Step 1 — Creating a User for Kafka

- `su useradd kafka -m`

  `su passwd kafka`

- Your **kafka** user is now ready. Log into this account using `su`:
- `su -l kafka`
- Now that we've created the Kafka-specific user, we can move on to downloading and extracting the Kafka binaries.

Step 2 — Downloading and Extracting the Kafka Binaries

- `/HOME/KAFKA`
- `mkdir ~/Downloads`
- Install `curl` using `apt-get` so that you'll be able to download remote files:
- `sudo apt-get update && sudo apt-get upgrade`
- Once `curl` is installed, use it to download the Kafka binaries:
- `wget "https://www.apache.org/dist/kafka/2.1.1/kafka_2.12-2.4.0.tgz"`
- Create a directory called `kafka` and change to this directory. This will be the base directory of the Kafka installation:
- `Cd kafka_2.12-2.4.0`
- Extract the archive you downloaded using the `tar` command:
- `tar -xvzf ~/Downloads/ kafka_2.12-2.4.0.tgz`
- .

Step 3 — Configuring the Kafka Server

Kafka's default behavior will not allow us to delete a *topic*, the category, group, or feed name to which messages can be published. To modify this, let's edit the configuration file.

Kafka's configuration options are specified in `server.properties`. Open this file with `nano` or your favorite editor:

- `nano ~/ kafka_2.12-2.4.0/config/server.properties`
- Let's add a setting that will allow us to delete Kafka topics. Add the following to the bottom of the file:

~/kafka/config/server.properties

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

Save the file, and exit `nano`. Now that we've configured Kafka, we can move on to creating systemd unit files for running and enabling it on startup.

Step 4 — Creating Systemd Unit Files and Starting the Kafka Server

In this section, we will create systemd unit files for the Kafka service. This will help us perform common service actions such as starting, stopping, and restarting Kafka in a manner consistent with other Linux services.

ZooKeeper is a service that Kafka uses to manage its cluster state and configurations. It is commonly used in many distributed systems as an integral component. If you would like to know more about it, visit the official ZooKeeper docs.

Create the unit file for `zookeeper`:

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

Enter the following unit definition into the file:

/etc/systemd/system/zookeeper.service

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

[Service]
Type=simple
User=kafka
ExecStart=/home/kafka/ kafka_2.12-2.4.0/bin/zookeeper-server-
start.sh /home/kafka/ kafka_2.12-2.4.0/config/zookeeper.properties
ExecStop=/home/kafka/ kafka_2.12-2.4.0/bin/zookeeper-server-stop.sh
Restart=on-abnormal

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

The `[Unit]` section specifies that ZooKeeper requires networking and the filesystem to be ready before it can start.

The `[Service]` section specifies that systemd should use the `zookeeper-server-start.sh` and `zookeeper-server-stop.sh` shell files for starting and stopping the service. It also specifies that ZooKeeper should be restarted automatically if it exits abnormally.

Next, create the systemd service file for `kafka`:

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

Enter the following unit definition into the file:

/etc/systemd/system/kafka.service

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

[Service]
Type=simple
```

```
User=kafka
ExecStart=/bin/sh -c '/home/kafka/ kafka_2.12-2.4.0/bin/kafka-server-start.sh /home/kafka/ kafka_2.12-2.4.0/config/server.properties > /home/kafka/ kafka_2.12-2.4.0/kafka.log 2>&1'
ExecStop=/home/kafka/ kafka_2.12-2.4.0/bin/kafka-server-stop.sh
Restart=on-abnormal

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

The `[Unit]` section specifies that this unit file depends on `zookeeper.service`. This will ensure that `zookeeper` gets started automatically when the `kafka` service starts.

The `[Service]` section specifies that systemd should use the `kafka-server-start.sh` and `kafka-server-stop.sh` shell files for starting and stopping the service. It also specifies that Kafka should be restarted automatically if it exits abnormally.

Now that the units have been defined, start Kafka with the following command:

- `sudo systemctl start kafka`
-

To ensure that the server has started successfully, check the journal logs for the `kafka` unit:

- `sudo journalctl -u kafka`
-

You should see output similar to the following:

```
Output
Mar 23 13:31:48 kafka systemd[1]: Started kafka.service.
```

You now have a Kafka server listening on port `9092`.

While we have started the `kafka` service, if we were to reboot our server, it would not be started automatically. To enable `kafka` on server boot, run:

- `sudo systemctl enable kafka`
-

Now that we've started and enabled the services, let's check the installation.

KAFKA CHEQUEO

Primero en un terminal
`bin/zookeeper-server-start.sh config/zookeeper.properties`
se lanza en un segundo terminal ---→ los dos quedarán bloquedos

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

Step 5 — Testing the Installation

Let's publish and consume a **"Hello World"** message to make sure the Kafka server is behaving correctly. Publishing messages in Kafka requires:

- A *producer*, which enables the publication of records and data to topics.
- A *consumer*, which reads messages and data from topics.

First, create a topic named `TutorialTopic` by typing:

```
/home/kafka/kafka_2.12-2.4.0/bin/kafka-topics.sh --create --
zookeeper localhost:2181 --replication-factor 1 --partitions 1 --
topic TutorialTopic
```

- You can create a producer from the command line using the `kafka-console-producer.sh` script. It expects the Kafka server's hostname, port, and a topic name as arguments.
  Publish the string `"Hello, World"` to the `TutorialTopic` topic by typing:

```
echo "Hello, World" | /home/kafka/ kafka_2.12-2.4.0/bin/kafka-
console-producer.sh --broker-list localhost:9092 --topic
TutorialTopic > /dev/null
```

- Next, you can create a Kafka consumer using the `kafka-console-consumer.sh` script. It expects the ZooKeeper server's hostname and port, along with a topic name as arguments.
  The following command consumes messages from `TutorialTopic`. Note the use of the `--from-beginning` flag, which allows the consumption of messages that were published before the consumer was started:

```
/home/kafka/ kafka_2.12-2.4.0/bin/kafka-console-consumer.sh --
bootstrap-server localhost:9092 --topic TutorialTopic --from-
beginning
```

- If there are no configuration issues, you should see `Hello, World` in your terminal:

```
Output
Hello, World
```