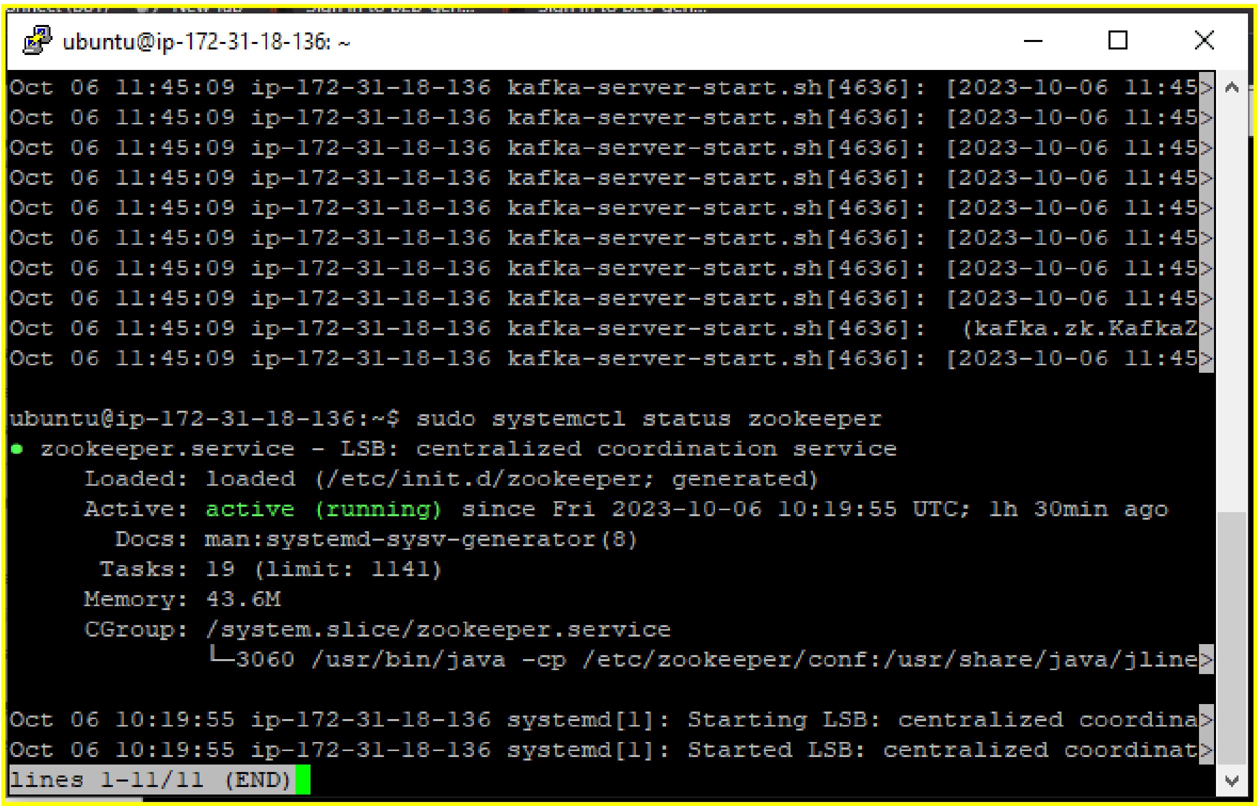
06/10/2023

Downloading kafka on cloud and checking status

Download Kafka latest:

