

How to Install Apache Kafka on Debian 11/10

tecadmin.net/install-apache-kafka-debian/

By Rahul

April 5, 2020



Apache Kafka is a distributed streaming platform. It is useful for building real-time streaming data pipelines to get data between the systems or applications. Another useful feature is real-time streaming applications that can transform streams of data or react on a stream of data.

This tutorial will help you to install Apache Kafka on Debian 11, Debian 10, and Debian 9 Linux systems.

Step 1 – Install Java

Apache Kafka required Java to run. You must have java installed on your system. Execute below command to install default OpenJDK on your system from the official PPA's.

```
sudo apt update  
sudo apt install default-jdk
```

Step 2 – Download Apache Kafka

Download the Apache Kafka binary files from its official [download](https://kafka.apache.org/downloads) website. You can also select any nearby mirror to download.

```
wget https://dlcdn.apache.org/kafka/3.2.0/kafka_2.13-3.2.0.tgz
```

Then extract the archive file

```
tar xzf kafka_2.13-3.2.0.tgz
sudo mv kafka_2.13-3.2.0 /usr/local/kafka
```

Step 3 – Create Systemd Unit Files

Next, create systemd unit files for the Zookeeper and Kafka service. This will help to manage Kafka services to start/stop using the systemctl command.

First, create systemd unit file for Zookeeper with below command:

```
vim /etc/systemd/system/zookeeper.service
```

Add below content:

```
[Unit]
Description=Apache Zookeeper server
Documentation=http://zookeeper.apache.org
Requires=network.target remote-fs.target
After=network.target remote-fs.target

[Service]
Type=simple
ExecStart=/usr/local/kafka/bin/zookeeper-server-start.sh
/usr/local/kafka/config/zookeeper.properties
ExecStop=/usr/local/kafka/bin/zookeeper-server-stop.sh
Restart=on-abnormal

[Install]
WantedBy=multi-user.target
```

Save the file and close it.

Next, to create a Kafka systemd unit file using the following command:

```
vim /etc/systemd/system/kafka.service
```

Add the below content. Make sure to set the correct **JAVA_HOME** path as per the Java installed on your system.

```
[Unit]
Description=Apache Kafka Server
Documentation=http://kafka.apache.org/documentation.html
Requires=zookeeper.service

[Service]
Type=simple
Environment="JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64"
ExecStart=/usr/local/kafka/bin/kafka-server-start.sh
/usr/local/kafka/config/server.properties
ExecStop=/usr/local/kafka/bin/kafka-server-stop.sh

[Install]
WantedBy=multi-user.target
```

Save file and close.

Reload the systemd daemon to apply new changes.

```
systemctl daemon-reload
```

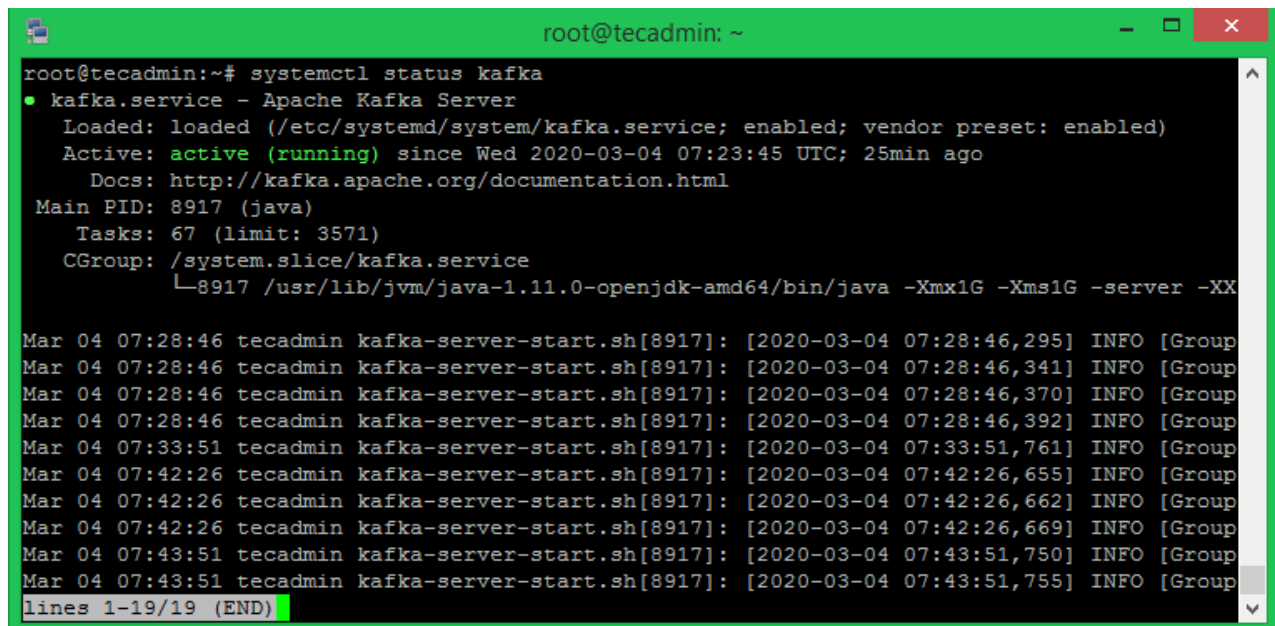
Step 4 – Start Kafka Server

Kafka required ZooKeeper so first, start a ZooKeeper server on your system. You can use the script available with Kafka to get start a single-node ZooKeeper instance.

```
sudo systemctl start zookeeper
```

