Step 1: Install Java and Bookeeper

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

```
sudo apt-get update
sudo apt-get install default-jre
```

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

```
sudo apt-get install zookeeperd
```

You will now need to check if Zookeeper is alive and if it’s OK

```
telnet localhost 2181
```

at Telnet prompt, You will have to enter

```
ruok
```

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

```
imok
```

Step 2: Create a Service User for Kafka

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

```
$ sudo adduser kafka
```

Follow the Tabs and set the password to create **Kafka User**. Now, you have to add the User to the Sudo Group, using the following command:

```
$ sudo adduser kafka sudo
```

Now, your User is ready, you need to log in using, the following command:

```
$ su -l kafka
```

Step 3: Download Apache Kafka

Now, you need to download and extract Kafka binaries in your Kafka user’s home directory. You can create your directory using the following command:

```
$ mkdir ~/Downloads
```

You need to download the Kafka binaries using Curl:

```
$ curl "https://downloads.apache.org/kafka/2.6.2/kafka_2.13-2.6.2.tgz" -o
~/Downloads/kafka.tgz
```

Create a new directory called Kafka and change your path to this directory to make it your base directory.

```
$ mkdir ~/kafka && cd ~/kafka
```

Now simply extract the archive you have downloaded using the following command:

```
$ tar -xvzf ~/Downloads/kafka.tgz --strip 1
```

–strip 1 is used to ensure that the archived data is extracted in **~/kafka/**.

Step 4: Configuring Kafka Server

The default behavior of Kafka prevents you from deleting a topic. Messages can be published to a Kafka topic, which is a category, group, or feed name. You must edit the configuration file to change this.

The **server.properties** file specifies Kafka's configuration options. Use nano or your favorite editor to open this file:

```
$ nano ~/kafka/config/server.properties
```

Add a setting that allows us to delete Kafka topics first. Add the following to the file's bottom:

```
delete.topic.enable = true
```

Now change the directory for storing logs:

```
log.dirs=/home/kafka/logs
```

Now you need to Save and Close the file. The next step is to set up Systemd Unit Files.

Step 5: Setting Up Kafka Systemd Unit Files

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

Create systemd unit file for Zookeeper with below command:

```
$ sudo nano /etc/systemd/system/zookeeper.service
```

Next, you need to add the below content:

```
[Unit]
Requires=network.target remote-fs.target
After=network.target remote-fs.target

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

[Install]
```

```
WantedBy=multi-user.target
```

Save this file and then close it. Then you need to create a Kafka systemd unit file using the following command snippet:

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

Now, you need to enter the following unit definition into the file:

```
[Unit]
Requires=zookeeper.service
After=zookeeper.service

[Service]
Type=simple
User=kafka
ExecStart=/bin/sh -c '/home/kafka/kafka/bin/kafka-server-start.sh
/home/kafka/kafka/config/server.properties > /home/kafka/kafka/kafka.log 2>&1'
ExecStop=/home/kafka/kafka/bin/kafka-server-stop.sh
Restart=on-abnormal

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

This unit file is dependent on **zookeeper.service,** as specified in the [Unit] section. This will ensure that zookeeper is started when the Kafka service is launched.
The [Service] line specifies that systemd should start and stop the service using the **kafka-server-start.sh** and **Kafka-server-stop.sh** shell files. It also indicates that if Kafka exits abnormally, it should be restarted.
After you've defined the units, use the following command to start Kafka:

```
$ sudo systemctl start kafka
```

Check the Kafka unit's journal logs to see if the server has started successfully:

```
$ sudo systemctl status kafka
```

Output:

```
kafka.service
     Loaded: loaded (/etc/systemd/system/kafka.service; disabled; vendor preset:
enabled)
     Active: active (running) since Wed 2021-02-10 00:09:38 UTC; 1min 58s ago
   Main PID: 55828 (sh)
      Tasks: 67 (limit: 4683)
     Memory: 315.8M
     CGroup: /system.slice/kafka.service
             ├─55828 /bin/sh -c /home/kafka/kafka/bin/kafka-server-start.sh
/home/kafka/kafka/config/server.properties > /home/kafka/kafka/kafka.log 2>&1
             └─55829 java -Xmx1G -Xms1G -server -XX:+UseG1GC -
XX:MaxGCPauseMillis=20 -XX:InitiatingHeapOccupancyPercent=35 -XX:
+ExplicitGCInvokesConcurrent -XX:MaxInlineLevel=15 -Djava.awt.headless=true -
Xlog:gc*:file=>

Feb 10 00:09:38 cart-67461-1 systemd[1]: Started kafka.service.
```

On port 9092, you now have a Kafka server listening.

The Kafka service has been begun. But if you rebooted your server, Kafka would not restart automatically. To enable the Kafka service on server boot, run the following commands:

```
$ sudo systemctl enable zookeeper
$ sudo systemctl enable kafka
```

You have successfully done the setup and installation of the Kafka server.

Step 6: Testing installation

In this stage, you'll put your Kafka setup to the test. To ensure that the Kafka server is functioning properly, you will publish and consume a "Hello World" message.

In order to publish messages in Kafka, you must first:

- A producer who allows records and data to be published to topics.
- A person who reads communications and data from different themes.

To get started, make a new topic called **MituResearch**:

```
$ ~/kafka/bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-
factor 1 --partitions 1 --topic MituResearch
```

The **kafka-console-producer.sh** script can be used to build a producer from the command line. As arguments, it expects the hostname, port, and topic of the Kafka server.

The string "Hello, World" should now be published to the **MituResearch** topic:

```
$ echo "Hello, World" | ~/kafka/bin/kafka-console-producer.sh --broker-list
localhost:9092 --topic MituResearch > /dev/null
```

Using the **Kafka-console-consumer.sh** script, establish a Kafka consumer. As parameters, it requests the ZooKeeper server's hostname and port, as well as a topic name.

Messages from **MituResearch** are consumed by the command below. Note the usage of the — from-beginning flag, which permits messages published before the consumer was launched to be consumed:

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

Hello, World will appear in your terminal if there are no configuration issues:

```
Hello, World
```

The script will keep running while it waits for further messages to be published. Open a new terminal window and log into your server to try this.
Start a producer in this new terminal to send out another message:

```
$ echo "Hello World from Sammy at DigitalOcean!" | ~/kafka/bin/kafka-console-
producer.sh --broker-list localhost:9092 --topic MituResearch > /dev/null
```

This message will appear in the consumer's output:

```
Hello, World
```

```
Hello World from Sammy at DigitalOcean!
```

To stop the consumer script, press CTRL+C once you've finished testing.
On Ubuntu 20.04, you've now installed and set up a Kafka server.

You'll do a few fast operations to tighten the security of your Kafka server in the next phase.

Step 7: Hardening Kafka Server

You can now delete the Kafka user's admin credentials after your installation is complete. Log out and back in as any other non-root sudo user before proceeding. Type exit if you're still in the same shell session as when you started this tutorial.

Remove the Kafka user from the sudo group:

```
$ sudo deluser kafka sudo
```

Lock the Kafka user's password with the **passwd** command to strengthen the security of your Kafka server even more. This ensures that no one may use this account to log into the server directly:

```
$ sudo passwd kafka -l
```

Only root or a sudo user can log in as Kafka at this time by entering the following command:

```
$ sudo su - kafka
```

If you want to unlock it in the future, use **passwd** with the **-u** option:

```
$ sudo passwd kafka -u
```

You've now successfully restricted the admin capabilities of the Kafka user.