# How to Install Kafka

## Defining other terms you will encounter

- Topic: A topic is a common name used to store and publish a particular stream of data. For example if you would wish to store all the data about a page being clicked, you can give the Topic a name such as “Added Customer”.
- Partition: Every topic is split up into partitions (“baskets”). When a topic is created, the number of partitions need to be specified but can be increased later as need arises. Each message gets stored into partitions with an incremental id known as its Offset value.
- Kafka Broker: Every server with Kafka installed in it is known as a broker. It is a container holding several topics having their partitions.
- Zookeeper: Zookeeper manages Kafka’s cluster state and configurations.

## Apache Kafka Use Cases

The following are some of the applications where you can take advantage of Apache Kafka:

- **Message Broking**: In comparison to most messaging systems Kafka has better throughput, built-in partitioning, replication, and fault-tolerance which makes it a good solution for large scale message processing applications
- **Website Activity Tracking**
- **Log Aggregation**: Kafka abstracts away the details of files and gives a cleaner abstraction of log or event data as a stream of messages.
- **Stream Processing**: capturing data in real-time from event sources; storing these event streams durably for later retrieval; and routing the event streams to different destination technologies as needed

- **Event Sourcing**: This is a style of application design where state changes are logged as a time-ordered sequence of records.
- **Commit Log**: Kafka can serve as a kind of external commit-log for a distributed system. The log helps replicate data between nodes and acts as a re-syncing mechanism for failed nodes to restore their data.
- **Metrics:** This involves aggregating statistics from distributed applications to produce centralized feeds of operational data.

## Installing Apache Kafka on Ubuntu 20.04

Apache Kafka requires Java for it to run so we shall prepare our server and get every pre-requisite installed

### Step 1: Preparing your Ubuntu Server

Create a new user for kafka

```
sudo adduser kafka1
```

After setting up the user, add it to sudoer by

```
sudo adduser kafka1 sudo
```

Then change the user to kafka1

```
su - kafka1
```

The rest instruction is all below this user

Update your fresh Ubuntu 20.04 server and get Java installed as illustrated below.

Notice that, if you have already installed jre and jdk, skip this step

```
sudo apt update && sudo apt upgrade
```

```
sudo apt install default-jre wget git unzip -y

sudo apt install default-jdk -y
```

### Step 2: Fetch Kafka on Ubuntu 20.04

After Java is well installed, let us now fetch Kafka sources. Head over to https://kafka.apache.org/downloads and look for the version you want and get the sources under Binary downloads. Click on the one that is recommended by Kafka and you will be redirected to a page that has a link you can use to fetch it.

```
cd ~

wget https://downloads.apache.org/kafka/2.6.0/kafka_2.13-2.6.0.tgz

mkdir /usr/local/kafka-server && cd /usr/local/kafka-server

sudo tar -xvzf ~/kafka_2.13-2.6.0.tgz --strip 1
```

Archive's contents will be extracted into /usr/local/kafka-server/ due to –strip 1 flag set.

### Step 3: Create Kafka and Zookeeper Systemd Unit Files

Systemd unit files for Kafka and Zookeeper will pretty much help in performing common service actions such as starting, stopping, and restarting Kafka. This makes it adapt to how other services are started, stopped, and restarted which is beneficial and consistent.

Let us begin with Zookeeper service:

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

Add the configuration below to the .service file

```
[Unit]

Description=Apache Zookeeper Server

Requires=network.target remote-fs.target

After=network.target remote-fs.target



[Service]

Type=simple

ExecStart=/usr/local/kafka-server/bin/zookeeper-server-start.sh
/usr/local/kafka-server/config/zookeeper.properties

ExecStop=/usr/local/kafka-server/bin/zookeeper-server-stop.sh

Restart=on-abnormal



[Install]

WantedBy=multi-user.target
```

Then for Kafka service. Make sure your JAVA_HOME configs are well inputted or Kafka will not start.

