**KAFKA DOCUMENTATION**

# Apache Kafka Cluster (3 Brokers) setup on Ubuntu

**Getting started**

We will be following below steps to setup our Kafka Cluster on 3 Ubuntu machines (each with kafka and zookeeper). Note that, steps guided below needs to be replicated in other 2 machines with only broker.id and advertised.listeners changed across machines (example demonstrated)

1. Download the latest Kafka binaries
2. Install Java
3. Disable RAM swap
4. Create a directory with appropriate permissions for Kafka logs
5. Create a directory with appropriate permissions for Zookeeper snapshots
6. Specify an ID for the Zookeeper servers to reach quorum
7. Modify the Kafka configuration file to reflect the appropriate settings
8. Modify the Zookeeper configuration file to reflect the appropriate settings
9. Create Zookeeper as a service, so it can run without manual intervention
10. Create Kafka as a service, so it can run without manual intervention
11. Create a topic to test that the cluster is working properly

**Download the latest Kafka binaries**

1. Use wget to download the tar file from the mirror:

wget http://mirror.cogentco.com/pub/apache/kafka/2.6.2/kafka\_2.12-2.6.2.tgz

1. Extract the tar file and move it into the /opt directory:

tar -xvf kafka\_2.12-2.6.2.tgz

sudo mv kafka\_2.12-2.6.2 /opt/kafka

1. Extract the tar file and move it into the /opt directory:

cd /opt/kafka

ls

**Install Java and Disable RAM swap**

1. Use the following command to install Java Developer Kit (As of now JDK 8 is used):

sudo apt install -y openjdk-8-jdk

1. Use the following command to verify that Java has been installed:

java -version

1. Use the following command to disable RAM swap:

swapoff -a

1. Use the following command to comment out swap in the /etc/fstab file:

sudo sed -i '/ swap / s/^/#/' /etc/fstab

**Create a New Directory for Kafka and Zookeeper**

1. Use the following command to create a new directory for Kafka message logs:

sudo mkdir -p /data/kafka

1. Use the following command to create a snapshot directory for Zookeeper:

sudo mkdir -p /data/zookeeper

1. Change ownership of those directories to allow the user ubuntu control:

sudo chown -R ubuntu:ubuntu /data/kafka

# Give permission to Zookeeper snapshot directory

sudo chown -R ubuntu:ubuntu /data/zookeeper

**Specify an ID for Each Zookeeper Server**

1. Use the following command to create a file in /data/zookeeper (on server #1) called myid with the contents "1" to specify Zookeeper server #1:

echo "1" > /data/zookeeper/myid

1. Use the following command to create a file in /data/zookeeper (on server #2) called myid with the contents "2" to specify Zookeeper server #2:

echo "2" > /data/zookeeper/myid

1. Use the following command to create a file in /data/zookeeper (on server #3) called myid with the contents "3" to specify Zookeeper server #3:

echo "3" > /data/zookeeper/myid

**Modify the Kafka and Zookeeper Configuration Files**

1. Use the following command to remove the existing server.properties file (in the config directory) and create a new server.properties file:

rm config/server.properties

vi config/server.properties

1. Copy and paste the following into the contents of the server.properties file and change the broker.id and the advertised.listeners:

# change this for each broker

broker.id=[broker\_number]

# example, broker.id=1 for server1, broker.id=2 for server2 and broker.id=3 for server 3

# change this to the hostname of each broker

# example advertised.listeners=PLAINTEXT://kafka1:9092

listeners=PLAINTEXT://192.168.100.90:9092

advertised.listeners=PLAINTEXT://192.168.100.90:9092

# example, hostname -> kafka1 for server1, hostname -> kafka2 for server2 and hostname -> kafka3 for server 3

# The ability to delete topics

delete.topic.enable=true

# Where logs are stored

log.dirs=/data/kafka

# default number of partitions

num.partitions=8

# default replica count based on the number of brokers

default.replication.factor=3

# to protect yourself against broker failure

min.insync.replicas=2

# logs will be deleted after how many hours

log.retention.hours=168

# size of the log files

log.segment.bytes=1073741824

# check to see if any data needs to be deleted

log.retention.check.interval.ms=300000

# location of all zookeeper instances and kafka directory

zookeeper.connect=zookeeper1:2181,zookeeper2:2181,zookeeper3:2181/kafka

# timeout for connecting with zookeeper

zookeeper.connection.timeout.ms=6000

# automatically create topics

auto.create.topics.enable=true

1. Use the following command to remove the existing zookeeper.properties file (in the config directory) and create a new zookeeper.properties file:

rm config/zookeeper.properties

vi config/zookeeper.properties

1. Copy and paste the following into the contents of the zookeeper.properties file (don't change this file):

# the directory where the snapshot is stored.

dataDir=/data/zookeeper

# the port at which the clients will connect

clientPort=2181

# setting number of connections to unlimited

maxClientCnxns=0

# keeps a heartbeat of zookeeper in milliseconds

tickTime=2000

# time for initial synchronization

initLimit=10

# how many ticks can pass before timeout

syncLimit=5

# define servers ip and internal ports to zookeeper

server.1=192.168.100.90:2888:3888

server.2=192.168.100.91:2888:3888

server.3=192.168.1.92:2888:3888

**Add custom hostnames to /etc/hosts file**

1. Since we are creating 3 Ubuntu Servers with custom hostnames for kafka and zookeeper, we are required to add this ip to hostname mapping in /etc/hosts file

sudo vi /etc/hosts

192.168.100.90 kafka1

192.168.100.90 zookeeper1

192.168.100.86 kafka2

192.168.100.86 zookeeper2

192.168.100.87 kafka3

192.168.100.87 zookeeper3

**Create the Kafka and Zookeeper Service**

1. Create the file /etc/init.d/zookeeper on each server and paste in the following contents:

sudo vi /etc/init.d/zookeeper

#!/bin/bash

#/etc/init.d/zookeeper

DAEMON\_PATH=/opt/kafka/bin

DAEMON\_NAME=zookeeper

# Check that networking is up.

#[ ${NETWORKING} = "no" ] && exit 0

PATH=$PATH:$DAEMON\_PATH

case "$1" in

start)

# Start daemon.

pid=`ps ax | grep -i 'org.apache.zookeeper' | grep -v grep | awk '{print $1}'`

if [ -n "$pid" ]

then

echo "Zookeeper is already running";

else

echo "Starting $DAEMON\_NAME";

$DAEMON\_PATH/zookeeper-server-start.sh -daemon /opt/kafka/config/zookeeper.properties

fi

;;

stop)

echo "Shutting down $DAEMON\_NAME";

$DAEMON\_PATH/zookeeper-server-stop.sh

;;

restart)

$0 stop

sleep 2

$0 start

;;

status)

pid=`ps ax | grep -i 'org.apache.zookeeper' | grep -v grep | awk '{print $1}'`

if [ -n "$pid" ]

then

echo "Zookeeper is Running as PID: $pid"

else

echo "Zookeeper is not Running"

fi

;;

\*)

echo "Usage: $0 {start|stop|restart|status}"

exit 1

esac

exit 0

1. Change the file to executable, change ownership, install, and start the service:

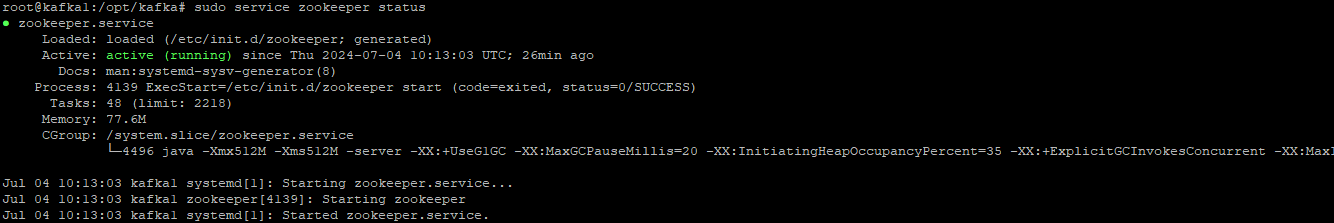
sudo chmod +x /etc/init.d/zookeeper

sudo chown root:root /etc/init.d/zookeeper

sudo update-rc.d zookeeper defaults

sudo service zookeeper start

sudo service zookeeper status



1. Create the file /etc/init.d/kafka on each server and paste in the following contents:

sudo vi /etc/init.d/kafka

#!/bin/bash

#/etc/init.d/kafka

DAEMON\_PATH=/opt/kafka/bin

DAEMON\_NAME=kafka

# Check that networking is up.

#[ ${NETWORKING} = "no" ] && exit 0

PATH=$PATH:$DAEMON\_PATH

# See how we were called.

case "$1" in

start)

