5.xlarge",
        "m5.2xlarge",
        "m5.4xlarge",
        "m3.medium",
        "m3.large",
        "m3.xlarge",
        "m3.2xlarge"
      ]
    },
    "CoreInstanceCount": {
      "Type": "String",
      "Description": "Number of core instances in the EMR cluster.",
      "Default": "2",
      "AllowedValues": [
        "2",
        "5",
        "10"
      ]
    },
    "EmrReleaseVersion": {
      "Type": "String",
      "Description": "The release version of EMR to launch.",
      "Default": "emr-5.33.1",
      "AllowedValues": [
        "emr-5.33.1",
        "emr-6.4.0"
      ]
    }
  }
```

After administrators have made the Amazon EMR CloudFormation templates available within Studio, data scientists can use them to self-provision Amazon EMR clusters. The `Parameters` section defined in the template translates into input fields on the cluster creation form within Studio or Studio Classic. For each parameter, data scientists can either enter a custom value into the input box or select from the predefined options listed in a dropdown menu, which corresponds to the `AllowedValues` specified in the template.

The following illustration shows the dynamic form assembled from a CloudFormation Amazon EMR template to create an Amazon EMR cluster in Studio or Studio Classic.


Create cluster

Select template | Enter cluster details

Configure your cluster.

EmrClusterName
Required

EmrReleaseVersion
emr-6.9.0
Required

CoreInstanceType
r4.xlarge
Required

IdleTimeout
7200
Required

MasterInstanceType
r4.xlarge
Required

Visit Launch an Amazon EMR cluster from Studio or Studio Classic to learn about how to launch a cluster from Studio or Studio Classic using those Amazon EMR templates.

Step 4: Set up the permissions to enable listing and launching Amazon EMR clusters from Studio

Last, attach the required IAM permissions to enable listing existing running Amazon EMR clusters and self-provisioning new clusters from Studio or Studio Classic.

The role(s) to which you must add those permissions depends on whether Studio or Studio Classic and Amazon EMR are deployed in the same account (choose *Single Account*) or in different accounts (choose *Cross account*).

Important

You can only discover and connect to Amazon EMR clusters for JupyterLab and Studio Classic applications that are launched from private spaces. Ensure that the Amazon EMR clusters are located in the same AWS region as your Studio environment.

Single account

If your Amazon EMR clusters and Studio or Studio Classic are deployed in the same AWS account, attach the following permissions to the SageMaker AI execution role accessing your cluster.

1. **Step 1**: Retrieve the ARN of the SageMaker AI execution role used by your private space.

   For information on spaces and execution roles in SageMaker AI, see Understanding domain space permissions and execution roles.

   For more information about how to retrieve the ARN of SageMaker AI's execution role, see Get your execution role.
2. **Step 2**: Attach the following permissions to the SageMaker AI execution role accessing your Amazon EMR clusters.

   a. Navigate to the IAM console.

   b. Choose **Roles** and then search for your execution role by name in the **Search** field. The role name is the last part of the ARN, after the last forward slash (/).

   c. Follow the link to your role.

   d. Choose **Add permissions** and then **Create inline policy**.

   e. In the **JSON** tab, add the Amazon EMR permissions allowing Amazon EMR access and operations. For details on the policy document, see *List Amazon EMR policies* in Reference policies. Replace the `region`, and `accountID` with their actual values before copying the list of statements to the inline policy of your role.

   f. Choose **Next** and then provide a **Policy name**.

   g. Choose **Create policy**.

   h. Repeat the **Create inline policy** step to add another policy granting the execution role the permissions to provision new Amazon EMR clusters using CloudFormation templates. For details on the policy document, see *Create Amazon EMRclusters policies* in Reference policies. Replace the `region` and `accountID` with their actual values before copying the list of statements to the inline policy of your role.

Note

Users of role-based access control (RBAC) connectivity to Amazon EMR clusters should also refer to the section called “Configure runtime role authentication when your Amazon EMR cluster and Studio are in the same account”.

Cross account

Before you get started, retrieve the ARN of the SageMaker AI execution role used by your private space.

For information on spaces and execution roles in SageMaker AI, see Understanding domain space permissions and execution roles.

For more information about how to retrieve the ARN of SageMaker AI's execution role, see Get your execution role.

If your Amazon EMR clusters and Studio or Studio Classic are deployed in separate AWS accounts, you configure the permissions on both accounts.

Note

Users of role-based access control (RBAC) connectivity to Amazon EMR clusters should also refer to the section called “Configure runtime role authentication when your cluster and Studio are in different accounts”.

On the Amazon EMR cluster account

Follow these steps to create the necessary roles and policies on the account where Amazon EMR is deployed, also referred to as the *trusting account*:

1. **Step 1**: Retrieve the ARN of the service role of your Amazon EMR cluster.

   To learn about how to find the ARN of the service role of a cluster, see Configure IAM service roles for Amazon EMR permissions to AWS services and resources.
2. **Step 2**: Create a custom IAM role named `AssumableRole` with the following configuration:

- Permissions: Grant the necessary permissions to `AssumableRole` to allow accessing Amazon EMR resources. This role is also known as an *Access role* in scenarios involving cross-account access.
- Trust relationship: Configure the trust policy for `AssumableRole` to allow assuming the execution role (The `SageMakerExecutionRole` in the cross-account diagram) from the Studio account that requires access.

By assuming the role, Studio or Studio Classic can gain temporary access to the permissions it needs in Amazon EMR.

For detailed instructions on how to create a new `AssumableRole` in your Amazon EMR AWS account, follow these steps:

a. Navigate to the IAM console.

b. In the left navigation pane, choose **Policy**, and then **Create policy**.

c. In the **JSON** tab, add the Amazon EMR permissions allowing Amazon EMR access and operations. For details on the policy document, see *List Amazon EMR policies* in Reference policies. Replace the `region`, and `accountID` with their actual values before copying the list of statements to the inline policy of your role.

d. Choose **Next** and then provide a **Policy name**.

e. Choose **Create policy**.

f. In the left navigation pane, choose **Roles** and then **Create role**.

g. On the **Create role** page, choose **Custom trust policy** as the trusted entity.

h. Paste in the following JSON document in the **Custom trust policy** section and then choose **Next**.

For users of Studio and JupyterLab

Replace `studio-account` with the Studio account ID, and `AmazonSageMaker-ExecutionRole` with the execution role used by your JupyterLab space.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/service-
role/AmazonSageMaker-ExecutionRole"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

For users of Studio Classic

Replace `studio-account` with the Studio Classic account ID.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

i. In the **Add permissions** page, add the permission you just created and then choose **Next**.

j. On the **Review** page, enter a name for the role such as `AssumableRole` and an optional description.

k. Review the role details and choose **Create role**.

For more information about creating a role on an AWS account, see Creating an IAM role (console).

On the Studio account

On the account where Studio is deployed, also referred to as the *trusted account*, update the SageMaker AI execution role accessing your clusters with the required permissions to access resources in the trusting account.

1. **Step 1**: Retrieve the ARN of the SageMaker AI execution role used by your private space.

   For information on spaces and execution roles in SageMaker AI, see Understanding domain space permissions and execution roles.

   For more information about how to retrieve the ARN of SageMaker AI's execution role, see Get your execution role.
2. **Step 2**: Attach the following permissions to the SageMaker AI execution role accessing your Amazon EMR clusters.

   a. Navigate to the IAM console.

   b. Choose **Roles** and then search for your execution role by name in the **Search** field. The role name is the last part of the ARN, after the last forward slash (/).

   c. Follow the link to your role.

   d. Choose **Add permissions** and then **Create inline policy**.

   e. In the **JSON** tab, add the inline policy granting the role permissions to update the domains, user profiles, and spaces. For details on the policy document, see *Domain, user profile, and space update actions policy* in Reference policies. Replace the `region` and `accountID` with their actual values before copying the list of statements to the inline policy of your role.

   f. Choose **Next** and then provide a **Policy name**.

   g. Choose **Create policy**.

   h. Repeat the **Create inline policy** step to add another policy granting the execution role the permissions to assume the `AssumableRole` and then perform actions permitted by the role's access policy. Replace `emr-account` with the Amazon EMR account ID, and `AssumableRole` with the name of the assumable role created in the Amazon EMR account.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRoleAssumptionForCrossAccountDiscovery",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": [
                "arn:aws:iam::111122223333:role/AssumableRole"
            ]
        }
    ]
}
```

i. Repeat the **Create inline policy** step to add another policy granting the execution role the permissions to provision new Amazon EMR clusters using CloudFormation templates. For details on the policy document, see *Create Amazon EMRclusters policies* in Reference policies. Replace the `region` and `accountID` with their actual values before copying the list of statements to the inline policy of your role.

j. (Optional) To allow listing Amazon EMR clusters deployed in the same account as Studio, add an additional inline policy to your Studio execution role as defined in *List Amazon EMR policies* in Reference policies.