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

Add the configuration below to the .service file

```
[Unit]

Description=Apache Kafka Server

Documentation=http://kafka.apache.org/documentation.html
```

```
Requires=zookeeper.service

After=zookeeper.service



[Service]

Type=simple

Environment="JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64"

ExecStart=/usr/local/kafka-server/bin/kafka-server-start.sh
/usr/local/kafka-server/config/server.properties

ExecStop=/usr/local/kafka-server/bin/kafka-server-stop.sh

Restart=on-abnormal



[Install]

WantedBy=multi-user.target
```

If you need help to locate the correct Java path, run the following command in your terminal window:

```
$ which javac
```

The resulting output provides the path to the Java binary directory.


Use the provided path to find the OpenJDK directory with the following command:

```
$ readlink -f /usr/bin/javac
```

The section of the path just before the /bin/javac directory needs to be assigned to the $JAVA_HOME variable.

After you are done adding the configurations, reload the systemd daemon to apply changes and then start the services. You can check their status as well.

```
sudo systemctl daemon-reload

sudo systemctl enable --now zookeeper

sudo systemctl enable --now kafka

sudo systemctl status kafka zookeeper
```

Use **:q** to quit

### Step 4: Install Cluster Manager for Apache Kafka (CMAK) | Kafka Manager

CMAK (previously known as Kafka Manager) is an opensource tool for managing Apache Kafka clusters developed by Yahoo.

```
cd ~

git clone https://github.com/yahoo/CMAK.git
```

### Step 5: Configure CMAK on Ubuntu 20.04

The minimum configuration is the zookeeper hosts which are to be used for CMAK (pka kafka manager) state. This can be found in the application.conf file in conf directory. Change cmak.zkhosts="my.zookeeper.host.com:2181" and you can also specify multiple zookeeper hosts by comma delimiting them, like

so: cmak.zkhosts="my.zookeeper.host.com:2181,other.zookeeper.host.com:2181". The host names can be ip addresses too.

```
$ nano ~/CMAK/conf/application.conf

cmak.zkhosts="localhost:2181
```

After you are done adding your zookeeper hosts, the command below will create a zip file which can be used to deploy the application. You should see a lot of output on your terminal as files are downloaded and compiled. Give it time to finish and compile because it takes a while.

```
cd ~/CMAK/

./sbt clean dist
```

When all is done, you should see a message like below:

```
[info] Your package is ready in
/home/tech/CMAK/target/universal/cmak-3.0.0.5.zip
```

Change into the directory where the zip file is located and unzip it:

```
$ cd ~/CMAK/target/universal

$ unzip cmak-3.0.0.5.zip

$ cd cmak-3.0.0.5
```

### Step 5: Starting the service and Accessing it

After extracting the produced zipfile, and changing the working directory to it as done in **Step 4**, you can run the service like this:

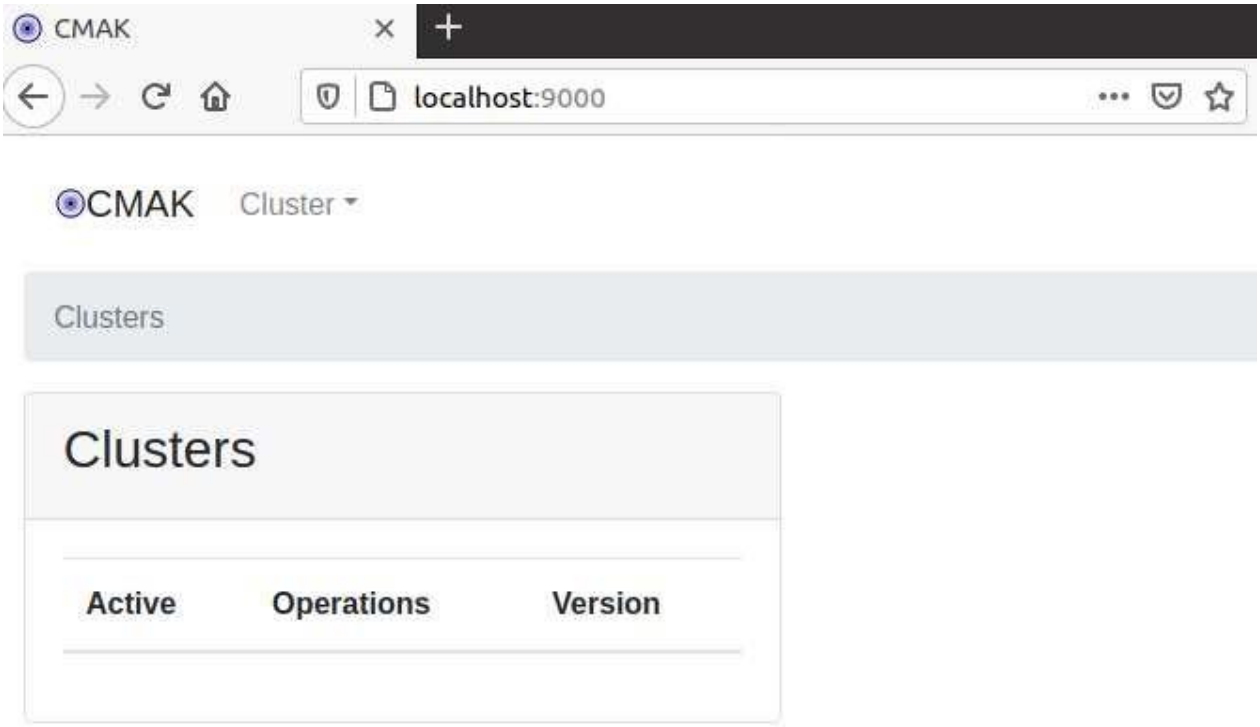
```
$ cd ~/CMAK/target/universal/cmak-3.0.0.5
```

```
$ bin/cmak
```

