

Project: Deploy 3 Node Zookeeper and 3 Node Kafka with Tools Zoonavigator, Kafka Manager, Prometheus, Grafana and Use external volume and mount persistent for data directory.

Deploy Instance For Servers:-

Firstly we will deploy Instances for servers, We will deploy 3 Instances for 3 Zookeeper nodes, 3 For Kafka and 1 more, For Admin Machine/Monitoring Machine. We will deploy total 7 VM with (t2.micro) Instance type for our cluster setup.

We have deployed multi node setup earlier before. So, things will be same with some modifications.

Download and Configure Zookeeper Nodes 1, 2 and 3

Firstly deploy EC2 Instance on AWS with t2.micro Instance type with Amazon Linux AMI. Details below:-

- AMI - Amazon Linux
- Instance Type - t2.micro
- Region - Europe (London) eu-west-2
- Zookeeper-1 Instance - Deploy on eu-west-2a
- Zookeeper-2 Instance - Deploy on eu-west-2b
- Zookeeper-3 Instance - Deploy on eu-west-2c
- Security Group Allow Port - 22, 8080, 80, 2181, 9100

Port Explained Below:-

- SSH - 22
- Prometheus agent - 8080
- HTTP - 80
- Zookeeper Client - 2181
- Prometheus Node Exporter Agent - 9100

Configure Zookeeper Node 1

Deploy EBS Volume For Zookeeper Node 1

Create an EBS Volume of 20 GB For Zookeeper Node 1. In this volume we will store data of zookeeper. Benefit of external volume is when or if our VM fails then our data will be safe and we can mount this EBS volume on another instance. Details given below:-

- Region - Europe (London) eu-west-2
- EBS Volume Zone - eu-west-2a

Goto AWS Console, Create and attach with Zookeeper-1 Node. Now let's configure zookeeper node 1 and EBS volume with persistent mount attachment.

Mount EBS Volume To Zookeeper Node 1

- `ssh -i <key-pair> ec2-user@<Public-Ip>`
- `sudo yum update -y`
- `sudo yum install java -y`
- `sudo mkdir -p /data/zookeeper`

- **sudo chown -R ec2-user:ec2-user /data**
- **lsblk**

If you see **XVDF** with 20G in this list then follow below commands for persistent mount.

- **sudo mkfs.xfs /dev/xvdf**
- **echo "/dev/xvdf /data xfs defaults,nofail 0 0" | sudo tee -a /etc/fstab**
- **sudo mount -a**

External volume mounted perfectly you if didn't get error after last command then it working perfect. Use below command for verify.

- **df -h /data**

If you get output of /dev/xvdf with 20G size volume then volume mounted successfully.

Configure Zookeeper

Now we will download Kafka and configure zookeeper. Kafka has zookeeper server start script and configuration file. So, We don't need to download zookeeper individually. We will download kafka binareis and run zookeeper.

Follow below commnads to download kafka and configure zookeeper.

- **cd**
- **wget https://downloads.apache.org/kafka/3.4.0/kafka_2.13-3.4.0.tgz**
- **tar -xvf kafka_2.13-3.4.0.tgz**
- **rm kafka_2.13-3.4.0.tgz**
- **ln -s kafka_2.13-3.4.0 kafka**
- **echo 1 > /data/zookeeper/myid**
- **nano kafka/config/zookeeper.properties**

Inside nano editor place cursor at the beginning of the first word and press below command for clean/delete the content of existing file in nano editor.

- **alt + t**

Copy and paste below configuration for zookeeper.properties and also replace server.1, server.2 and server.3 private IP with your own instance private IP.

Example - server.1=<enter private IP of Zookeeper 1 Node>:2888:3888 Do this for all nodes. For 2 and 3.

```
dataDir=/data/zookeeper/
clientPort=2181
maxClientCnxns=0
tickTime=2000
initLimit=10
syncLimit=5
4lw.commands.whitelist=*
server.1=172.31.17.155:2888:3888
server.2=172.31.38.114:2888:3888
server.3=172.31.14.212:2888:3888
```

- **Ctrl + o**
- **Ctrl +x**

Zookeeper-1 has been configured. Now Need to create a service file for start and stop zookeeper. So, follow below commands

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

Copy and paste below codes in zookeeper.service file

[Unit]

Description=zookeeper

After=network.target

[Service]

