# Table of Contents

What Is Amazon MSK? .......................................................................................................................... 1
Setting Up .................................................................................................................................................. 3
Sign Up for AWS ...................................................................................................................................... 3
Download Libraries and Tools .................................................................................................................. 3
Getting Started ......................................................................................................................................... 4
  Step 1: Create a VPC ............................................................................................................................ 4
  Step 2: Enable High Availability and Fault Tolerance ................................................................. 7
  Step 3: Create a Cluster ....................................................................................................................... 8
  Step 4: Create a Client Machine ....................................................................................................... 10
  Step 5: Create a Topic .......................................................................................................................... 11
  Step 6: Produce and Consume Data ............................................................................................... 12
  Step 7: View Metrics .......................................................................................................................... 13
  Step 8: Delete the Cluster ............................................................................................................... 14
How It Works .......................................................................................................................................... 15
  Creating a Cluster ............................................................................................................................... 15
    Creating a Cluster Using the AWS Management Console ......................................................... 15
    Creating a Cluster Using the AWS CLI .......................................................................................... 16
    Creating a Cluster with a Custom MSK Configuration Using the AWS CLI ....................... 17
    Creating a Cluster Using the API ................................................................................................. 18
  Deleting a Cluster ............................................................................................................................. 18
    Deleting a Cluster Using the AWS Management Console ....................................................... 18
    Deleting a Cluster Using the AWS CLI ....................................................................................... 18
    Deleting a Cluster using the API ................................................................................................. 18
  Getting the Apache ZooKeeper Connection String ..................................................................... 18
    Getting the Apache ZooKeeper Connection String Using the AWS Management Console .... 18
    Getting the Apache ZooKeeper Connection String Using the AWS CLI ............................... 19
    Getting the Apache ZooKeeper Connection String Using the API ....................................... 20
  Getting the Bootstrap Brokers .......................................................................................................... 20
    Getting the Bootstrap Brokers Using the AWS Management Console .................................. 20
    Getting the Bootstrap Brokers Using the AWS CLI ................................................................. 20
    Getting the Bootstrap Brokers Using the API ............................................................................ 20
  Listing Clusters ................................................................................................................................. 20
    Listing Clusters Using the AWS Management Console ............................................................ 20
    Listing Clusters Using the AWS CLI ............................................................................................. 21
    Listing Clusters Using the API ........................................................................................................ 21
  Scaling Up Broker Storage ............................................................................................................... 21
    Scaling Up Broker Storage Using the AWS Management Console ......................................... 21
    Scaling Up Broker Storage Using the AWS CLI ........................................................................ 21
    Scaling Up Broker Storage Using the API .................................................................................. 22
  Updating the Configuration of a Cluster ......................................................................................... 22
    Updating the Configuration of a Cluster Using the AWS CLI .................................................. 22
    Updating the Configuration of a Cluster Using the API ............................................................ 23
  Expanding a Cluster ............................................................................................................................ 23
    Expanding a Cluster Using the AWS Management Console ..................................................... 24
    Expanding a Cluster Using the AWS CLI ...................................................................................... 24
    Expanding a Cluster Using the API ............................................................................................... 25
  Monitoring a Cluster ........................................................................................................................... 25
    Amazon MSK Monitoring Levels for CloudWatch Metrics ...................................................... 25
    Amazon MSK Metrics for Monitoring with CloudWatch ............................................................ 26
    Viewing Amazon MSK Metrics Using CloudWatch ................................................................. 31
    Consumer-Lag Checking with Burrow ......................................................................................... 31
    Open Monitoring with Prometheus .............................................................................................. 33
  Tagging a Cluster ............................................................................................................................... 36
    Tag Basics .......................................................................................................................................... 36
Right-size Your Cluster .............................................................................................................. 78
Monitor Disk Space ................................................................................................................... 78
Adjust the Data Retention Parameters ........................................................................................ 78
Don't Add Non-MSK Brokers .................................................................................................... 79
Enable In-Transit Encryption ..................................................................................................... 79
Reassign Partitions ................................................................................................................... 79
Document History ..................................................................................................................... 80
AWS Glossary ............................................................................................................................ 81
What Is Amazon MSK?

Amazon Managed Streaming for Apache Kafka (Amazon MSK) is a fully managed service that enables you to build and run applications that use Apache Kafka to process streaming data. Amazon MSK provides the control-plane operations, such as those for creating, updating, and deleting clusters. It lets you use Apache Kafka data-plane operations, such as those for producing and consuming data. It runs open-source versions of Apache Kafka. This means existing applications, tooling, and plugins from partners and the Apache Kafka community are supported without requiring changes to application code. You can use Amazon MSK to create clusters that use Apache Kafka version 1.1.1, 2.2.1, or 2.3.1.

The following diagram provides an overview of how Amazon MSK works.

The diagram demonstrates the interaction between the following components:

- **Broker nodes** — When creating an Amazon MSK cluster, you specify how many broker nodes you want Amazon MSK to create in each Availability Zone. In the example cluster shown in this diagram, there's one broker per Availability Zone. Each Availability Zone has its own virtual private cloud (VPC) subnet.
- **ZooKeeper nodes** — Amazon MSK also creates the Apache ZooKeeper nodes for you. Apache ZooKeeper is an open-source server that enables highly reliable distributed coordination.
- **Producers, consumers, and topic creators** — Amazon MSK lets you use Apache Kafka data-plane operations to create topics and to produce and consume data.
- **AWS CLI** — You can use the AWS Command Line Interface (AWS CLI) or the APIs in the SDK to perform control-plane operations. For example, you can use the AWS CLI or the SDK to create or delete an Amazon MSK cluster, list all the clusters in an account, or view the properties of a cluster.
Amazon MSK detects and automatically recovers from the most common failure scenarios for clusters so that your producer and consumer applications can continue their write and read operations with minimal impact. When Amazon MSK detects a broker failure, it mitigates the failure or replaces the unhealthy or unreachable broker with a new one. In addition, where possible, it reuses the storage from the older broker to reduce the data that Apache Kafka needs to replicate. Your availability impact is limited to the time required for Amazon MSK to complete the detection and recovery. After a recovery, your producer and consumer apps can continue to communicate with the same broker IP addresses that they used before the failure.

To get started using Amazon MSK, see Getting Started (p. 4).

To see the control-plane operations available through Amazon MSK, see the Amazon MSK API Reference.

After you create a cluster, you can use Amazon CloudWatch to monitor it. For more information about monitoring your cluster using metrics, see the section called “Monitoring a Cluster” (p. 25).
Setting Up Amazon MSK

Before you use Amazon MSK for the first time, complete the following tasks.

Tasks

• Sign Up for AWS (p. 3)
• Download Libraries and Tools (p. 3)

Sign Up for AWS

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all services in AWS, including Amazon MSK. You are charged only for the services that you use.

If you have an AWS account already, skip to the next task. If you don't have an AWS account, use the following procedure to create one.

To sign up for an AWS account

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

Download Libraries and Tools

The following libraries and tools can help you work with Amazon MSK:

• The AWS Command Line Interface (AWS CLI) supports Amazon MSK. The AWS CLI enables you to control multiple AWS services from the command line and automate them through scripts. Upgrade your AWS CLI to the latest version to ensure that it has support for Amazon MSK. For detailed instructions on how to upgrade the AWS CLI, see Installing the AWS Command Line Interface.
• The Amazon Managed Streaming for Kafka API Reference documents the API operations that Amazon MSK supports.
• The AWS SDKs for Go, Java, JavaScript, .NET, Node.js, PHP, Python, and Ruby include Amazon MSK support and samples.
Getting Started Using Amazon MSK

This section shows you an example of how you can create an MSK cluster, produce and consume data, and monitor the health of your cluster using metrics. This example doesn't represent all the options you can choose when you create an MSK cluster. In different parts of this tutorial, we choose default options for simplicity. This doesn't mean that they're the only options that work for setting up an MSK cluster or client instances.

This is a step-by-step tutorial that uses the AWS Management Console and the AWS CLI. If you prefer to follow a different approach, you can perform the types of AWS operations described in this tutorial using only the AWS Management Console, only the AWS CLI, only the AWS API, or a combination of the three environments. Alternatively, you can use AWS CloudFormation to set up an MSK cluster. For some example AWS CloudFormation templates, see Amazon MSK CloudFormation Examples.

Prerequisites

Before you start, ensure that you have an AWS account and that you have the AWS Command Line Interface (AWS CLI) installed on your computer. For more information about these prerequisites, see Setting Up (p. 3).

Important
Make sure that you have the latest version of the AWS CLI. Earlier versions might not have the full functionality of Amazon MSK.

Topics
- Step 1: Create a VPC for Your MSK Cluster (p. 4)
- Step 2: Enable High Availability and Fault Tolerance (p. 7)
- Step 3: Create an Amazon MSK Cluster (p. 8)
- Step 4: Create a Client Machine (p. 10)
- Step 5: Create a Topic (p. 11)
- Step 6: Produce and Consume Data (p. 12)
- Step 7: Use Amazon CloudWatch to View Amazon MSK Metrics (p. 13)
- Step 8: Delete the Amazon MSK Cluster (p. 14)

Step 1: Create a VPC for Your MSK Cluster

In the first step of Getting Started Using Amazon MSK (p. 4), you use the Amazon VPC Console to create an Amazon Virtual Private Cloud (Amazon VPC). You create an MSK cluster in this VPC in a later step.

To create a VPC

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Choose Launch VPC Wizard.
3. Choose **Select** to accept the default Amazon VPC configuration named **VPC with a Single Public Subnet**.
4. For **VPC name**, enter `AWSKafkaTutorialVPC`. If you copy this name and paste it into the console, delete any white space that gets into the text field before the name.

5. For **Availability Zone**, choose `us-east-1a`.

6. For **Subnet name**, enter `AWSKafkaTutorialSubnet-1`. If you copy this name and paste it into the console, delete any white space that gets into the text field before the name.

7. Choose **Create VPC**, and then choose **OK**.

8. In the list of VPCs, find `AWSKafkaTutorialVPC` and copy its ID from the **VPC ID** column. Save this ID somewhere because you need it in some of the following steps.

**Next Step**

**Step 2: Enable High Availability and Fault Tolerance (p. 7)**
Step 2: Enable High Availability and Fault Tolerance

In this step of Getting Started Using Amazon MSK (p. 4), you enable high availability and fault tolerance. To do so, you add two subnets to the VPC that you created with one subnet in the previous step. After you complete this step, you will have three subnets in three different Availability Zones.

To add subnets to your VPC

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Subnets.
3. In the list of subnets, find AWSKafkaTutorialSubnet-1, and then find the column named Route table. Copy the value associated with AWSKafkaTutorialSubnet-1 in that column and save it for later.
4. Choose Create subnet.
5. For the Name tag, enter AWSKafkaTutorialSubnet-2. If you copy this name and paste it into the console, delete any white space that gets into the text field before the name.
6. For VPC, choose AWSKafkaTutorialVPC.
7. For Availability Zone, choose us-east-1b.
8. For IPv4 CIDR block, enter 10.0.1.0/24.
9. Choose Create, and then choose Close.
10. Choose AWSKafkaTutorialSubnet-2 from the list of subnets by selecting the check box next to it. Ensure that no other check boxes in the list are selected.
11. In the subnet view near the bottom of the page, choose the Route Table tab, and then choose Edit route table association.
Step 3: Create a Cluster

12. In the Route Table ID list, choose the route table whose value you copied earlier in this procedure.
13. Choose Save, and then choose Close.
14. Repeat this procedure to create another subnet with the name AWSKafkaTutorialSubnet-3, in the us-east-1c Availability Zone, and with the IPv4 CIDR block set to 10.0.2.0/24. If you copy the name of the subnet and paste it into the console, delete any white space that gets into the text field before the name.
15. Edit the route table for AWSKafkaTutorialSubnet-3 to ensure that it has the same route table used for AWSKafkaTutorialSubnet-1 and AWSKafkaTutorialSubnet-2.

Next Step
Step 3: Create an Amazon MSK Cluster (p. 8)

Step 3: Create an Amazon MSK Cluster

In this step of Getting Started Using Amazon MSK (p. 4), you create an Amazon MSK cluster in the VPC.

To create an Amazon MSK cluster using the AWS CLI

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Subnets, and then copy the subnet IDs of the three subnets you created previously. Save these IDs because you need them later in this procedure.
3. In the navigation pane, choose Security Groups. Then in the table of security groups, find the group for which the VPC ID column has the ID you saved for AWSKafkaTutorialVPC. Copy the ID of this security group and save it because you need it later in this procedure.
4. Copy the following JSON and save it to a file. Name the file clusterinfo.json.
5. In the `clusterinfo.json` file, replace the placeholders for the three subnet IDs and the security group ID with the values that you saved in previous steps.

6. In `clusterinfo.json`, replace `your-CMK` with a customer managed CMK. You can also remove EncryptionAtRest and let Amazon MSK create a CMK and use it on your behalf. Setting InCluster to `true` means that you want Amazon MSK to encrypt your data as it travels between brokers within the cluster. For ClientBroker you can choose one of the following settings: TLS, TLS_PLAINTEXT, or PLAINTEXT. In this exercise, we use TLS to indicate that we want data to be encrypted as it travels between clients and brokers. For more information about encryption settings, see the section called “Encryption” (p. 49).

7. Upgrade your AWS CLI to the latest version to ensure that it has support for Amazon MSK. For detailed instructions on how to upgrade the AWS CLI, see Installing the AWS Command Line Interface.

8. Run the following AWS CLI command in the directory where you saved the `clusterinfo.json` file.

   ```bash
   aws kafka create-cluster --cli-input-json file://clusterinfo.json
   ```

   The output of the command looks like the following JSON:

   ```json
   {
       "ClusterArn": "...",
       "ClusterName": "AWSKafkaTutorialCluster",
       "State": "CREATING"
   }
   ```

9. Save the value of the ClusterArn key because you need it later.

   **Important**
   Ensure that you saved ClusterArn before you proceed.

Next Step
Step 4: Create a Client Machine

In this step of Getting Started Using Amazon MSK (p. 4), you create a client machine. You use this client machine to create a topic that produces and consumes data. For simplicity, we'll put this client machine in the same VPC as the Amazon MSK cluster. But a client machine doesn't have to be in the same VPC as the cluster.

To create a client machine

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Choose Launch Instance.
3. Choose Select to create an instance of Amazon Linux 2 AMI (HVM), SSD Volume Type.
4. Choose the t2.xlarge instance type by selecting the check box next to it.
5. Choose Next: Configure Instance Details.
6. In the Network list, choose AWSKafkaTutorialVPC.
7. In the Auto-assign Public IP list, choose Enable.
8. In the menu near the top, choose 5. Add Tags.
9. Choose Add Tag.
10. Enter Name for the Key and AWSKafkaTutorialClient for the Value.
11. Choose Review and Launch, and then choose Launch.
12. Choose Create a new key pair, enter MSKKeyPair for Key pair name, and then choose Download Key Pair. Alternatively, you can use an existing key pair if you prefer.
13. Read the acknowledgement, select the check box next to it, and choose Launch Instances.
14. Choose View Instances. Then, in the Security Groups column, choose the security group that is associated with the AWSKafkaTutorialClient instance.
15. Copy the value of Group ID (and not the group name) that is associated with the security group, and save it for later.
17. In the navigation pane, choose Security Groups. In the VPC ID column of the security groups, find the row that contains the ID you saved for AWSKafkaTutorialVPC, and the Description column has the value default VPC security group. Choose this row by selecting the check box in the first column.
18. In the Inbound Rules tab, choose Edit rules.
20. In the new rule, choose All traffic in the Type column. In the second field in the Source column, enter the ID of the security group of the client machine. This is the group ID that you saved earlier.
21. Choose Save rules.
22. Repeat these steps to add an inbound rule in the security group that corresponds to your client machine to allow it to receive traffic from the AWSKafkaTutorialVPC security group. Now your client machine can communicate back and forth with your MSK cluster.

Next Step

Step 5: Create a Topic (p. 11)
Step 5: Create a Topic

In this step of Getting Started Using Amazon MSK (p. 4), you install Apache Kafka client libraries and tools on the client machine, and then you create a topic.