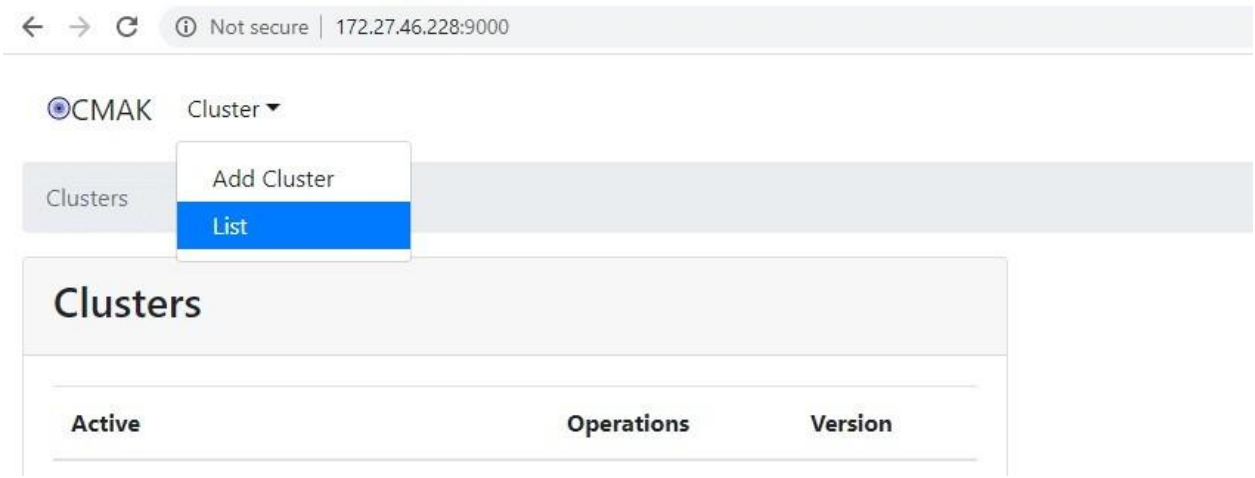
By default, it will choose port 9000, so open your favorite browser and point it to http://localhost:9000. In case your firewall is running, kindly allow the port to be accessed externally.

```
sudo ufw allow 9000
```

You should see an interface as shown below once everything is okay:

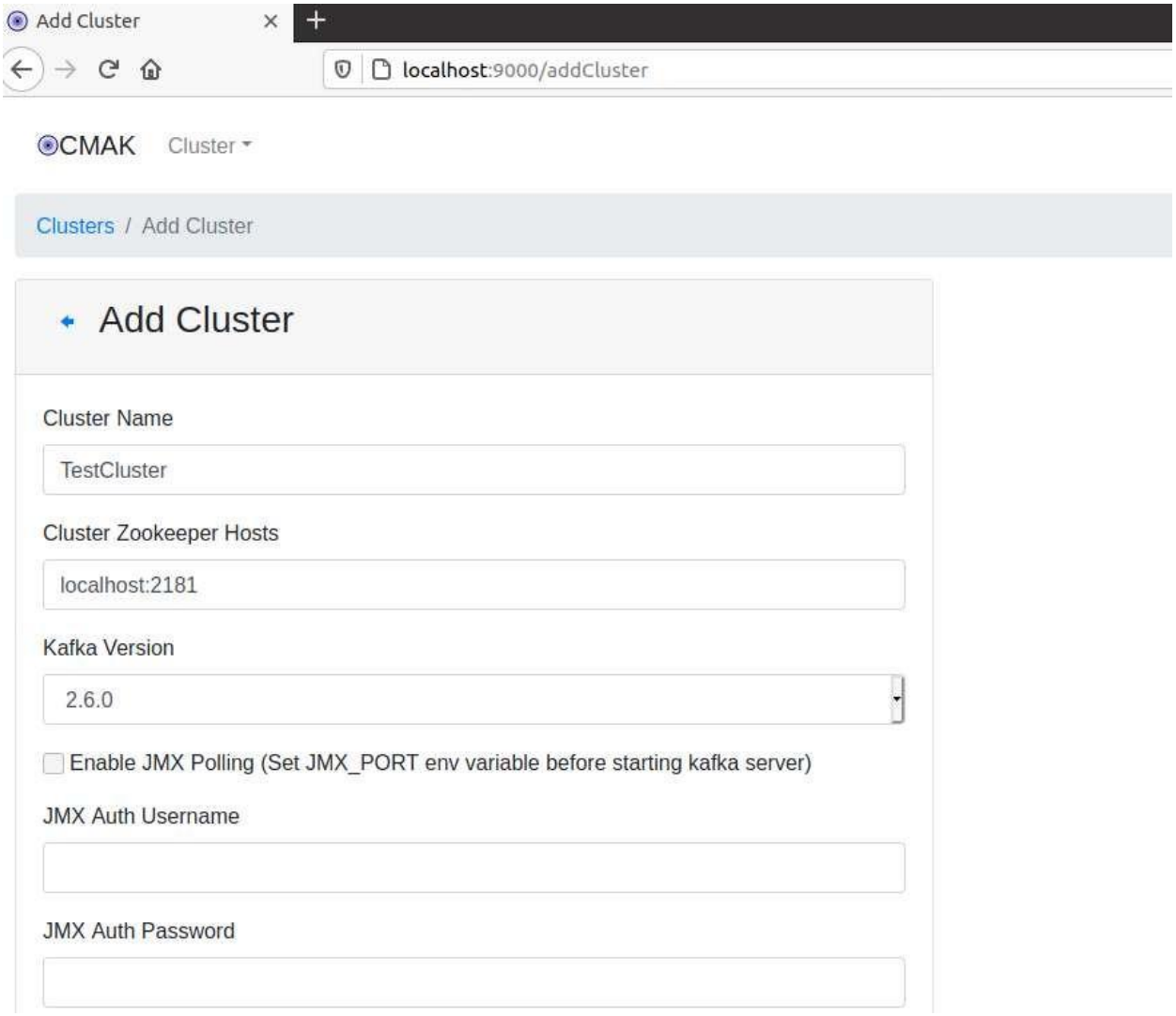



You will immediately notice that there is no cluster available when we first get into the interface as shown above. Therefore, we shall proceed to create a new cluster. Click on the “Cluster” drop-down list and then choose “Add Cluster“.