3. **Step 3**: Associate your assumable role(s) (access role) with your domain or user profile. JupyterLab users in Studio can use the SageMaker AI console or the provided script.

   Choose the tab that corresponds to your use case.

   Associate your assumable roles in JupyterLab using the SageMaker AI console

   To associate your assumable roles with your user profile or domain using the SageMaker AI console:

   1. Navigate to the SageMaker AI console at https://console.aws.amazon.com/sagemaker/.
   2. In the left navigation pane, choose **domain**, and then select the domain using the SageMaker AI execution role whose permissions you updated.

3. - To add your assumable role(s) (access role) to your domain: In the **App Configurations** tab of the **Domain details** page, navigate to the **JupyterLab** section.
   - To add your assumable role(s) (access role) to your user profile: On the **Domain details** page, chose the **User profiles** tab, select the user profile using the SageMaker AI execution role whose permissions you updated. In the **App Configurations** tab, navigate to the **JupyterLab** section.
4. Choose **Edit** and add the ARNs of your assumable role (access role).
5. Choose **Submit**.

Associate your assumable roles in JupyterLab using a Python script

In a JupyterLab application started from a space using the SageMaker AI execution role whose permissions you updated, run the following command in a terminal. Replace the `domainID`, `user-profile-name`, `emr-accountID`, and `AssumableRole` (`EMRServiceRole` for RBAC runtime roles) with their proper values. This code snippet updates the user profile settings for a specific user profile (use `client.update_userprofile`) or domain settings (use `client.update_domain`) within a SageMaker AI domain. Specifically, it allows the JupyterLab application to assume a particular IAM role (`AssumableRole`) for running Amazon EMR clusters within the Amazon EMR account.

```
import botocore.session
import json
sess = botocore.session.get_session()
client = sess.create_client('sagemaker')

client.update_userprofile(
DomainId="domainID",
UserProfileName="user-profile-name",
DefaultUserSettings={
    'JupyterLabAppSettings': {
        'EmrSettings': {
            'AssumableRoleArns': ["arn:aws:iam::emr-
accountID:role/AssumableRole"],
            'ExecutionRoleArns': ["arn:aws:iam::emr-
accountID:role/EMRServiceRole",
                                 "arn:aws:iam::emr-
accountID:role/AnotherServiceRole"]
        }
    }
})
resp = client.describe_user_profile(DomainId="domainID", UserProfileName=user-
profile-name")

resp['CreationTime'] = str(resp['CreationTime'])
resp['LastModifiedTime'] = str(resp['LastModifiedTime'])
print(json.dumps(resp, indent=2))
```

For users of Studio Classic

Provide the ARN of the `AssumableRole` to your Studio Classic execution role. The ARN is loaded by the Jupyter server at launch. The execution role used by Studio assumes that cross-account role to discover and connect to Amazon EMR clusters in the *trusting account*.

You can specify this information by using Lifecycle Configuration (LCC) scripts. You can attach the LCC to your domain or a specific user profile. The LCC script that you use must be a JupyterServer configuration. For more information on how to create an LCC script, see Use Lifecycle Configurations with Studio Classic.

The following is an example LCC script. To modify the script, replace `AssumableRole` and `emr-account` with their respective values. The number of cross-accounts is limited to five.

```
# This script creates the file that informs Studio Classic that the role
 "arn:aws:iam::emr-account:role/AssumableRole" in remote account "emr-account"
 must be assumed to list and describe Amazon EMR clusters in the remote account.

#!/bin/bash

set -eux

FILE_DIRECTORY="/home/sagemaker-user/.cross-account-configuration-DO_NOT_DELETE"
FILE_NAME="emr-discovery-iam-role-arns-DO_NOT_DELETE.json"
FILE="$FILE_DIRECTORY/$FILE_NAME"

mkdir -p $FILE_DIRECTORY

cat > "$FILE" <<- "EOF"
{
  emr-cross-account1: "arn:aws:iam::emr-cross-account1:role/AssumableRole",
  emr-cross-account2: "arn:aws:iam::emr-cross-account2:role/AssumableRole"
}
EOF
```

After the LCC runs and the files are written, the server reads the file `/home/sagemaker-user/.cross-account-configuration-DO_NOT_DELETE/emr-discovery-iam-role-arns-DO_NOT_DELETE.json` and stores the cross-account ARN.

Configure listing Amazon EMR clusters

Administrators can configure permissions for the SageMaker Studio execution role to grant users the ability to view the list of Amazon EMR clusters they have access to, allowing them to connect to these clusters. The clusters to which you want access can be deployed in the same AWS account as Studio (choose *Single account*) or in separate accounts (choose *Cross account*). The following page describes how to grant the permissions for viewing Amazon EMR clusters from Studio or Studio Classic.

> **Important**
>
> You can only discover and connect to Amazon EMR clusters for JupyterLab and Studio Classic applications that are launched from private spaces. Ensure that the Amazon EMR clusters are located in the same AWS region as your Studio environment.

To let data scientists discover and then connect to Amazon EMRclusters from Studio or Studio Classic, follow these steps.

Single account

If your Amazon EMR clusters and Studio or Studio Classic are deployed in the same AWS account, attach the following permissions to the SageMaker AI execution role accessing your cluster.

1. **Step 1**: Retrieve the ARN of the SageMaker AI execution role used by your private space.

   For information on spaces and execution roles in SageMaker AI, see Understanding domain space permissions and execution roles.

   For more information about how to retrieve the ARN of SageMaker AI's execution role, see Get your execution role.

2. **Step 2**: Attach the following permissions to the SageMaker AI execution role accessing your Amazon EMR clusters.

    a. Navigate to the IAM console.

    b. Choose **Roles** and then search for your execution role by name in the **Search** field. The role name is the last part of the ARN, after the last forward slash (/).

    c. Follow the link to your role.

    d. Choose **Add permissions** and then **Create inline policy**.

    e. In the **JSON** tab, add the Amazon EMR permissions allowing Amazon EMR access and operations. For details on the policy document, see *List Amazon EMR policies* in Reference policies. Replace the `region`, and `accountID` with their actual values before copying the list of statements to the inline policy of your role.

    f. Choose **Next** and then provide a **Policy name**.

    g. Choose **Create policy**.

Note

Users of role-based access control (RBAC) connectivity to Amazon EMR clusters should also refer to the section called “Configure runtime role authentication when your Amazon EMR cluster and Studio are in the same account”.

Cross account

Before you get started, retrieve the ARN of the SageMaker AI execution role used by your private space.

For information on spaces and execution roles in SageMaker AI, see Understanding domain space permissions and execution roles.

For more information about how to retrieve the ARN of SageMaker AI's execution role, see Get your execution role.

If your Amazon EMR clusters and Studio or Studio Classic are deployed in separate AWS accounts, you configure the permissions on both accounts.

Note

Users of role-based access control (RBAC) connectivity to Amazon EMR clusters should also refer to the section called “Configure runtime role authentication when your cluster and Studio are in different accounts”.

On the Amazon EMR cluster account

Follow these steps to create the necessary roles and policies on the account where Amazon EMR is deployed, also referred to as the *trusting account*:

1. **Step 1**: Retrieve the ARN of the service role of your Amazon EMR cluster.

   To learn about how to find the ARN of the service role of a cluster, see Configure IAM service roles for Amazon EMR permissions to AWS services and resources.
2. **Step 2**: Create a custom IAM role named `AssumableRole` with the following configuration:
   - Permissions: Grant the necessary permissions to `AssumableRole` to allow accessing Amazon EMR resources. This role is also known as an *Access role* in scenarios involving cross-account access.
   - Trust relationship: Configure the trust policy for `AssumableRole` to allow assuming the execution role (The `SageMakerExecutionRole` in the cross-account diagram) from the Studio account that requires access.

   By assuming the role, Studio or Studio Classic can gain temporary access to the permissions it needs in Amazon EMR.

   For detailed instructions on how to create a new `AssumableRole` in your Amazon EMR AWS account, follow these steps:

   a. Navigate to the IAM console.

   b. In the left navigation pane, choose **Policy**, and then **Create policy**.

   c. In the **JSON** tab, add the Amazon EMR permissions allowing Amazon EMR access and operations. For details on the policy document, see *List Amazon EMR policies* in Reference policies. Replace the `region`, and `accountID` with their actual values before copying the list of statements to the inline policy of your role.

d. Choose **Next** and then provide a **Policy name**.

e. Choose **Create policy**.

f. In the left navigation pane, choose **Roles** and then **Create role**.

g. On the **Create role** page, choose **Custom trust policy** as the trusted entity.

h. Paste in the following JSON document in the **Custom trust policy** section and then choose **Next**.