#User=ec2-user

#Group=ec2-user

Environment="KAFKA_HEAP_OPTS=-Xmx256M -Xms128M"

#Environment="KAFKA_OPTS=-javaagent:/home/ec2-user/prometheus/jmx_prometheus_javaagent-0.3.1.jar=8080:/home/ec2-user/prometheus/zookeeper.yml"

ExecStart=/home/ec2-user/kafka/bin/zookeeper-server-start.sh

/home/ec2-user/kafka/config/zookeeper.properties

SuccessExitStatus=143

[Install]

WantedBy=multi-user.target

- **Ctrl + o**
- **Ctrl + x**
- **sudo systemctl daemon-reload**
- **sudo systemctl enable zookeeper**

This service file has some configuration for start/stop/restart zookeeper, Java heap memory and prometheus java agent. But I have comment it because we are not using currently. We will use when we configure prometheus then we will uncomment it.

We have also enabled zookeeper.service. Benefit of this - When we reboot ec2-instance then it will auto start zookeeper after boot.

Configure Zookeeper Node 2

Deploy EBS Volume For Zookeeper Node 2

Create an EBS Volume of 20 GB For Zookeeper Node 2. In this volume we will store data of zookeeper. Benefit of external volume is when or if our VM fails then our data will be safe and we can mount this EBS volume on another instance. Details given below:-

- Region - Europe (London) eu-west-2
- EBS Volume Zone - eu-west-2b

Goto AWS Console, Create and attach with Zookeeper-1 Node. Now let's configure zookeeper node 1 and EBS volume with persistent mount attachment.

Mount EBS Volume To Zookeeper Node 2

- **ssh -i <key-pair> ec2-user@<Public-Ip>**

- `sudo yum update -y`
- `sudo yum install java -y`
- `sudo mkdir -p /data/zookeeper`
- `sudo chown -R ec2-user:ec2-user /data`
- `lsblk`

If you see **XVDF** with 20G in this list then follow below commands for persistent mount.

- `sudo mkfs.xfs /dev/xvdf`
- `echo "/dev/xvdf /data xfs defaults,nofail 0 0" | sudo tee -a /etc/fstab`
- `sudo mount -a`

External volume mounted perfectly you if didn't get error after last command then it working perfect. Use below command for verify.

- `df -h /data`

If you get output of /dev/xvdf with 20G size volume then volume mounted successfully.

Configure Zookeeper

Now we will download Kafka and configure zookeeper. Kafka has zookeeper server start script and configuration file. So, We don't need to download zookeeper individually. We will download kafka binareis and run zookeeper.

Follow below commnads to download kafka and configure zookeeper.

- `cd`
- `wget https://downloads.apache.org/kafka/3.4.0/kafka_2.13-3.4.0.tgz`
- `tar -xvf kafka_2.13-3.4.0.tgz`
- `rm kafka_2.13-3.4.0.tgz`
- `ln -s kafka_2.13-3.4.0 kafka`
- `echo 2 > /data/zookeeper/myid`
- `nano kafka/config/zookeeper.properties`

Inside nano editor place cursor at the beginning of the first word and press below command for clean/delete the content of existing file in nano editor.

- `alt + t`

Copy and paste below configuration for zookeeper.properties and also replace server.1, server.2 and server.3 private IP with your own instance private IP.

Example - server.2=<enter private IP of Zookeeper 2 Node>:2888:3888 Do this for all nodes. For 1 and 3.

```
dataDir=/data/zookeeper/
clientPort=2181
maxClientCnxns=0
tickTime=2000
initLimit=10
syncLimit=5
4lw.commands.whitelist=*
server.1=172.31.17.155:2888:3888
server.2=172.31.38.114:2888:3888
server.3=172.31.14.212:2888:3888
```

- **Ctrl + o**
- **Ctrl +x**

Zookeepe-2 has been configured. Now Need to create a service file for start and stop zookeeper. So, follow below commands

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

Copy and paste below codes in zookeeper.service file

[Unit]

Description=zookeeper

After=network.target

[Service]

#User=ec2-user

#Group=ec2-user

Environment="KAFKA_HEAP_OPTS=-Xmx256M -Xms128M"

#Environment="KAFKA_OPTS=-javaagent:/home/ec2-user/prometheus/jmx_prometheus_javaagent-0.3.1.jar=8080:/home/ec2-user/prometheus/zookeeper.yml"

ExecStart=/home/ec2-user/kafka/bin/zookeeper-server-start.sh

/home/ec2-user/kafka/config/zookeeper.properties

SuccessExitStatus=143

[Install]

WantedBy=multi-user.target

- **Ctrl + o**
- **Ctrl + x**
- **sudo systemctl daemon-reload**
- **sudo systemctl enable zookeeper**

This service file has some configuration for start/stop/restart zookeeper, Java heap memory and prometheus java agent. But I have commet it because we are not using currently. We will use when we configure prometheus then we will uncomment it.