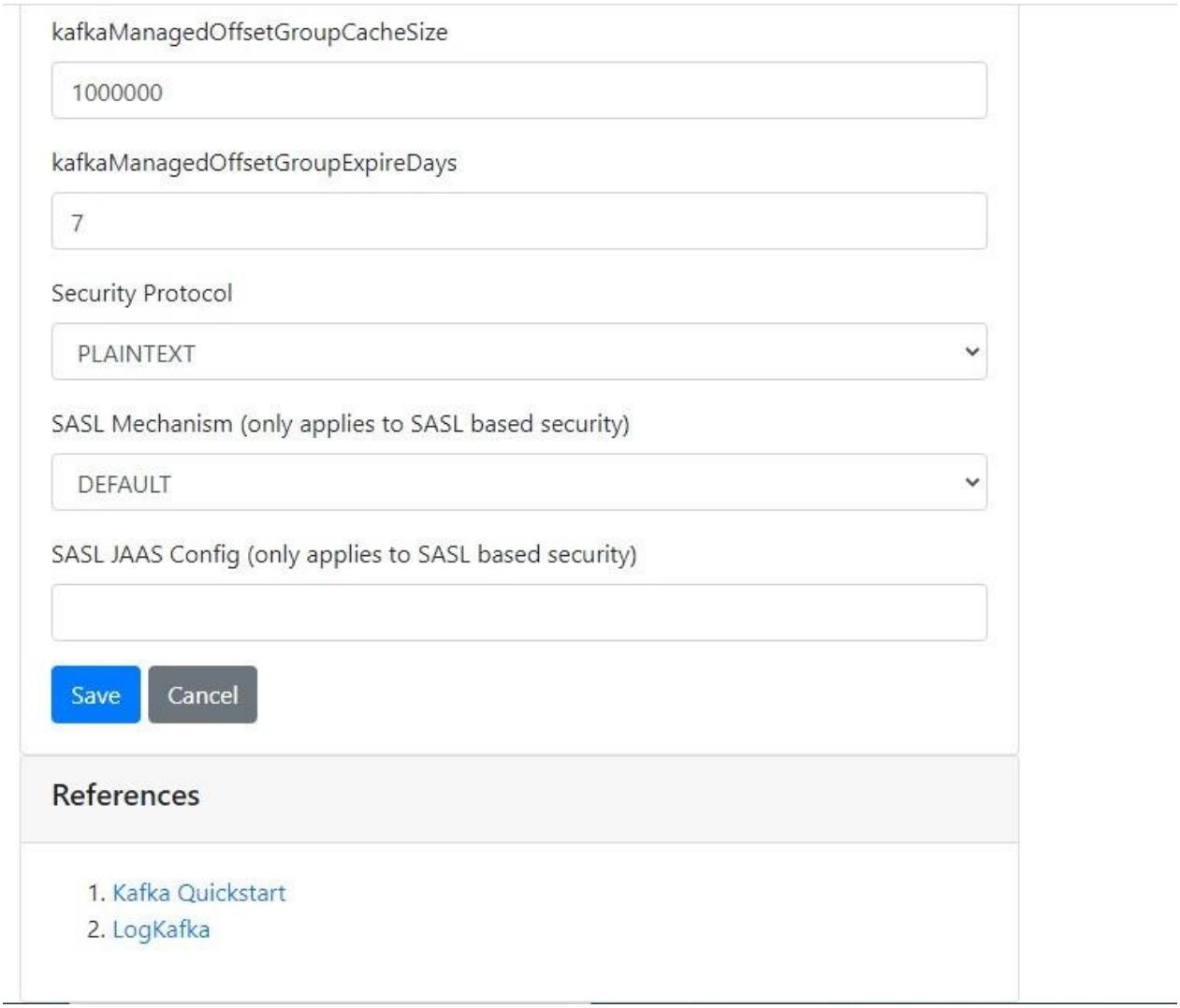



You will be presented with a page as shown below. Fill in the form with the details being requested (Cluster Name, Zookeeper Hosts etc). In case you have several Zookeeper Hosts, add them delimited by a comma. You can fill in the other details depending on your needs.

Sample filled fields




After everything is well filled to your satisfaction, scroll down and hit “Save“.