dsh -a wget https://archive.apache.org/dist/kafka/3.5.1/kafka_2.13-3.5.1.tgz

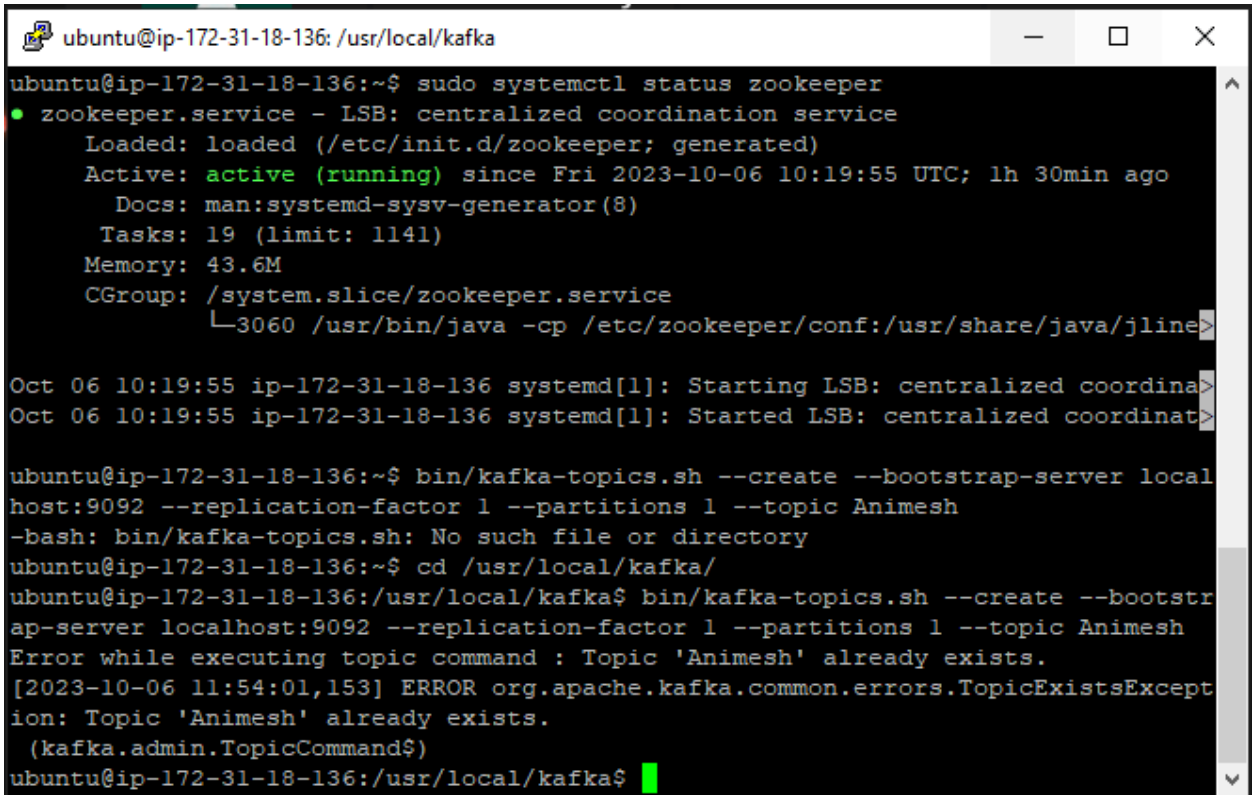


Then extract that using tar -xvzf

We also need zookeeper for kafka
Install zookeeperd:

Sudo apt install zookeeperd -y

Checking Zookeper status



Connect with **telnet localhost 2181**
Are you okay

Configure zookeeper:
Sudo nano /etc/systemd/zookeeper.services

And enter the config detail as:

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

[Service]
Type=simple
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

Move the kafka to the same location as hadoop:
Sudo mv kafka version /usr/local/kafka

Add the path to bashrc file:
export PATH=$PATH:/usr/local/kafka/bin

Set config for kafka service

Then add the config setting as:

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

[Service]
Type=simple
Type=simple
Environment="JAVA_HOME=/usr/lib/jvm/java-11-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
Restart=on-abnormal