# Start daemon.

pid=`ps ax | grep -i 'kafka.Kafka' | grep -v grep | awk '{print $1}'`

if [ -n "$pid" ]

then

echo "Kafka is already running"

else

echo "Starting $DAEMON\_NAME"

$DAEMON\_PATH/kafka-server-start.sh -daemon /opt/kafka/config/server.properties

fi

;;

stop)

echo "Shutting down $DAEMON\_NAME"

$DAEMON\_PATH/kafka-server-stop.sh

;;

restart)

$0 stop

sleep 2

$0 start

;;

status)

pid=`ps ax | grep -i 'kafka.Kafka' | grep -v grep | awk '{print $1}'`

if [ -n "$pid" ]

then

echo "Kafka is Running as PID: $pid"

else

echo "Kafka is not Running"

fi

;;

\*)

echo "Usage: $0 {start|stop|restart|status}"

exit 1

esac

exit 0

1. Change the file to executable, change ownership, install, and start the service:

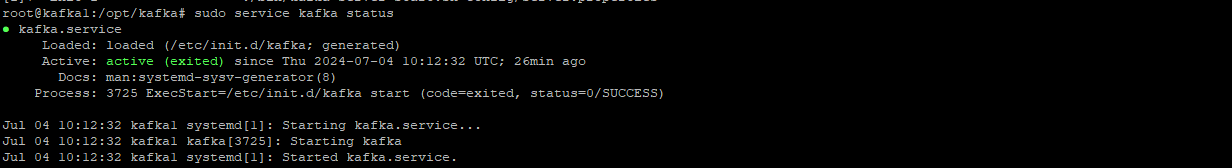
sudo chmod +x /etc/init.d/kafka

sudo chown root:root /etc/init.d/kafka

sudo update-rc.d kafka defaults

sudo service kafka start

sudo service kafka status



**Create a Topic**

1. Use the following command to create a topic named awesome-kafka:

./bin/kafka-topics.sh --zookeeper zookeeper1:2181/kafka --create --topic myfirsttopic-kafka --replication-factor 1 --partitions 3

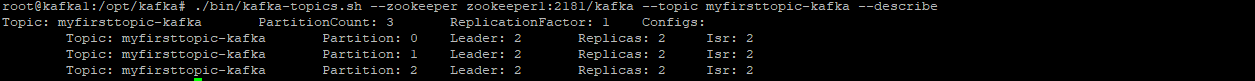


(or)

bin/kafka-topics.sh --create --topic my\_topic --bootstrap-server 192.168.100.85:9092 --partitions 3 --replication-factor 1(**In newer versions of Kafka (2.4.0 and later), ZooKeeper is not directly used for topic management; instead, the --bootstrap-server option is used**)

1. Use the following command to describe the topic:

./bin/kafka-topics.sh --zookeeper zookeeper1:2181/kafka --topic awesome-kafka –describe



(or)

bin/kafka-topics.sh --describe --topic my\_topic --bootstrap-server 192.168.100.85:9092 **(bin/kafka-topics.sh --describe --topic my\_topic --bootstrap-server localhost:9092)**

**Run a Producer**

1. Use the following command to run a Producer and publish messages

./bin/kafka-console-producer.sh --broker-list kafka1:9092 --topic awesome-kafka

>hello kafka

>this is from your producer

>are you alright?

