**Amazon Managed Streaming for Apache Kafka**

Amazon MSK is a fully managed service that makes it easy to build and run applications that use [Apache Kafka](https://aws.amazon.com/streaming-data/what-is-kafka/) to process streaming data. Apache Kafka is an open-source platform for building real-time streaming data pipelines and applications. With Amazon MSK, you can use native Apache Kafka APIs to populate data lakes, stream changes to and from databases, and power machine learning and analytics applications.

Apache Kafka clusters are challenging to setup, scale, and manage in production. When you run Apache Kafka on your own, you need to provision servers, configure Apache Kafka manually, replace servers when they fail, orchestrate server patches and upgrades, architect the cluster for high availability, ensure data is durably stored and secured, setup monitoring and alarms, and carefully plan scaling events to support load changes. Amazon MSK makes it easy to build and run production applications on Apache Kafka without needing Apache Kafka infrastructure management expertise. That means you spend less time managing infrastructure and more time building applications.

With a few clicks in the [Amazon MSK console](https://console.aws.amazon.com/msk/home?region=us-east-1) you can create highly available Apache Kafka clusters with settings and configuration based on Apache Kafka’s deployment best practices. Amazon MSK automatically provisions and runs Apache Kafka clusters. Amazon MSK continuously monitors cluster health and automatically replaces unhealthy nodes with no downtime to application. In addition, Amazon MSK secures Apache Kafka cluster by encrypting data at rest.

**Benefits of Amazon MSK**

### **Fully compatible**

Amazon MSK runs and manages [Apache Kafka](https://aws.amazon.com/streaming-data/what-is-kafka/) . This makes it easy to migrate and run existing Apache Kafka applications on AWS without changes to the application code. By using Amazon MSK, you maintain open source compatibility and can continue to use familiar custom and community-built tools such as [MirrorMaker](https://cwiki.apache.org/confluence/pages/viewpage.action?pageId=27846330" \t "_blank), [Apache Flink](https://flink.apache.org/), and [Prometheus](https://prometheus.io/).

### **Fully managed**

Amazon MSK lets you focus on creating streaming applications without having to worry about the operational overhead of managing Apache Kafka environment. Amazon MSK manages the provisioning, configuration, and maintenance of Apache Kafka clusters and Apache ZooKeeper nodes for you. Amazon MSK also shows key Apache Kafka performance metrics in the AWS console.

### **Elastic stream processing**

Apache Flink is a powerful, open-source stream processing framework for stateful computations of streaming data. You can run fully managed [Apache Flink applications](https://aws.amazon.com/kinesis/data-analytics/) written in SQL, Java, or Scala that elastically scale to process data streams within Amazon MSK.

### **Highly available**

Amazon MSK creates an Apache Kafka cluster and offers multi-AZ replication within an AWS Region. Amazon MSK continuously monitors cluster health, and if a component fails, Amazon MSK will automatically replace it.

### **Highly secure**

Amazon MSK provides multiple levels of security for Apache Kafka clusters including VPC network isolation, AWS IAM for control-plane API authorization, encryption at rest, TLS encryption in-transit, TLS based certificate authentication, and supports Apache Kafka Access Control Lists (ACLs) for data-plane authorization.

### **Apache ZooKeeper included**

[Apache ZooKeeper](https://zookeeper.apache.org/) is required to run Apache Kafka, coordinate cluster tasks, and maintain state for resources interacting with the cluster. Amazon MSK manages the Apache ZooKeeper nodes for you. Each Amazon MSK cluster includes the appropriate number of Apache ZooKeeper nodes for your Apache Kafka cluster at no additional cost.

### **Automatic recovery and patching**

Amazon MSK continuously monitors the health of your clusters and replaces unhealthy brokers without downtime for your applications. Amazon MSK manages the availability of your Apache ZooKeeper nodes so you will not need to start, stop, or directly access the nodes yourself. Amazon MSK also deploys software patches as needed to keep your cluster up to date and running smoothly.

## **Scalable**

### **Broker scaling**

You can start with a few brokers within an Amazon MSK cluster. Then, using the AWS management console or AWS CLI, you can scale up to 100’s of brokers per cluster. Submit a limit increase request if you need more than 15 brokers per cluster or more than 30 brokers per account.