[Install]
WantedBy=multi-user.target

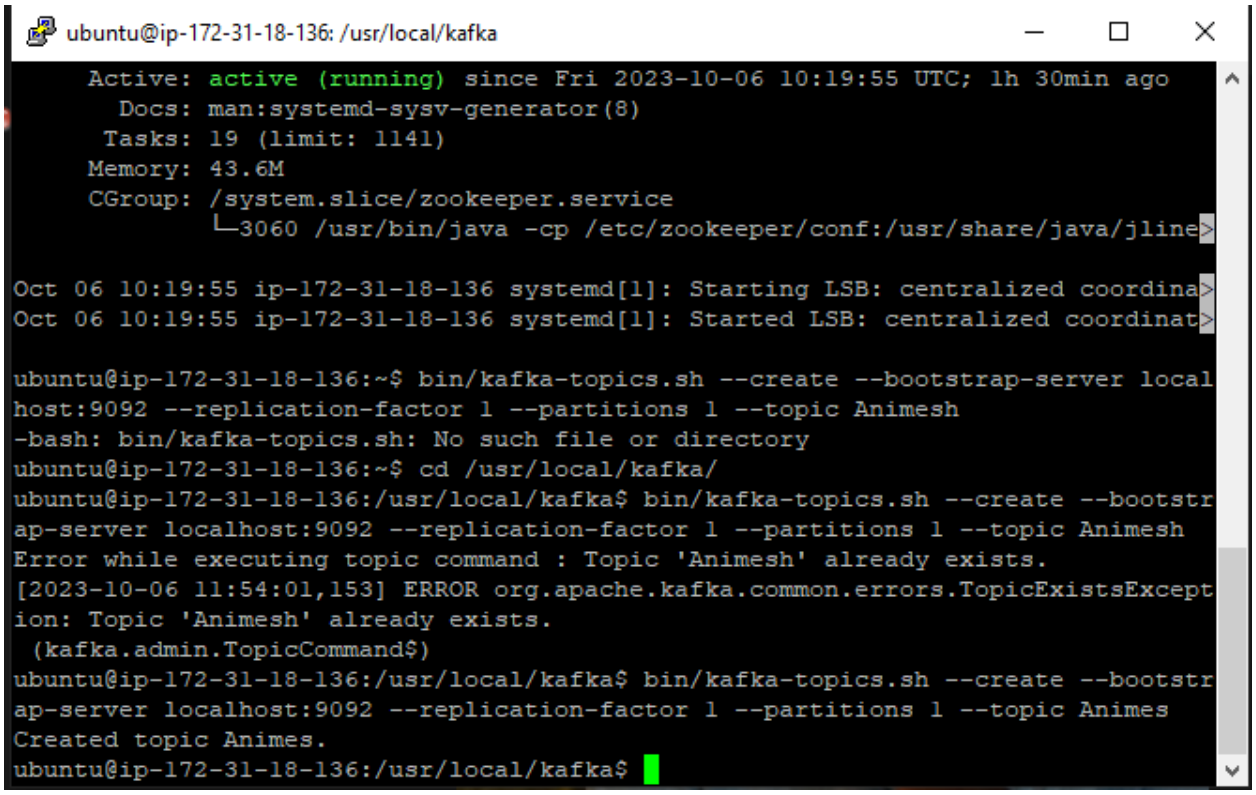
Set up the heap size for kafka:

export KAFKA_HEAP_OPTS="-Xmx256M -Xms128M"

Reload the daemon services:
Sudo systemctl daemon-reload

Start the kafka server :
Sudo systemctl start kafka

Check the status of the kafka :



Create a topic

bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic ktf

==========

“ −-bin/kafka-topics.sh: This is the script used to interact with Kafka topics.

--create : This flag indicates that we want to create a new topic.

--bootstrap-server localhost:9092 : This specifies the address of the Kafka broker(s) that will be used for bootstrapping and connecting to the cluster. In this case, it is set to localhost:9092 , assuming your Kafka broker is running on the local machine with default port 9092.

--replication-factor 1 : This parameter determines the number of replicas for each partition in the topic. In this example, we set it to 1, meaning there will be only one replica per partition.

--partitions 1 : This parameter specifies the number of partitions for the topic. Partitions are used for parallelism and scalability in Kafka. Here, we set it to 1, indicating that there will be only one partition for this topic.