### Step 6: Adding Sample Topic

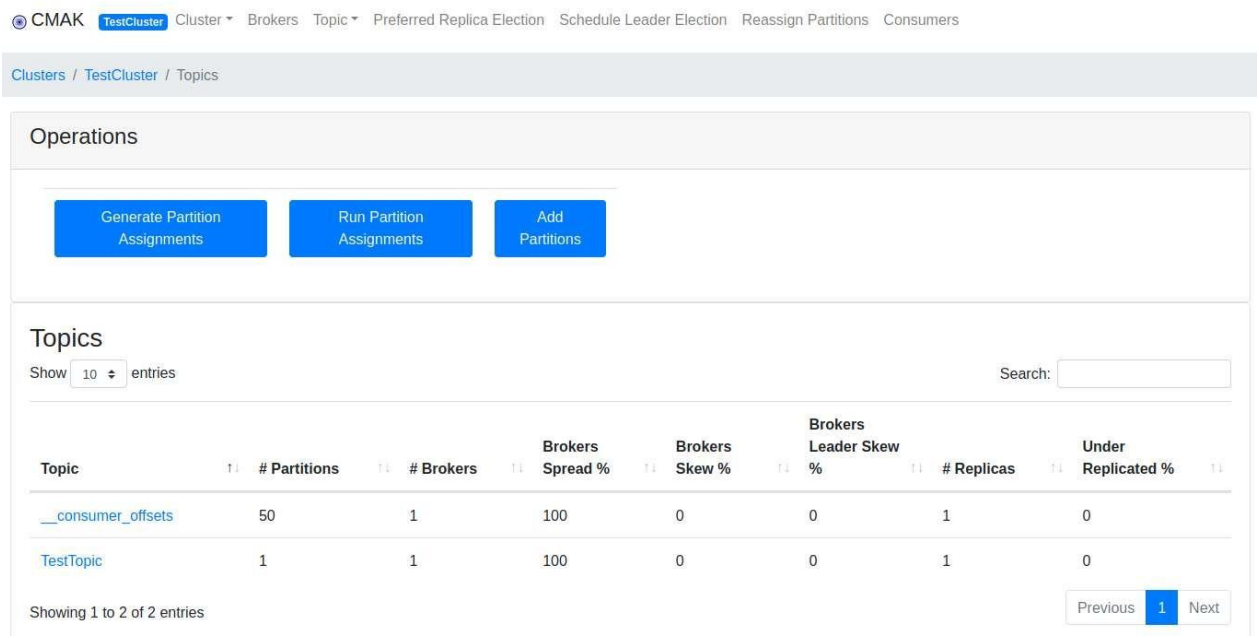
Apache Kafka provides multiple shell scripts to work with. Let us first create a sample topic called "ComputingForGeeksTopic" with a single partition with single replica. Open a new terminal leaving CMAK running and issue the command below:

```
cd /usr/local/kafka-server

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