We have also enabled zookeeper.service. Benefit of this - When we reboot ec2-instance then it will auto start zookeeper after boot.

Configure Zookeeper Node 3

Deploy EBS Volume For Zookeeper Node 3

Create an EBS Volume of 20 GB For Zookeeper Node 3. In this volume we will store data of zookeeper. Benefit of external volume is when or if our VM fails then our data will be safe and we can mount this EBS volume on another instance. Details given below:-

- Region - Europe (London) eu-west-2
- EBS Volume Zone - eu-west-2c

Goto AWS Console, Create and attach with Zookeeper-3 Node. Now let's configure zookeeper node 3 and EBS volume with persistent mount attachment.

Mount EBS Volume To Zookeeper Node 3

- `ssh -i <key-pair> ec2-user@<Public-Ip>`
- `sudo yum update -y`
- `sudo yum install java -y`
- `sudo mkdir -p /data/zookeeper`
- `sudo chown -R ec2-user:ec2-user /data`
- `lsblk`

If you see **XVDF** with 20G in this list then follow below commands for persistent mount.

- `sudo mkfs.xfs /dev/xvdf`
- `echo "/dev/xvdf /data xfs defaults,nofail 0 0" | sudo tee -a /etc/fstab`
- `sudo mount -a`

External volume mounted perfectly you if didn't get error after last command then it working perfect. Use below command for verify.

- `df -h /data`

If you get output of `/dev/xvdf` with 20G size volume then volume mounted successfully.

Configure Zookeeper

Now we will download Kafka and configure zookeeper. Kafka has zookeeper server start script and configuration file. So, We don't need to download zookeeper individually. We will download kafka binareis and run zookeeper.

Follow below commnads to download kafka and configure zookeeper.

- `cd`
- `wget https://downloads.apache.org/kafka/3.4.0/kafka_2.13-3.4.0.tgz`
- `tar -xvf kafka_2.13-3.4.0.tgz`
- `rm kafka_2.13-3.4.0.tgz`
- `ln -s kafka_2.13-3.4.0 kafka`
- `echo 3 > /data/zookeeper/myid`
- `nano kafka/config/zookeeper.properties`

Inside nano editor place cursor at the beginning of the first word and press below command for clean/delete the content of existing file in nano editor.

- `alt + t`

Copy and paste below configuration for `zookeeper.properties` and also replace `server.1`, `server.2` and `server.3` private IP with your own instance private IP.

Example - `server.3=<enter private IP of Zookeeper 1 Node>:2888:3888` Do this for all nodes. For 1 and 2.

```
dataDir=/data/zookeeper/  
clientPort=2181  
maxClientCnxns=0  
tickTime=2000  
initLimit=10  
syncLimit=5  
4lw.commands.whitelist=*
```

server.1=172.31.17.155:2888:3888
server.2=172.31.38.114:2888:3888
server.3=172.31.14.212:2888:3888

- **Ctrl + o**
- **Ctrl + x**

Zookeeper-3 has been configured. Now Need to create a service file for start and stop zookeeper. So, follow below commands

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

Copy and paste below codes in zookeeper.service file

[Unit]

Description=zookeeper

After=network.target

[Service]

#User=ec2-user

#Group=ec2-user

Environment="KAFKA_HEAP_OPTS=-Xmx256M -Xms128M"

#Environment="KAFKA_OPTS=-javaagent:/home/ec2-user/prometheus/jmx_prometheus_javaagent-0.3.1.jar=8080:/home/ec2-user/prometheus/zookeeper.yml"

ExecStart=/home/ec2-user/kafka/bin/zookeeper-server-start.sh

/home/ec2-user/kafka/config/zookeeper.properties

SuccessExitStatus=143

[Install]

WantedBy=multi-user.target

- **Ctrl + o**
- **Ctrl + x**
- **sudo systemctl daemon-reload**
- **sudo systemctl enable zookeeper**

This service file has some configuration for start/stop/restart zookeeper, Java heap memory and prometheus java agent. But I have comment it because we are not using currently. We will use when we configure prometheus then we will uncomment it.