To create a topic on the client machine

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose Instances, and then choose AWSKafkaTutorialClient by selecting the check box next to it.
3. Choose Actions, and then choose Connect. Follow the instructions to connect to the client machine AWSKafkaTutorialClient.
4. Install Java on the client machine by running the following command:
   ```bash
   sudo yum install java-1.8.0
   ```
5. Run the following command to download Apache Kafka.
   ```bash
   wget https://archive.apache.org/dist/kafka/2.2.1/kafka_2.12-2.2.1.tgz
   ```
   **Note**
   If you want to use a mirror site other than the one used in this command, you can choose a different one on the Apache website.
6. Run the following command in the directory where you downloaded the TAR file in the previous step.
   ```bash
   tar -xzf kafka_2.12-2.2.1.tgz
   ```
7. Go to the `kafka_2.12-2.2.1` directory.
8. Cluster creation can take a few minutes. To find out whether the cluster you created is ready, run the following command, replacing `ClusterArn` with the Amazon Resource Name (ARN) that you obtained at the end of the section called “Step 3: Create a Cluster” (p. 8).
   ```bash
   aws kafka describe-cluster --region us-east-1 --cluster-arn "ClusterArn"
   ```

The result of running this command looks like the following JSON:

```json
{
   "ClusterInfo": {
      "BrokerNodeGroupInfo": {
         "BrokerAZDistribution": "DEFAULT",
         "ClientSubnets": [
            "subnet-0d44a1567c2ce409a",
            "subnet-051201cac65561565",
            "subnet-08b4ec6b2bd3bd8c2"
         ],
         "InstanceType": "kafka.m5.large",
         "SecurityGroups": [
            "sg-041e78b0a8ba7f834"
         ],
         "StorageInfo": {
            "EbsStorageInfo": {
               "VolumeSize": 1000
            }
         }
      }
   }
}
```
If the output of the command shows that the state of the cluster is still **CREATING**, wait a few minutes, and then run the command again. Keep running this command every few minutes until the state turns to **ACTIVE**. When the state is **ACTIVE**, the output of this `describe-cluster` command includes an additional key named `ZookeeperConnectString`. Copy the entire value associated with this key because you need it to create an Apache Kafka topic in the following command.

9. Run the following command, replacing `ZookeeperConnectString` with the value that you saved after you ran the `describe-cluster` command.

```
bin/kafka-topics.sh --create --zookeeper ZookeeperConnectString --replication-factor 3 --partitions 1 --topic AWSKafkaTutorialTopic
```

If the command succeeds, you see the following message: Created topic `AWSKafkaTutorialTopic`.

**Next Step**

**Step 6: Produce and Consume Data (p. 12)**

### Step 6: Produce and Consume Data

In this step of Getting Started Using Amazon MSK (p. 4), you produce and consume data.

**To produce and consume messages**

1. In this example we use the JVM truststore to talk to the MSK cluster. To do this, first create a folder named `/tmp` on the client machine. Then, go to the `bin` folder of the Apache Kafka installation and run the following command, replacing `JDKFolder` with the name of your JDK folder. For example, the name of the JDK folder on your instance might be `java-1.8.0-openjdk-1.8.0.201.b09-0.amzn2.x86_64`.

   ```
   cp /usr/lib/jvm/JDKFolder/jre/lib/security/cacerts /tmp/kafka.client.truststore.jks
   ```

2. While still in the `bin` folder of the Apache Kafka installation on the client machine, create a text file named `client.properties` with the following contents.

   ```
   security.protocol=SSL
   ssl.truststore.location=/tmp/kafka.client.truststore.jks
   ```
3. Run the following command, replacing `ClusterArn` with the Amazon Resource Name (ARN) that you obtained at the end of the section called “Step 3: Create a Cluster” (p. 8).

```
aws kafka get-bootstrap-brokers --region us-east-1 --cluster-arn ClusterArn
```

From the JSON result of the command, save the value associated with the string named "BootstrapBrokerString" because you need it in the following commands.

4. Run the following command, replacing `BootstrapBrokerString` with the value that you obtained when you ran the previous command.

```
kafka-console-producer.sh --broker-list BootstrapBrokerString --producer.config client.properties --topic AWSKafkaTutorialTopic
```

5. Enter any message that you want, and press Enter. Repeat this step two or three times. Every time you enter a line and press Enter, that line is sent to your Apache Kafka cluster as a separate message.

6. Keep the connection to the client machine open, and then open a second, separate connection to that machine in a new window.

7. In the following command, replace `BootstrapBrokerString` with the value that you saved earlier. Then, go to the `bin` folder and run the command using your second connection to the client machine.

```
kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --consumer.config client.properties --topic AWSKafkaTutorialTopic --from-beginning
```

You start seeing the messages you entered earlier when you used the console producer command. These messages are TLS encrypted in transit.

8. Enter more messages in the producer window, and watch them appear in the consumer window.

**Next Step**

**Step 7: Use Amazon CloudWatch to View Amazon MSK Metrics (p. 13)**

**Step 7: Use Amazon CloudWatch to View Amazon MSK Metrics**

In this step of Getting Started Using Amazon MSK (p. 4), you look at the Amazon MSK metrics in Amazon CloudWatch.

**To view Amazon MSK metrics in CloudWatch**

2. In the navigation pane, choose Metrics.
3. Choose the All metrics tab, and then choose AWS/Kafka.
4. To view broker-level metrics, choose Broker ID, Cluster Name. For cluster-level metrics, choose Cluster Name.
5. (Optional) In the graph pane, select a statistic and a time period, and then create a CloudWatch alarm using these settings.

**Next Step**
Step 8: Delete the Amazon MSK Cluster

In the final step of Getting Started Using Amazon MSK (p. 4), you delete the MSK cluster that you created in Step 3: Create an Amazon MSK Cluster (p. 8).

To delete the Amazon MSK cluster using the AWS CLI

1. Run the following command on the computer where you have the AWS CLI installed.
   ```bash
   aws kafka list-clusters --region us-east-1
   ```

2. In the output of the `list-clusters` command, look for the cluster Amazon Resource Name (ARN) that corresponds to the cluster that you want to delete. Copy that ARN.

3. Run the following command, replacing `ClusterArn` with the ARN that you obtained when you ran the previous command.
   ```bash
   aws kafka delete-cluster --region us-east-1 --cluster-arn ClusterArn
   ```
Amazon MSK: How It Works

An Amazon MSK cluster is the primary Amazon MSK resource that you can create in your account. The topics in this section describe how to perform common Amazon MSK operations. For a list of all the operations that you can perform on an MSK cluster, see the following:

- The AWS Management Console
- The Amazon MSK API Reference
- The Amazon MSK CLI Command Reference

Topics
- Creating an Amazon MSK Cluster (p. 15)
- Deleting an Amazon MSK Cluster (p. 18)
- Getting the Apache ZooKeeper Connection String for an Amazon MSK Cluster (p. 18)
- Getting the Bootstrap Brokers for an Amazon MSK Cluster (p. 20)
- Listing Amazon MSK Clusters (p. 20)
- Scaling Up Broker Storage (p. 21)
- Updating the Configuration of an Amazon MSK Cluster (p. 22)
- Expanding an Amazon MSK Cluster (p. 23)
- Monitoring an Amazon MSK Cluster (p. 25)
- Tagging an Amazon MSK Cluster (p. 36)

Creating an Amazon MSK Cluster

Before you can create an Amazon MSK cluster you need to have a VPC and set up subnets within that VPC. You need two subnets in two different Availability Zones in the following Regions: South America (São Paulo), Canada (Central), and US West (N. California). In all other Regions where Amazon MSK is available, you can specify either two or three subnets. Your subnets must all be in different Availability Zones. When you create a cluster, Amazon MSK distributes the broker nodes evenly over the subnets that you specify. For an example of how to set up a VPC and subnets for an MSK cluster, see ??? (p. 4) and ??? (p. 7).

Creating a Cluster Using the AWS Management Console

1. Open the Amazon MSK console at https://console.aws.amazon.com/msk/.
2. Choose Create cluster.
3. Specify a name for the cluster.
4. In the VPC list, choose the VPC you want to use for the cluster. You can also specify which version of Apache Kafka you want Amazon MSK to use to create the cluster.
5. Specify two subnets if you're using one of the following Regions: South America (São Paulo), Canada (Central), and US West (N. California). In other Regions where Amazon MSK is available, you can specify either two or three subnets. The subnets that you specify must be in different Availability Zones.
6. Choose the kind of configuration you want. For information about MSK configurations, see Configuration (p. 38).
7. Specify the type and number of brokers you want MSK to create in each Availability Zone. The minimum is one broker per Availability Zone and the maximum is five brokers per Availability Zone.

8. (Optional) Assign tags to your cluster. Tags are optional. For more information, see the section called “Tagging a Cluster” (p. 36).

9. You can adjust the storage volume per broker. After you create the cluster, you can increase the storage volume per broker but you can't decrease it.

10. Choose the settings you want for encrypting data in transit. By default, MSK encrypts data as it transits between brokers within a cluster. If you don't want to encrypt data as it transits between brokers, clear the check box labeled Enable encryption within the cluster.

11. Choose one of the three settings for encrypting data as it transits between clients and brokers. For more information, see the section called “Encryption in Transit” (p. 49).

12. Choose the kind of CMK that you want to use for encrypting data at rest. For more information, see the section called “Encryption at Rest” (p. 49).

13. If you want to authenticate the identity of clients, choose Enable TLS client authentication by selecting the box next to it. For more information about authentication, see the section called “Client Authentication” (p. 59).

14. Choose the monitoring level you want. This determines the set of metrics you get. For more information, see the section called “Monitoring a Cluster” (p. 25).

15. (Optional) Choose Advanced settings, and then choose Customize settings. You can specify one or more security groups that you want to give access to your cluster (for example, the security groups of client machines). If you specify security groups that were shared with you, you must ensure that you have permissions to them. Specifically, you need the ec2:DescribeSecurityGroups permission. For an example, see Amazon EC2: Allows Managing EC2 Security Groups Associated With a Specific VPC, Programmatically and in the Console.

16. Choose Create cluster.

Creating a Cluster Using the AWS CLI

1. Copy the following JSON and save it to a file. Name the file brokernodegroupinfo.json. Replace the subnet IDs in the JSON with the values that correspond to your subnets. These subnets must be in different Availability Zones. Replace "Security-Group-ID" with the ID of one or more security groups of the client VPC. Clients associated with these security groups get access to the cluster. If you specify security groups that were shared with you, you must ensure that you have permissions to them. Specifically, you need the ec2:DescribeSecurityGroups permission. For an example, see Amazon EC2: Allows Managing EC2 Security Groups Associated With a Specific VPC, Programmatically and in the Console. Finally, save the updated JSON file on the computer where you have the AWS CLI installed.

```json
{
    "InstanceType": "kafka.m5.large",
    "ClientSubnets": [
        "Subnet-1-ID",
        "Subnet-2-ID"
    ],
    "SecurityGroups": [
        "Security-Group-ID"
    ]
}
```

**Important**
Specify exactly two subnets if you are using one of the following Regions: South America (São Paulo), Canada (Central), and US West (N. California). For other Regions where Amazon MSK is available, you can specify either two or three subnets. The subnets that you specify
must be in distinct Availability Zones. When you create a cluster, Amazon MSK distributes
the broker nodes evenly across the subnets that you specify.

2. Run the following AWS CLI command in the directory where you saved the
brokernodegroupinfo.json file, replacing "Your-Cluster-Name" with a name of your
choice. For "Monitoring-Level", you can specify one of the following three values: DEFAULT,
PER_BROKER, or PER_TOPIC_PER_BROKER. For information about these three different levels of
monitoring, see ?? (p. 25). The enhanced-monitoring parameter is optional. If you don’t
specify it in the create-cluster command, you get the DEFAULT level of monitoring.

   aws kafka create-cluster --cluster-name "Your-Cluster-Name" --broker-node-group-info
   file://brokernodegroupinfo.json --kafka-version "2.2.1" --number-of-broker-nodes 3 --
   enhanced-monitoring "Monitoring-Level"

The output of the command looks like the following JSON:

   {
     "ClusterArn": "...",
     "ClusterName": "AWSKafkaTutorialCluster",
     "State": "CREATING"
   }

Note
   The create-cluster command might return an error stating that one or more subnets
   belong to unsupported Availability Zones. When this happens, the error indicates which
   Availability Zones are unsupported. Create subnets that don't use the unsupported
   Availability Zones and try the create-cluster command again.

3. Save the value of the ClusterArn key because you need it to perform other actions on your cluster.

Creating a Cluster with a Custom MSK Configuration
Using the AWS CLI

For information about custom MSK configurations and how to create them, see Configuration (p. 38).

1. Save the following JSON to a file, replacing configuration-arn with the ARN of the configuration
   that you want to use to create the cluster.

   {
     "Arn": configuration-arn,
     "Revision": 1
   }

2. Run the create-cluster command and use the configuration-info option to point to the
   JSON file you saved in the previous step. The following is an example.

   aws kafka create-cluster --cluster-name ExampleClusterName --broker-node-group-info
   file://brokernodegroupinfo.json --kafka-version "1.1.1" --number-of-broker-nodes 3 --
   enhanced-monitoring PER_TOPIC_PER_BROKER --configuration-info file://configuration.json

The following is an example of a successful response after running this command.

   {
     "ClusterArn": "arn:aws:kafka:us-east-1:123456789012:cluster/
      CustomConfigExampleCluster/abcd1234-abcd-dcba-4321-a1b2abcd9f9f-2",
   }
Creating a Cluster Using the API

To create a cluster using the API, see CreateCluster.

Deleting an Amazon MSK Cluster

Deleting a Cluster Using the AWS Management Console

1. Open the Amazon MSK console at https://console.aws.amazon.com/msk/.
2. Choose the MSK cluster that you want to delete by selecting the check box next to it.
3. Choose Delete, and then confirm deletion.

Deleting a Cluster Using the AWS CLI

Run the following command, replacing ClusterArn with the Amazon Resource Name (ARN) that you obtained when you created your cluster. If you don't have the ARN for your cluster, you can find it by listing all clusters. For more information, see the section called "Listing Clusters" (p. 20).

aws kafka delete-cluster --cluster-arn ClusterArn

Deleting a Cluster using the API

To delete a cluster using the API, see DeleteCluster.

Getting the Apache ZooKeeper Connection String for an Amazon MSK Cluster

Getting the Apache ZooKeeper Connection String Using the AWS Management Console

1. Open the Amazon MSK console at https://console.aws.amazon.com/msk/.
2. The table shows all the clusters for the current region under this account. Choose the name of a cluster to view its description.
3. On the Cluster summary page, choose View client information. This shows you the bootstrap servers, as well as the Apache ZooKeeper connection string.
Getting the Apache ZooKeeper Connection String Using the AWS CLI

1. If you don’t know the Amazon Resource Name (ARN) of your cluster, you can find it by listing all the clusters in your account. For more information, see the section called “Listing Clusters” (p. 20).

2. To get the Apache ZooKeeper connection string, along with other information about your cluster, run the following command, replacing \texttt{ClusterArn} with the ARN of your cluster.

\begin{verbatim}
aws kafka describe-cluster --cluster-arn \texttt{ClusterArn}
\end{verbatim}

The output of this \texttt{describe-cluster} command looks like the following JSON example.

\begin{verbatim}
{
   "ClusterInfo": { 
      "BrokerNodeGroupInfo": { 
         "BrokerAZDistribution": "DEFAULT", 
         "ClientSubnets": [ 
            "subnet-0123456789abcdef0", 
            "subnet-2468013579abcdef1", 
            "subnet-1357902468abcdef2" 
         ],  
         "InstanceType": "kafka.m5.large", 
         "StorageInfo": { 
            "EbsStorageInfo": { 
               "VolumeSize": 1000 
            }
         }
      },  
      "ClusterName": "testcluster", 
      "CreationTime": "2018-12-02T17:38:36.75Z", 
      "CurrentBrokerSoftwareInfo": { 
         "KafkaVersion": "2.2.1" 
      },  
      "CurrentVersion": "K13V1IB3VIYZZH", 
      "EncryptionInfo": { 
         "EncryptionAtRest": { 
            "DataVolumeKMSKeyId": "arn:aws:kms:us-east-1:555555555555:key/12345678-abcd-2345-ef01-abcd123456" 
         }
      },  
      "EnhancedMonitoring": "DEFAULT", 
      "NumberOfBrokerNodes": 3, 
      "State": "ACTIVE", 
      "ZookeeperConnectString": "10.0.1.101:2018,10.0.2.101:2018,10.0.3.101:2018"
   }
}
\end{verbatim}

The previous JSON example shows the \texttt{ZookeeperConnectString} key in the output of the \texttt{describe-cluster} command. Copy the value corresponding to this key and save it for when you need to create a topic on your cluster.