```
Created topic TestTopic.
```

Confirm whether the topic is updated in CMAK interface. To do this, whilst in your cluster, Click Topic>List

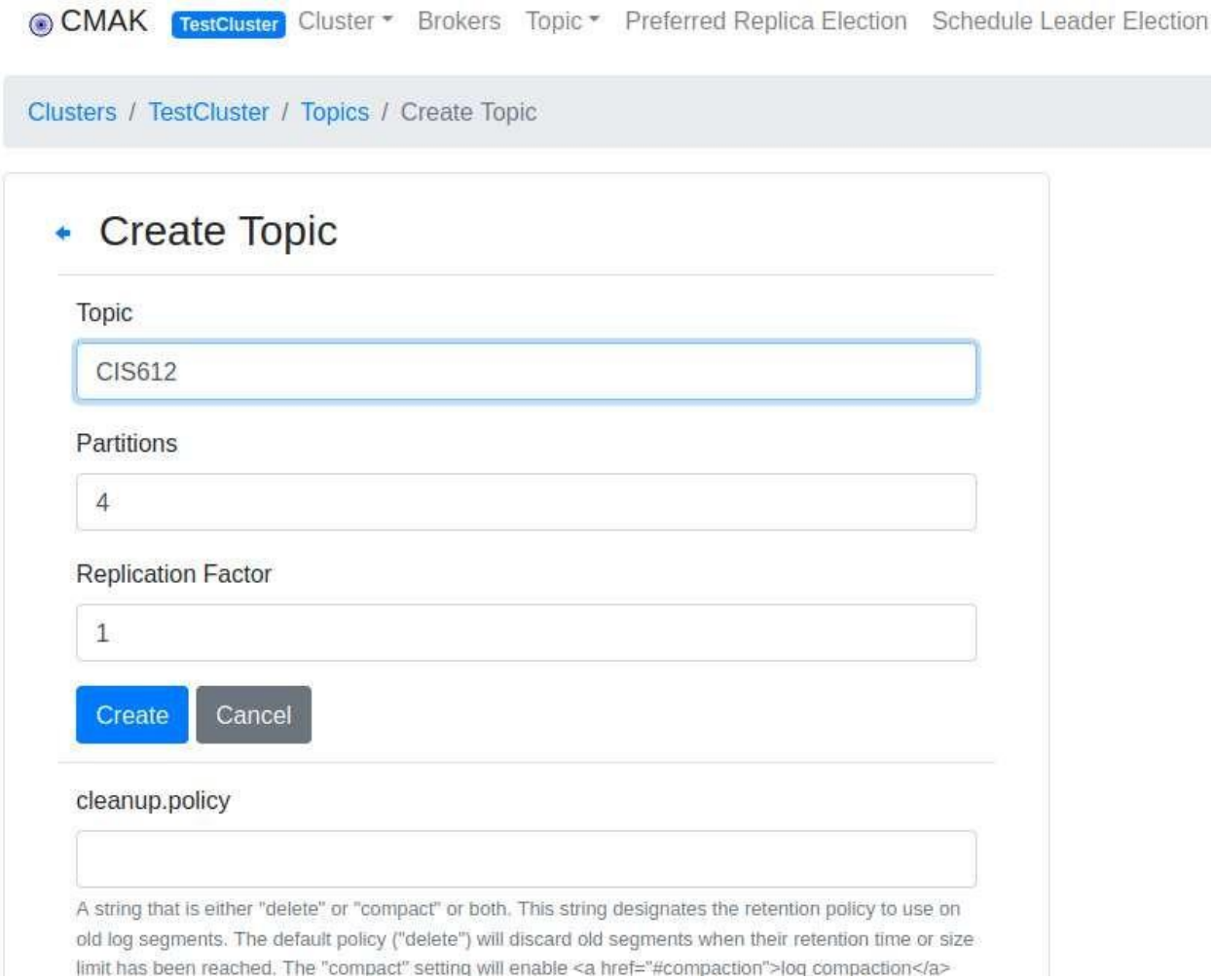



## Step 7: Create Topic in CMAK interface

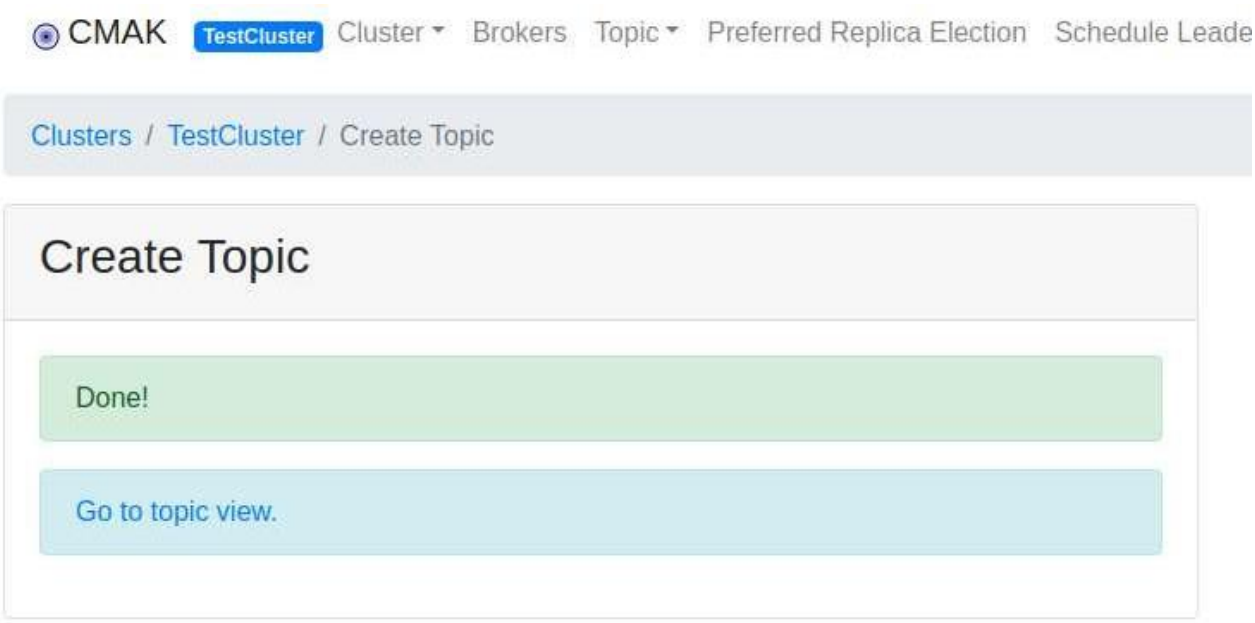
Another simpler way of creating a Topic is via the CMAK web interface. Simply click on "Topic" drop-down list and click on "Create". This is illustrated below. You will be required to input all the details you need about the new Topic (Replication Factor, Partitions and others). Fill in the form then click "Create" below it.

