**Introduction**

Apache Kafka is a distributed message broker designed to handle large volumes of real-time data efficiently Kafka runs as a cluster of one or more servers which makes it highly scalable and due to this distributed nature it has inbuilt fault-tolerance while delivering higher throughput when compared to its counterparts.

Steps to install Kafka cluster on Ubuntu 16.04

**Step 1. Install Java**

Kafka is written in Java and Scala and requires jre 1.7 and above to run it. In this step, we will ensure Java is installed.

sudo apt-get update

sudo apt-get install default-jre

**Step 2. Install Zookeeper**

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. Kafka uses Zookeeper for maintaining heart beats of its nodes, maintain configuration, and most importantly to elect leaders.

sudo apt-get install zookeeperd

we will now check if Zookeeper is alive and if it’s OK

telnet localhost 2181

at Telnet prompt, we will enter

ruok

(are you okay) if it’s all okay it will end telnet session and reply with

imok

**Step 3. Create a service User for Kafka**

As Kafka is a network application creating a non sudo user specifically for Kafka minimizes the risk if the machine is to be compromised.

sudo adduser --system --no-create-home --disabled-password --disabled-login kafka

**Step 4. Installing Kafka**

Download Kafka

cd ~

wget "<http://www-us.apache.org/dist/kafka/0.11.0.0/kafka_2.12-0.11.0.0.tgz>"

Optionally check integrity of downloaded file

curl <http://kafka.apache.org/KEYS> | gpg --import

wget <https://dist.apache.org/repos/dist/release/kafka/0.11.0.0/kafka_2.11-0.11.0.0.tgz.asc>

gpg --verify kafka\_2.11-0.11.0.0.tgz.asc kafka\_2.12-0.11.0.0.tgz

Create directory for extracting Kafka

sudo mkdir /opt/kafka

sudo tar -xvzf kafka\_2.12-0.11.0.0.tgz --directory /opt/kafka --strip-components 1

Optionally delete Kafka tarball and .asc file

rm -rf kafka\_2.12-0.11.0.0.tgz kafka\_2.11-0.11.0.0.tgz.asc

**Step 5. Configuring Kafka Server**

Kafka persists data to disk so we will now make a directory for it.

sudo mkdir /var/lib/kafka

sudo mkdir /var/lib/kafka/data

open /opt/kafka/config/server.properties in your favourite editor.

sudo nano /opt/kafka/config/server.properties

By default, Kafka doesn’t allow us to delete topics. To be able to delete topics, find the line and change it.

delete.topic.enable = true

next, we will change log directory

log.dirs=/var/lib/kafka/data

Kafka auto-deletes oldest logs after a particular time period or after logs reach a certain size. To adjust it according to the time change the line below.

log.retention.hours=168  #other accepted keys are(log.retention.ms, log.retention.minutes)

Or according to disk size.

log.retention.bytes=104857600

**Step 6. Ensure Permission of Directories**

first, we will ensure kafka user which we created at 3rd step has permission to all of the Kafka related directories.

sudo chown -R kafka:nogroup /opt/kafka

sudo chown -R kafka:nogroup /var/lib/kafka

**Step 7. Testing installation**

In a terminal start a kafka server

sudo /opt/kafka/bin/kafka-server-start.sh /opt/kafka/config/server.properties

In another terminal create a topic

/opt/kafka/bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test

List all topics with the command below and it will print test the topic we just created

/opt/kafka/bin/kafka-topics.sh --list --zookeeper localhost:2181

Let’s start publishing messages on test topic

/opt/kafka/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test

We will now create a subscriber on test topic and listen from the beginning of the topic.

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

Enter some message in the producer

Hello world!!!

You will see the messages appearing the subscriber terminal.

As we had started Kafka in the first step as sudo, we should change the user of directories back to kafka. Basically, we would require rerunning step 6

**Step 8. Launching Kafka as a service on startup**

In order to launch Kafka as a service on Ubuntu 16.04, we will create a unit file describing the service.

For this, we will create a unit file in /etc/systemd/system directory

sudo nano /etc/systemd/system/kafka.service

[Unit]

Description=High-available, distributed message broker

After=network.target

[Service]

User=kafka

ExecStart=/opt/kafka/bin/kafka-server-start.sh /opt/kafka/config/server.properties

[Install]

WantedBy=multi-user.target

Start the newly created service

sudo systemctl start kafka

Check status of the service by

sudo systemctl status kafka

**Step 9. Setting up multi-node cluster**

Although Kafka can run on a single node, we can run it on multiple nodes for data redundancy and accidentally fail over.

For ease of understanding let’s assume the first node we installed in is

node-1 with ip 10.0.0.1

And nodes we will install next is

node-2 with ip 10.0.0.2

node-3 with ip 10.0.0.3

For setting up a multi-node cluster we will first follow step 1 through step 5 and in step 5 with other settings in

/opt/kafka/config/server.properties

We will change the following settings.

broker.id should be unique for each node in the cluster.

for node-2 broker.id=1

for node-3 broker.id=2

change zookeeper.connect value to have such that it lists all zookeeper hosts with port

zookeeper.connect=10.0.0.1:2181,10.0.0.2:2181,10.0.0.3:2181

Note: Although Zookeeper is not required on every node it’s a good practice

We will also require changing zookeeper settings in

/etc/zookeeper/conf/zoo.cfg

to mention all the nodes of zookeeper

server.0=10.0.0.1:2888:3888

server.1=10.0.0.2:2888:3888

server.2=10.0.0.3:2888:3888

Restart zookeeper

sudo systemctl restart zookeeper.service

Now we can follow step 6 onwards to complete the installation.