We have also enabled zookeeper.service. Benefit of this - When we reboot ec2-instance then it will auto start zookeeper after boot.

Configure Kafka Broker 1, 2 and 3

We have configured Zookeeper Node 1, 2 and 3. But didn't started yet. We will configure Kafka brokers then we will start them.

Firstly deploy EC2 Instance on AWS with t2.micro Instance type with Amazon Linux AMI. Details below:-

- AMI - Amazon Linux
- Instance Type - t2.micro
- Region - Europe (London) eu-west-2

- Kafka-1 Instance - Deploy on eu-west-2a
- Kafka-2 Instance - Deploy on eu-west-2b
- Kafka-3 Instance - Deploy on eu-west-2c
- Security Group Allow Port - 22, 80, 8080, 9100, 8778, 9092

Security Group Port Explained Below:-

- SSH - 22
- HTTP - 80
- Prometheus Agent - 8080
- Prometheus Node Exporter Agent - 9100
- Jolokia Agent - 8778
- Broker Listen - 9092

Configure Kafka Broker/Node 1

Deploy EBS Volume For Kafka Node 1

Create an EBS Volume of 20 GB For Kafka Node 1. In this volume we will store data of Kafka-1. Benefit of external volume is when or if our VM fails then our data will be safe and we can mount this EBS volume on another instance. Details given below:-

- Region - Europe (London) eu-west-2
- EBS Volume Zone - eu-west-2a

Goto AWS Console, Create and attach with Kafka Broker-1 Node. Now let's configure Kafka node 1 and EBS volume with persistent mount attachment.

Mount EBS Volume To Kafka Broker/Node 1

- `ssh -i <key-pair> ec2-user@<Public-Ip>`
- `sudo yum update -y`
- `sudo yum install java -y`
- `sudo mkdir -p /data/kafka`
- `sudo chown -R ec2-user:ec2-user /data`
- `lsblk`

If you see **XVDF** with 20G in this list then follow below commands for persistent mount.

- `sudo mkfs.xfs /dev/xvdf`
- `echo "/dev/xvdf /data xfs defaults,nofail 0 0" | sudo tee -a /etc/fstab`
- `sudo mount -a`

External volume mounted perfectly you if didn't get error after last command then it working perfect. Use below command for verify.

- `df -h /data`

If you get output of /dev/xvdf with 20G size volume then volume mounted successfully.

Configure Kafka Broker/Node 1

Now we will download Kafka and configure Kafka Broker 1.

Follow below commands to download kafka and configure Broker-1.

- **cd**
- **wget https://downloads.apache.org/kafka/3.4.0/kafka_2.13-3.4.0.tgz**
- **tar -xvf kafka_2.13-3.4.0.tgz**
- **rm kafka_2.13-3.4.0.tgz**
- **ln -s kafka_2.13-3.4.0 kafka**
- **nano kafka/config/server.properties**

Inside nano editor place cursor at the beginning of the first word and press below command for clean/delete the content of existing file in nano editor.

- **alt + t**

Copy and paste below configuration for server.properties.

```
#Default configuration
num.network.threads=3
num.io.threads=8
socket.send.buffer.bytes=102400
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600
num.partitions=1
num.recovery.threads.per.data.dir=1
offsets.topic.replication.factor=1
transaction.state.log.replication.factor=1
transaction.state.log.min.isr=1
log.retention.hours=168
log.retention.check.interval.ms=300000
zookeeper.connection.timeout.ms=18000
group.initial.rebalance.delay.ms=0
```

```
#CustomConfiguration
broker.id=1
listeners=PLAINTEXT://172.31.19.35:9092
log.dirs=/data/kafka
zookeeper.connect=172.31.17.155:2181,172.31.38.114:2181,172.31.14.212:2181
```

In configuration file lines after #CustomConfigurations will modify with your own IP. Syntax below:

- **listeners=PLAINTEXT://<your broker/node-1 instance private IP>:9092**
- **zookeeper.connect=<zookeeper-1 instance private IP>:2181,<zookeeper-2 instance private IP>:2181,<zookeeper-3 instance private IP>:2181**
- **Ctrl + o**
- **Ctrl + x**

Kafka Broker 1 has been configured. Now, Create a service file for Start/Stop/Restart Kafka. Follow below command.