--topic ktf : This sets the name of the topic as "ktf". You can replace "ktf" with your desired topic name.

”==========

List available topics

bin/kafka-topics.sh --list --bootstrap-server localhost:9092

==========

bin/kafka-topics.sh : This is the script used to interact with Kafka topics.

--list : This flag indicates that we want to list the topics.

--bootstrap-server localhost:9092 : This specifies the address of the Kafka broker(s) that will be used for bootstrapping and connecting to the cluster. In this case, it is set to localhost:9092 , assuming your Kafka broker is running on the local machine with default port 9092.

To list the available topics using this command, make sure you have Kafka installed and running, and then execute this command in your terminal or command prompt. It will connect to your local Kafka cluster and retrieve a list of all the topics present in that cluster. The output will display the names of all available topics.

==========

Start production of messages

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

==========

bin/kafka-console-producer.sh : This is the script used to start the console producer in Kafka.

--broker-list localhost:9092 : This parameter specifies the address of the Kafka broker(s) that will be used for producing messages. In this case, it is set to localhost:9092 , assuming your Kafka broker is running on the local machine with default port 9092.
--topic ktf : This parameter specifies the name of the topic to which you want to produce messages. In this example, it is set to "ktf". You can replace "ktf" with your desired topic name.
To start producing messages using this command, make sure you have Kafka installed and running, and then execute this command in your terminal or command prompt. It will connect to your local Kafka cluster and open a console where you can enter messages. Each line you enter will be treated as a separate message and will be sent to the specified topic ("ktf" in this case). Press Enter after each message to send it.
Once you start producing messages, they will be published to the specified topic and can be consumed by consumers subscribed to that topic.