\textbf{Important}

Your Amazon MSK cluster must be in the \texttt{ACTIVE} state for you to be able to obtain the ZooKeeper connection string. When a cluster is still in the \texttt{CREATING} state, the output of the \texttt{describe-cluster} command doesn’t include \texttt{ZookeeperConnectString}. If this is the case, wait a few minutes and then run the \texttt{describe-cluster} again after your cluster reaches the \texttt{ACTIVE} state.
Getting the Apache ZooKeeper Connection String Using the API

To get the Apache ZooKeeper connection string using the API, see DescribeCluster.

Getting the Bootstrap Brokers for an Amazon MSK Cluster

Getting the Bootstrap Brokers Using the AWS Management Console

1. Open the Amazon MSK console at https://console.aws.amazon.com/msk/.
2. The table shows all the clusters for the current region under this account. Choose the name of a cluster to view its description.
3. On the Cluster summary page, choose View client information. This shows you the bootstrap servers, as well as the Apache ZooKeeper connection string.

Getting the Bootstrap Brokers Using the AWS CLI

Run the following command, replacing ClusterArn with the Amazon Resource Name (ARN) that you obtained when you created your cluster. If you don't have the ARN for your cluster, you can find it by listing all clusters. For more information, see the section called "Listing Clusters" (p. 20).

```
aws kafka get-bootstrap-brokers --cluster-arn ClusterArn
```

The output of this command looks like the following JSON example.

```
{
   "BootstrapBrokerStringTls": "b-3.exampleClusterName.abcde.c2.kafka.us-east-1.amazonaws.com:9094,b-1.exampleClusterName.abcde.c2.kafka.us-east-1.amazonaws.com:9094,b-2.exampleClusterName.abcde.c2.kafka.us-east-1.amazonaws.com:9094"
}
```

Getting the Bootstrap Brokers Using the API

To get the bootstrap brokers using the API, see GetBootstrapBrokers.

Listing Amazon MSK Clusters

Listing Clusters Using the AWS Management Console

1. Open the Amazon MSK console at https://console.aws.amazon.com/msk/.
2. The table shows all the clusters for the current region under this account. Choose the name of a cluster to view its details.

Listing Clusters Using the AWS CLI

Run the following command.

```
aws kafka list-clusters
```

Listing Clusters Using the API

To list clusters using the API, see ListClusters.

Scaling Up Broker Storage

You can increase the amount of EBS storage per broker. You can't decrease the storage.

Scaling Up Broker Storage Using the AWS Management Console

1. Open the Amazon MSK console at https://console.aws.amazon.com/msk/.
2. Choose the MSK cluster for which you want to update broker storage.
3. In the Storage section, choose Edit.
4. Specify the storage volume you want. You can only increase the amount of storage, you can't decrease it.
5. Choose Save changes.

Scaling Up Broker Storage Using the AWS CLI

Run the following command, replacing ClusterArn with the Amazon Resource Name (ARN) that you obtained when you created your cluster. If you don't have the ARN for your cluster, you can find it by listing all clusters. For more information, see the section called “Listing Clusters” (p. 20).

Replace Current-Cluster-Version with the current version of the cluster.

**Important**
Cluster versions aren't simple integers. You can obtain the current version by describing the cluster. An example version is KTVPDKIKX0DER.

The Target-Volume-in-GiB parameter represents the amount of storage that you want each broker to have. It is only possible to update the storage for all the brokers. You can't specify individual brokers for which to update storage. The value you specify for Target-Volume-in-GiB must be a whole number that is greater than 100 GiB. The storage per broker after the update operation can't exceed 16384 GiB.

```
```
Scaling Up Broker Storage Using the API

To update a broker storage using the API, see UpdateBrokerStorage.

Updating the Configuration of an Amazon MSK Cluster

For information about MSK configuration, including how to create a custom configuration, which properties you can update, and what happens when you update the configuration of an existing cluster, see Configuration (p. 38).

Updating the Configuration of a Cluster Using the AWS CLI

1. Copy the following JSON and save it to a file. Name the file configuration-info.json. Replace ConfigurationArn with the Amazon Resource Name (ARN) of the configuration that you want to use to update the cluster. The ARN string must be in quotes in the following JSON.

   ```json
   {
       "Arn": ConfigurationArn,
       "Revision": Configuration-Revision
   }
   ```

2. Run the following command, replacing ClusterArn with the ARN that you obtained when you created your cluster. If you don't have the ARN for your cluster, you can find it by listing all clusters. For more information, see the section called "Listing Clusters" (p. 20).

   Replace Path-to-Config-Info-File with the path to your configuration info file. If you named the file that you created in the previous step configuration-info.json and saved it in the current directory, then Path-to-Config-Info-File is configuration-info.json.

   Replace Current-Cluster-Version with the current version of the cluster.

   ```bash
   ```

   The following is an example of how to use this command:

   ```bash
   aws kafka update-cluster-configuration --cluster-arn arn:aws:kafka:us-east-1:0123456789012:cluster/exampleName/abcd1234-0123-abcd-5678-1234abcd-1 --configuration-info file://c:\users\tester\msk\configuration-info.json --current-version "K1X5R6FFK87"
   ```

   The output of this update-cluster-configuration command looks like the following JSON example.
3. To get the result of the update-cluster-configuration operation, run the following command, replacing `ClusterOperationArn` with the ARN that you obtained in the output of the update-cluster-configuration command.

```bash
aws kafka describe-cluster-operation --cluster-operation-arn ClusterOperationArn
```

The output of this describe-cluster-operation command looks like the following JSON example.

```json
{
   "ClusterOperationInfo": {
      "ClientRequestId": "982168a3-939f-11e9-8a62-538df00285db",
      "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
      "CreationTime": "2019-06-20T21:08:57.735Z",
      "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-abcd-4f7f-1234-9876543210ef",
      "OperationState": "UPDATE_COMPLETE",
      "OperationType": "UPDATE_CLUSTER_CONFIGURATION",
      "SourceClusterInfo": {},
      "TargetClusterInfo": {
         "ConfigurationInfo": {
            "Arn": "arn:aws:kafka:us-east-1:123456789012:configuration/ExampleConfigurationName/abcdabcd-abcd-1234-abcd-abcd123e8e8e-1",
            "Revision": 1
         }
      }
   }
}
```

In this output, `OperationType` is `UPDATE_CLUSTER_CONFIGURATION`. If `OperationState` has the value `UPDATE_IN_PROGRESS`, wait a while, then run the describe-cluster-operation command again.

### Updating the Configuration of a Cluster Using the API

To create a cluster using the API, see [UpdateClusterConfiguration](#).

### Expanding an Amazon MSK Cluster

Use this Amazon MSK operation when you want to increase the number of brokers in your MSK cluster.

**Important**

If you want to expand an MSK cluster, make sure to use this Amazon MSK operation. Don't try to add brokers to a cluster without using this operation.
Expanding a Cluster Using the AWS Management Console

1. Open the Amazon MSK console at https://console.aws.amazon.com/msk/.
2. Choose the MSK cluster whose number of brokers you want to increase.
3. On the cluster details page, choose the Edit button next to the Cluster-Level Broker Details heading.
4. Enter the number of brokers that you want the cluster to have per Availability Zone and then choose Save changes.

Expanding a Cluster Using the AWS CLI

1. Run the following command, replacing ClusterArn with the Amazon Resource Name (ARN) that you obtained when you created your cluster. If you don’t have the ARN for your cluster, you can find it by listing all clusters. For more information, see the section called “Listing Clusters” (p. 20).

   Replace Current-Cluster-Version with the current version of the cluster.

   Important
   Cluster versions aren’t simple integers. You can obtain the current version by describing the cluster. An example version is KTVPDKIKX0DER.

   The Target-Number-of-Brokers parameter represents the total number of broker nodes that you want the cluster to have when this operation completes successfully. The value you specify for Target-Number-of-Brokers must be a whole number that is greater than the current number of brokers in the cluster. It must also be a multiple of the number of Availability Zones.


   The output of this update-broker-count operation looks like the following JSON.

   ```json
   {
     "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
     "ClusterOperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-abcd-4f7f-1234-9876543210ef"
   }
   ```

2. To get the result of the update-broker-count operation, run the following command, replacing ClusterOperationArn with the ARN that you obtained in the output of the update-broker-count command.

   aws kafka describe-cluster-operation --cluster-operation-arn ClusterOperationArn

   The output of this describe-cluster-operation command looks like the following JSON example.

   ```json
   {
   ```
Expanding a Cluster Using the API

To increase the number of brokers in a cluster using the API, see UpdateBrokerCount.

Monitoring an Amazon MSK Cluster

Amazon MSK gathers Apache Kafka metrics and sends them to Amazon CloudWatch where you can view them. For more information about Apache Kafka metrics, including the ones that Amazon MSK surfaces, see Monitoring in the Apache Kafka documentation.

You can also monitor your MSK cluster with Prometheus, an open-source monitoring application. For information about Prometheus, see Overview in the Prometheus documentation. To learn how to monitor your cluster with Prometheus, see the section called “Open Monitoring with Prometheus” (p. 33).

Topics
- Amazon MSK Monitoring Levels for CloudWatch Metrics (p. 25)
- Amazon MSK Metrics for Monitoring with CloudWatch (p. 26)
- Viewing Amazon MSK Metrics Using CloudWatch (p. 31)
- Consumer-Lag Checking with Burrow (p. 31)
- Open Monitoring with Prometheus (p. 33)

Amazon MSK Monitoring Levels for CloudWatch Metrics

When creating an Amazon MSK cluster, you can set the enhancedMonitoring property to one of three monitoring levels: DEFAULT, PER_BROKER, or PER_TOPIC_PER_BROKER. The tables in the following section show all the metrics that Amazon MSK makes available starting at each monitoring level.
Amazon MSK Metrics for Monitoring with CloudWatch

Amazon MSK integrates with Amazon CloudWatch metrics so that you can collect, view, and analyze CloudWatch metrics for your Amazon MSK cluster. The metrics that you configure for your MSK cluster are automatically collected and pushed to CloudWatch. The following three tables show the metrics that become available at each of the three monitoring levels.

**DEFAULT Level Monitoring**

The metrics described in the following table are available at the DEFAULT monitoring level. They are free.

**Metrics available at the DEFAULT monitoring level**

<table>
<thead>
<tr>
<th>Name</th>
<th>When Visible</th>
<th>Dimensions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveControllerCount</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name</td>
<td>Only one controller per cluster should be active at any given time.</td>
</tr>
<tr>
<td>CpuIdle</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The percentage of CPU idle time.</td>
</tr>
<tr>
<td>CpuSystem</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The percentage of CPU in kernel space.</td>
</tr>
<tr>
<td>CpuUser</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The percentage of CPU in user space.</td>
</tr>
<tr>
<td>GlobalPartitionCount</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name</td>
<td>Total number of partitions across all brokers in the cluster.</td>
</tr>
<tr>
<td>GlobalTopicCount</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name</td>
<td>Total number of topics across all brokers in the cluster.</td>
</tr>
<tr>
<td>KafkaAppLogsDiskUsed</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The percentage of disk space used for application logs.</td>
</tr>
<tr>
<td>KafkaDataLogsDiskUsed</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The percentage of disk space used for data logs.</td>
</tr>
<tr>
<td>MemoryBuffered</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The size in bytes of buffered memory for the broker.</td>
</tr>
<tr>
<td>MemoryCached</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The size in bytes of cached memory for the broker.</td>
</tr>
<tr>
<td>Name</td>
<td>When Visible</td>
<td>Dimension</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------</td>
<td>-----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>MemoryFree</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The size in bytes of memory that is free and available for the broker.</td>
</tr>
<tr>
<td>MemoryUsed</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The size in bytes of memory that is in use for the broker.</td>
</tr>
<tr>
<td>NetworkRxDropped</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The number of dropped receive packages.</td>
</tr>
<tr>
<td>NetworkRxErrors</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The number of network receive errors for the broker.</td>
</tr>
<tr>
<td>NetworkRxPackets</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The number of packets received by the broker.</td>
</tr>
<tr>
<td>NetworkTxDropped</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The number of dropped transmit packages.</td>
</tr>
<tr>
<td>NetworkTxErrors</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The number of network transmit errors for the broker.</td>
</tr>
<tr>
<td>NetworkTxPackets</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The number of packets transmitted by the broker.</td>
</tr>
<tr>
<td>OfflinePartitionsCount</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name</td>
<td>Total number of partitions that are offline in the cluster.</td>
</tr>
<tr>
<td>RootDiskUsed</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The percentage of the root disk used by the broker.</td>
</tr>
<tr>
<td>SwapFree</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The size in bytes of swap memory that is available for the broker.</td>
</tr>
</tbody>
</table>
### Amazon Managed Streaming for Apache Kafka Developer Guide

Amazon MSK Metrics for Monitoring with CloudWatch

<table>
<thead>
<tr>
<th>Name</th>
<th>When Visible</th>
<th>Dimensions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SwapUsed</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>The size in bytes of swap memory that is in use for the broker.</td>
</tr>
<tr>
<td>ZooKeeperRequestLatencyMsMean</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>Mean latency in milliseconds for ZooKeeper requests from broker.</td>
</tr>
<tr>
<td>ZooKeeperSessionState</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>Cluster Name, Broker ID</td>
<td>Connection status of broker's ZooKeeper session which may be one of the following: NOT_CONNECTED: '0.0', ASSOCIATING: '0.1', CONNECTING: '0.5', CONNECTEDREADONLY: '0.8', CONNECTED: '1.0', CLOSED: '5.0', AUTH_FAILED: '10.0'.</td>
</tr>
</tbody>
</table>

### PER_BROKER Level Monitoring

When you set the monitoring level to PER_BROKER, you get the metrics described in the following table in addition to all the DEFAULT level metrics. You pay for the metrics in the following table, whereas the DEFAULT level metrics continue to be free. The metrics in this table have the following dimensions: Cluster Name, Broker ID.