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

[Unit]

Description=Kafka

After=network.target

```
[Service]
#User=ec2-user
#Group=ec2-user
Environment="KAFKA_HEAP_OPTS=-Xmx256M -Xms128M"
#Environment="KAFKA_OPTS=-javaagent:/home/ec2-user/prometheus/jmx_prometheus_javaagent-0.3.1.jar=8080:/home/ec2-user/prometheus/kafka-0-8-2.yml"
ExecStart=/home/ec2-user/kafka/bin/kafka-server-start.sh /home/ec2-user/kafka/config/server.properties
SuccessExitStatus=143
```

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

- **Ctrl + o**
- **Ctrl + x**
- **sudo systemctl daemon-reload**
- **sudo systemctl enable zookeeper**

This service file has some configuration for start/stop/restart Kafka Broker, Java heap memory, prometheus java agent and jolokia agent. But I have comment it because we are not using currently. We will use when we configure prometheus and jolokia then we will uncomment it.

We have also enabled kafka.service. Benefit of this - When we reboot ec2-instance then it will auto start kafka broker after boot.

Configure Kafka Broker/Node 2

Deploy EBS Volume For Kafka Node 2

Create an EBS Volume of 20 GB For Kafka Node 1. In this volume we will store data of Kafka-1. Benefit of external volume is when or if our VM fails then our data will be safe and we can mount this EBS volume on another instance. Details given below:-

- Region - Europe (London) eu-west-2
- EBS Volume Zone - eu-west-2b

Goto AWS Console, Create and attach with Kafka Broker-2 Node. Now let's configure Kafka node 2 and EBS volume with persistent mount attachment.

Mount EBS Volume To Kafka Broker/Node 2

- **ssh -i <key-pair> ec2-user@<Public-Ip>**
- **sudo yum update -y**
- **sudo yum install java -y**
- **sudo mkdir -p /data/kafka**
- **sudo chown -R ec2-user:ec2-user /data**
- **lsblk**

If you see **XVDF** with 20G in this list then follow below commands for persistent mount.

- **sudo mkfs.xfs /dev/xvdf**
- **echo "/dev/xvdf /data xfs defaults,nofail 0 0" | sudo tee -a /etc/fstab**
- **sudo mount -a**

External volume mounted perfectly you if didn't get error after last command then it working perfect. Use below command for verify.

- **df -h /data**

If you get output of /dev/xvdf with 20G size volume then volume mounted successfully.

Configure Kafka Broker/Node 2

Now we will download Kafka and configure Kafka Broker 2.

Follow below commnads to download kafka and configure Broker-2.

- **cd**
- **wget https://downloads.apache.org/kafka/3.4.0/kafka_2.13-3.4.0.tgz**
- **tar -xvf kafka_2.13-3.4.0.tgz**
- **rm kafka_2.13-3.4.0.tgz**
- **ln -s kafka_2.13-3.4.0 kafka**
- **nano kafka/config/server.properties**

Inside nano editor place cursor at the beginning of the first word and press below command for clean/delete the content of existing file in nano editor.

- **alt + t**

Copy and paste below configuration for server.properties.

```
#Default configuration
num.network.threads=3
num.io.threads=8
socket.send.buffer.bytes=102400
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600
num.partitions=1
num.recovery.threads.per.data.dir=1
offsets.topic.replication.factor=1
transaction.state.log.replication.factor=1
transaction.state.log.min.isr=1
log.retention.hours=168
log.retention.check.interval.ms=300000
zookeeper.connection.timeout.ms=18000
group.initial.rebalance.delay.ms=0

#CustomConfiguration
broker.id=2
listeners=PLAINTEXT://172.31.32.42:9092
log.dirs=/data/kafka
zookeeper.connect=172.31.17.155:2181,172.31.38.114:2181,172.31.14.212:2181
```

In configuration file lines after #CustomConfigurations will modify with your own IP. Syntax below:

- **listeners=PLAINTEXT://<your broker/node-2 instance private IP>:9092**
- **zookeeper.connect=<zookeeper-1 instance private IP>:2181,<zookeeper-2 instance private IP>:2181,<zookeeper-3 instance private IP>:2181**

- **Ctrl + o**
- **Ctrl + x**

Kafka Broker 1 has been configured. Now, Create a service file for Start/Stop/Restart Kafka. Follow below command.