For users of Studio and JupyterLab

Replace `studio-account` with the Studio account ID, and `AmazonSageMaker-ExecutionRole` with the execution role used by your JupyterLab space.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/service-role/AmazonSageMaker-ExecutionRole"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

For users of Studio Classic

Replace `studio-account` with the Studio Classic account ID.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

i. In the **Add permissions** page, add the permission you just created and then choose **Next**.

j. On the **Review** page, enter a name for the role such as `AssumableRole` and an optional description.

k. Review the role details and choose **Create role**.

For more information about creating a role on an AWS account, see Creating an IAM role (console).

On the Studio account

On the account where Studio is deployed, also referred to as the *trusted account*, update the SageMaker AI execution role accessing your clusters with the required permissions to access resources in the trusting account.

1. **Step 1**: Retrieve the ARN of the SageMaker AI execution role used by your private space.

   For information on spaces and execution roles in SageMaker AI, see Understanding domain space permissions and execution roles.

   For more information about how to retrieve the ARN of SageMaker AI's execution role, see Get your execution role.

2. **Step 2**: Attach the following permissions to the SageMaker AI execution role accessing your Amazon EMR clusters.

   a. Navigate to the IAM console.

   b. Choose **Roles** and then search for your execution role by name in the **Search** field. The role name is the last part of the ARN, after the last forward slash (/).

   c. Follow the link to your role.

   d. Choose **Add permissions** and then **Create inline policy**.

e. In the **JSON** tab, add the inline policy granting the role permissions to update the domains, user profiles, and spaces. For details on the policy document, see *Domain, user profile, and space update actions policy* in Reference policies. Replace the `region` and `accountID` with their actual values before copying the list of statements to the inline policy of your role.

f. Choose **Next** and then provide a **Policy name**.

g. Choose **Create policy**.

h. Repeat the **Create inline policy** step to add another policy granting the execution role the permissions to assume the `AssumableRole` and then perform actions permitted by the role's access policy. Replace `emr-account` with the Amazon EMR account ID, and `AssumableRole` with the name of the assumable role created in the Amazon EMR account.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRoleAssumptionForCrossAccountDiscovery",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": [
                "arn:aws:iam::111122223333:role/AssumableRole"
            ]
        }
    ]
}
```

i. (Optional) To allow listing Amazon EMR clusters deployed in the same account as Studio, add an additional inline policy to your Studio execution role as defined in *List Amazon EMR policies* in Reference policies.

3. **Step 3**: Associate your assumable role(s) (access role) with your domain or user profile. JupyterLab users in Studio can use the SageMaker AI console or the provided script.

   Choose the tab that corresponds to your use case.

Associate your assumable roles in JupyterLab using the SageMaker AI console

To associate your assumable roles with your user profile or domain using the SageMaker AI console:

1. Navigate to the SageMaker AI console at https://console.aws.amazon.com/sagemaker/.
2. In the left navigation pane, choose **domain**, and then select the domain using the SageMaker AI execution role whose permissions you updated.
3. - To add your assumable role(s) (access role) to your domain: In the **App Configurations** tab of the **Domain details** page, navigate to the **JupyterLab** section.
   - To add your assumable role(s) (access role) to your user profile: On the **Domain details** page, chose the **User profiles** tab, select the user profile using the SageMaker AI execution role whose permissions you updated. In the **App Configurations** tab, navigate to the **JupyterLab** section.
4. Choose **Edit** and add the ARNs of your assumable role (access role).
5. Choose **Submit**.

Associate your assumable roles in JupyterLab using a Python script

In a JupyterLab application started from a space using the SageMaker AI execution role whose permissions you updated, run the following command in a terminal. Replace the `domainID`, `user-profile-name`, `emr-accountID`, and `AssumableRole` (`EMRServiceRole` for RBAC runtime roles) with their proper values. This code snippet updates the user profile settings for a specific user profile (use `client.update_userprofile`) or domain settings (use `client.update_domain`) within a SageMaker AI domain. Specifically, it allows the JupyterLab application to assume a particular IAM role (`AssumableRole`) for running Amazon EMR clusters within the Amazon EMR account.

```
import botocore.session
import json
sess = botocore.session.get_session()
client = sess.create_client('sagemaker')

client.update_userprofile(
DomainId="domainID",
UserProfileName="user-profile-name",
DefaultUserSettings={
    'JupyterLabAppSettings': {
        'EmrSettings': {
            'AssumableRoleArns': ["arn:aws:iam::emr-
accountID:role/AssumableRole"],
            'ExecutionRoleArns': ["arn:aws:iam::emr-
accountID:role/EMRServiceRole",
                                 "arn:aws:iam::emr-
accountID:role/AnotherServiceRole"]
        }

    }
})
resp = client.describe_user_profile(DomainId="domainID", UserProfileName=user-
profile-name")

resp['CreationTime'] = str(resp['CreationTime'])
resp['LastModifiedTime'] = str(resp['LastModifiedTime'])
print(json.dumps(resp, indent=2))
```

For users of Studio Classic

Provide the ARN of the `AssumableRole` to your Studio Classic execution role. The ARN is loaded by the Jupyter server at launch. The execution role used by Studio assumes that cross-account role to discover and connect to Amazon EMR clusters in the *trusting account*.

You can specify this information by using Lifecycle Configuration (LCC) scripts. You can attach the LCC to your domain or a specific user profile. The LCC script that you use must be a JupyterServer configuration. For more information on how to create an LCC script, see Use Lifecycle Configurations with Studio Classic.

The following is an example LCC script. To modify the script, replace `AssumableRole` and `emr-account` with their respective values. The number of cross-accounts is limited to five.

```
# This script creates the file that informs Studio Classic that the role
 "arn:aws:iam::emr-account:role/AssumableRole" in remote account "emr-account"
 must be assumed to list and describe Amazon EMR clusters in the remote account.

#!/bin/bash

set -eux
```

```
FILE_DIRECTORY="/home/sagemaker-user/.cross-account-configuration-DO_NOT_DELETE"
FILE_NAME="emr-discovery-iam-role-arns-DO_NOT_DELETE.json"
FILE="$FILE_DIRECTORY/$FILE_NAME"

