The following aspects of running Kafka clusters on AWS:

1.Deployment considerations and patterns

2.Storage options

3.Instance types

4.Networking

5.Upgrades

6.Performance tuning

7.Monitoring

8.Security

9.Backup and restore

Note: While implementing Kafka clusters in a production environment, make sure also to consider factors like your number of messages, message size, monitoring, failure handling, and any operational issues.

Deployment considerations and patterns.

In this section, we discuss various deployment options available for Kafka on AWS. A successful deployment starts with thoughtful consideration of these options. Considering availability, consistency, and operational overhead of the deployment helps when choosing the right option.

Single AWS Region, Three Availability Zones, All Active.

One typical deployment pattern (all active) is in a single AWS Region with three Availability Zones (AZs). One Kafka cluster is deployed in each AZ along with Apache ZooKeeper and Kafka producer and consumer instances as shown in the illustration following.

Graphical user interface, diagram

Description automatically generated

In this pattern, this is the Kafka cluster deployment:

Kafka producers and Kafka cluster are deployed on each AZ.

Data is distributed evenly across three Kafka clusters by using Elastic Load Balancer.

Kafka consumers aggregate data from all three Kafka clusters.

Kafka cluster failover occurs this way:

Mark down all Kafka producers

Stop consumers

Debug and restack Kafka

Restart consumers

Restart Kafka producers

2. Storage options:

There are two storage options for file storage in Amazon EC2:

--Ephemeral storage (instance store)

--Amazon Elastic Block Store (Amazon EBS)

Ephemeral storage is local to the Amazon EC2 instance. It can provide high IOPS based on the instance type. On the other hand, Amazon EBS volumes offer higher resiliency and you can configure IOPS based on your storage needs. EBS volumes also offer some distinct advantages in terms of recovery time. Your choice of storage is closely related to the type of workload supported by your Kafka cluster.

Kafka provides built-in fault tolerance by replicating data partitions across a configurable number of instances. If a broker fails, you can recover it by fetching all the data from other brokers in the cluster that host the other replicas. Depending on the size of the data transfer, it can affect recovery process and network traffic. These in turn eventually affect the cluster�s performance.

The following table contrasts the benefits of using an instance store versus using EBS for storage:

Table

Description automatically generated

Intuit chose **EBS** because of their frequent instance **restacking** requirements and also other benefits provided by **EBS**.

Generally, **Kafka deployments** use a **replication** factor of three. **EBS** offers **replication** within their service, so **Intuit** chose a replication factor of two instead of three.

**3. Networking:**

The network plays a very important role in a distributed system like **Kafka**. A fast and reliable network ensures that nodes can communicate with each other easily. The available network throughput controls the maximum amount of traffic that **Kafka** can handle. Network throughput, combined with disk storage, is often the governing factor for **cluster** sizing.

If you expect your cluster to receive high read/write traffic, select an instance type that offers **10-Gb/s performance**.

In addition, choose an option that keeps **interbroker** network traffic on the private subnet, because this approach allows clients to connect to the brokers. Communication between brokers and clients uses the same network interface and port. For more details, see the documentation about IP addressing for EC2 instances.

If you are deploying in more than one AWS Region, you can connect the two VPCs in the two AWS Regions using cross-region VPC peering. However, be aware of the networking costs associated with **cross-AZ deployments**.

Setting up kafka on AWS:

Graphical user interface, text, application, email

Description automatically generated

mkdir ~/Downloads

curl "[https://www.apache.org/dist/kafka/2.1.1/kafka\_2.11-2.1.1.tgz"](https://www.apache.org/dist/kafka/2.1.1/kafka_2.11-2.1.1.tgz%22) -o ~/Downloads/kafka.tgz

Graphical user interface, text, application, email

Description automatically generated