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

[Unit]

Description=Kafka

After=network.target

[Service]

#User=ec2-user

#Group=ec2-user

Environment="KAFKA_HEAP_OPTS=-Xmx256M -Xms128M"

#Environment="KAFKA_OPTS=-javaagent:/home/ec2-user/prometheus/jmx_prometheus_javaagent-0.3.1.jar=8080:/home/ec2-user/prometheus/kafka-0-8-2.yml"

ExecStart=/home/ec2-user/kafka/bin/kafka-server-start.sh /home/ec2-user/kafka/config/server.properties

SuccessExitStatus=143

[Install]

WantedBy=multi-user.target

- **Ctrl + o**
- **Ctrl + x**
- **sudo systemctl daemon-reload**
- **sudo systemctl enable zookeeper**

This service file has some configuration for start/stop/restart Kafka Broker, Java heap memory, prometheus java agent and jolokia agent. But I have comment it because we are not using currently. We will use when we configure prometheus and jolokia then we will uncomment it.

We have also enabled kafka.service. Benefit of this - When we reboot ec2-instance then it will auto start kafka broker after boot.

Configure Kafka Node/Broker 3

Deploy EBS Volume For Kafka Node 3

Create an EBS Volume of 20 GB For Kafka Node 3. In this volume we will store data of Kafka-3. Benefit of external volume is when or if our VM fails then our data will be safe and we can mount this EBS volume on another instance. Details given below:-

- Region - Europe (London) eu-west-2
- EBS Volume Zone - eu-west-2c

Goto AWS Console, Create and attach with Kafka Broker-3 Node. Now let's configure Kafka node 3 and EBS volume with persistent mount attachment.

Mount EBS Volume To Kafka Broker/Node 3

- **ssh -i <key-pair> ec2-user@<Public-Ip>**

- **sudo yum update -y**
- **sudo yum install java -y**
- **sudo mkdir -p /data/kafka**
- **sudo chown -R ec2-user:ec2-user /data**
- **lsblk**

If you see **XVDF** with 20G in this list then follow below commands for persistent mount.

- **sudo mkfs.xfs /dev/xvdf**
- **echo "/dev/xvdf /data xfs defaults,nofail 0 0" | sudo tee -a /etc/fstab**
- **sudo mount -a**

External volume mounted perfectly you if didn't get error after last command then it working perfect. Use below command for verify.

- **df -h /data**

If you get output of /dev/xvdf with 20G size volume then volume mounted successfully.

Configure Kafka Broker/Node 3

Now we will download Kafka and configure Kafka Broker 3.

Follow below commnads to download kafka and configure Broker-3.

- **cd**
- **wget https://downloads.apache.org/kafka/3.4.0/kafka_2.13-3.4.0.tgz**
- **tar -xvf kafka_2.13-3.4.0.tgz**
- **rm kafka_2.13-3.4.0.tgz**
- **ln -s kafka_2.13-3.4.0 kafka**
- **nano kafka/config/server.properties**

Inside nano editor place cursor at the beginning of the first word and press below command for clean/delete the content of existing file in nano editor.

- **alt + t**

Copy and paste below configuration for server.properties.

```
#Default configuration
num.network.threads=3
num.io.threads=8
socket.send.buffer.bytes=102400
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600
num.partitions=1
num.recovery.threads.per.data.dir=1
offsets.topic.replication.factor=1
transaction.state.log.replication.factor=1
transaction.state.log.min.isr=1
log.retention.hours=168
log.retention.check.interval.ms=300000
zookeeper.connection.timeout.ms=18000
group.initial.rebalance.delay.ms=0
```

```
#CustomConfiguration
broker.id=3
listeners=PLAINTEXT://172.31.5.194:9092
log.dirs=/data/kafka
zookeeper.connect=172.31.17.155:2181,172.31.38.114:2181,172.31.14.212:2181
```

In configuration file lines after #CustomConfigurations will modify with your own IP. Syntax below:

- listeners=PLAINTEXT://<your broker/node-3 instance private IP>:9092
- zookeeper.connect=<zookeeper-1 instance private IP>:2181,<zookeeper-2 instance private IP>:2181,<zookeeper-3 instance private IP>:2181
- **Ctrl + o**
- **Ctrl + x**

Kafka Broker 1 has been configured. Now, Create a service file for Start/Stop/Restart Kafka. Follow below command.