mkdir -p $FILE_DIRECTORY

cat > "$FILE" <<- "EOF"
{
  emr-cross-account1: "arn:aws:iam::emr-cross-account1:role/AssumableRole",
  emr-cross-account2: "arn:aws:iam::emr-cross-account2:role/AssumableRole"
}
EOF
```

After the LCC runs and the files are written, the server reads the file `/home/sagemaker-user/.cross-account-configuration-DO_NOT_DELETE/emr-discovery-iam-role-arns-DO_NOT_DELETE.json` and stores the cross-account ARN.

Refer to List Amazon EMR clusters from Studio or Studio Classic to learn about how to discover and connect to Amazon EMR clusters from Studio or Studio Classic notebooks.

Configure IAM runtime roles for Amazon EMR cluster access in Studio

When you connect to an Amazon EMR cluster from your Studio or Studio Classic notebooks, you can visually browse a list of IAM roles, known as runtime roles, and select one on the fly. Subsequently, all your Apache Spark, Apache Hive, or Presto jobs created from your notebook access only the data and resources permitted by policies attached to the runtime role. Also, when data is accessed from data lakes managed with AWS Lake Formation, you can enforce table-level and column-level access using policies attached to the runtime role.

With this capability, you and your teammates can connect to the same cluster, each using a runtime role scoped with permissions matching your individual level of access to data. Your sessions are also isolated from one another on the shared cluster.

To try out this feature using Studio Classic, see Apply fine-grained data access controls with AWS Lake Formation and Amazon EMR from Amazon SageMaker Studio Classic. This blog post helps you set up a demo environment where you can try using preconfigured runtime roles to connect to Amazon EMR clusters.

Prerequisites

Before you get started, make sure you meet the following prerequisites:

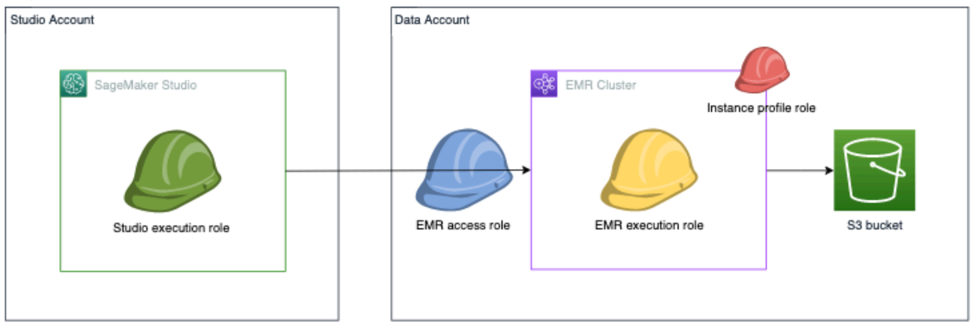
- Use Amazon EMR version 6.9 or above.
- **For Studio Classic users**: Use JupyterLab version 3 in the Studio Classic Jupyter server application configuration. This version supports Studio Classic connection to Amazon EMR clusters using runtime roles.

  For Studio users: Use a SageMaker distribution image version `1.10` or above.
- Allow the use of runtime roles in your cluster's security configuration. For more information, see Runtime roles for Amazon EMR steps.
- Create a notebook with any of the kernels listed in Supported images and kernels to connect to an Amazon EMR cluster from Studio or Studio Classic.
- Make sure you review the instructions in Set up Studio to use runtime IAM roles to configure your runtime roles.

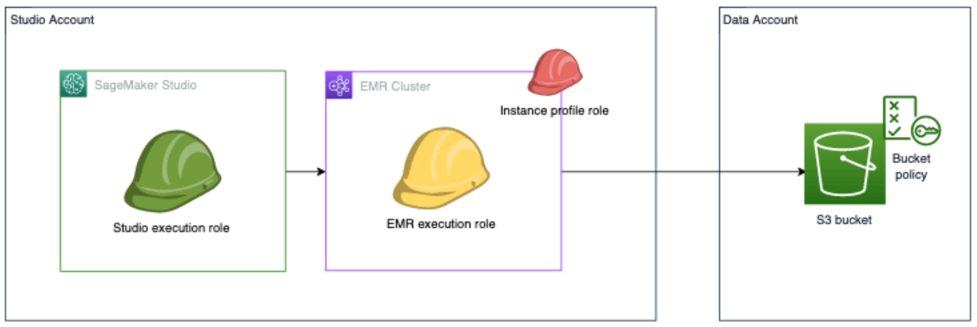
Cross-account connection scenarios

Runtime role authentication supports a variety of cross-account connection scenarios when your data resides outside of your Studio account. The following image shows three different ways you can assign your Amazon EMR cluster, data, and even Amazon EMR runtime execution role between your Studio and data accounts:

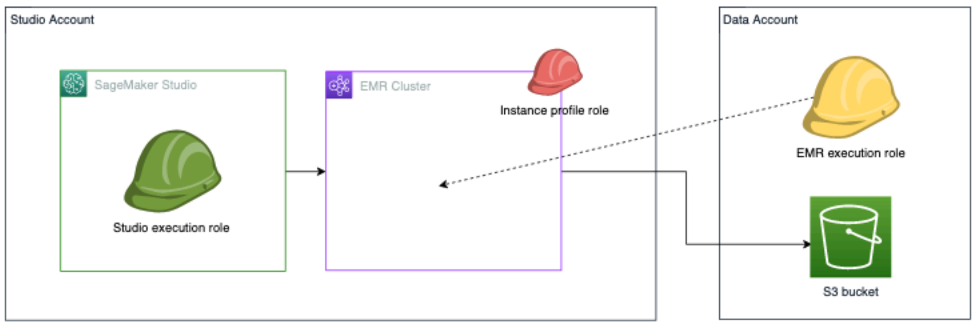
Option 1



Option 2



Option 3



In option 1, your Amazon EMR cluster and Amazon EMR runtime execution role are in a separate data account from the Studio account. You define a separate Amazon EMR access role (also referred to as `Assumable role`) permission policy which grants permission to Studio or Studio Classic execution role to assume the Amazon EMR access role. The Amazon EMR access role then calls the Amazon EMR API `GetClusterSessionCredentials` on behalf of your Studio or Studio Classic execution role, giving you access to the cluster.

In option 2, your Amazon EMR cluster and Amazon EMR runtime execution role are in your Studio account. Your Studio execution role has permission to use the Amazon EMR API `GetClusterSessionCredentials` to gain access to your cluster. To access the Amazon S3 bucket, give the Amazon EMR runtime execution role cross-account Amazon S3 bucket access permissions — you grant these permissions within your Amazon S3 bucket policy.

In option 3, your Amazon EMR clusters are in your Studio account, and the Amazon EMR runtime execution role is in the data account. Your Studio or Studio Classic execution role has permission to use the Amazon EMR API `GetClusterSessionCredentials` to gain access to your cluster. Add the Amazon EMR runtime execution role into the execution role configuration JSON. Then you can select the role in the UI when you choose your cluster. For details about how to set up your execution role configuration JSON file, see Preload your execution roles into Studio or Studio Classic.

Set up Studio to use runtime IAM roles

To establish runtime role authentication for your Amazon EMR clusters, configure the required IAM policies, network, and usability enhancements. Your setup depends on whether you handle any cross-account arrangements if your Amazon EMR clusters, Amazon EMR runtime execution role, or both, reside outside of your Studio account. The following section guides you through the policies to install, how to configure the network to allow traffic between cross-accounts, and the local configuration file to set up to automate your Amazon EMR connection.

Configure runtime role authentication when your Amazon EMR cluster and Studio are in the same account

If your Amazon EMR cluster resides in your Studio account, complete the following steps to add necessary permissions to your Studio execution policy:

1. Add the required IAM policy to connect to Amazon EMR clusters. For details, see Configure listing Amazon EMR clusters.
2. Grant permission to call the Amazon EMR API `GetClusterSessionCredentials` when you pass one or more permitted Amazon EMR runtime execution roles specified in the policy.
3. (Optional) Grant permission to pass IAM roles that follow any user-defined naming conventions.
4. (Optional) Grant permission to access Amazon EMR clusters that are tagged with specific user-defined strings.
5. Preload your IAM roles so you can select the role to use when you connect to your Amazon EMR cluster. For details about how to preload your IAM roles, see Preload your execution roles into Studio or Studio Classic.

The following example policy permits Amazon EMR runtime execution roles belonging to the modeling and training groups to call `GetClusterSessionCredentials`. In addition, the policyholder can access Amazon EMR clusters tagged with the strings `modeling` or `training`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "elasticmapreduce:GetClusterSessionCredentials",
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "elasticmapreduce:ExecutionRoleArn": [
                        "arn:aws:iam::111122223333:role/emr-execution-role-ml-
modeling*",
                        "arn:aws:iam::111122223333:role/emr-execution-role-ml-
training*"
   ]},
  "StringLike":{
                    "elasticmapreduce:ResourceTag/group": [
                        "*modeling*",
                        "*training*"
                    ]
                }
            }
        }
    ]
}
```

Configure runtime role authentication when your cluster and Studio are in different accounts

If your Amazon EMR cluster is not in your Studio account, allow your SageMaker AI execution role to assume the cross-account Amazon EMR access role so you can connect to the cluster. Complete the following steps to set up your cross-account configuration:

1. Create your SageMaker AI execution role permission policy so that the execution role can assume the Amazon EMR access role. The following policy is an example:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAssumeCrossAccountEMRAccessRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::111122223333:role/emr-access-role-name"
        }
    ]
}
```

2. Create the trust policy to specify which Studio account IDs are trusted to assume the Amazon EMR access role. The following policy is an example:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
      {
        "Sid": "AllowCrossAccountSageMakerExecutionRoleToAssumeThisRole",
        "Effect": "Allow",
        "Principal": {
          "AWS": "arn:aws:iam::111122223333:role/studio_execution_role"
        },
        "Action": "sts:AssumeRole"
      }
    ]
}
```

3. Create the Amazon EMR access role permission policy, which grants the Amazon EMR runtime execution role the needed permissions to carry out the intended tasks on the cluster. Configure the Amazon EMR access role to call the API `GetClusterSessionCredentials` with the Amazon EMR runtime execution roles specified in the access role permission policy. The following policy is an example:

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCallingEmrGetClusterSessionCredentialsAPI",
            "Effect": "Allow",
            "Action": "elasticmapreduce:GetClusterSessionCredentials",
            "Resource": "arn:aws:elasticmapreduce:us-
east-1:111122223333:cluster/cluster-id",
            "Condition": {
                "StringLike": {
                    "elasticmapreduce:ExecutionRoleArn": [
                        "arn:aws:iam::111122223333:role/emr-execution-role-
name"
                    ]
                }
            }
        }
    ]
}
```

4. Set up the cross-account network so that traffic can move back and forth between your accounts. For guided instruction, see *the section called "Configure network access"Set up the* . The steps in this section help you complete the following tasks:

   a. VPC-peer your Studio account and your Amazon EMR account to establish a connection.

   b. Manually add routes to the private subnet route tables in both accounts. This permits creation and connection of Amazon EMR clusters from the Studio account to the remote account's private subnet.

   c. Set up the security group attached to your Studio domain to allow outbound traffic and the security group of the Amazon EMR primary node to allow inbound TCP traffic from the Studio instance security group.

5. Preload your IAM runtime roles so you can select the role to use when you connect to your Amazon EMR cluster. For details about how to preload your IAM roles, see Preload your execution roles into Studio or Studio Classic.

Configure Lake Formation access

When you access data from data lakes managed by AWS Lake Formation, you can enforce table-level and column-level access using policies attached to your runtime role. To configure permission for Lake Formation access, see Integrate Amazon EMR with AWS Lake Formation.

Preload your execution roles into Studio or Studio Classic

You can preload your IAM runtime roles so you can select the role to use when you connect to your Amazon EMR cluster. Users of JupyterLab in Studio can use the SageMaker AI console or the provided script.

Preload runtime roles in JupyterLab using the SageMaker AI console

To associate your runtime roles with your user profile or domain using the SageMaker AI console:

1. Navigate to the SageMaker AI console at https://console.aws.amazon.com/sagemaker/.
2. In the left navigation pane, choose **domain**, and then select the domain using the SageMaker AI execution role whose permissions you updated.
3. - To add your runtime (and access roles for cross-account use case) to your domain: In the **App Configurations** tab of the **Domain details** page, navigate to the **JupyterLab** section.
   - To add your runtime (and access roles for cross-account use case) to your user profile: On the **Domain details** page, chose the **User profiles** tab, select the user profile using the SageMaker AI execution role whose permissions you updated. In the **App Configurations** tab, navigate to the **JupyterLab** section.
4. Choose **Edit** and add the ARNs of your access role (assumable role) and EMR Serverless runtime execution roles.
5. Choose **Submit**.

When you next connect to an Amazon EMR server, the runtime roles should appear in a drop-down menu for selection.

Preload runtime roles in JupyterLab using a Python script

In a JupyterLab application started from a space using the SageMaker AI execution role whose permissions you updated, run the following command in a terminal. Replace the `domainID`, `user-profile-name`, `emr-accountID`, and `EMRServiceRole` with their proper values. This code snippet updates a user profile settings (`client.update_user_profile`) within a SageMaker AI domain in a cross account use case. Specifically, it sets the service roles for Amazon EMR. It also allows the JupyterLab application to assume a particular IAM role (`AssumableRole` or `AccessRole`) for running Amazon EMR within the Amazon EMR account.

Alternatively, use `client.update_domain` to update the domain settings if your space uses an execution role set at the domain level.

```
import botocore.session
import json
sess = botocore.session.get_session()
client = sess.create_client('sagemaker')

client.update_user_profile(
DomainId="domainID",
UserProfileName="user-profile-name",
UserSettings={
    'JupyterLabAppSettings': {
        'EmrSettings': {
            'AssumableRoleArns': ["arn:aws:iam::emr-accountID:role/AssumableRole"],
            'ExecutionRoleArns': ["arn:aws:iam::emr-accountID:role/EMRServiceRole",
                                 "arn:aws:iam::emr-accountID:role/AnotherServiceRole"]
        }

    }
})
resp = client.describe_user_profile(DomainId="domainID", UserProfileName=user-
profile-name")

resp['CreationTime'] = str(resp['CreationTime'])
resp['LastModifiedTime'] = str(resp['LastModifiedTime'])
print(json.dumps(resp, indent=2))
```

Preload runtime roles in Studio Classic

Provide the ARN of the `AccessRole` (`AssumableRole`) to your SageMaker AI execution role. The ARN is loaded by the Jupyter server at launch. The execution role used by Studio assumes that cross-account role to discover and connect to Amazon EMR clusters in the *trusting account*.

You can specify this information by using Lifecycle Configuration (LCC) scripts. You can attach the LCC to your domain or a specific user profile. The LCC script that you use must be a JupyterServer configuration. For more information on how to create an LCC script, see Use Lifecycle Configurations with Studio Classic.

The following is an example LCC script. To modify the script, replace `AssumableRole` and `emr-account` with their respective values. The number of cross-accounts is limited to five.

The following snippet is an example LCC bash script you can apply if your Studio Classic application and cluster are in the same account:

```
#!/bin/bash

set -eux

FILE_DIRECTORY="/home/sagemaker-user/.sagemaker-analytics-configuration-
DO_NOT_DELETE"
FILE_NAME="emr-configurations-DO_NOT_DELETE.json"
FILE="$FILE_DIRECTORY/$FILE_NAME"

mkdir -p $FILE_DIRECTORY

cat << 'EOF' > "$FILE"
{
    "emr-execution-role-arns":
    {
      "123456789012": [
          "arn:aws:iam::123456789012:role/emr-execution-role-1",
          "arn:aws:iam::123456789012:role/emr-execution-role-2"
      ]
    }
}
EOF
```

If your Studio Classic application and clusters are in different accounts, specify the Amazon EMR access roles that can use the cluster. In the following example policy, *123456789012* is the Amazon EMR cluster account ID, and *212121212121* and *434343434343* are the ARNs for the permitted Amazon EMR access roles.

```
#!/bin/bash

set -eux

FILE_DIRECTORY="/home/sagemaker-user/.sagemaker-analytics-configuration-
DO_NOT_DELETE"
FILE_NAME="emr-configurations-DO_NOT_DELETE.json"
FILE="$FILE_DIRECTORY/$FILE_NAME"

mkdir -p $FILE_DIRECTORY
```

```
cat << 'EOF' > "$FILE"
{
    "emr-execution-role-arns":
    {
      "123456789012": [
          "arn:aws:iam::212121212121:role/emr-execution-role-1",
          "arn:aws:iam::434343434343:role/emr-execution-role-2"
      ]
    }
}
EOF

# add your cross-account EMR access role
FILE_DIRECTORY="/home/sagemaker-user/.cross-account-configuration-DO_NOT_DELETE"
FILE_NAME="emr-discovery-iam-role-arns-DO_NOT_DELETE.json"
FILE="$FILE_DIRECTORY/$FILE_NAME"

mkdir -p $FILE_DIRECTORY

cat << 'EOF' > "$FILE"
{
    "123456789012": "arn:aws:iam::123456789012:role/cross-account-emr-access-role"
}
EOF
```

Reference policies

- **List Amazon EMR policies**: This policy allows performing the following actions:
  - `AllowPresignedUrl` allows generating pre-signed URLs for accessing the Spark UI from within Studio.
  - `AllowClusterDiscovery` and `AllowClusterDetailsDiscovery` allows listing and describing Amazon EMR clusters in the provided region and account.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPresignedUrl",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:CreatePersistentAppUI",
                "elasticmapreduce:DescribePersistentAppUI",
                "elasticmapreduce:GetPersistentAppUIPresignedURL",
                "elasticmapreduce:GetOnClusterAppUIPresignedURL"
            ],
            "Resource": [
                "arn:aws:elasticmapreduce:us-east-1:111122223333:cluster/*"
            ]
        },
        {
            "Sid": "AllowClusterDetailsDiscovery",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:ListInstances",
                "elasticmapreduce:ListInstanceGroups",
                "elasticmapreduce:DescribeSecurityConfiguration"
            ],
            "Resource": [
                "arn:aws:elasticmapreduce:us-east-1:111122223333:cluster/*"
            ]
        },
        {
            "Sid": "AllowClusterDiscovery",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

- **Create Amazon EMR clusters policies**: This policy allows performing the following actions:
  - `AllowEMRTemplateDiscovery` allows searching for Amazon EMR templates in the Service Catalog. Studio and Studio Classic use this to show available templates.
  - `AllowSagemakerProjectManagement` enables the creation of What is a SageMaker AI Project?. In Studio or Studio Classic, access to the AWS Service Catalog is managed through What is a SageMaker AI Project?.

The IAM policy defined in the provided JSON grants those permissions. Replace *region* and *accountID* with your actual region and AWS account ID values before copying the list of statements to the inline policy of your role.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEMRTemplateDiscovery",
            "Effect": "Allow",
            "Action": [
                "servicecatalog:SearchProducts"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowSagemakerProjectManagement",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreateProject",
                "sagemaker:DeleteProject"
            ],
            "Resource": "arn:aws:sagemaker:us-east-1:111122223333:project/*"
        }
    ]
}
```

