# How to Install Apache Kafka on CentOS 7

Last Updated: Wed, May 18, 2016

CentOS Linux Guides Programming Server Apps System Admin

Apache Kafka is a scalable and high-throughtput messaging system which is capable of efficiently handling a huge amount of data.

You can either deploy Kafka on one server or build a distributed Kafka cluster for greater performance. As a starter, this article explains how to install Apache Kafka on one single Vultr CentOS 7 server instance.

## Prerequisites:

Before moving on, you should:

- Deploy a Vultr CentOS 7 server instance. Depending on your needs, you may need to increase the available memory.
- Use a sudo user to log in from your SSH terminal.

## Step 1: Update the CentOS 7 system

Use the command below to update your system to the latest stable status:

```
sudo yum update -y && sudo reboot
```

After the reboot has finished, use the same sudo user to log in again.

## Step 2: Install OpenJDK Runtime

You need to setup a Java virtual machine on your system before you can run Apache Kafka properly. Here, you can install OpenJDK Runtime Environment 1.8.0 using YUM:

```
sudo yum install java-1.8.0-openjdk.x86_64
```

Validate your installation with:

```
java -version
```

The output should resemble:

```
openjdk version "1.8.0_91"

OpenJDK Runtime Environment (build 1.8.0_91-b14)

OpenJDK 64-Bit Server VM (build 25.91-b14, mixed mc
```

You also need to setup the "JAVA_HOME" and "JRE_HOME" environment variables:

```
sudo vi /etc/profile
```

Append the following lines to the original content of the file:

```
export JAVA_HOME=/usr/lib/jvm/jre-1.8.0-openjdk

export JRE_HOME=/usr/lib/jvm/jre
```

Save and quit:

```
:wq
```

Reload the profile to put your changes into effect:

```
source /etc/profile
```

## Step 3: Download Apache Kafka

Download the latest stable version of Apache Kafka from the official website. At the time of writing, it's `0.9.0.1` .

```
cd ~

wget http://www-us.apache.org/dist/kafka/0.9.0.1/ka
```

Unzip the archive to a preferred location, such as `/opt` :

```
tar -xvf kafka_2.11-0.9.0.1.tgz

sudo mv kafka_2.11-0.9.0.1 /opt
```

## Step 4: Start and test Apache Kafka

At this point, Apache Kafka is available on your system. Let's give it a test drive.

### 4.1: Get into the Kafka directory

```
cd /opt/kafka_2.11-0.9.0.1
```

## 4.2: Start the Zookeeper server

```
bin/zookeeper-server-start.sh -daemon config/zookee
```

## 4.3: Modify the configuration of your Kafka server

```
 vi bin/kafka-server-start.sh
```

Adjust the memory usage according to your specific system parameters. For example, if you are using a Vultr server instance with 768MB memory in the test environment, you need to locate the following line:

```
export KAFKA_HEAP_OPTS="-Xmx1G -Xms1G"
```

Replace it with:

```
export KAFKA_HEAP_OPTS="-Xmx256M -Xms128M"
```

Save an quit:

```
:wq
```

## 4.4: Start the Kafka server

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

If everything went successfully, you will see several messages about the Kafka server's status, and the last one will read:

```
INFO [Kafka Server 0], started (kafka.server.KafkaS
```

This means that you have started the Kafka server.

## 4.5: Create a topic "test" in a new SSH connection

Open a new SSH connection, use the following commands to create a topic "test":

```
 --replication-factor 1 --partitions 1 --topic test
```

You can view your topics with the following command:

```
ı/kafka-topics.sh --list --zookeeper localhost:2181
```

In our case, The output will read:

```
test
```

## 4.6: Produce messages using the topic "test"

```
bin/kafka-console-producer.sh --broker-list localhc
```

Using the command above, you can input any number of messages as you wish, such as:

```
Welcome aboard!

Bonjour!
```

If you receive an error similar to `"WARN Error while fetching metadata with correlation id"` while inputting a message, you'll need to update the `server.properties` file with the following info:

```
port = 9092

advertised.host.name = localhost
```

## 4.7: Display messages

Open a third SSH connection, and then run the following commands:

```
cd /opt/kafka_2.11-0.9.0.1

bin/kafka-console-consumer.sh --zookeeper localhost
```

Ta-da! The messages you produced earlier will display in the third SSH connection. Of course, if you input more messages from the second SSH connection now, you will immediately see them on the third SSH connection.

Finally, you can press Ctrl+C on each SSH connection to stop these scripts.

That's it. You can learn more about Apache Kafka on the official website. Have fun!