>Looks you are not responding back

>Bye :)

**Run a Consumer**

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

Stop ZooKeeper

sudo systemctl stop zookeeper

sudo systemctl status zookeeper

Start the ZooKeeper Service:

sudo systemctl start zookeeper

sudo systemctl status zookeeper

**Create Topic**

**./bin/kafka-topics.sh --zookeeper zookeeper1:2181/kafka --create --topic awesome-kafka --replication-factor 1 --partitions 3**

**Describe Topic**

**./bin/kafka-topics.sh --zookeeper zookeeper1:2181/kafka --topic awesome-kafka --describe**

**From producer**

**./bin/kafka-console-producer.sh --broker-list localhost:9092 --topic awesome-kafka**

**From Consumer**

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

**Reference :**

**https://github.com/naddym/kafka-setup-on-ubuntu?tab=readme-ov-file#apache-kafka-cluster-3-brokers-setup-on-ubuntu**

Troubleshooting Steps

**1) If you encounter an error stating bash: ./bin/kafka-topics.sh: No such file or directory** **after restarting your Kafka server, it likely means that you are not in the correct directory where Kafka was installed, or the Kafka installation might have been corrupted or moved.**

Here’s how to resolve the issue:

### Step-by-Step Guide

1. **Find the Kafka Installation Directory**

Locate the directory where Kafka is installed. Common installation directories include /usr/local/kafka or /opt/kafka. You can use the find command to locate the Kafka installation directory if you’re unsure:

bash

Copy code

sudo find / -name "kafka-topics.sh"

This command will search your entire filesystem for the kafka-topics.sh script and output its path.

1. **Navigate to the Kafka Installation Directory**

Once you’ve located the Kafka installation directory, navigate to it using the cd command. For example, if Kafka is installed in /usr/local/kafka, you would run:

bash

Copy code

cd /usr/local/kafka

1. **Verify the bin Directory Exists**

Verify that the bin directory exists and contains the necessary scripts:

bash

Copy code

ls bin

You should see a list of scripts like kafka-server-start.sh, kafka-topics.sh, zookeeper-server-start.sh, etc.

1. **Run the Kafka Commands**

Now, you can run the Kafka commands using the appropriate script. For example, to create a topic:

bash

Copy code

./bin/kafka-topics.sh --create --topic awesome-kafka --zookeeper zookeeper1:2181/kafka --replication-factor 1 --partitions 1

### Example Commands

1. **Start ZooKeeper (if not already running)**

bash

Copy code

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

1. **Start Kafka Server**

bash

Copy code

./bin/kafka-server-start.sh config/server.properties &

1. **Create a Topic**

bash

Copy code

./bin/kafka-topics.sh --create --topic awesome-kafka --zookeeper zookeeper1:2181/kafka --replication-factor 1 --partitions 1

1. **List Topics**

bash

Copy code

./bin/kafka-topics.sh --list --zookeeper zookeeper1:2181/kafka

1. **Describe a Topic**

bash

Copy code

./bin/kafka-topics.sh --describe --topic awesome-kafka --zookeeper zookeeper1:2181/kafka

**2)The error message you're seeing indicates that there are no available Kafka brokers to create the topic with the specified replication factor. Here’s how you can troubleshoot and resolve this issue:**

### Step-by-Step Guide to Resolve the Issue

1. **Ensure Kafka Broker is Running**

First, make sure that your Kafka broker is up and running. If it is not running, start it using the following command:

bash

Copy code

./bin/kafka-server-start.sh config/server.properties &

1. **Check Kafka Broker Status**

Verify that the Kafka broker is running by checking the logs:

bash

Copy code

tail -f /opt/kafka/logs/server.log

Look for messages indicating that the broker has started successfully.

1. **Ensure ZooKeeper is Running**

If your Kafka setup uses ZooKeeper, ensure that the ZooKeeper server is also running:

bash

Copy code

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

Check ZooKeeper logs to ensure it is running without issues:

bash

Copy code

tail -f /opt/kafka/logs/zookeeper.log

1. **Verify Kafka Broker Registration with ZooKeeper**

Kafka brokers register themselves with ZooKeeper. Ensure that the broker has registered by using the ZooKeeper shell to list the brokers:

bash

Copy code

./bin/zookeeper-shell.sh zookeeper1:2181

Once in the ZooKeeper shell, list the brokers:

bash

Copy code

ls /brokers/ids

You should see a list of broker IDs. If the list is empty, your broker has not registered correctly.