- **Domain, user profile, and space update actions policy** : The following policy grants permissions to update SageMaker AI domains, user profiles, and spaces within the specified region and AWS account.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SageMakerUpdateResourcesPolicy",
            "Effect": "Allow",
            "Action": [
                "sagemaker:UpdateDomain",
                "sagemaker:UpdateUserprofile",
                "sagemaker:UpdateSpace"
            ],
            "Resource": [
                "arn:aws:sagemaker:us-east-1:111122223333:domain/*",
                "arn:aws:sagemaker:us-east-1:111122223333:user-profile/*"
            ]
        }
    ]
}
```

User guide

This section covers how data scientist and data engineers can launch, discover, connect to, or terminate an Amazon EMR cluster from Studio or Studio Classic.

Before users can list or launch clusters, administrators must have configured the necessary settings in the Studio environment. For information on how administrators can configure a Studio environment to allow self-provisioning and listing of Amazon EMR clusters, see the section called “Admin guide”.

Topics

Supported images and kernels to connect to an Amazon EMR cluster from Studio or Studio Classic

The following images and kernels come with sagemaker-studio-analytics-extension, the JupyterLab extension that connects to a remote Spark (Amazon EMR) cluster via the SparkMagic library using Apache Livy.

- **For Studio users:** SageMaker Distribution is a Docker environment for data science used as the default image of JupyterLab notebook instances. All versions of SageMaker AI Distribution come with `sagemaker-studio-analytics-extension` pre-installed.
- **For Studio Classic users:** The following images come pre-installed with `sagemaker-studio-analytics-extension`:
  - DataScience – Python 3 kernel
  - DataScience 2.0 – Python 3 kernel
  - DataScience 3.0 – Python 3 kernel
  - SparkAnalytics 1.0 – SparkMagic and PySpark kernels
  - SparkAnalytics 2.0 – SparkMagic and PySpark kernels
  - SparkMagic – SparkMagic and PySpark kernels
  - PyTorch 1.8 – Python 3 kernels
  - TensorFlow 2.6 – Python 3 kernel
  - TensorFlow 2.11 – Python 3 kernel

To connect to Amazon EMR clusters using another built-in image or your own image, follow the instructions in Bring your own image.

Bring your own image

To bring your own image in Studio or Studio Classic and allow your notebooks to connect to Amazon EMR clusters, install the following sagemaker-studio-analytics-extension extension to your kernel. It supports connecting SageMaker Studio or Studio Classic notebooks to Spark(Amazon EMR) clusters through the SparkMagic library.

```
pip install sparkmagic
pip install sagemaker-studio-sparkmagic-lib
pip install sagemaker-studio-analytics-extension
```

Additionally, to connect to Amazon EMR with Kerberos authentication, you must install the kinit client. Depending on your OS, the command to install the kinit client can vary. To bring an Ubuntu (Debian based) image, use the `apt-get install -y -qq krb5-user` command.

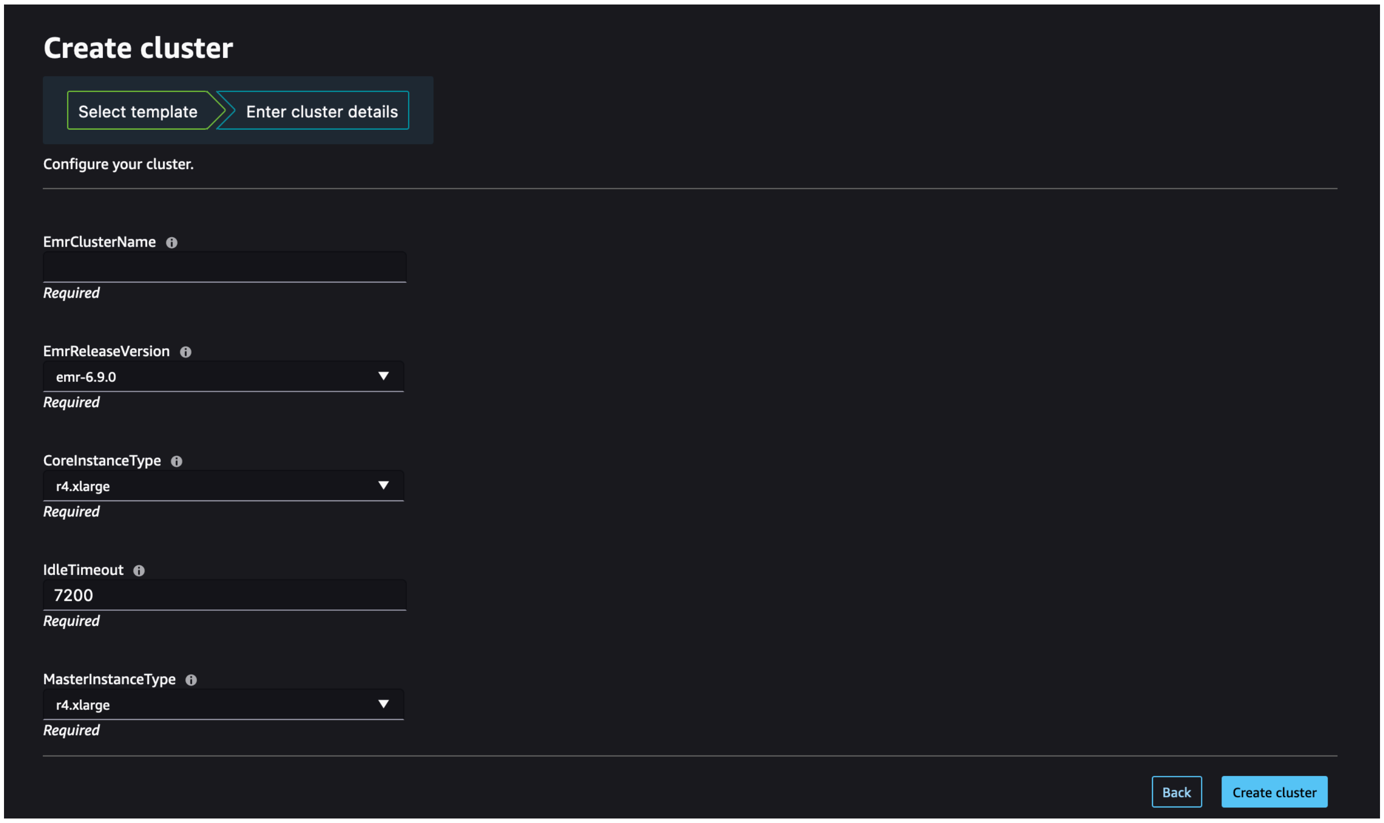
For more information on bringing your own image in SageMaker Studio or Studio Classic, see Bring your own SageMaker image.

Launch an Amazon EMR cluster from Studio or Studio Classic

Data scientists and data engineers can self-provision Amazon EMR clusters from Studio or Studio Classic using CloudFormation templates set up by their administrators. Before users can launch a cluster, administrators must have configured the necessary settings in the Studio environment. For information on how administrators can configure a Studio environment to allow self-provisioning Amazon EMR clusters, see Configure Amazon EMR CloudFormation templates in the Service Catalog.

To provision a new Amazon EMR cluster from Studio or Studio Classic:

1. In the Studio or Studio Classic UI's left-side panel, select the **Data** node in the left navigation menu. Navigate down to **Amazon EMR Clusters**. This opens up a page listing the Amazon EMR clusters that you can access from Studio or Studio Classic.
2. Choose the **Create** button at the top right corner. This opens up a new modal listing the cluster templates available to you.
3. Select a cluster template by choosing a template name and then choose **Next**.
4. Enter the cluster's details, such as a cluster name and any specific configurable parameter set by your administrator, and then choose **Create cluster**. The creation of the cluster might take a couple of minutes.



Configure your cluster.

EmrClusterName

Required

EmrReleaseVersion
emr-6.9.0
Required

CoreInstanceType
r4.xlarge
Required

IdleTimeout
7200
Required

MasterInstanceType
r4.xlarge
Required

Once the cluster is provisioned, the Studio or Studio Classic UI displays a *The cluster has been successfully created* message.

To connect to your cluster, see Connect to an Amazon EMR cluster from SageMaker Studio or Studio Classic

List Amazon EMR clusters from Studio or Studio Classic

Data scientists and data engineers can discover, and then connect to Amazon EMR clusters from Studio. The Amazon EMR clusters may be in the same AWS account as Studio or in a different AWS account.

Before users can list or connect to clusters, administrators must have configured the necessary settings in the Studio environment. For information on how administrators can configure a Studio environment to allow discovering running Amazon EMR clusters, see the section called "Admin guide". If your administrator configured the cross-account discovery of Amazon EMR clusters, you can view a consolidated list of clusters. The list includes clusters from the AWS account used by Studio as well as clusters from remote accounts that you have been granted access to.

To view the list of available Amazon EMR clusters from within Studio:

1. In the Studio UI's left navigation menu, scroll down to **EMR Clusters**. This opens up a page listing the Amazon EMR clusters that you have access to.

   The list displays clusters in the following stages: **Bootstrapping**, **Starting Running**, **Waiting**. You can narrow down the displayed clusters by their current status using the filter icon.
2. Choose a particular **Running** cluster you want to connect to, and then refer to Connect to an Amazon EMR cluster from SageMaker Studio or Studio Classic.

Connect to an Amazon EMR cluster from SageMaker Studio or Studio Classic

Data scientists and data engineers can discover and then connect to an Amazon EMR cluster directly from the Studio user interface. Before you begin, ensure that you have configured the necessary permissions as described in the Step 4: Set up the permissions to enable listing and launching Amazon EMR clusters from Studio section. These permissions grant Studio the ability to create, start, view, access, and terminate clusters.

You can connect an Amazon EMR cluster to a new JupyterLab notebook directly from the Studio UI, or choose to initiate the connection in a notebook of a running JupyterLab application.

Important

You can only discover and connect to Amazon EMR clusters for JupyterLab and Studio Classic applications that are launched from private spaces. Ensure that the Amazon EMR clusters are located in the same AWS region as your Studio environment. Your JupyterLab space must use a SageMaker Distribution image version `1.10` or higher.

Connect to an Amazon EMR cluster using the Studio UI

To connect to your cluster using the Studio or Studio Classic UI, you can either initiate a connection from the list of clusters accessed in List Amazon EMR clusters from Studio or Studio Classic, or from a notebook in SageMaker Studio or Studio Classic.

To connect an Amazon EMR cluster to a new JupyterLab notebook from the Studio UI:

1. In the Studio UI's left-side panel, select the **Data** node in the left navigation menu. Navigate down to **Amazon EMR applications and clusters**. This opens up a page listing the Amazon EMR clusters that you can access from Studio in the **Amazon EMR clusters** tab.

> **Note**
>
> If you or your administrator have configured the permissions to allow cross-account access to Amazon EMR clusters, you can view a consolidated list of clusters across all accounts that you have granted access to Studio.

2. Select an Amazon EMR cluster you want to connect to a new notebook, and then choose **Attach to notebook**. This opens up a modal window displaying the list of your JupyterLab spaces.
3.
   - Select the space from which you want to launch a JupyterLab application, and then choose **Open notebook**. This launches a JupyterLab application from your chosen space and opens a new notebook.

     > **Note**
     >
     > Users of Studio Classic need to select an image and kernel. For a list of supported images, see Supported images and kernels to connect to an Amazon EMR cluster from Studio or Studio Classic or refer to Bring your own image.

   - Alternatively, you can create a new private space by choosing the **Create new space** button at the top of the modal window. Enter a name for your space and then choose **Create space and open notebook**. This creates a private space with the default instance type and latest SageMaker distribution image available, launches a JupyterLab application, and opens a new notebook.
4. If the cluster you select does not use Kerberos, LDAP, or runtime role authentication, Studio prompts you to select the credential type. Choose from **Http basic authentication** or **No credentials**, then enter your credentials, if applicable.

   If the cluster you select supports runtime roles, choose the name of the IAM role that your Amazon EMR cluster can assume for the job run.

   > **Important**
   >
   > To successfully connect a JupyterLab notebook to an Amazon EMR cluster supporting runtime roles, you must first associate the list of runtime roles with your domain or user profile, as outlined in the section called “Configure IAM runtime roles for Amazon

EMR cluster access”. Failing to complete this step will prevent you from establishing the connection.

Upon selection, a connection command populates the first cell of your notebook and initiates the connection with the Amazon EMR cluster.

Once the connection succeeds, a message confirms the connection and the start of the Spark application.

Alternatively, you can connect to a cluster from a JupyterLab or Studio Classic notebook.

1. Choose the **Cluster** button at the top of your notebook. This opens a modal window listing the Amazon EMR clusters in a `Running` state that you can access. You can see the `Running` Amazon EMR clusters in the **Amazon EMR clusters** tab.

   Note

   For the users of Studio Classic, **Cluster** is only visible when you use a kernel from Supported images and kernels to connect to an Amazon EMR cluster from Studio or Studio Classic or from Bring your own image. If you cannot see **Cluster** at the top of your notebook, ensure that your administrator has configured the discoverability of your clusters and switch to a supported kernel.

2. Select the cluster to which you want to connect, then choose **Connect**.
3. If you configured your Amazon EMR clusters to support runtime IAM roles, you can select your role from the **Amazon EMR execution role** drop down menu.

   Important

   To successfully connect a JupyterLab notebook to an Amazon EMR cluster supporting runtime roles, you must first associate the list of runtime roles with your domain or user profile, as outlined in the section called “Configure IAM runtime roles for Amazon EMR cluster access”. Failing to complete this step will prevent you from establishing the connection.

Otherwise, if the cluster you choose does not use Kerberos, LDAP, or runtime role authentication, Studio or Studio Classic prompts you to select the credential type. You can choose **HTTP basic authentication** or **No credential**.

4. Studio adds and then run a code block to an active cell to establish the connection. This cell contains the connection magic command to connect your notebook to your application according to your authentication type.

   Once the connection succeeds, a message confirms the connection and the start of the Spark application.

Connect to an Amazon EMR cluster using a connection command

To establish a connection to an Amazon EMR cluster, you can execute connection commands within a notebook cell.

When establishing the connection, you can authenticate using Kerberos, Lightweight Directory Access Protocol (LDAP), or runtime IAM role authentication. The authentication method you choose depends on your cluster configuration.

You can refer to this example Access Apache Livy using a Network Load Balancer on a Kerberos-enabled Amazon EMR cluster to set up an Amazon EMR cluster that uses Kerberos authentication. Alternatively, you can explore the CloudFormation example templates using Kerberos or LDAP authentication in the aws-samples/sagemaker-studio-emr GitHub repository.

If your administrator has enabled cross-account access, you can connect to your Amazon EMR cluster from a Studio Classic notebook, regardless of whether your Studio Classic application and cluster reside in the same AWS account or different accounts.

For each of the following authentication types, use the specified command to connect to your cluster from your Studio or Studio Classic notebook.

- **Kerberos**

  Append the `--assumable-role-arn` argument if you need cross-account Amazon EMR access. Append the `--verify-certificate` argument if you connect to your cluster with HTTPS.