#### Additional metrics that are available starting at the PER_BROKER monitoring level

<table>
<thead>
<tr>
<th>Name</th>
<th>When Visible</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesInPerSec</td>
<td>After you create a topic.</td>
<td>The number of bytes per second received from clients.</td>
</tr>
<tr>
<td>BytesOutPerSec</td>
<td>After you create a topic.</td>
<td>The number of bytes per second sent to clients.</td>
</tr>
<tr>
<td>FetchConsumerLocalTimeMsMean</td>
<td>After there's a producer/consumer.</td>
<td>The mean time in milliseconds that the consumer request is processed at the leader.</td>
</tr>
<tr>
<td>FetchConsumerRequestQueueTimeMsMean</td>
<td>After there's a producer/consumer.</td>
<td>The mean time in milliseconds that the consumer request waits in the request queue.</td>
</tr>
<tr>
<td>FetchConsumerResponseQueueTimeMsMean</td>
<td>After there's a producer/consumer.</td>
<td>The mean time in milliseconds that the consumer request waits in the response queue.</td>
</tr>
<tr>
<td>FetchConsumerResponseSendTimeMsMean</td>
<td>After there's a producer/consumer.</td>
<td>The mean time in milliseconds for the consumer to send a response.</td>
</tr>
<tr>
<td>FetchConsumerTotalTimeMsMean</td>
<td>After there's a producer/consumer.</td>
<td>The mean total time in milliseconds that consumers spend on fetching data from the broker.</td>
</tr>
<tr>
<td>Name</td>
<td>When Visible</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FetchFollowerLocalTimeMsMean</td>
<td>After there's a producer/consumer.</td>
<td>The mean time in milliseconds that the follower request is processed at the leader.</td>
</tr>
<tr>
<td>FetchFollowerRequestQueueTimeMsMean</td>
<td>After there's a producer/consumer.</td>
<td>The mean time in milliseconds that the follower request waits in the request queue.</td>
</tr>
<tr>
<td>FetchFollowerResponseQueueTimeMsMean</td>
<td>After there's a producer/consumer.</td>
<td>The mean time in milliseconds that the follower request waits in the response queue.</td>
</tr>
<tr>
<td>FetchFollowerResponseSendTimeMsMean</td>
<td>After there's a producer/consumer.</td>
<td>The mean time in milliseconds for the follower to send a response.</td>
</tr>
<tr>
<td>FetchFollowerTotalTimeMsMean</td>
<td>After there's a producer/consumer.</td>
<td>The mean total time in milliseconds that followers spend on fetching data from the broker.</td>
</tr>
<tr>
<td>FetchMessageConversionsPerSec</td>
<td>After you create a topic.</td>
<td>The number of fetch message conversions per second for the broker.</td>
</tr>
<tr>
<td>FetchThrottleByteRate</td>
<td>After bandwidth throttling is applied.</td>
<td>The number of throttled bytes per second.</td>
</tr>
<tr>
<td>FetchThrottleQueueSize</td>
<td>After bandwidth throttling is applied.</td>
<td>The number of messages in the throttle queue.</td>
</tr>
<tr>
<td>FetchThrottleTime</td>
<td>After bandwidth throttling is applied.</td>
<td>The average fetch throttle time in milliseconds.</td>
</tr>
<tr>
<td>LeaderCount</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>The number of leader replicas.</td>
</tr>
<tr>
<td>MessagesInPerSec</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>The number of incoming messages per second for the broker.</td>
</tr>
<tr>
<td>NetworkProcessorAvgIdlePercent</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>The average percentage of the time the network processors are idle.</td>
</tr>
<tr>
<td>PartitionCount</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>The number of partitions for the broker.</td>
</tr>
<tr>
<td>ProduceLocalTimeMsMean</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>The mean time in milliseconds for the follower to send a response.</td>
</tr>
<tr>
<td>ProduceMessageConversionsPerSec</td>
<td>After you create a topic.</td>
<td>The number of produce message conversions per second for the broker.</td>
</tr>
<tr>
<td>ProduceMessageConversionsTimeMsMean</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>The mean time in milliseconds spent on message format conversions.</td>
</tr>
<tr>
<td>ProduceRequestQueueTimeMsMean</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>The mean time in milliseconds that request messages spend in the queue.</td>
</tr>
<tr>
<td>ProduceResponseQueueTimeMsMean</td>
<td>After the cluster gets to the ACTIVE state.</td>
<td>The mean time in milliseconds that response messages spend in the queue.</td>
</tr>
</tbody>
</table>
### Name | When Visible | Description
---|---|---
ProduceResponseSendTimeMsMean | After the cluster gets to the ACTIVE state. | The mean time in milliseconds spent on sending response messages.
ProduceThrottleByteRate | After bandwidth throttling is applied. | The number of throttled bytes per second.
ProduceThrottleQueueSize | After bandwidth throttling is applied. | The number of messages in the throttle queue.
ProduceThrottleTime | After bandwidth throttling is applied. | The average produce throttle time in milliseconds.
ProduceTotalTimeMsMean | After the cluster gets to the ACTIVE state. | The mean produce time in milliseconds.
RequestBytesMean | After the cluster gets to the ACTIVE state. | The mean number of request bytes for the broker.
RequestExemptFromThrottleTime | After request throttling is applied. | The average time in milliseconds spent in broker network and I/O threads to process requests that are exempt from throttling.
RequestHandlerAvgIdlePercent | After the cluster gets to the ACTIVE state. | The average percentage of the time the request handler threads are idle.
RequestThrottleQueueSize | After request throttling is applied. | The number of messages in the throttle queue.
RequestThrottleTime | After request throttling is applied. | The average request throttle time in milliseconds.
RequestTime | After request throttling is applied. | The average time in milliseconds spent in broker network and I/O threads to process requests.
UnderMinIsrPartitionCount | After the cluster gets to the ACTIVE state. | The number of under minIsr partitions for the broker.
UnderReplicatedPartitions | After the cluster gets to the ACTIVE state. | The number of under-replicated partitions for the broker.

**PER_TOPIC_PER_BROKER Level Monitoring**

When you set the monitoring level to `PER_TOPIC_PER_BROKER`, you get the metrics described in the following table, in addition to all the metrics from the `PER_BROKER` and `DEFAULT` levels. Only the `DEFAULT` level metrics are free. The metrics in this table have the following dimensions: Cluster Name, Broker ID, Topic.

**Additional metrics that are available starting at the PER_TOPIC_PER_BROKER monitoring level**

<table>
<thead>
<tr>
<th>Name</th>
<th>When Visible</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesInPerSec</td>
<td>After you create a topic.</td>
<td>The number of bytes received per second.</td>
</tr>
</tbody>
</table>
### Viewing Amazon MSK Metrics Using CloudWatch

You can monitor metrics for Amazon MSK using the CloudWatch console, the command line, or the CloudWatch API. The following procedures show you how to access metrics using these different methods.

**To access metrics using the CloudWatch console**

Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

1. In the navigation pane, choose **Metrics**.
2. Choose the **All metrics** tab, and then choose **AWS/Kafka**.
3. To view topic-level metrics, choose **Topic, Broker ID, Cluster Name**; for broker-level metrics, choose **Broker ID, Cluster Name**; and for cluster-level metrics, choose **Cluster Name**.
4. (Optional) In the graph pane, select a statistic and a time period, and then create a CloudWatch alarm using these settings.

**To access metrics using the AWS CLI**

Use the `list-metrics` and `get-metric-statistics` commands.

**To access metrics using the CloudWatch CLI**

Use the `mon-list-metrics` and `mon-get-stats` commands.

**To access metrics using the CloudWatch API**

Use the `ListMetrics` and `GetMetricStatistics` operations.

### Consumer-Lag Checking with Burrow

Burrow is a monitoring companion for Apache Kafka that provides consumer-lag checking. Burrow has a modular design that includes the following subsystems:

- Clusters run an Apache Kafka client that periodically updates topic lists and the current HEAD offset (the most recent offset) for every partition.
- Consumers fetch information about consumer groups from a repository. This repository can be an Apache Kafka cluster (consuming the `__consumer_offsets` topic), ZooKeeper, or some other repository.
- The storage subsystem stores all of this information in Burrow.

<table>
<thead>
<tr>
<th>Name</th>
<th>When Visible</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesOutPerSec</td>
<td>After you create a topic.</td>
<td>The number of bytes sent per second.</td>
</tr>
<tr>
<td>FetchMessageConversionsPerSec</td>
<td>After you create a topic.</td>
<td>The number of fetched messages converted per second.</td>
</tr>
<tr>
<td>MessagesInPerSec</td>
<td>After you create a topic.</td>
<td>The number of messages received per second.</td>
</tr>
<tr>
<td>ProduceMessageConversionsPerSec</td>
<td>After you create a topic.</td>
<td>The number of conversions per second for produced messages.</td>
</tr>
</tbody>
</table>
• The evaluator subsystem retrieves information from the storage subsystem for a specific consumer group and calculates the status of that group. This follows the consumer lag evaluation rules.

• The notifier subsystem requests status on consumer groups according to a configured interval and sends out notifications (Email, HTTP, or some other method) for groups that meet the configured criteria.

• The HTTP Server subsystem provides an API interface to Burrow for fetching information about clusters and consumers.

For more information about Burrow, see Burrow - Kafka Consumer Lag Checking.

**Important**

Make sure that Burrow is compatible with the version of Apache Kafka that you are using for your MSK cluster.

**To set up and use Burrow with Amazon MSK**

1. Create an MSK cluster and launch a client machine in the same VPC as the cluster. For example, you can follow the instructions at [Getting Started](#) (p. 4).

2. Run the following command on the EC2 instance that serves as your client machine.

   ```
   sudo yum install go
   ```

3. Run the following command on the client machine to get the Burrow project t.

   ```
   go get github.com/linkedin/Burrow
   ```

4. Run the following command to install dep. It installs it in the `/home/ec2-user/go/bin/dep` folder.

   ```
   curl https://raw.githubusercontent.com/golang/dep/master/install.sh | sh
   ```

5. Go to the `/home/ec2-user/go/src/github.com/linkedin/Burrow` folder and run the following command.

   ```
   /home/ec2-user/go/bin/dep ensure
   ```

6. Run the following command in the same folder.

   ```
   go install
   ```

7. Open the `/home/ec2-user/go/src/github.com/linkedin/Burrow/config/burrow.toml` configuration file for editing. In the following sections of the configuration file, replace the placeholders with the name of your MSK cluster, the host:port pairs for your ZooKeeper servers, and your bootstrap brokers.

   To get your ZooKeeper host:port pairs, describe your MSK cluster and look for the value of `ZookeeperConnectString`. See the section called “Getting the Apache ZooKeeper Connection String” (p. 18).

   To get your bootstrap brokers, see the section called “Getting the Bootstrap Brokers” (p. 20).

   Follow the formatting shown below when you edit the configuration file.

   ```
   [zookeeper]
   servers=[ "ZooKeeper-host-port-pair-1", "ZooKeeper-host-port-pair-2", "ZooKeeper-host-port-pair-3" ]
   timeout=6
   ```
Open Monitoring with Prometheus

You can monitor your MSK cluster with Prometheus, an open-source monitoring system for time-series metric data. You can also use tools that are compatible with Prometheus-formatted metrics or tools that integrate with Amazon MSK Open Monitoring, like Datadog, Lenses, New Relic, and Sumo logic. Open monitoring is available for free but charges apply for the transfer of data across Availability Zones. For information about Prometheus, see the Prometheus documentation.

Creating an Amazon MSK Cluster with Open Monitoring Enabled

Using the AWS Management Console

1. Sign in to the AWS Management Console, and open the Amazon MSK console at https://console.aws.amazon.com/msk/home?region=us-east-1#/home/.
2. In the Monitoring section, select the check box next to Enable open monitoring with Prometheus.
3. Provide the required information in all the sections of the page, and review all the available options.
4. Choose Create cluster.

Using the AWS CLI

- Invoke the create-cluster command and specify its open-monitoring option. Enable the JmxExporter, the NodeExporter, or both. If you specify open-monitoring, the two exporters can’t be disabled at the same time.

Using the API

- Invoke the CreateCluster operation and specify OpenMonitoring. Enable the jmxExporter, the nodeExporter, or both. If you specify OpenMonitoring, the two exporters can’t be disabled at the same time.

Enabling Open Monitoring for an Existing Amazon MSK Cluster

Using the AWS Management Console

1. Sign in to the AWS Management Console, and open the Amazon MSK console at https://console.aws.amazon.com/msk/home?region=us-east-1#/home/.
2. Choose the name of the cluster that you want to update. This takes you to the Details page for the cluster.
3. On the Details tab, scroll down to find the Monitoring section.
4. Choose Edit.
5. Select the check box next to Enable open monitoring with Prometheus.
6. Choose Save changes.

Using the AWS CLI

- Invoke the update-monitoring command and specify its open-monitoring option. Enable the JmxExporter, the NodeExporter, or both. If you specify open-monitoring, the two exporters can’t be disabled at the same time.

Using the API

- Invoke the UpdateMonitoring operation and specify OpenMonitoring. Enable the jmxExporter, the nodeExporter, or both. If you specify OpenMonitoring, the two exporters can’t be disabled at the same time.

Setting Up a Prometheus Host on an Amazon EC2 Instance

1. Download the Prometheus server from https://prometheus.io/download/#prometheus to your Amazon EC2 instance.
2. Extract the downloaded file to a directory and go to that directory.
3. Create a file with the following contents and name it prometheus.yml.

```
# file: prometheus.yml
# my global config
global:
```
scrape_interval: 10s

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from
  # this config.
  - job_name: 'prometheus'
    static_configs:
      # 9090 is the prometheus server port
      - targets: ['localhost:9090']
  - job_name: 'broker'
    file_sd_configs:
      - files:
          - 'targets.json'

4. Use the `ListNodes` operation to get a list of your cluster's brokers.

5. Create a file named `targets.json` with the following JSON. Replace `broker_dns_1`, `broker_dns_2`, and the rest of the broker DNS names with the DNS names you obtained for your brokers in the previous step. Include all of the brokers you obtained in the previous step. Amazon MSK uses port 11001 for the JMX Exporter and port 11002 for the Node Exporter.

```json
[
  {
    "labels": {
      "job": "jmx"
    },
    "targets": [
      "broker_dns_1:11001",
      "broker_dns_2:11001",
      ...
      "broker_dns_N:11001"
    ]
  },
  {
    "labels": {
      "job": "node"
    },
    "targets": [
      "broker_dns_1:11002",
      "broker_dns_2:11002",
      ...
      "broker_dns_N:11002"
    ]
  }
]
```

6. To start the Prometheus server on your Amazon EC2 instance, run the following command in the directory where you extracted the Prometheus files and saved `prometheus.yml` and `targets.json`.

```
./prometheus
```

7. Find the IPv4 public IP address of the Amazon EC2 instance where you ran Prometheus in the previous step. You need this public IP address in the following step.

8. To access the Prometheus web UI, open a browser that can access your Amazon EC2 instance, and go to `Prometheus-Instance-Public-IP:9090`, where `Prometheus-Instance-Public-IP` is the public IP address you got in the previous step.
Prometheus Metrics

All metrics emitted by Apache Kafka to JMX are accessible using open monitoring with Prometheus. For information about Apache Kafka metrics, see Monitoring in the Apache Kafka documentation. In addition, you can use the Prometheus Node Exporter to get CPU and disk metrics for the broker nodes.

Tagging an Amazon MSK Cluster

You can assign your own metadata in the form of tags to an Amazon MSK resource, such as an MSK cluster. A tag is a key-value pair that you define for the resource. Using tags is a simple yet powerful way to manage AWS resources and organize data, including billing data.

Topics
- Tag Basics (p. 36)
- Tracking Costs Using Tagging (p. 36)
- Tag Restrictions (p. 36)
- Tagging Resources Using the Amazon MSK API (p. 37)

Tag Basics

You can use the Amazon MSK API to complete the following tasks:

- Add tags to an Amazon MSK resource.
- List the tags for an Amazon MSK resource.
- Remove tags from an Amazon MSK resource.

You can use tags to categorize your Amazon MSK resources. For example, you can categorize your Amazon MSK clusters by purpose, owner, or environment. Because you define the key and value for each tag, you can create a custom set of categories to meet your specific needs. For example, you might define a set of tags that help you track clusters by owner and associated application.

The following are several examples of tags:

- Project: Project name
- Owner: Name
- Purpose: Load testing
- Environment: Production

Tracking Costs Using Tagging

You can use tags to categorize and track your AWS costs. When you apply tags to your AWS resources, including Amazon MSK clusters, your AWS cost allocation report includes usage and costs aggregated by tags. You can organize your costs across multiple services by applying tags that represent business categories (such as cost centers, application names, or owners). For more information, see Use Cost Allocation Tags for Custom Billing Reports in the AWS Billing and Cost Management User Guide.

Tag Restrictions

The following restrictions apply to tags in Amazon MSK.
Basic restrictions

• The maximum number of tags per resource is 50.
• Tag keys and values are case-sensitive.
• You can't change or edit tags for a deleted resource.

Tag key restrictions

• Each tag key must be unique. If you add a tag with a key that's already in use, your new tag overwrites the existing key-value pair.
• You can't start a tag key with `aws:` because this prefix is reserved for use by AWS. AWS creates tags that begin with this prefix on your behalf, but you can't edit or delete them.
• Tag keys must be between 1 and 128 Unicode characters in length.
• Tag keys must consist of the following characters: Unicode letters, digits, white space, and the following special characters: _ . / = + - @.

Tag value restrictions

• Tag values must be between 0 and 255 Unicode characters in length.
• Tag values can be blank. Otherwise, they must consist of the following characters: Unicode letters, digits, white space, and any of the following special characters: _ . / = + - @.

Tagging Resources Using the Amazon MSK API

You can use the following operations to tag or untag an Amazon MSK resource or to list the current set of tags for a resource:

• ListTagsForResource
• TagResource
• UntagResource
Amazon MSK Configuration

Amazon MSK provides a default configuration for brokers, topics, and Apache ZooKeeper nodes. You can also create custom configurations and use them to create new MSK clusters or to update existing clusters. An MSK configuration consists of a set of properties and their corresponding values.

Topics
- Custom MSK Configurations (p. 38)
- The Default Amazon MSK Configuration (p. 42)
- Amazon MSK Configuration Operations (p. 44)

Custom MSK Configurations

Amazon MSK enables you to create a custom MSK configuration where you set the following properties. Properties that you don't set explicitly get the values they have in the section called "Default Configuration" (p. 42). For more information about configuration properties, see Apache Kafka Configuration.