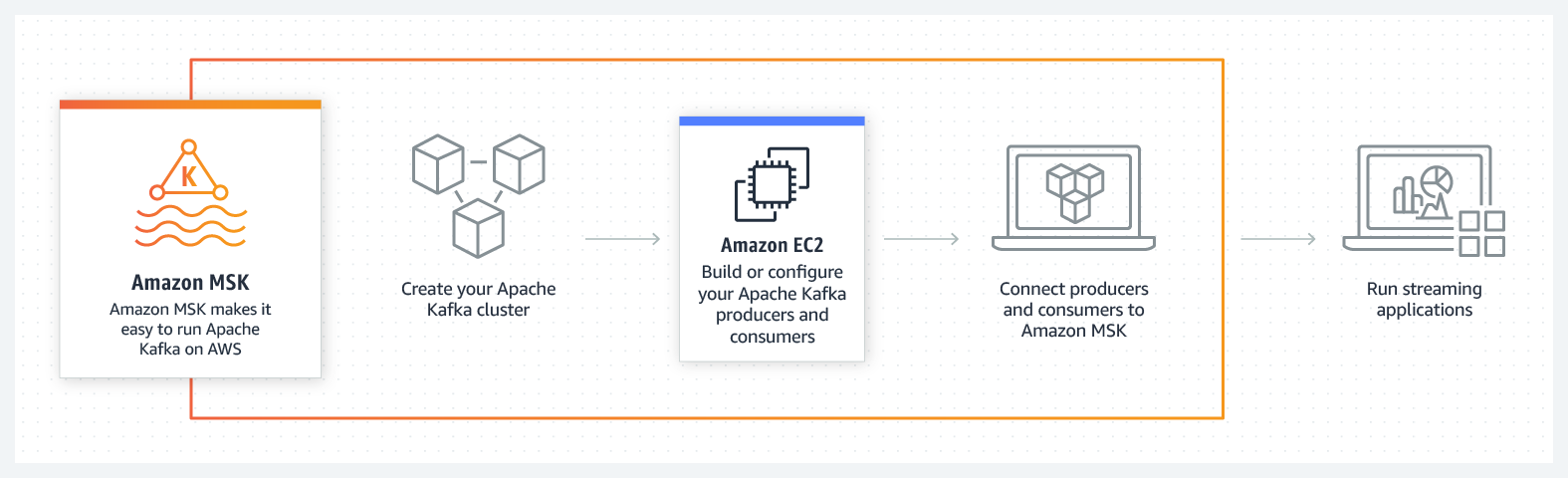
## **Deeply integrated**

Amazon MSK makes it easier for AWS customers to build end-to-end solutions by providing native AWS integrations out-of-the-box. You can run fully managed Apache Flink applications on data within Amazon MSK, encrypt data at rest using AWS KMS, authenticate clients to Amazon MSK using AWS Certificate Manager Private CAs, deploy Amazon MSK using code with AWS CloudFormation, privately connect clients within an Amazon VPC to Amazon MSK, and leverage AWS Identity and Access Management (IAM) for fine-grained service-level API control.

## **How it works**

Apache Kafka is a streaming data store that decouples applications producing streaming data (producers) into its data store from applications consuming streaming data (consumers) from its data store. Organizations use Apache Kafka as a data source for applications that continuously analyze and react to streaming data.

With a few clicks in the [Amazon MSK Console](https://console.aws.amazon.com/msk/home?region=us-east-1) Amazon MSK provisions Apache Kafka cluster and with support for version upgrades you can always be using the latest version of Apache Kafka that Amazon MSK supports. Once you configure clusters, applications can stream data from producers to a topic, where this data is read in real-time by consumers.



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


       Diagram showing the architecture of an example Amazon MSK Cluster.
     

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.

### **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.

# 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](https://kafka.apache.org/documentation/#configuration).

# 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 Default Amazon MSK Configuration](https://docs.aws.amazon.com/msk/latest/developerguide/msk-default-configuration.html). For more information about configuration properties, see [Apache Kafka Configuration](https://kafka.apache.org/documentation/#configuration).

To learn how you can create a custom MSK configuration, list all configurations, or describe them, see [Amazon MSK Configuration Operations](https://docs.aws.amazon.com/msk/latest/developerguide/msk-configuration-operations.html). To create an MSK cluster using a custom MSK configuration or to update a cluster with a new custom configuration, see [Amazon MSK: How It Works](https://docs.aws.amazon.com/msk/latest/developerguide/operations.html).

When you update your existing MSK cluster with a custom MSK configuration, Amazon MSK does rolling restarts when necessary, using best practices to minimize customer downtime. For example, after Amazon MSK restarts each broker, it tries to let the broker catch up on data that the broker might have missed during the configuration update before it moves to the next broker.

# 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](https://kafka.apache.org/documentation/#brokerconfigs) in the Apache Kafka documentation. For information about dynamic configuration and example commands, see [Updating Broker Configs](https://kafka.apache.org/documentation/#dynamicbrokerconfigs) in the Apache Kafka documentation.

# 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](https://kafka.apache.org/documentation/#topicconfigs) in the Apache Kafka documentation.