  ```
  %load_ext sagemaker_studio_analytics_extension.magics
  %sm_analytics emr connect --cluster-id cluster_id \
  ```

```
--auth-type Kerberos --language python
[--assumable-role-arn EMR_access_role_ARN ]
[--verify-certificate /home/user/certificateKey.pem]
```

- **LDAP**

  Append the `--assumable-role-arn` argument if you need cross-account Amazon EMR access. Append the `--verify-certificate` argument if you connect to your cluster with HTTPS.

  ```
  %load_ext sagemaker_studio_analytics_extension.magics
  %sm_analytics emr connect --cluster-id cluster_id \
  --auth-type Basic_Access --language python
  [--assumable-role-arn EMR_access_role_ARN ]
  [--verify-certificate /home/user/certificateKey.pem]
  ```

- **NoAuth**

  Append the `--assumable-role-arn` argument if you need cross-account Amazon EMR access. Append the `--verify-certificate` argument if you connect to your cluster with HTTPS.

  ```
  %load_ext sagemaker_studio_analytics_extension.magics
  %sm_analytics emr connect --cluster-id cluster_id \
  --auth-type None --language python
  [--assumable-role-arn EMR_access_role_ARN ]
  [--verify-certificate /home/user/certificateKey.pem]
  ```

- **Runtime IAM roles**

  Append the `--assumable-role-arn` argument if you need cross-account Amazon EMR access. Append the `--verify-certificate` argument if you connect to your cluster with HTTPS.

  For more information on connecting to an Amazon EMR cluster using runtime IAM roles, see Configure IAM runtime roles for Amazon EMR cluster access in Studio.