Apache Kafka Configuration Properties That You Can Set

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto.create.topics.enable</td>
<td>Enables topic autocreation on the server.</td>
</tr>
<tr>
<td>compression.type</td>
<td>The final compression type for a given topic. You can set this property to the standard compression codecs (gzip, snappy, lz4, and zstd). It additionally accepts uncompressed, which is equivalent to no compression; and producer, which means retain the original compression codec set by the producer.</td>
</tr>
<tr>
<td>default.replication.factor</td>
<td>The default replication factor for automatically created topics.</td>
</tr>
<tr>
<td>delete.topic.enable</td>
<td>Enables the delete topic operation. If this config is turned off, you can't delete a topic through the admin tool.</td>
</tr>
<tr>
<td>group.initial.rebalance.delay.ms</td>
<td>Amount of time the group coordinator waits for more consumers to join a new group before performing the first rebalance. A longer delay means potentially fewer rebalances, but increases the time until processing begins.</td>
</tr>
<tr>
<td>group.max.session.timeout.ms</td>
<td>Maximum session timeout for registered consumers. Longer timeouts give consumers more time to process messages in between heartbeats at the cost of a longer time to detect failures.</td>
</tr>
<tr>
<td>group.min.session.timeout.ms</td>
<td>Minimum session timeout for registered consumers. Shorter timeouts result in quicker failure detection at the cost of more frequent consumer heartbeating, which can overwhelm broker resources.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>log.cleaner.delete.retention.ms</td>
<td>Amount of time that you want Apache Kafka to retain deleted records.</td>
</tr>
<tr>
<td>log.cleaner.min.cleanable.ratio</td>
<td>Minimum ratio of dirty log to total log for a log to be eligible for cleaning. A higher ratio means fewer, more efficient cleanings, but it also means more wasted space in the log.</td>
</tr>
<tr>
<td>log.cleanup.policy</td>
<td>The default cleanup policy for segments beyond the retention window. A comma-separated list of valid policies. Valid policies are <code>delete</code> and <code>compact</code>.</td>
</tr>
<tr>
<td>log.flush.interval.messages</td>
<td>Number of messages accumulated on a log partition before messages are flushed to disk.</td>
</tr>
<tr>
<td>log.flush.interval.ms</td>
<td>Maximum time in ms that a message in any topic is kept in memory before flushed to disk. If not set, the value in log.flush.scheduler.interval.ms is used.</td>
</tr>
<tr>
<td>log.retention.bytes</td>
<td>Maximum size of the log before deleting it.</td>
</tr>
<tr>
<td>log.retention.hours</td>
<td>Number of hours to keep a log file before deleting it, tertiary to the log.retention.ms property.</td>
</tr>
<tr>
<td>log.retention.minutes</td>
<td>Number of minutes to keep a log file before deleting it, secondary to log.retention.ms property. If not set, the value in log.retention.hours is used.</td>
</tr>
<tr>
<td>log.retention.ms</td>
<td>Number of milliseconds to keep a log file before deleting it (in milliseconds). If not set, the value in log.retention.minutes is used.</td>
</tr>
<tr>
<td>log.roll.ms</td>
<td>Maximum time before a new log segment is rolled out (in milliseconds). If not set, the value in log.roll.hours is used.</td>
</tr>
<tr>
<td>log.segment.bytes</td>
<td>Maximum size of a single log file.</td>
</tr>
<tr>
<td>max.incremental.fetch.session.cache.slots</td>
<td>Maximum number of incremental fetch sessions that are maintained.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>message.max.bytes</td>
<td>Largest record batch size allowed by Kafka. If this is increased and there are consumers older than 0.10.2, the consumers' fetch size must also be increased so that the they can fetch record batches this large.</td>
</tr>
<tr>
<td></td>
<td>In the latest message format version, records are always grouped into batches for efficiency. In previous message format versions, uncompressed records are not grouped into batches and this limit only applies to a single record in that case.</td>
</tr>
<tr>
<td></td>
<td>This can be set per topic with the topic level max.message.bytes config.</td>
</tr>
<tr>
<td>message.timestamp.difference.max.ms</td>
<td>The maximum difference allowed between the timestamp when a broker receives a message and the timestamp specified in the message. If message.timestamp.type=CreateTime, a message is rejected if the difference in timestamp exceeds this threshold. This configuration is ignored if message.timestamp.type=LogAppendTime.</td>
</tr>
<tr>
<td>message.timestamp.type</td>
<td>Specifies whether the timestamp in the message is the message creation time or the log append time. The allowed values are CreateTime and LogAppendTime.</td>
</tr>
<tr>
<td>min.insync.replicas</td>
<td>When a producer sets acks to &quot;all&quot; (or &quot;-1&quot;), min.insync.replicas specifies the minimum number of replicas that must acknowledge a write for the write to be considered successful. If this minimum cannot be met, the producer raises an exception (either NotEnoughReplicas or NotEnoughReplicasAfterAppend). When used together, min.insync.replicas and acks enable you to enforce greater durability guarantees. A typical scenario would be to create a topic with a replication factor of 3, set min.insync.replicas to 2, and produce with acks of &quot;all&quot;. This ensures that the producer raises an exception if a majority of replicas don't receive a write.</td>
</tr>
<tr>
<td>num.io.threads</td>
<td>The number of threads that the server uses for processing requests, which may include disk I/O.</td>
</tr>
<tr>
<td>num.network.threads</td>
<td>The number of threads that the server uses for receiving requests from the network and sending responses to it.</td>
</tr>
<tr>
<td>num.recovery.threads.per.data.dir</td>
<td>The number of threads per data directory to be used for log recovery at startup and for flushing at shutdown.</td>
</tr>
</tbody>
</table>
### Custom Configurations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>num.replica.fetchers</td>
<td>The number of fetcher threads used to replicate messages from a source broker. Increasing this value can increase the degree of I/O parallelism in the follower broker.</td>
</tr>
<tr>
<td>num.partitions</td>
<td>Default number of log partitions per topic.</td>
</tr>
<tr>
<td>offsets.retention.minutes</td>
<td>After a consumer group loses all its consumers (that is, it becomes empty) its offsets are kept for this retention period before getting discarded. For standalone consumers (that is, using manual assignment), offsets are expired after the time of the last commit plus this retention period.</td>
</tr>
<tr>
<td>offsets.topic.replication.factor</td>
<td>The replication factor for the offsets topic (set higher to ensure availability). Internal topic creation fails until the cluster size meets this replication factor requirement.</td>
</tr>
<tr>
<td>replica.fetch.max.bytes</td>
<td>Number of bytes of messages to attempt to fetch for each partition. This is not an absolute maximum. If the first record batch in the first non-empty partition of the fetch is larger than this value, the record batch is returned to ensure that progress can be made. The maximum record batch size accepted by the broker is defined via message.max.bytes (broker config) or max.message.bytes (topic config).</td>
</tr>
<tr>
<td>replica.fetch.response.max.bytes</td>
<td>The maximum number of bytes expected for the entire fetch response. Records are fetched in batches, and if the first record batch in the first non-empty partition of the fetch is larger than this value, the record batch will still be returned to ensure that progress can be made. This isn’t an absolute maximum. The message.max.bytes (broker config) or max.message.bytes (topic config) properties specify the maximum record batch size that the broker accepts.</td>
</tr>
<tr>
<td>socket.request.max.bytes</td>
<td>The maximum number of bytes in a socket request.</td>
</tr>
<tr>
<td>transaction.max.timeout.ms</td>
<td>Maximum timeout for transactions. If a client’s requested transaction time exceed this, the broker returns an error in InitProducerIdRequest. This prevents a client from too large of a timeout, which can stall consumers reading from topics included in the transaction.</td>
</tr>
<tr>
<td>transaction.state.log.min.isr</td>
<td>Overridden min.insync.replicas config for the transaction topic.</td>
</tr>
</tbody>
</table>
Dynamic Configuration

In addition to the configuration properties that Amazon MSK provides, you can dynamically set cluster- and broker-level configuration properties that don’t require a broker restart. You can dynamically set configuration properties that aren’t marked as read-only in the table under Broker Configs in the Apache Kafka documentation. For information about dynamic configuration and example commands, see Updating Broker Configs in the Apache Kafka documentation.

**Note**

You can set the advertised.listeners property, but not the listeners property.

**Topic-Level Configuration**

You can use Apache Kafka commands to set or modify topic-level configuration properties for new and existing topics. For more information about topic-level configuration properties and examples on how to set them, see Topic-Level Configs in the Apache Kafka documentation.

**The Default Amazon MSK Configuration**

When you create an MSK cluster without specifying a custom MSK configuration, Amazon MSK creates and uses a default configuration with the values shown in the following table. For properties that aren’t in this table, Amazon MSK uses the defaults associated with your version of Apache Kafka. For a list of these default values, see Apache Kafka Configuration.

### Default Configuration Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>allow.everyone.if.no.acl.found</td>
<td>If no resource patterns match a specific resource, the resource has no associated ACLs. In this</td>
<td>true</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Default Value</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>name, if this property is set to true, everyone is allowed to access the resource, not just the super users.</td>
<td>auto.create.topics.enable Enables autocreation of a topic on the server.</td>
<td>false</td>
</tr>
<tr>
<td>auto.leader.rebalance.enable Enables auto leader balancing. A background thread checks and triggers leader balance if required at regular intervals.</td>
<td>auto.leader.rebalance.enable Enables auto leader balancing. A background thread checks and triggers leader balance if required at regular intervals.</td>
<td>true</td>
</tr>
<tr>
<td>default.replication.factor Default replication factors for automatically created topics.</td>
<td>default.replication.factor Default replication factors for automatically created topics.</td>
<td>3</td>
</tr>
<tr>
<td>min.insync.replicas When a producer sets acks to &quot;all&quot; (or &quot;-1&quot;), min.insync.replicas specifies the minimum number of replicas that must acknowledge a write for the write to be considered successful. If this minimum can't be met, the producer raises an exception (either NotEnoughReplicas or NotEnoughReplicasAfterAppend). When used together, min.insync.replicas and acks enable you to enforce greater durability guarantees. A typical scenario would be to create a topic with a replication factor of 3, set min.insync.replicas to 2, and produce with acks of &quot;all&quot;. This ensures that the producer raises an exception if a majority of replicas do not receive a write.</td>
<td>min.insync.replicas When a producer sets acks to &quot;all&quot; (or &quot;-1&quot;), min.insync.replicas specifies the minimum number of replicas that must acknowledge a write for the write to be considered successful. If this minimum can't be met, the producer raises an exception (either NotEnoughReplicas or NotEnoughReplicasAfterAppend). When used together, min.insync.replicas and acks enable you to enforce greater durability guarantees. A typical scenario would be to create a topic with a replication factor of 3, set min.insync.replicas to 2, and produce with acks of &quot;all&quot;. This ensures that the producer raises an exception if a majority of replicas do not receive a write.</td>
<td>2</td>
</tr>
<tr>
<td>num.io.threads Number of threads that the server uses for processing requests, which may include disk I/O.</td>
<td>num.io.threads Number of threads that the server uses for processing requests, which may include disk I/O.</td>
<td>8</td>
</tr>
<tr>
<td>num.network.threads Number of threads that the server uses for receiving requests from the network and sending responses to the network.</td>
<td>num.network.threads Number of threads that the server uses for receiving requests from the network and sending responses to the network.</td>
<td>5</td>
</tr>
<tr>
<td>num.partitions Default number of log partitions per topic.</td>
<td>num.partitions Default number of log partitions per topic.</td>
<td>1</td>
</tr>
</tbody>
</table>
### Amazon Managed Streaming for Apache Kafka Developer Guide

#### Configuration Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>num.replica.fetchers</td>
<td>Number of fetcher threads used to replicate messages from a source broker. Increasing this value can increase the degree of I/O parallelism in the follower broker.</td>
<td>2</td>
</tr>
<tr>
<td>socket.receive.buffer.bytes</td>
<td>SO_RCVBUF buffer of the socket sever sockets. If the value is -1, the OS default is used.</td>
<td>102400</td>
</tr>
<tr>
<td>socket.request.max.bytes</td>
<td>Maximum number of bytes in a socket request.</td>
<td>104857600</td>
</tr>
<tr>
<td>socket.send.buffer.bytes</td>
<td>SO_SNDBUF buffer of the socket sever sockets. If the value is -1, the OS default is used.</td>
<td>102400</td>
</tr>
<tr>
<td>unclean.leader.election.enable</td>
<td>Indicates whether to enable replicas not in the ISR set to be elected as leader as a last resort, even though doing so may result in data loss.</td>
<td>true</td>
</tr>
<tr>
<td>zookeeper.set.acl</td>
<td>Set client to use secure ACLs.</td>
<td>false</td>
</tr>
</tbody>
</table>

---

### Amazon MSK Configuration Operations

This topic describes how to create custom MSK configurations and how to perform operations on them. It isn't currently possible to delete a custom configuration. For information about how to use MSK configurations to create or update clusters, see *How It Works* (p. 15).

#### To create an MSK configuration

1. Create a file where you specify the configuration properties that you want to set and the values that you want to assign to them. The following are the contents of an example configuration file.

   ```
   auto.create.topics.enable = true
   zookeeper.connection.timeout.ms = 1000
   log.roll.ms = 604800000
   ```

2. Run the following AWS CLI command, replacing `config-file-path` with the path to the file where you saved your configuration in the previous step.

   ```
   aws kafka create-configuration --name "ExampleConfigurationName" --description "Example configuration description." --kafka-versions "1.1.1" --server-properties file://config-file-path
   ```

   The following is an example of a successful response after running this command.
3. The previous command returns an Amazon Resource Name (ARN) for the newly created configuration. Save this ARN because you need it to refer to this configuration in other commands. If you lose your configuration ARN, you can find it again by listing all the configurations in your account.

To describe an MSK configuration

- This command returns metadata about the configuration. To get a detailed description of the configuration, run the `describe-configuration-revision`.

To run this example, replace `configuration-arn` with the ARN you obtained when you created the configuration. If you didn’t save the ARN when you created the configuration, you can use the `list-configurations` command to list all configuration in your account, and find the configuration that you want in the list that appears in the response. The ARN of the configuration also appears in that list.

```bash
aws kafka describe-configuration --arn configuration-arn
```

The following is an example of a successful response after running this command.

```json
{
  "Name": "SomeTest",
  "KafkaVersions": ["1.1.1"],
  "CreationTime": "2019-05-21T00:54:23.591Z",
  "Description": "Example configuration description."
}
```

To describe an MSK configuration revision

Describing an MSK configuration using the `describe-configuration-revision` command, gives you the metadata of the configuration. To see a description of the configuration, use this command, `describe-configuration-revision`, instead.

- Run the following command, replacing `configuration-arn` with the ARN you obtained when you created the configuration. If you didn’t save the ARN when you created the configuration, you can use the `list-configurations` command to list all configuration in your account, and find the
configuration that you want in the list that appears in the response. The ARN of the configuration also appears in that list.

```
aws kafka describe-configuration-revision --arn configuration-arn --revision 1
```

The following is an example of a successful response after running this command.

```
{
    "CreationTime": "2019-05-21T00:54:23.591Z",
    "Description": "Example configuration description.",
    "Revision": 1,
    "ServerProperties": "YXV0by5jcmVhdGUudG9waWNzLmVuYWJsZS91c2VyZ29tcG9ydWdpc3Rlci5Gb3VuZGVyb20ucmVjdCBhbmQtdmFsdWVz",
}
```

The value of ServerProperties is encoded using Base64. If you use a Base64 decoder (for example, https://www.base64decode.org/) to manually decode it, you get the contents of the original configuration file that you used to create the custom configuration. In this case, you get the following:

```
auto.create.topics.enable = true
zookeeper.connection.timeout.ms = 1000
log.roll.ms = 604800000
```

To list all MSK configurations in your account for the current Region

- Run the following command.

```
aws kafka list-configurations
```

The following is an example of a successful response after running this command.

```
{
    "Configurations": [
        {
            "CreationTime": "2019-05-21T00:54:23.591Z",
            "Description": "Example configuration description.",
            "KafkaVersions": [
                "1.1.1"
            ],
            "LatestRevision": {
                "CreationTime": "2019-05-21T00:54:23.591Z",
                "Description": "Example configuration description.",
                "Revision": 1
            },
            "Name": "SomeTest"
        },
        {
            "Arn": "arn:aws:kafka:us-east-1:123456789012:configuration/SomeTest/abcdabcd-1234-abcd-1234-abcd123e8e8e-1",
            "CreationTime": "2019-05-03T23:08:29.446Z",
            "Description": "Example configuration description.",
        }
    ]
}
```
"KafkaVersions": [
  "1.1.1"
],
"LatestRevision": {
  "CreationTime": "2019-05-03T23:08:29.446Z",
  "Description": "Example configuration description.",
  "Revision": 1
},
"Name": "ExampleConfigurationName"
]
Data Protection in Amazon Managed Streaming for Apache Kafka

Amazon Managed Streaming for Apache Kafka conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:
• Use multi-factor authentication (MFA) with each account.
• Use SSL/TLS to communicate with AWS resources.

• Set up API and user activity logging with AWS CloudTrail.
• Use AWS encryption solutions, along with all default security controls within AWS services.
• Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

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

For more information about data protection, see the [AWS Shared Responsibility Model and GDPR blog post on the AWS Security Blog](https://aws.amazon.com/security/gdpr/).