You will be required to input all the details you need about the new Topic (Replication Factor, Partitions and others). Fill in the form then click "Create" below the page you will

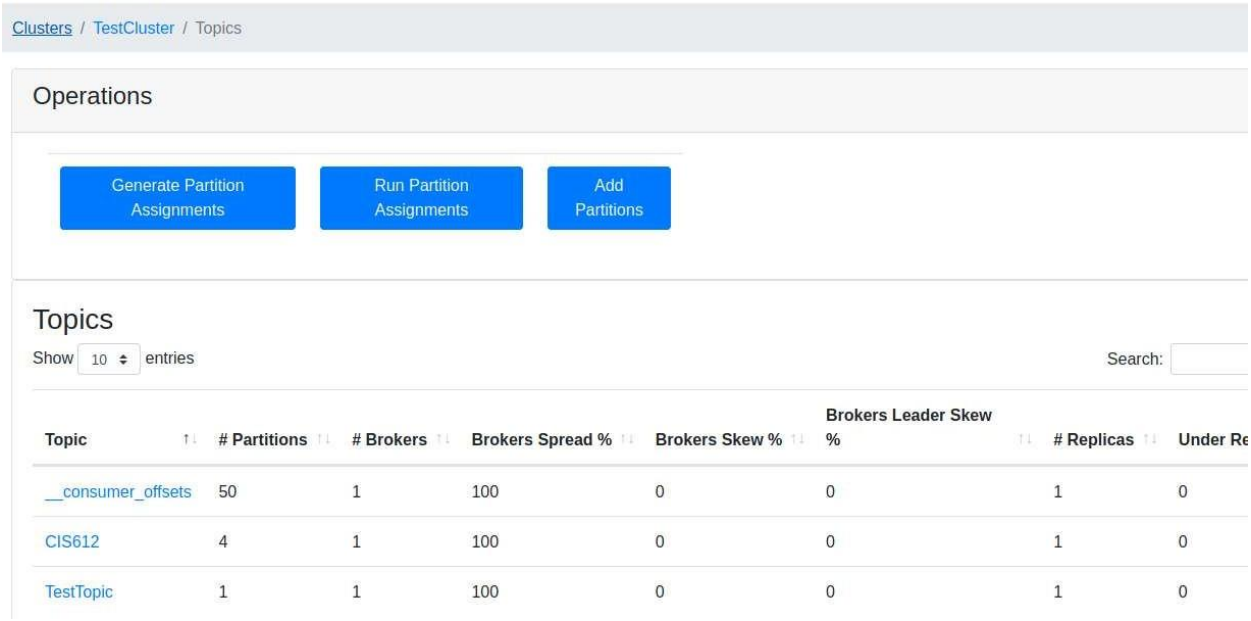
be presented with as illustrated below:




You will see a message that your topic was created as shown below. A link to view it will be availed as well. Click on it to view your new Topic




And there it is:




Apache Kafka is now installed on Ubuntu 20.04 server. It should be noted that it is possible to install Kafka on multi-servers to create a cluster.