  ```
  %load_ext sagemaker_studio_analytics_extension.magics
  %sm_analytics emr connect --cluster-id cluster_id \
  --auth-type Basic_Access \
  --emr-execution-role-arn arn:aws:iam::studio_account_id:role/emr-execution-role-name
  [--assumable-role-arn EMR_access_role_ARN]
  [--verify-certificate /home/user/certificateKey.pem]
  ```

Connect to an Amazon EMR cluster over HTTPS

If you have configured your Amazon EMR cluster with transit encryption enabled and Apache Livy server for HTTPS and would like Studio or Studio Classic to communicate with Amazon EMR using HTTPS, you need to configure Studio or Studio Classic to access your certificate key.

For self-signed or local Certificate Authority (CA) signed certificates, you can do this in two steps:

1. Download the PEM file of your certificate to your local file system using one of the following options:
   - Jupyter's built-in file upload function.
   - A notebook cell.
   - (For Studio Classic users only) A lifecycle configuration (LCC) script.

     For information on how to use an LCC script, see Customize a Notebook Instance Using a Lifecycle Configuration Script
2. Enable the validation of the certificate by providing the path to your certificate in the `--verify-certificate` argument of your connection command.

   ```
   %sm_analytics emr connect --cluster-id cluster_id \
   --verify-certificate /home/user/certificateKey.pem ...
   ```

For public CA issued certificates, set the certificate validation by setting the `--verify-certificate` parameter as `true`.

Alternatively, you can disable the certificate validation by setting the `--verify-certificate` parameter as `false`.

You can find the list of available connection commands to an Amazon EMR cluster in Connect to an Amazon EMR cluster using a connection command.

Terminate an Amazon EMR cluster from Studio or Studio Classic

The following procedure shows how to terminate an Amazon EMR cluster from a Studio or Studio Classic notebook.

To terminate a cluster in a `Running` state, navigate to the list of available Amazon EMR clusters.

1. In the Studio UI, scroll down to the **Data** node in the left navigation menu.
2. Navigate down to the **EMR Clusters** node. This opens up a page listing the Amazon EMR clusters that you have access to.
3. Select the name of the cluster that you want to terminate, and then choose **Terminate**.
4. This opens up a confirmation window informing you that any pending work or data on your cluster will be lost permanently after termination. Confirm by choosing **Terminate** again.

Access Spark UI from Studio or Studio Classic

The following sections give instructions for accessing the Spark UI from SageMaker AI Studio or Studio Classic notebooks. The Spark UI allows you to monitor and debug your Spark Jobs submitted to run on Amazon EMR from Studio or Studio Classic notebooks. SSH tunneling and presigned URLs are two ways for accessing the Spark UI.

Set up SSH tunneling for Spark UI access

To set up SSH tunneling to access the Spark UI, follow one of the two options in this section.

Options for setting up SSH tunneling:

- Option 1: Set up an SSH tunnel to the master node using local port forwarding
- Option 2, part 1: Set up an SSH tunnel to the master node using dynamic port forwarding

  Option 2, part 2: Configure proxy settings to view websites hosted on the master node

For information about viewing web interfaces hosted on Amazon EMR clusters, see View web interfaces hosted on Amazon EMR Clusters. You can also visit your Amazon EMR console to get access to the Spark UI.

Note

You can set up an SSH tunnel even if presigned URLs are not available to you.

Presigned URLs

To create one-click URLs that can access Spark UI on Amazon EMR from SageMaker Studio or Studio Classic notebooks, you must enable the following IAM permissions. Choose the option that applies to you:

- **For Amazon EMR clusters that are in the same account as the SageMaker Studio or Studio Classic notebook: Add the following permissions to the SageMaker Studio or Studio Classic IAM execution role.**
- **For Amazon EMR clusters that are in a different account (not SageMaker Studio or Studio Classic notebook): Add the following permissions to the cross-account role that you created for List Amazon EMR clusters from Studio or Studio Classic.**

> **Note**
>
> You can access presigned URLs from the console in the following regions:
>
> - US East (N. Virginia) Region
> - US West (N. California) Region
> - Canada (Central) Region
> - Europe (Frankfurt) Region
> - Europe (Stockholm) Region
> - Europe (Ireland) Region
> - Europe (London) Region
> - Europe (Paris) Region
> - Asia Pacific (Tokyo) Region
> - Asia Pacific (Seoul) Region
> - Asia Pacific (Sydney) Region
> - Asia Pacific (Mumbai) Region
> - Asia Pacific (Singapore) Region
> - South America (São Paulo)

The following policy gives access to presigned URLs for your execution role.

```
{
        "Sid": "AllowPresignedUrl",
        "Effect": "Allow",
        "Action": [
            "elasticmapreduce:DescribeCluster",
            "elasticmapreduce:ListInstanceGroups",
            "elasticmapreduce:CreatePersistentAppUI",
            "elasticmapreduce:DescribePersistentAppUI",
            "elasticmapreduce:GetPersistentAppUIPresignedURL",
            "elasticmapreduce:GetOnClusterAppUIPresignedURL"
        ],
        "Resource": [
            "arn:aws:elasticmapreduce:region:account-id:cluster/*"
        ]
}
```

Blogs and whitepapers

The following blogs use a case study of sentiment prediction for a movie review to illustrate the process of executing a complete machine learning workflow. This includes data preparation, monitoring Spark jobs, and training and deploying a ML model to get predictions directly from your Studio or Studio Classic notebook.

- Create and manage Amazon EMR clusters from SageMaker Studio or Studio Classic to run interactive Spark and ML workloads.
- To extend the use case to a cross-account configuration where SageMaker Studio or Studio Classic and your Amazon EMR cluster are deployed in separate AWS accounts, see Create and manage Amazon EMR clusters from SageMaker Studio or Studio Classic to run interactive Spark and ML workloads - Part 2.

See also:

- A walkthrough of the configuration of Access Apache Livy using a Network Load Balancer on a Kerberos-enabled Amazon EMR cluster.
- AWS whitepapers for SageMaker Studio or Studio Classic best practices.

Troubleshooting

When working with Amazon EMR clusters from Studio or Studio Classic notebooks, you may encounter various potential issues or challenges during the connection or usage process. To help you troubleshoot and resolve these errors, this section provides guidance on common problems that can arise.

The following are common errors that might occur while connecting or using Amazon EMR clusters from Studio or Studio Classic notebooks.

Troubleshoot Livy connections hanging or failing

The following are Livy connectivity issues that might occur while using Amazon EMR clusters from Studio or Studio Classic notebooks.

- **Your Amazon EMR cluster encountered an out-of-memory error.**

  A possible reason for a Livy connection via `sparkmagic` hanging or failing is if your Amazon EMR cluster encountered an out-of-memory error.

  By default, the Java configuration parameter of the Apache Spark driver, `spark.driver.defaultJavaOptions`, is set to `-XX:OnOutOfMemoryError='kill -9 %p'`. This means that the default action taken when the driver program encounters an `OutOfMemoryError` is to terminate the driver program by sending a SIGKILL signal. When the Apache Spark driver is terminated, any Livy connection via `sparkmagic` that depends on that driver hangs or fails. This is because the Spark driver is responsible for managing the Spark application's resources, including task scheduling and execution. Without the driver, the Spark application cannot function, and any attempts to interact with it fails.

  If you suspect that your Spark cluster is experiencing memory issues, you can check Amazon EMR logs. Containers killed due to out-of-memory errors typically exit with a code of 137. In such cases, you need to restart the Spark application and establish a new Livy connection to resume interaction with the Spark cluster.

  You can refer to the knowledge base article How do I resolve the error "Container killed by YARN for exceeding memory limits" in Spark on Amazon EMR? on AWS re:Post to learn about various strategies and parameters that can be used to address an out-of-memory issue.

  We recommend reviewing the Amazon EMR Best Practices Guides for best practices and tuning guidance on running Apache Spark workloads on your Amazon EMR clusters.

- **Your Livy session times out when connecting to an Amazon EMR cluster for the first time.**

  When you initially connect to an Amazon EMR cluster using sagemaker-studio-analytics-extension, which enables connection to a remote Spark (Amazon EMR) cluster via the SparkMagic library using Apache Livy, you may encounter a connection timeout error:

  ```
  An error was encountered: Session 0 did not start up in 60 seconds.
  ```

  If your Amazon EMR cluster requires the initialization of a Spark application upon establishing a connection, there is an increased chance of seeing connection timeout errors.

  To reduce the chances of getting timeouts when connecting to an Amazon EMR cluster using Livy through the analytics extension, `sagemaker-studio-analytics-extension` version `0.0.19` and later override the default server session timeout to `120` seconds instead of `sparkmagic`'s default of `60` seconds.

  We recommend upgrading your extension `0.0.18` and sooner by running the following upgrade command.

  ```
  pip install --upgrade sagemaker-studio-analytics-extension
  ```

  Note that when providing a custom timeout configuration in `sparkmagic`, `sagemaker-studio-analytics-extension` honors this override. However, setting the session timeout to `60` seconds automatically triggers the default server session timeout of `120` seconds in `sagemaker-studio-analytics-extension`.

Data preparation using AWS Glue interactive sessions

AWS Glue interactive sessions is a serverless service that you can enlist to collect, transform, clean, and prepare data for storage in your data lakes and data pipelines. AWS Glue interactive sessions provides an on-demand, serverless Apache Spark runtime environment that you can initialize in seconds on a dedicated Data Processing Unit (DPU) without having to provision and manage complex compute cluster infrastructure. After initialization, you can browse the AWS Glue data catalog, run large queries, access data governed by AWS Lake Formation, and interactively analyze and prepare data using Spark, right in your Studio or Studio Classic notebooks. You can then use the prepared data to train, tune, and deploy models using the purpose-built ML tools within SageMaker Studio or Studio Classic. You should consider AWS Glue Interactive Sessions for your