**Topics**
- Amazon MSK Encryption (p. 49)
- How Do I Get Started with Encryption? (p. 50)

**Amazon MSK Encryption**

Amazon MSK provides data encryption options that you can use to meet strict data management requirements.

**Encryption at Rest**

Amazon MSK integrates with [AWS Key Management Service (KMS)](https://aws.amazon.com/kms/) to offer transparent server-side encryption. Amazon MSK always encrypts your data at rest. When you create a MSK cluster, you can specify the AWS KMS customer master key (CMK) that you want Amazon MSK to use to encrypt your data at rest. If you don't specify a CMK, Amazon MSK creates an **AWS managed CMK** for you and uses it on your behalf. For more information about CMKs, see [Customer Master Keys (CMKs)](https://docs.aws.amazon.com/kms/latest/developerguide/customer-master-key.html) in the [AWS Key Management Service Developer Guide](https://docs.aws.amazon.com/kms/latest/developerguide/).

**Encryption in Transit**

Amazon MSK uses TLS 1.2. By default, it encrypts data in transit between the brokers of your MSK cluster. You can override this default at the time you create the cluster.

For communication between clients and brokers, you must specify one of the following three settings:

• Only allow TLS encrypted data. This is the default setting.
• Allow both plaintext, as well as TLS encrypted data.
• Only allow plaintext data.

Amazon MSK brokers use public AWS Certificate Manager certificates. Therefore, any truststore that trusts Amazon Trust Services also trusts the certificates of Amazon MSK brokers.

Enabling encryption reduces performance by approximately 30%. However, the exact percentage depends on the configuration of your cluster and clients.
How Do I Get Started with Encryption?

When creating an MSK cluster, you can specify encryption settings in JSON format. The following is an example.

```
{
    "EncryptionAtRest": {
        "DataVolumeKMSKeyId": "arn:aws:kms:us-east-1:123456789012:key/abcdabcd-1234-abcd-1234-abcd123e8e8e",
    },
    "EncryptionInTransit": {
        "InCluster": true,
        "ClientBroker": "TLS"
    }
}
```

For `DataVolumeKMSKeyId`, you can specify a customer managed CMK or the AWS managed CMK for MSK in your account (alias/aws/kafka). If you don't specify `EncryptionAtRest`, Amazon MSK still encrypts your data at rest under the AWS managed CMK. To determine which CMK your cluster is using, send a GET request or invoke the `DescribeCluster` API operation.

For `EncryptionInTransit`, the default value of `InCluster` is true, but you can set it to false if you don't want Amazon MSK to encrypt your data as it passes between brokers.

To specify the encryption mode for data in transit between clients and brokers, set `ClientBroker` to one of three values: `TLS`, `TLS_PLAINTEXT`, or `PLAINTEXT`.

**To specify encryption settings when creating a cluster**

1. Save the contents of the previous example in a file and give the file any name that you want. For example, call it `encryption-settings.json`.
2. Run the `create-cluster` command and use the `encryption-info` option to point to the file where you saved your configuration JSON. The following is an example.

   ```
   aws kafka create-cluster --cluster-name "ExampleClusterName" --broker-node-group-info file://brokernodegroupinfo.json --encryption-info file://encryptioninfo.json --kafka-version "2.2.1" --number-of-broker-nodes 3
   ```

   The following is an example of a successful response after running this command.

   ```
   {
       "ClusterArn": "arn:aws:kafka:us-east-1:123456789012:cluster/SecondTLSTest/abcdabcd-1234-abcd-1234-abcd123e8e8e",
       "ClusterName": "ExampleClusterName",
       "State": "CREATING"
   }
   ```

**To test TLS encryption**

1. Create a client machine following the guidance in the section called “Step 4: Create a Client Machine” (p. 10).
2. Install Apache Kafka on the client machine.
3. Run the following command on a machine that has the AWS CLI installed, replacing `clusterARN` with the ARN of your cluster (a cluster created with `ClientBroker` set to TLS like the example in the previous procedure).
How Do I Get Started with Encryption?

1. Run the following command to describe the cluster:
   ```bash
   aws kafka describe-cluster --cluster-arn clusterARN
   ```

2. In the result, look for the value of `ZookeeperConnectString` and save it because you need it in the next step.

3. Go to the `bin` folder of the Apache Kafka installation on the client machine. To create a topic, run the following command, replacing `ZookeeperConnectString` with the value you obtained for `ZookeeperConnectString` in the previous step.
   ```bash
   kafka-topics.sh --create --zookeeper ZookeeperConnectString --replication-factor 3 --partitions 1 --topic TLSTestTopic
   ```

4. In this example we use the JVM truststore to talk to the MSK cluster. To do this, first create a folder named `/tmp` on the client machine. Then, go to the `bin` folder of the Apache Kafka installation, and run the following command. (Your JVM path might be different.)
   ```bash
   cp /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.201.b09-0.amzn2.x86_64/jre/lib/security/cacerts /tmp/kafka.client.truststore.jks
   ```

5. While still in the `bin` folder of the Apache Kafka installation on the client machine, create a text file named `client.properties` with the following contents.
   ```properties
   security.protocol=SSL
   ssl.truststore.location=/tmp/kafka.client.truststore.jks
   ```

6. Run the following command on a machine that has the AWS CLI installed, replacing `clusterARN` with the ARN of your cluster.
   ```bash
   aws kafka get-bootstrap-brokers --cluster-arn clusterARN
   ```

   A successful result looks like the following. Save this result because you need it for the next step.
   ```json
   {
     "BootstrapBrokerStringTls": "a-1.example.g7oein.c2.kafka.us-east-1.amazonaws.com:0123,a-3.example.g7oein.c2.kafka.us-east-1.amazonaws.com:0123,a-2.example.g7oein.c2.kafka.us-east-1.amazonaws.com:0123"
   }
   ```

7. In the `bin` folder of the Apache Kafka installation on the client machine, run the following, replacing `BootstrapBrokerStringTls` with the value you obtained in the previous step. Leave this producer command running.
   ```bash
   kafka-console-producer.sh --broker-list BootstrapBrokerStringTls --producer.config client.properties --topic TLSTestTopic
   ```

8. Open a new command window on the same client machine, go to the `bin` folder of the Apache Kafka installation, and run the following command to create a consumer.
   ```bash
   kafka-console-consumer.sh --bootstrap-server BootstrapBrokerStringTls --consumer.config client.properties --topic TLSTestTopic
   ```

9. In the producer window, type a text message followed by a return, and look for the same message in the consumer window. Amazon MSK encrypted this message in transit.

For more information about configuring Apache Kafka clients to work with encrypted data, see [Configuring Kafka Clients](#).
Identity and Access Management for Amazon Managed Streaming for Apache Kafka

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use Amazon MSK resources. IAM is an AWS service that you can use with no additional charge.

Topics
- How Amazon Managed Streaming for Apache Kafka Works with IAM (p. 52)
- Amazon Managed Streaming for Apache Kafka Identity-Based Policy Examples (p. 54)
- Using Service-Linked Roles for Amazon MSK (p. 57)
- Troubleshooting Amazon Managed Streaming for Apache Kafka Identity and Access (p. 58)

How Amazon Managed Streaming for Apache Kafka Works with IAM

Before you use IAM to manage access to Amazon MSK, you should understand what IAM features are available to use with Amazon MSK. To get a high-level view of how Amazon MSK and other AWS services work with IAM, see AWS Services That Work with IAM in the IAM User Guide.

Topics
- Amazon MSK Identity-Based Policies (p. 52)
- Amazon MSK Resource-Based Policies (p. 54)
- Authorization Based on Amazon MSK Tags (p. 54)
- Amazon MSK IAM Roles (p. 54)

Amazon MSK Identity-Based Policies

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. Amazon MSK supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the IAM User Guide.

Actions

The Action element of an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions usually have the same name as the associated AWS API operation. The action is used in a policy to grant permissions to perform the associated operation.

Policy actions in Amazon MSK use the following prefix before the action: kafka:. For example, to grant someone permission to describe an MSK cluster with the Amazon MSK DescribeCluster API operation, you include the kafka:DescribeCluster action in their policy. Policy statements must include either an Action or NotAction element. Amazon MSK defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows:

"Action": ["kafka:action1", "kafka:action2"]
You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word Describe, include the following action:

```
"Action": "kafka:Describe*"
```

To see a list of Amazon MSK actions, see Actions Defined by Amazon Managed Streaming for Apache Kafka in the IAM User Guide.

**Resources**

The Resource element specifies the object or objects to which the action applies. Statements must include either a Resource or a NotResource element. You specify a resource using an ARN or using the wildcard (*) to indicate that the statement applies to all resources.

The Amazon MSK instance resource has the following ARN:

```
arn:${Partition}:kafka:${Region}:${Account}:cluster/${ClusterName}/${UUID}
```

For more information about the format of ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces.

For example, to specify the CustomerMessages instance in your statement, use the following ARN:

```
"Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/CustomerMessages/abcd1234-abcd-dcbe-4321-a1b2abcd9f9f-2"
```

To specify all instances that belong to a specific account, use the wildcard (*):

```
"Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/**"
```

Some Amazon MSK actions, such as those for creating resources, cannot be performed on a specific resource. In those cases, you must use the wildcard (*).

```
"Resource": "*"
```

To specify multiple resources in a single statement, separate the ARNs with commas.

```
"Resource": ["resource1", "resource2"]
```

To see a list of Amazon MSK resource types and their ARNs, see Resources Defined by Amazon Managed Streaming for Apache Kafka in the IAM User Guide. To learn with which actions you can specify the ARN of each resource, see Actions Defined by Amazon Managed Streaming for Apache Kafka.

**Condition Keys**

The Condition element (or Condition block) lets you specify conditions in which a statement is in effect. The Condition element is optional. You can build conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple Condition elements in a statement, or multiple keys in a single Condition element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.
You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM Policy Elements: Variables and Tags in the IAM User Guide.

Amazon MSK defines its own set of condition keys and also supports using some global condition keys. To see all AWS global condition keys, see AWS Global Condition Context Keys in the IAM User Guide.

To see a list of Amazon MSK condition keys, see Condition Keys for Amazon Managed Streaming for Apache Kafka in the IAM User Guide. To learn with which actions and resources you can use a condition key, see Actions Defined by Amazon Managed Streaming for Apache Kafka.

Examples

To view examples of Amazon MSK identity-based policies, see Amazon Managed Streaming for Apache Kafka Identity-Based Policy Examples (p. 54).

Amazon MSK Resource-Based Policies

Amazon MSK does not support resource-based policies.

Authorization Based on Amazon MSK Tags

You can attach tags to Amazon MSK clusters. To control access based on tags, you provide tag information in the condition element of a policy using the kafka:ResourceTag/key-name, aws:RequestTag/key-name, or aws:TagKeys condition keys. For more information about tagging Amazon MSK resources, see the section called “Tagging a Cluster” (p. 36).

To view an example identity-based policy for limiting access to a cluster based on the tags on that cluster, see Accessing Amazon MSK Clusters Based on Tags (p. 56).

Amazon MSK IAM Roles

An IAM role is an entity within your AWS account that has specific permissions.

Using Temporary Credentials with Amazon MSK

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as AssumeRole or GetFederationToken.

Amazon MSK supports using temporary credentials.

Service-Linked Roles

Service-linked roles allow AWS services to access resources in other services to complete an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view but not edit the permissions for service-linked roles.

Amazon MSK supports service-linked roles. For details about creating or managing Amazon MSK service-linked roles, the section called “Service-Linked Roles” (p. 57).

Amazon Managed Streaming for Apache Kafka Identity-Based Policy Examples

By default, IAM users and roles don't have permission to create or modify Amazon MSK resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API
operations on the specified resources they need. The administrator must then attach those policies to
the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see
Creating Policies on the JSON Tab in the IAM User Guide.

Topics

• Policy Best Practices (p. 55)
• Allow Users to View Their Own Permissions (p. 55)
• Accessing One Amazon MSK Cluster (p. 56)
• Accessing Amazon MSK Clusters Based on Tags (p. 56)

Policy Best Practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete
Amazon MSK resources in your account. These actions can incur costs for your AWS account. When you
create or edit identity-based policies, follow these guidelines and recommendations:

• Get Started Using AWS Managed Policies – To start using Amazon MSK quickly, use AWS managed
  policies to give your employees the permissions they need. These policies are already available in
  your account and are maintained and updated by AWS. For more information, see Get Started Using
  Permissions With AWS Managed Policies in the IAM User Guide.

• Grant Least Privilege – When you create custom policies, grant only the permissions required
  to perform a task. Start with a minimum set of permissions and grant additional permissions as
  necessary. Doing so is more secure than starting with permissions that are too lenient and then trying
  to tighten them later. For more information, see Grant Least Privilege in the IAM User Guide.

• Enable MFA for Sensitive Operations – For extra security, require IAM users to use multi-factor
  authentication (MFA) to access sensitive resources or API operations. For more information, see Using
  Multi-Factor Authentication (MFA) in AWS in the IAM User Guide.

• Use Policy Conditions for Extra Security – To the extent that it's practical, define the conditions under
  which your identity-based policies allow access to a resource. For example, you can write conditions
to specify a range of allowable IP addresses that a request must come from. You can also write conditions
  to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For
  more information, see IAM JSON Policy Elements: Condition in the IAM User Guide.

Allow Users to View Their Own Permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed
policies that are attached to their user identity. This policy includes permissions to complete this action
on the console or programmatically using the AWS CLI or AWS API.

```json
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "ViewOwnUserInfo",
        "Effect": "Allow",
        "Action": [
          "iam:GetUserPolicy",
          "iam:ListGroupsForUser",
          "iam:ListAttachedUserPolicies",
          "iam:ListUserPolicies",
          "iam:GetUser"
        ],
        "Resource": [  
```
Accessing One Amazon MSK Cluster

In this example, you want to grant an IAM user in your AWS account access to one of your clusters, `purchaseQueriesCluster`. This policy allows the user to describe the cluster, get its bootstrap brokers, list its broker nodes, and update it.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "UpdateCluster",
            "Effect": "Allow",
            "Action": [
                "kafka:Describe*",
                "kafka:Get*",
                "kafka:List*",
                "kafka:Update*"
            ],
        }
    ]
}
```

Accessing Amazon MSK Clusters Based on Tags

You can use conditions in your identity-based policy to control access to Amazon MSK resources based on tags. This example shows how you might create a policy that allows the user to describe the cluster, get its bootstrap brokers, list its broker nodes, update it, and delete it. However, permission is granted only if the cluster tag `Owner` has the value of that user's user name.