1. **Check Network Configuration**

Ensure that the broker's advertised listeners are correctly configured in the server.properties file. The advertised.listeners property should be set to the broker's IP address and port:

properties

Copy code

advertised.listeners=PLAINTEXT://192.168.100.81:9092

Save any changes and restart the Kafka broker:

bash

Copy code

./bin/kafka-server-stop.sh

./bin/kafka-server-start.sh config/server.properties &

1. **Create the Topic Again**

Once you have verified that the Kafka broker is running and registered with ZooKeeper, try creating the topic again:

bash

Copy code

./bin/kafka-topics.sh --create --topic awesome-kafka --zookeeper zookeeper1:2181/kafka

**3)The error message indicates that the broker.id configuration value in your Kafka server.properties file is not correctly set as an integer. It seems there's a comment or an invalid value within the configuration line. Here’s how you can resolve this issue:**

### Step-by-Step Guide to Fix broker.id Configuration

1. **Open the server.properties File**

Edit the server.properties file located in your Kafka configuration directory:

bash

Copy code

nano /opt/kafka/config/server.properties

1. **Locate the broker.id Configuration**

Find the line that sets the broker.id. It should look something like this:

properties

Copy code

broker.id=1 # example, broker.id=1 for server1, broker.id=2 for server2 and broker.id=3 for server 3

1. **Correct the broker.id Configuration**

Remove the comment from the line or place it on a separate line. The broker.id should be a single integer value. Correct it to look like this:

properties

Copy code

broker.id=1

If you need to add a comment, place it on a separate line:

properties

Copy code

broker.id=1

# example, broker.id=1 for server1, broker.id=2 for server2 and broker.id=3 for server 3

1. **Save the Changes**

Save the changes and exit the editor. In nano, you can do this by pressing Ctrl + O, then Enter to save, and Ctrl + X to exit.

1. **Restart Kafka Broker**

Restart the Kafka broker to apply the changes:

bash

Copy code

./bin/kafka-server-stop.sh

./bin/kafka-server-start.sh config/server.properties &

**4) The error message indicates that the zookeeper option is not recognized when using the Kafka TopicCommand. This typically happens when there is a version mismatch or a configuration issue. Here are the steps to troubleshoot and resolve the issue:**

**1. Verify Kafka Version**

First, ensure that you are using a compatible version of Kafka. Check the version of Kafka you have installed:

bash

Copy code

bin/kafka-topics.sh –version

**2. Check Command Syntax**

The kafka-topics.sh command has different options depending on the version of Kafka you are using. In newer versions of Kafka (2.4.0 and later), ZooKeeper is not directly used for topic management; instead, the --bootstrap-server option is used.

**3. Correct Command Usage**

If you are using a recent version of Kafka, use the --bootstrap-server option instead of --zookeeper. Here’s an example:

bash

Copy code

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

**4. If Using Older Kafka Versions**

If you are using an older version of Kafka (prior to 2.4.0) that still uses ZooKeeper, you would use:

bash

Copy code

bin/kafka-topics.sh --create --topic my\_topic --zookeeper localhost:2181 --partitions 1 --replication-factor 1

**5. Ensure Correct Path and Permissions**

Ensure that you are in the correct directory and have the necessary permissions. If you installed Kafka in /usr/local/kafka, navigate to that directory:

bash

Copy code

cd /usr/local/kafka

Ensure that you have the right permissions to execute the script:

bash

Copy code

chmod +x bin/kafka-topics.sh

**Example Commands**

**Creating a Topic (Recent Versions of Kafka):**

bash

Copy code

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

**Listing Topics (Recent Versions of Kafka):**

bash

Copy code

bin/kafka-topics.sh --list --bootstrap-server localhost:9092

**Describing a Topic (Recent Versions of Kafka):**

bash

Copy code

bin/kafka-topics.sh --describe --topic my\_topic --bootstrap-server localhost:9092

**Summary**

* Use --bootstrap-server for Kafka versions 2.4.0 and later.
* Use --zookeeper for older versions of Kafka.
* Verify the Kafka version and ensure compatibility.
* Ensure you are in the correct directory and have necessary permissions.