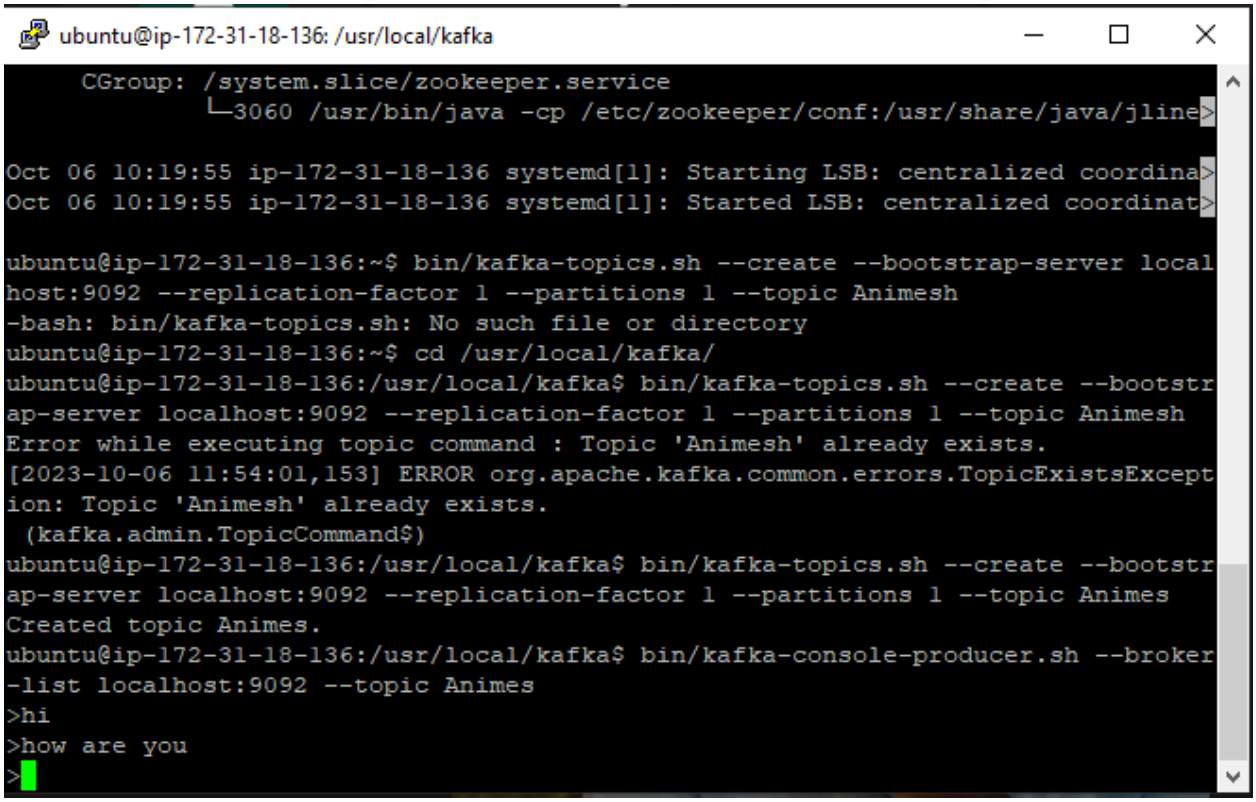
==========

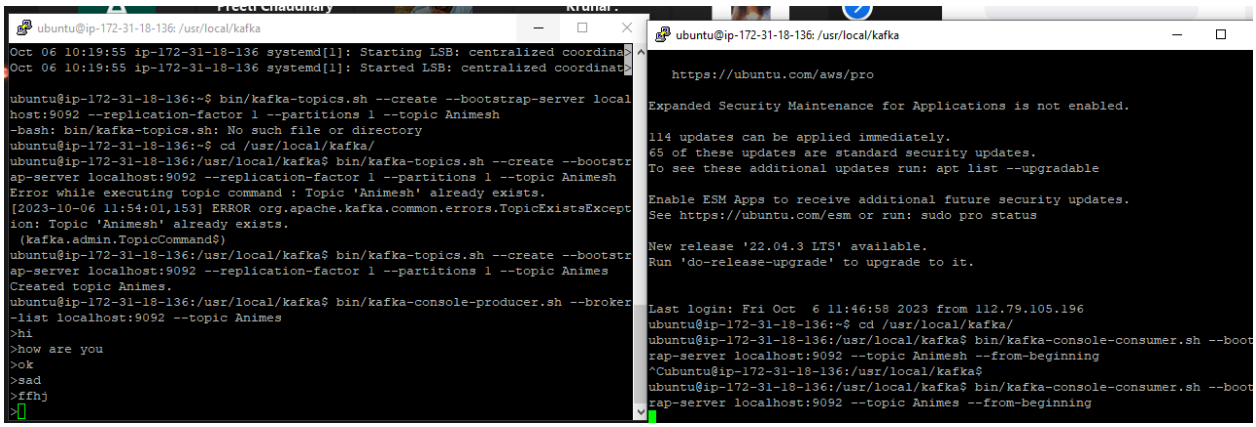
Start consumer for collection

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

==========

bin/kafka-console-consumer.sh : This is the script used to start the console consumer in Kafka.
--bootstrap-server localhost:9092 : This parameter specifies the address of the Kafka broker(s) that will be used for consuming messages. In this case, it is set to localhost:9092 , assuming your Kafka broker is running on the local machine with default port 9092.
--topic ktf : This parameter specifies the name of the topic from which you want to consume messages. In this example, it is set to "ktf". You can replace "ktf" with your desired topic name.
--from-beginning : This flag indicates that you want to consume messages from the beginning of the topic. It ensures that all existing messages in the topic are consumed, not just new ones.
To start consuming messages using this command, make sure you have Kafka installed and running, and then execute this command in your terminal or command prompt. It will connect to your local Kafka cluster and start consuming messages from the specified topic ("ktf" in this case). The consumed messages will be displayed in the console.
By default, each message consumed will be displayed along with its offset, key (if present), and value. You can use additional flags and options with this command to customize how messages are consumed and displayed.