```
{
    "Version": "2012-10-17",
    "Statement": [  
        {
            "Sid": "AccessClusterIfOwner",
            "Effect": "Allow",
            "Action": [
                "kafka:Describe*",
                "kafka:Get*"
            ],
        }
    ]
}
```
You can attach this policy to the IAM users in your account. If a user named richard-roo attempts to update an MSK cluster, the cluster must be tagged Owner=richard-roo or owner=richard-roo. Otherwise, he is denied access. The condition tag key Owner matches both Owner and owner because condition key names are not case-sensitive. For more information, see IAM JSON Policy Elements: Condition in the IAM User Guide.

Using Service-Linked Roles for Amazon MSK

Amazon MSK uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to Amazon MSK. Service-linked roles are predefined by Amazon MSK and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up Amazon MSK easier because you do not have to manually add the necessary permissions. Amazon MSK defines the permissions of its service-linked roles. Unless defined otherwise, only Amazon MSK can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

For information about other services that support service-linked roles, see AWS Services That Work with IAM, and look for the services that have Yes in the Service-Linked Role column. Choose a Yes with a link to view the service-linked role documentation for that service.

Topics
- Service-Linked Role Permissions for Amazon MSK (p. 57)
- Creating a Service-Linked Role for Amazon MSK (p. 58)
- Editing a Service-Linked Role for Amazon MSK (p. 58)
- Supported Regions for Amazon MSK Service-Linked Roles (p. 58)

Service-Linked Role Permissions for Amazon MSK

Amazon MSK uses the service-linked role named AWSServiceRoleForKafka – Allows Amazon MSK to access AWS resources on your behalf.

The AWSServiceRoleForKafka service-linked role trusts the following services to assume the role:

- kafka.amazonaws.com

The role permissions policy allows Amazon MSK to complete the following actions on the specified resources:

- Action: ec2:CreateNetworkInterface on *
- Action: ec2:DescribeNetworkInterfaces on *
Troubleshooting Amazon Managed Streaming for Apache Kafka

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-Linked Role Permissions in the IAM User Guide.

Creating a Service-Linked Role for Amazon MSK

You don't need to create a service-linked role manually. When you create an Amazon MSK cluster in the AWS Management Console, the AWS CLI, or the AWS API, Amazon MSK creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create an Amazon MSK cluster, Amazon MSK creates the service-linked role for you again.

Editing a Service-Linked Role for Amazon MSK

Amazon MSK does not allow you to edit the AWSServiceRoleForKafka service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a Service-Linked Role in the IAM User Guide.

Supported Regions for Amazon MSK Service-Linked Roles

Amazon MSK supports using service-linked roles in all of the AWS Regions where the service is available. For more information, see AWS Regions and Endpoints.

Troubleshooting Amazon Managed Streaming for Apache Kafka Identity and Access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon MSK and IAM.

Topics

- I Am Not Authorized to Perform an Action in Amazon MSK (p. 58)

I Am Not Authorized to Perform an Action in Amazon MSK

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the mateojackson IAM user tries to use the console to delete a cluster but does not have kafka:DeleteCluster permissions.

User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: kafka:DeleteCluster on resource: purchaseQueriesCluster
In this case, Mateo asks his administrator to update his policies to allow him to access the purchaseQueriesCluster resource using the kafka:DeleteCluster action.

## Client Authentication

You can enable client authentication with TLS for connections from your applications to your Amazon MSK brokers. To use client authentication, you need an ACM Private CA. For information about private CAs, see Creating and Managing a Private CA.

### To create a cluster that supports client authentication

This procedure shows you how to enable client authentication using a CA that is hosted by ACM.

1. Create a file named `clientauthinfo.json` with the following contents. Replace `Private-CA-ARN` with the ARN of your PCA.

   ```json
   {
     "Tls": {
       "CertificateAuthorityArnList": ["Private-CA-ARN"]
     }
   }
   ```

2. Create a file named `brokernodegroupinfo.json` as described in the section called “Creating a Cluster Using the AWS CLI” (p. 16).

3. Client authentication requires that you also enable encryption in transit between clients and brokers. Create a file named `encryptioninfo.json` with the following contents. Replace `KMS-Key-ARN` with the ARN of your KMS key. You can set `ClientBroker` to TLS or TLS_PLAINTEXT.

   ```json
   {
     "EncryptionAtRest": {
       "DataVolumeKMSKeyId": "KMS-Key-ARN"
     },
     "EncryptionInTransit": {
       "InCluster": true,
       "ClientBroker": "TLS"
     }
   }
   ```

   For more information about encryption, see the section called “Encryption” (p. 49).

4. On a machine where you have the AWS CLI installed, run the following command to create a cluster with authentication and in-transit encryption enabled. Save the cluster ARN provided in the response.

   ```bash
   ```

### To set up a client to use authentication

1. Create an Amazon EC2 instance to use as a client machine. For simplicity, create this instance in the same VPC you used for the cluster. See the section called “Step 4: Create a Client Machine” (p. 10) for an example of how to create such a client machine.

2. Create a topic. For an example, see the instructions under the section called “Step 5: Create a Topic” (p. 11).
3. On a machine where you have the AWS CLI installed, run the following command to get the bootstrap brokers of the cluster. Replace `Cluster-ARN` with the ARN of your cluster.

```
aws kafka get-bootstrap-brokers --cluster-arn Cluster-ARN
```

Save the string associated with `BootstrapBrokerStringTls` in the response.

4. On your client machine, run the following command to use the JVM trust store to create your client trust store. If your JVM path is different, adjust the command accordingly.

```
cp /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.201.b09-0.amzn2.x86_64/jre/lib/security/cacerts kafka.client.truststore.jks
```

5. On your client machine, run the following command to create a private key for your client. Replace `Distinguished-Name`, `Example-Alias`, `Your-Store-Pass`, and `Your-Key-Pass` with strings of your choice.

```
keytool -genkey -keystore kafka.client.keystore.jks -validity 300 -storepass Your-Store-Pass -keypass Your-Key-Pass -dname "CN=Distinguished-Name" -alias Example-Alias -storetype pkcs12
```

6. On your client machine, run the following command to create a certificate request with the private key you created in the previous step.

```
```

7. Open the `client-cert-sign-request` file and ensure that it starts with `-----BEGIN CERTIFICATE REQUEST-----` and ends with `-----END CERTIFICATE REQUEST-----`. If it starts with `-----BEGIN NEW CERTIFICATE REQUEST-----`, delete the word `NEW` (and the single space that follows it) from the beginning and the end of the file.

8. On a machine where you have the AWS CLI installed, run the following command to sign your certificate request. Replace `Private-CA-ARN` with the ARN of your PCA. You can change the validity value if you want. Here we use 300 as an example.

```
aws acm-pca issue-certificate --certificate-authority-arn Private-CA-ARN --csr file://client-cert-sign-request --signing-algorithm "SHA256WITHRSA" --validity Value=300,Type="DAYS"
```

Save the certificate ARN provided in the response.

9. Run the following command to get the certificate that ACM signed for you. Replace `Certificate-ARN` with the ARN you obtained from the response to the previous command.

```
aws acm-pca get-certificate --certificate-authority-arn Private-CA-ARN --certificate-arn Certificate-ARN
```

10. From the JSON result of running the previous command, copy the strings associated with `Certificate` and `CertificateChain`. Paste these two strings in a new file named signed-certificate-from-acm. Paste the string associated with `Certificate` first, followed by the string associated with `CertificateChain`. Replace the `\n` characters with new lines. The following is the structure of the file after you paste the certificate and certificate chain in it.

```
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
...
```
11. Run the following command on the client machine to add this certificate to your keystore so you can present it when you talk to the MSK brokers.

```
keytool -keystore kafka.client.keystore.jks -import -file signed-certificate-from-acm -alias Example-Alias -storepass Your-Store-Pass -keypass Your-Key-Pass
```

12. Create a file named `client.properties` with the following contents. Adjust the truststore and keystore locations to the paths where you saved `kafka.client.truststore.jks`.

```properties
security.protocol=SSL
ssl.truststore.location=/tmp/kafka_2.12-2.2.1/kafka.client.truststore.jks
ssl.keystore.location=/tmp/kafka_2.12-2.2.1/kafka.client.keystore.jks
ssl.keystore.password=Your-Store-Pass
ssl.key.password=Your-Key-Pass
```

To produce and consume messages using authentication

1. Run the following command to create a topic.

```
bin/kafka-topics.sh --create --zookeeper ZooKeeper-Connection-String --replication-factor 3 --partitions 1 --topic ExampleTopic
```

2. Run the following command to start a console producer. The file named `client.properties` is the one you created in the previous procedure.

```
bin/kafka-console-producer.sh --broker-list BootstrapBroker-String --topic ExampleTopic --producer.config client.properties
```

3. In a new command window on your client machine, run the following command to start a console consumer.

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBroker-String --topic ExampleTopic --consumer.config client.properties
```

4. Type messages in the producer window and watch them appear in the consumer window.

To add or remove read and write access to a topic

1. To grant read access to a topic, run the following command on your client machine. Use the same `Distinguished-Name` you used when you created the private key.

```
bin/kafka-acls.sh --authorizer-properties zookeeper.connect=ZooKeeper-Connection-String --add --allow-principal "User:CN=Distinguished-Name" --operation Read --groups* --topic Topic-Name
```

To remove read access, you can run the same command, replacing `--add` with `--remove`.

2. To grant write access to a topic, run the following command on your client machine. Use the same `Distinguished-Name` you used when you created the private key.
Controlling Access to Apache ZooKeeper

For security reasons you can limit access to the Apache ZooKeeper nodes that are part of your Amazon MSK cluster. To limit access to the nodes, you can assign a separate security group to them. You can then decide who gets access to that security group.

To place your Apache ZooKeeper nodes in a separate security group

1. Get the Apache ZooKeeper connection string for your cluster. To learn how, see the section called “Getting the Apache ZooKeeper Connection String” (p. 18). The connection string contains the DNS names of your Apache ZooKeeper nodes.
2. Use a tool like `host` or `ping` to convert the DNS names you obtained in the previous step to IP addresses. Save these IP addresses because you need them later in this procedure.
3. Save the IP addresses of the Apache ZooKeeper nodes because you need them later in this procedure.
4. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
5. In the left pane, under NETWORK & SECURITY, choose Network Interfaces.
6. In the search field above the table of network interfaces, type the name of your cluster, then type return. This limits the number of network interfaces that appear in the table to those interfaces that are associated with your cluster.
7. Select the check box at the beginning of the row that corresponds to the first network interface in the list.
8. In the details pane at the bottom of the page, look for the Primary private IPv4 IP. If this IP address matches one of the IP addresses you obtained in the first step of this procedure, this means that this network interface is assigned to an Apache ZooKeeper node that is part of your cluster. Otherwise, deselect the check box next to this network interface, and select the next network interface in the list. The order in which you select the network interfaces doesn’t matter. In the next steps, you will perform the same operations on all network interfaces that are assigned to Apache ZooKeeper nodes, one by one.
9. When you select a network interface that corresponds to an Apache ZooKeeper node, choose the Actions menu at the top of the page, then choose Change Security Groups. Assign a new security group to this network interface. For information about creating security groups, see Creating a Security Group in the Amazon VPC documentation.
10. Repeat the previous step to assign the same new security group to all the network interfaces that are associated with the Apache ZooKeeper nodes of your cluster.
11. You can now choose who has access to this new security group. For information about setting security group rules, see Adding, Removing, and Updating Rules in the Amazon VPC documentation.
Logging

You can deliver Apache Kafka broker logs to one or more of the following destination types: Amazon CloudWatch Logs, Amazon S3, Amazon Kinesis Data Firehose. You can also log Amazon MSK API calls with AWS CloudTrail.

Broker Logs

Broker logs enable you to troubleshoot your Apache Kafka applications and to analyze their communications with your MSK cluster. You can configure your new or existing MSK cluster to deliver INFO-level broker logs to one or more of the following types of destination resources: a CloudWatch log group, an S3 bucket, a Kinesis Data Firehose delivery stream. Through Kinesis Data Firehose you can then deliver the log data from your delivery stream to Amazon ES. You must create a destination resource before you configure your cluster to deliver broker logs to it. Amazon MSK doesn't create these destination resources for you if they don't already exist. For information about these three types of destination resources and how to create them, see the following documentation:

- Amazon CloudWatch Logs
- Amazon S3
- Amazon Kinesis Data Firehose

Note

Amazon Kinesis Data Firehose isn't available in the Asia Pacific (Hong Kong) and Middle East (Bahrain) Regions.

Required Permissions

For Amazon MSK to deliver broker logs to the destinations that you configure, ensure that the AmazonMSKFullAccess policy is attached to your IAM role. To stream broker logs to an S3 bucket, you also need the s3:PutBucketPolicy permission attached to your IAM role. For information about S3 bucket policies, see How Do I Add an S3 Bucket Policy? in the Amazon S3 Console User Guide. For information about IAM policies in general, see Access Management in the IAM User Guide.

Configuring Broker Logs Using the AWS Management Console

If you are creating a new cluster, look for the Broker log delivery heading in the Monitoring section. You can specify the destinations to which you want Amazon MSK to deliver your broker logs.

For an existing cluster, choose the cluster from your list of clusters, then choose the Details tab. Scroll down to the Monitoring section and then choose its Edit button. You can specify the destinations to which you want Amazon MSK to deliver your broker logs.

Configuring Broker Logs Using the AWS CLI

When you use the create-cluster or the update-monitoring commands, you can optionally specify the logging-info parameter and pass to it a JSON structure like the following example. In this JSON, all three destination types are optional.

```json
{
  "BrokerLogs": {
    "S3": {
      "Bucket": "ExampleBucketName",
      "Prefix": "ExamplePrefix",
      "Enabled": true
    }
  }
}
```
“Firehose”: {
   "DeliveryStream": "ExampleDeliveryStreamName",
   "Enabled": true
 },
"CloudWatchLogs": {
   "Enabled": true,
   "LogGroup": "ExampleLogGroupName"
}
}

Configuring Broker Logs Using the API

You can specify the optional loggingInfo structure in the JSON that you pass to the CreateCluster or UpdateMonitoring operations.

Logging Amazon MSK API Calls with AWS CloudTrail

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

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

To learn more about CloudTrail, including how to configure and enable it, see the AWS CloudTrail User Guide.

Amazon MSK Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in Amazon MSK, that activity is recorded in a CloudTrail event along with other AWS service events in Event history. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

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

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

Amazon MSK supports logging the following actions as events in CloudTrail log files:

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

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

For more information, see the CloudTrail userIdentity Element.

Example: Amazon MSK Log File Entries

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

The following example shows CloudTrail log entries that demonstrate the DescribeCluster and DeleteCluster actions.

```json
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "ABCDEF0123456789ABCDE",
        "arn": "arn:aws:iam::012345678901:user/Joe",
        "accountId": "012345678901",
        "accessKeyId": "AIDACKCEVSQ6G2EXAMPLE",
        "userName": "Joe"
      },
      "eventTime": "2018-12-12T02:29:24Z",
      "eventSource": "kafka.amazonaws.com",
      "eventName": "DescribeCluster",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "aws-cli/1.14.67 Python/3.6.0 Windows/10 botocore/1.9.20",
      "requestParameters": {
      },
      "responseElements": null,
      "requestID": "bd83f638-fdb5-0123-157e2fbf2bde",
      "eventID": "60052aba-0123-4511-bcde-3e18dbd42aa4",
      "readOnly": true,
      "eventType": "AwsApiCall",
      "recipientAccountId": "012345678901"
    },
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "ABCDEF0123456789ABCDE",
        "arn": "arn:aws:iam::012345678901:user/Joe",
        "accountId": "012345678901",
        "accessKeyId": "AIDACKCEVSQ6G2EXAMPLE",
        "userName": "Joe"
      },
      "eventTime": "2018-12-12T02:29:24Z",
      "eventSource": "kafka.amazonaws.com",
      "eventName": "DeleteCluster",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "aws-cli/1.14.67 Python/3.6.0 Windows/10 botocore/1.9.20",
      "requestParameters": {
      },
      "responseElements": null,
      "requestID": "bd83f638-fdb5-0123-157e2fbf2bde",
      "eventID": "60052aba-0123-4511-bcde-3e18dbd42aa4",
      "readOnly": true,
      "eventType": "AwsApiCall",
      "recipientAccountId": "012345678901"
    }
  ]
}
```
Compliance Validation for Amazon Managed Streaming for Apache Kafka

Third-party auditors assess the security and compliance of Amazon Managed Streaming for Apache Kafka as part of AWS compliance programs. These include PCI and HIPAA BAA.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

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

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

- **Security and Compliance Quick Start Guides** – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- **Architecting for HIPAA Security and Compliance Whitepaper** – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- **AWS Compliance Resources** – This collection of workbooks and guides might apply to your industry and location.
- **Evaluating Resources with Rules** in the AWS Config Developer Guide – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
Resilience in Amazon Managed Streaming for Apache Kafka

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, Amazon MSK offers several features to help support your data resiliency and backup needs.

Infrastructure Security in Amazon Managed Streaming for Apache Kafka

As a managed service, Amazon Managed Streaming for Apache Kafka is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access Amazon MSK through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

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

To access your Amazon MSK cluster from a client that’s in the same Amazon VPC as the cluster, make sure the cluster's security group has an inbound rule that accepts traffic from the client's security group. For information about setting up these rules, see Security Group Rules. For an example of how to access a cluster from an Amazon EC2 instance that's in the same VPC as the cluster, see Getting Started (p. 4).

To access an MSK cluster from outside its Amazon VPC, the following options exist.

Amazon VPC Peering

To access your MSK cluster from a VPC that's different from the cluster's VPC, you can create a peering connection between the two VPCs. For information about VPC peering, see the Amazon VPC Peering Guide.

AWS Direct Connect

AWS Direct Connect links your internal network to an AWS Direct Connect location over a standard 1 gigabit or 10 gigabit Ethernet fiber-optic cable. One end of the cable is connected to your router, the other to an AWS Direct Connect router. With this connection in place, you can create virtual interfaces directly to the AWS cloud and Amazon VPC, bypassing Internet service providers in your network path. For more information, see AWS Direct Connect.

AWS Transit Gateway

AWS Transit Gateway is a service that enables you to connect their VPCs and your on-premises networks to a single gateway. For information about how to use AWS Transit Gateway, see AWS Transit Gateway.

VPN Connections

You can connect your MSK cluster's VPC to remote networks and users using the VPN connectivity options described in the following topic: VPN Connections.

