Setting up a 3-node Apache Kafka cluster

Java Installation:

- Install Java on all three nodes

|  |
| --- |
| sudo apt update  sudo apt install openjdk-8-jdk |

Kafka Installation:

Download Kafka:

- Downloaded the latest version of Apache Kafka **link**

2. Extract Kafka:

- Extract the downloaded Kafka archive:

-use Kafka with 2.13 version only

|  |
| --- |
| tar -xzf kafka\_2.13-<version>.tgz  cd kafka\_2.13-<version> |

3. Configure Server Properties:

- Edit the server properties file for each node:

|  |
| --- |
| cp config/server.properties config/server-1.properties  cp config/server.properties config/server-2.properties |

- Edit `config/server-1.properties` and `config/server-2.properties`

Properties

|  |
| --- |
| broker.id=1 # Change for each server (1, 2, 3)  listeners=PLAINTEXT://:9093 # Use different ports for each node (e.g., 9093, 9094)  log.dirs=/tmp/kafka-logs-1 # Use different directories for each node |

4. Start Kafka Servers:

- Open three terminal windows, one for each server, and run the following commands:

|  |
| --- |
| bin/kafka-server-start.sh config/server.properties  bin/kafka-server-start.sh config/server-1.properties  bin/kafka-server-start.sh config/server-2.properties |

Zookeeper config set up:

Cd to zookeeper properties

|  |
| --- |
| properties  tickTime=2000  dataDir=/tmp/zookeeper  clientPort=2181 # Use a different port if needed  initLimit=5  syncLimit=2  server.1=<ip-of-node-1>:2888:3888  server.2=<ip-of-node-2>:2888:3888  server.3=<ip-of-node-3>:2888:3888 |

3. Start Zookeeper:

- Start Zookeeper on each node:

|  |
| --- |
| bin/zookeeper-server-start.sh config/zookeeper.properties |

Start Kafka on each node

|  |
| --- |
| command |

Testing the Cluster:

1. Create a Topic:

- Create a test topic. Replace `my-topic` with your desired topic name.

|  |
| --- |
| bin/kafka-topics.sh --create --topic my-topic --bootstrap-server localhost:9092 --replication-factor 3 --partitions 1 |

2. Produce and Consume Messages:

- Open two new terminal windows, one for producing and one for consuming messages:

|  |
| --- |
| bin/kafka-console-producer.sh --topic my-topic --bootstrap-server localhost:9092 |

|  |
| --- |
| bin/kafka-console-consumer.sh --topic my-topic --bootstrap-server localhost:9092 --from-beginning |

- Start producing and consuming messages to test the cluster.

a 3-node Apache Kafka cluster on Ubuntu.

# 3-Node Setup with Apache Kafka: Implementation, Limitations, and Security Considerations

Implementation

A 3-node setup with Apache Kafka involves deploying Kafka across three different nodes to achieve high availability and fault tolerance. Here's a step-by-step guide to implementing this setup:

1. Node Configuration:

- Set up three separate nodes, ensuring proper network connectivity.

- Install Java on each node, as Kafka is a Java-based platform.

2. Kafka Installation:

- Download and install Apache Kafka on each node.

- Configure Kafka properties, such as broker IDs, listeners, and data directories, to suit your environment.

3. Zookeeper Configuration:

- Set up Apache Zookeeper as it is a crucial component for Kafka's coordination.

- Configure Zookeeper ensemble across the three nodes.

4. Broker Configuration:

- Adjust Kafka broker configurations to establish inter-broker communication.

- Ensure replication factors are appropriately set for fault tolerance.

5. Start Kafka Cluster:

- Start Zookeeper ensemble on the designated nodes.

- Launch Kafka brokers on each node, creating a distributed and fault-tolerant Kafka cluster.

Limitations

While a 3-node setup enhances Kafka's reliability, there are certain limitations to consider:

Resource Intensiveness:

Deploying Kafka across multiple nodes requires significant resources, and smaller environments may face challenges in resource allocation.

Complexity:

Managing a multi-node setup introduces complexity in terms of configuration and maintenance, especially for those unfamiliar with distributed systems.

Cost:

Scaling horizontally with more nodes may result in increased infrastructure costs.

Disadvantages of Apache Kafka Binary vs. Docker-Compose Set Up using Confluent Platform (CP)

Apache Kafka Binary:

Manual Configuration:

Requires manual configuration of Kafka and Zookeeper properties, which may be error-prone.

Dependency Management:

Dependencies on external tools for monitoring and management need to be set up separately.

Docker-Compose Set Up using CP:

Simplified Deployment:

Docker-compose simplifies deployment by packaging Kafka, Zookeeper, and other necessary components into a unified setup.

Integrated Tools:

Confluent Platform includes additional tools (Confluent Control Center, Schema Registry) for a comprehensive Kafka ecosystem.

Issues in Connect Set Up when Scaling

When scaling Kafka Connect horizontally, several issues may arise:

Configuration Management:

Coordinating configurations across multiple Connect nodes can be challenging.

Data Consistency:

Ensuring consistency in data processing across multiple instances requires careful attention to connector configurations.

Monitoring and Scaling Tools:

Additional tools may be needed for effective monitoring and scaling of Kafka Connect in a distributed environment.

3-Node Set Up Security

Ensuring the security of your 3-node Kafka setup is paramount. Consider the following security measures:

Network Security:

Implement firewalls and network policies to restrict unauthorized access to Kafka nodes.

Encryption:

Enable SSL/TLS for securing data in transit between Kafka brokers and clients.

Authentication and Authorization:

Utilize mechanisms like SASL for authentication and configure ACLs to control access to Kafka resources.

Regular Audits:

Conduct regular security audits to identify and address potential vulnerabilities.

More Effort is Needed for Security

While Kafka provides security features, additional effort is required to ensure a robust security posture:

Regular Updates:

Keep Kafka, Zookeeper, and other components updated to patch security vulnerabilities.

User Education:

Educate users on security best practices, including the proper handling of credentials and secure communication.

Monitoring and Incident Response:

Implement monitoring tools and a robust incident response plan to detect and mitigate security threats promptly.

Viewing Topics

To view Kafka topics in a 3-node setup:

- Use the Kafka command-line tools like `kafka-topics.sh` to list, describe, or manage topics.

- Leverage Confluent Control Center or other monitoring tools for a more comprehensive view of topics and their metrics.

a 3-node setup with Apache Kafka offers scalability and fault tolerance but comes with its set of complexities and security considerations. By carefully implementing and managing the system, you can harness the full potential of Kafka in a distributed environment.