ubuntu@ip-172-31-18-136: /usr/local/kafka
CGroup: /system.slice/zookeeper.service
└─3060 /usr/bin/java -cp /etc/zookeeper/conf:/usr/share/java/jline
Oct 06 10:19:55 ip-172-31-18-136 systemd[1]: Starting LSB: centralized coordina
Oct 06 10:19:55 ip-172-31-18-136 systemd[1]: Started LSB: centralized coordinat
ubuntu@ip-172-31-18-136:~$ bin/kafka-topics.sh --create --bootstrap-server local
host:9092 --replication-factor 1 --partitions 1 --topic Animesh
-bash: bin/kafka-topics.sh: No such file or directory
ubuntu@ip-172-31-18-136:~$ cd /usr/local/kafka/
ubuntu@ip-172-31-18-136:/usr/local/kafka$ bin/kafka-topics.sh --create --bootstr
ap-server localhost:9092 --replication-factor 1 --partitions 1 --topic Animesh
Error while executing topic command : Topic 'Animesh' already exists.
[2023-10-06 11:54:01,153] ERROR org.apache.kafka.common.errors.TopicExistsExcept
ion: Topic 'Animesh' already exists.
(kafka.admin.TopicCommand$)
ubuntu@ip-172-31-18-136:/usr/local/kafka$ bin/kafka-topics.sh --create --bootstr
ap-server localhost:9092 --replication-factor 1 --partitions 1 --topic Animes
Created topic Animes.
ubuntu@ip-172-31-18-136:/usr/local/kafka$ bin/kafka-console-producer.sh --broker
-list localhost:9092 --topic Animes
>hi
>how are you
>


ubuntu@ip-172-31-18-136: /usr/local/kafka
Oct 06 10:19:55 ip-172-31-18-136 systemd[1]: Starting LSB: centralized coordina
Oct 06 10:19:55 ip-172-31-18-136 systemd[1]: Started LSB: centralized coordinat
ubuntu@ip-172-31-18-136:~$ bin/kafka-topics.sh --create --bootstrap-server local
host:9092 --replication-factor 1 --partitions 1 --topic Animesh
-bash: bin/kafka-topics.sh: No such file or directory
ubuntu@ip-172-31-18-136:~$ cd /usr/local/kafka/
ubuntu@ip-172-31-18-136:/usr/local/kafka$ bin/kafka-topics.sh --create --bootstr
ap-server localhost:9092 --replication-factor 1 --partitions 1 --topic Animesh
Error while executing topic command : Topic 'Animesh' already exists.
[2023-10-06 11:54:01,153] ERROR org.apache.kafka.common.errors.TopicExistsExcept
ion: Topic 'Animesh' already exists.
(kafka.admin.TopicCommand$)
ubuntu@ip-172-31-18-136:/usr/local/kafka$ bin/kafka-topics.sh --create --bootstr
ap-server localhost:9092 --replication-factor 1 --partitions 1 --topic Animes
Created topic Animes.
ubuntu@ip-172-31-18-136:/usr/local/kafka$ bin/kafka-console-producer.sh --broker
-list localhost:9092 --topic Animes
>hi
>how are you
>ok
>sad
>ffhj
>
ubuntu@ip-172-31-18-136: /usr/local/kafka
https://ubuntu.com/aws/pro
Expanded Security Maintenance for Applications is not enabled.
114 updates can be applied immediately.
65 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
Last login: Fri Oct 6 11:46:58 2023 from 112.79.105.196
ubuntu@ip-172-31-18-136:~$ cd /usr/local/kafka/
ubuntu@ip-172-31-18-136:/usr/local/kafka$ bin/kafka-console-consumer.sh --boot
rap-server localhost:9092 --topic Animesh --from-beginning
^Cubuntu@ip-172-31-18-136:/usr/local/kafka$
ubuntu@ip-172-31-18-136:/usr/local/kafka$ bin/kafka-console-consumer.sh --boot
rap-server localhost:9092 --topic Animes --from-beginning