REST Proxies

You can install a REST proxy on an instance running within your cluster's Amazon VPC. REST proxies enable your producers and consumers to communicate with the cluster through HTTP API requests.

Multiple Region Multi-VPC Connectivity

The following document describes connectivity options for multiple VPCs that reside in different AWS Regions: Multiple Region Multi-VPC Connectivity.
EC2-Classic

Use the following procedure to access your cluster from an EC2-Classic instance.

1. Follow the guidance described in ClassicLink to connect your EC2-Classic instance to your cluster’s VPC.
2. Find and copy the private IP associated with your EC2-Classic instance.
3. Using the AWS CLI, run the following command, replacing `ClusterArn` with the Amazon Resource Name (ARN) for your MSK cluster.

   ```
   aws kafka describe-cluster --region us-east-1 --cluster-arn "ClusterArn"
   ```

4. In the output of the `describe-cluster` command, look for `SecurityGroups` and save the ID of the security group for your MSK cluster.
5. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
6. In the left pane, choose `Security Groups`.
7. Choose the security group whose ID you saved after you ran the `describe-cluster` command. Select the box at the beginning of the row corresponding to this security group.
8. In the lower half of the page, choose `Inbound Rules`.
9. Choose `Edit rules`, then choose `Add Rule`.
10. For the `Type` field, choose `All traffic` in the drop-down list.
11. Leave the `Source` set to `Custom` and enter the private IP of your EC2-Classic instance, followed immediately by `/32` with no intervening spaces.
12. Choose `Save rules`.

Port Information

The following list provides the numbers of the ports that Amazon MSK uses to communicate with client machines.

- To communicate with producers and consumers in plaintext, brokers use port 9092.
- To communicate with producers and consumers in TLS, brokers use port 9094.
- Apache ZooKeeper nodes use port 2181.
Migrating Clusters Using MirrorMaker

You can mirror or migrate your cluster using MirrorMaker, which is part of Apache Kafka. For example, you can use it to migrate your Apache Kafka cluster to Amazon MSK or to migrate from one MSK cluster to another. For information about how to use MirrorMaker, see Mirroring data between clusters in the Apache Kafka documentation.

An outline of the steps to follow when using MirrorMaker to migrate to an MSK cluster

1. Create the destination MSK cluster
2. Start MirrorMaker from an Amazon EC2 instance within the same Amazon VPC as the destination cluster.
3. Inspect the MirrorMaker lag.
4. After MirrorMaker catches up, redirect producers and consumers to the new cluster using the MSK cluster bootstrap brokers.
5. Shut down MirrorMaker.

Migrating Your Apache Kafka Cluster to Amazon MSK

Suppose that you have an Apache Kafka cluster named CLUSTER_ONPREM. That cluster is populated with topics and data. If you want to migrate that cluster to a newly created Amazon MSK cluster named CLUSTER_AWSMSK, this procedure provides a high-level view of the steps that you need to follow.

To migrate your existing Apache Kafka cluster to Amazon MSK

1. In CLUSTER_AWSMSK, create all the topics that you want to migrate.

   You can’t use MirrorMaker for this step because it doesn’t automatically re-create the topics that you want to migrate with the right replication level. You can create the topics in Amazon MSK with the same replication factors and numbers of partitions that they had in CLUSTER_ONPREM. You can also create the topics with different replication factors and numbers of partitions.

2. Start MirrorMaker from an instance that has read access to CLUSTER_ONPREM and write access to CLUSTER_AWSMSK.

3. Run the following command to mirror all topics:

   ```bash
   ./bin/kafka-mirror-maker.sh --consumer.config config/mirrormaker-consumer.properties --producer.config config/mirrormaker-producer.properties --whitelist '.*'
   ```

   In this command, config/mirrormaker-consumer.properties points to a bootstrap broker in CLUSTER_ONPREM; for example, bootstrap.servers=localhost:9092. And config/mirrormaker-producer.properties points to a bootstrap broker in CLUSTER_AWSMSK; for example, bootstrap.servers=10.0.0.237:9092,10.0.2.196:9092,10.0.1.233:9092.

4. Keep MirrorMaker running in the background, and continue to use CLUSTER_ONPREM. MirrorMaker mirrors all new data.
5. Check the progress of mirroring by inspecting the lag between the last offset for each topic and the current offset from which MirrorMaker is consuming.

Remember that MirrorMaker is simply using a consumer and a producer. So, you can check the lag using the `kafka-consumer-groups.sh` tool. To find the consumer group name, look inside the `mirrormaker-consumer.properties` file for the `group.id` and use its value. If there is no such key in the file, you can create it. For example, set `group.id=mirrormaker-consumer-group`.

6. After MirrorMaker finishes mirroring all topics, stop all producers and consumers, and then stop MirrorMaker. Then redirect the producers and consumers to the CLUSTER_AWSMSK cluster by changing their producer and consumer bootstrap brokers values. Restart all producers and consumers on CLUSTER_AWSMSK.

Migrating From One Amazon MSK Cluster to Another

You can use Apache MirrorMaker to migrate an MSK cluster to another cluster. For example, you can migrate from one version of Apache Kafka to another. For an example of how to use AWS CloudFormation to do this, see AWS::MSK::Cluster Examples (search for the example titled Create Two MSK Clusters To Use With Apache MirrorMaker).

MirrorMaker 1.0 Best Practices

This list of best practices applies to MirrorMaker 1.0.

- Run MirrorMaker on the destination cluster. This way, if a network problem happens, the messages are still available in the source cluster. If you run MirrorMaker on the source cluster and events are buffered in the producer and there is a network issue, events might be lost.
- If encryption is required in transit, run it in the source cluster.
- For consumers, set `auto.commit.enabled=false`
- For producers, set
  - `max.in-flight.requests.per.connection=1`
  - `retries=Int.Max_Value`
  - `acks=all`
  - `max.block.ms = Long.Max_Value`
- For a high producer throughput:
  - Buffer messages and fill message batches — tune `buffer.memory`, `batch.size`, `linger.ms`
  - Tune socket buffers — receive.buffer.bytes, `send.buffer.bytes`
- To avoid data loss, turn off auto commit at the source, so that MirrorMaker can control the commits, which it typically does after it receives the ack from the destination cluster. If the producer has `acks=all` and the destination cluster has `min.insync.replicas` set to more than 1, the messages are persisted on more than one broker at the destination before the MirrorMaker consumer commits the offset at the source.
- If order is important, you can set `retries` to 0. Alternatively, for a production environment, set `max.inflight.connections` to 1 to ensure that the batches sent out are not committed out of order if a batch fails in the middle. This way, each batch sent is retried until the next batch is sent out. If `max.block.ms` is not set to the maximum value, and if the producer buffer is full, there can be data loss (depending on some of the other settings). This can block and back-pressure the consumer.
- For high throughput
  - Increase `buffer.memory`. 

71
MirrorMaker 2.* Advantages

- Makes use of the Apache Kafka Connect framework and ecosystem.
- Detects new topics and partitions.
- Automatically syncs topic configuration between clusters.
- Supports "active/active" cluster pairs, as well as any number of active clusters.
- Provides new metrics including end-to-end replication latency across multiple data centers and clusters.
- Emits offsets required to migrate consumers between clusters and provides tooling for offset translation.
- Supports a high-level configuration file for specifying multiple clusters and replication flows in one place, compared to low-level producer/consumer properties for each MirrorMaker 1.* process.
Amazon MSK Limits

Amazon MSK has the following limits. If you require higher limits, you can create a support case.

- Up to 90 brokers per account and 30 brokers per cluster
- A minimum of 1 GiB of storage per broker
- A maximum of 16384 GiB of storage per broker
- Up to 100 configurations per account
Apache Kafka Versions

Amazon Managed Streaming for Apache Kafka (Amazon MSK) supports the following Apache Kafka versions.

Apache Kafka Versions 2.3.1

For information about Apache Kafka version 2.3.1, see its release notes on the Apache Kafka downloads site.

Apache Kafka Versions 2.2.1

For information about Apache Kafka version 2.2.1, see its release notes on the Apache Kafka downloads site.

Apache Kafka Versions 1.1.1

For information about Apache Kafka version 1.1.1, see its release notes on the Apache Kafka downloads site.
Troubleshooting Your Amazon MSK Cluster

The following information can help you troubleshoot problems that you might have with your Amazon MSK cluster. You can also post your issue to the Amazon MSK forum.

Topics
- No Default Security Group (p. 75)
- Cluster Appears Stuck in the CREATING State (p. 75)
- Cluster State Goes from CREATING to FAILED (p. 75)
- Cluster State is ACTIVE but Producers Cannot Send Data or Consumers Cannot Receive Data (p. 75)
- AWS CLI Doesn't Recognize Amazon MSK (p. 76)
- Partitions Go Offline or Replicas Are Out of Sync (p. 76)
- Disk Space is Running Low (p. 76)
- Memory is Running Low (p. 76)
- Producer Gets NotLeaderForPartitionException (p. 76)
- Under-Replicated Partitions (URP) Greater Than Zero (p. 76)

No Default Security Group

If you try to create a cluster and get an error indicating that there's no default security group, it might be because you are using a VPC that was shared with you. Ask your administrator to grant you permission to describe the security groups on this VPC and try again. For an example of a policy that allows this action, see Amazon EC2: Allows Managing EC2 Security Groups Associated With a Specific VPC, Programmatically and in the Console.

Cluster Appears Stuck in the CREATING State

Sometimes cluster creation can take up to 30 minutes. Wait for 30 minutes and check the state of the cluster again.

Cluster State Goes from CREATING to FAILED

Try creating the cluster again.

Cluster State is ACTIVE but Producers Cannot Send Data or Consumers Cannot Receive Data

- If the cluster creation succeeds (the cluster state is ACTIVE), but you can't send or receive data, ensure that your producer and consumer applications have access to the cluster. For more information, see the guidance in the section called “Step 4: Create a Client Machine” (p. 10).
• If your producers and consumers have access to the cluster but still experience problems producing and consuming data, the cause might be KAFKA-7697, which affects Apache Kafka version 2.1.0 and can lead to a deadlock in one or more brokers. Consider migrating to Apache Kafka 2.2.1, which is not affected by this bug. For information about how to migrate, see Migration (p. 70).

AWS CLI Doesn't Recognize Amazon MSK

If you have the AWS CLI installed, but it doesn't recognize the Amazon MSK commands, upgrade your AWS CLI to the latest version. For detailed instructions on how to upgrade the AWS CLI, see Installing the AWS Command Line Interface. For information about how to use the AWS CLI to run Amazon MSK commands, see How It Works (p. 15).

Partitions Go Offline or Replicas Are Out of Sync

These can be symptoms of low disk space. See the section called “Disk Space is Running Low” (p. 76).

Disk Space is Running Low

See the following best practices for managing disk space: the section called “Monitor Disk Space” (p. 78) and the section called “Adjust the Data Retention Parameters” (p. 78).

Memory is Running Low

If you see the MemoryUsed metric running high or MemoryFree running low, that doesn't mean there's a problem. Apache Kafka is designed to use as much memory as possible, and it manages it optimally.

Producer Gets NotLeaderForPartitionException

This is often a transient error. Set the producer's retries configuration parameter to a value that's higher than its current value.

Under-Replicated Partitions (URP) Greater Than Zero

The UnderReplicatedPartitions metric is an important one to monitor. In a healthy MSK cluster, this metric has the value 0. If it's greater than zero, that might be for one of the following reasons.

• If UnderReplicatedPartitions is spiky, the issue might be that the cluster isn't provisioned at the right size to handle incoming and outgoing traffic. See the section called “Right-size Your Cluster” (p. 78).
• If UnderReplicatedPartitions is consistently greater than 0 including during low-traffic periods, the issue might be that you've set restrictive ACLs that don't grant topic access to brokers. To replicate partitions, brokers must be authorized to both READ and DESCRIBE topics. DESCRIBE is granted by
default with the READ authorization. For information about setting ACLs, see Authorization and ACLs in the Apache Kafka documentation.
Best Practices

This topic outlines some best practices to follow when using Amazon MSK.

Right-size Your Cluster

To determine the right size for your MSK cluster and understand costs, see the MSK Sizing and Pricing spreadsheet. This spreadsheet provides an estimate for sizing an MSK cluster and the associated costs of Amazon MSK compared to a similar, self-managed, EC2-based Apache Kafka cluster. For more information about the input parameters in the spreadsheet, hover over the parameter descriptions. This spreadsheet was the result of running a test workload with three producers and three consumers, and ensuring that P99 write latencies were below 100 ms. This might not reflect your workload or performance expectations. Therefore, we recommend that you test your workloads after provisioning the cluster.

Monitor Disk Space

To avoid running out of disk space for messages, create a CloudWatch alarm that watches the KafkaDataLogsDiskUsed metric. When the value of this metric reaches or exceeds 85%, perform one or more of the following actions:

- Increase broker storage. For information on how to do this, see the section called “Scaling Up Broker Storage” (p. 21).
- Reduce the message retention period or log size. For information on how to do that, see the section called “Adjust the Data Retention Parameters” (p. 78).
- Delete unused topics.

For information on how to set up and use alarms, see Using Amazon CloudWatch Alarms. For a full list of Amazon MSK metrics, see the section called “Monitoring a Cluster” (p. 25).

Adjust the Data Retention Parameters

Consuming messages doesn't remove them from the log. To free up disk space regularly, you can explicitly specify a retention time period, which is how long messages stay in the log. You can also specify a retention log size. When either the retention time period or the retention log size are reached, Apache Kafka starts removing inactive segments from the log.

To specify a retention policy at the cluster level, set one or more of the following parameters: log.retention.hours, log.retention.minutes, log.retention.ms, or log.retention.bytes. For more information, see the section called “Custom Configurations” (p. 38).

You can also specify retention parameters at the topic level:

- To specify a retention time period per topic, use the following command.

  ```bash
  kafka-configs.sh --zookeeper ZooKeeperConnectionString --alter --entity-type topics --
  entity-name TopicName --add-config retention.ms=DesiredRetentionTimePeriod
  ```
• To specify a retention log size per topic, use the following command.

```
kafka-configs.sh --zookeeper ZooKeeperConnectionString --alter --entity-type topics --entity-name TopicName --add-config retention.bytes=DesiredRetentionLogSize
```

The retention parameters that you specify at the topic level take precedence over cluster-level parameters.

**Don't Add Non-MSK Brokers**

If you use Apache ZooKeeper commands to add brokers, these brokers don't get added to your MSK cluster, and your Apache ZooKeeper will contain incorrect information about the cluster. This might result in data loss. For supported cluster operations, see *How It Works* (p. 15).

**Enable In-Transit Encryption**

For information about encryption in transit and how to enable it, see the section called “Encryption in Transit” (p. 49).

**Reassign Partitions**

To move partitions to different brokers on the same cluster, you can use the partition reassignment tool named `kafka-reassign-partitions.sh`. For example, after you add new brokers to expand a cluster, you can rebalance that cluster by reassigning partitions to the new brokers. For information about how to add brokers to a cluster, see the section called “Expanding a Cluster” (p. 23). For information about the partition reassignment tool, see *Expanding your cluster* in the Apache Kafka documentation.
Document History for Amazon MSK Developer Guide

The following table describes the important changes to the Amazon MSK Developer Guide.

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for Streaming Broker Logs</td>
<td>Amazon MSK can now stream broker logs to CloudWatch Logs, Amazon S3, and Amazon Kinesis Data Firehose. Kinesis Data Firehose can, in turn, deliver these logs to the destinations that it supports, such as Amazon ES.</td>
<td>2020-02-25</td>
</tr>
<tr>
<td>Support for Apache Kafka 2.3.1</td>
<td>Amazon MSK now supports Apache Kafka version 2.3.1.</td>
<td>2019-12-19</td>
</tr>
<tr>
<td>Open Monitoring</td>
<td>Amazon MSK now supports open monitoring with Prometheus.</td>
<td>2019-12-04</td>
</tr>
<tr>
<td>Support for Apache Kafka 2.2.1</td>
<td>Amazon MSK now supports Apache Kafka version 2.2.1.</td>
<td>2019-07-31</td>
</tr>
<tr>
<td>General Availability</td>
<td>New features include tagging support, authentication, TLS encryption, configurations, and the ability to update broker storage.</td>
<td>2019-05-30</td>
</tr>
<tr>
<td>Support for Apache Kafka 2.1.0</td>
<td>Amazon MSK now supports Apache Kafka version 2.1.0.</td>
<td>2019-02-05</td>
</tr>
</tbody>
</table>
AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the AWS General Reference.