Now start the Kafka server and view the running status:

```
sudo systemctl start kafka
sudo systemctl status kafka
```

A terminal window titled 'root@tecadmin: ~' with a green title bar. The terminal shows the command 'systemctl status kafka' and its output. The output indicates that the 'kafka.service' is 'loaded' and 'active (running)'. It also shows the main PID as 8917 (java) and the tasks as 67. Below this, there are several log entries from 'kafka-server-start.sh' showing INFO messages for various groups. The terminal text is as follows:

```
root@tecadmin:~# systemctl status kafka
● kafka.service - Apache Kafka Server
   Loaded: loaded (/etc/systemd/system/kafka.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2020-03-04 07:23:45 UTC; 25min ago
     Docs: http://kafka.apache.org/documentation.html
   Main PID: 8917 (java)
    Tasks: 67 (limit: 3571)
   CGroup: /system.slice/kafka.service
           └─8917 /usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -Xmx1G -Xms1G -server -XX

Mar 04 07:28:46 tecadmin kafka-server-start.sh[8917]: [2020-03-04 07:28:46,295] INFO [Group
Mar 04 07:28:46 tecadmin kafka-server-start.sh[8917]: [2020-03-04 07:28:46,341] INFO [Group
Mar 04 07:28:46 tecadmin kafka-server-start.sh[8917]: [2020-03-04 07:28:46,370] INFO [Group
Mar 04 07:28:46 tecadmin kafka-server-start.sh[8917]: [2020-03-04 07:28:46,392] INFO [Group
Mar 04 07:33:51 tecadmin kafka-server-start.sh[8917]: [2020-03-04 07:33:51,761] INFO [Group
Mar 04 07:42:26 tecadmin kafka-server-start.sh[8917]: [2020-03-04 07:42:26,655] INFO [Group
Mar 04 07:42:26 tecadmin kafka-server-start.sh[8917]: [2020-03-04 07:42:26,662] INFO [Group
Mar 04 07:42:26 tecadmin kafka-server-start.sh[8917]: [2020-03-04 07:42:26,669] INFO [Group
Mar 04 07:43:51 tecadmin kafka-server-start.sh[8917]: [2020-03-04 07:43:51,750] INFO [Group
Mar 04 07:43:51 tecadmin kafka-server-start.sh[8917]: [2020-03-04 07:43:51,755] INFO [Group
lines 1-19/19 (END)
```

All done. The Kafka installation has been successfully completed. The part of this tutorial will help you to work with the Kafka server.

Step 5 – Create a Topic in Kafka

Kafka provides multiple pre-built shell script to work on it. First, create a topic named “testTopic” with a single partition with single replica:

```
cd /usr/local/kafka
bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-
factor 1 --partitions 1 --topic testTopic
```

Created topic testTopic.

The replication-factor describes how many copies of data will be created. As we are running with a single instance keep this value 1.

Set the partitions options as the number of brokers you want your data to be split between. As we are running with a single broker keep this value 1.

You can create multiple topics by running the same command as above. After that, you can see the created topics on Kafka by the running below command:

```
bin/kafka-topics.sh --list --zookeeper localhost:9092

testTopic
TecAdminTutorial1
TecAdminTutorial2
```

Alternatively, instead of manually creating topics you can also configure your brokers to auto-create topics when a non-existent topic is published to.

Step 6 – Send Messages to Kafka

The “producer” is the process responsible for put data into our Kafka. The Kafka comes with a command-line client that will take input from a file or from standard input and send it out as messages to the Kafka cluster. The default Kafka sends each line as a separate message.

Let’s run the producer and then type a few messages into the console to send to the server.

```
bin/kafka-console-producer.sh --broker-list localhost:9092 --topic testTopic

>Welcome to kafka
>This is my first topic
>
```

You can exit this command or keep this terminal running for further testing. Now open a new terminal to the Kafka consumer process on the next step.

Step 7 – Using Kafka Consumer

Kafka also has a command-line consumer to read data from the Kafka cluster and display messages to standard output.

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

Welcome to kafka
This is my first topic
```

Now, If you have still running Kafka producer (Step #6) in another terminal. Just type some text on that producer terminal. it will immediately visible on consumer terminal. See the below screenshot of Kafka producer and consumer in working:

```
root@tecadmin: /usr/local/kafka
root@tecadmin:/usr/local/kafka# bin/kafka-console-producer.sh --broker-list localhost:9092 --topic testTopic
>hello
>tecadmin.net
>it's working
>
```

(Producer Node)

```
root@tecadmin: /usr/local/kafka
root@tecadmin:/usr/local/kafka# bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic testTopic --from-beginning
Welcome to kafka
This is my first topic
hi
hello
tecadmin.net
it's working
```

(Consumer Node)

Conclusion

You have successfully installed and configured the Kafka service on a Debian Linux system.