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

[Unit]

Description=Kafka

After=network.target

[Service]

#User=ec2-user

#Group=ec2-user

Environment="KAFKA_HEAP_OPTS=-Xmx256M -Xms128M"

#Environment="KAFKA_OPTS=-javaagent:/home/ec2-user/prometheus/jmx_prometheus_javaagent-0.3.1.jar=8080:/home/ec2-user/prometheus/kafka-0-8-2.yml"

ExecStart=/home/ec2-user/kafka/bin/kafka-server-start.sh /home/ec2-user/kafka/config/server.properties

SuccessExitStatus=143

[Install]

WantedBy=multi-user.target

- **Ctrl + o**
- **Ctrl + x**
- **sudo systemctl daemon-reload**
- **sudo systemctl enable zookeeper**

This service file has some configuration for start/stop/restart Kafka Broker, Java heap memory, prometheus java agent and jolokia agent. But I have comment it because we are not using currently. We will use when we configure prometheus and jolokia then we will uncomment it.

We have also enabled kafka.service. Benefit of this - When we reboot ec2-instance then it will auto start kafka broker after boot.

Start Zookeeper and Kafka

Zookeeper Node 1, 2 and 3 has been properly configured and Kafka Broker/Node 1, 2 and 3 has been properly configured. Let's start one by one for run the kafka cluster. So, Do this follow below instructions:-

Remember we have configured and created zookeeper.service file in all 3 nodes and kafka.service file in all 3 nodes. So, We will use service files for start zookeeper and kafka.

How To Use Service File For Start Zookeeper and Kafka

Firstly learn systemctl command for start services. Whenever you modify the service file of Zookeeper or Kafka then you have to reload the daemon services for update the service file. To do this use below command

- **sudo systemctl daemon-reload**

But we have already reload the daemon services. When we change or modify the service file we will do. We will do later when we configure prometheus and jolokia agent. For now read below commands that is important.

Below command is for start zookeeper.service

- **sudo systemctl start zookeeper**

Below command is for stop zookeeper

- **sudo systemctl stop zookeeper**

Below commands is for check status of service. i.e. zookeeper.service is running or not

- **sudo systemctl status zookeeper**

Start Zookeeper Servers

Go to Zookeeper Node 1 using ssh into Zookeeper Node 1 Instance and use below command.

- **sudo systemctl start zookeeper**
- **sudo systemctl status zookeeper**

```
[ec2-user@Zookeeper-1 ~]$ sudo systemctl status zookeeper
● zookeeper.service - zookeeper
   Loaded: loaded (/etc/systemd/system/zookeeper.service; enabled; preset: disabled)
   Active: active (running) since Mon 2023-05-22 05:01:14 UTC; 5h 47min ago
     Main PID: 1920 (java)
       Tasks: 52 (limit: 1108)
      Memory: 223.7M
         CPU: 1min 19.490s
        CGroup: /system.slice/zookeeper.service
                └─1920 java -Xmx256M -Xms128M -server -XX:+UseG1GC -XX:MaxGCPauseMillis=20

May 22 05:01:47 Zookeeper-1 zookeeper-server-start.sh[1920]: [2023-05-22 05:01:47,208]
May 22 05:01:47 Zookeeper-1 zookeeper-server-start.sh[1920]: [2023-05-22 05:01:47,209]
May 22 05:04:47 Zookeeper-1 zookeeper-server-start.sh[1920]: [2023-05-22 05:04:47,338]
May 22 05:04:50 Zookeeper-1 zookeeper-server-start.sh[1920]: [2023-05-22 05:04:50,000]
May 22 05:04:59 Zookeeper-1 zookeeper-server-start.sh[1920]: [2023-05-22 05:04:59,708]
May 22 05:05:00 Zookeeper-1 zookeeper-server-start.sh[1920]: [2023-05-22 05:05:00,423]
May 22 05:05:00 Zookeeper-1 zookeeper-server-start.sh[1920]: [2023-05-22 05:05:00,424]
May 22 05:05:17 Zookeeper-1 zookeeper-server-start.sh[1920]: [2023-05-22 05:05:17,321]
May 22 05:05:20 Zookeeper-1 zookeeper-server-start.sh[1920]: [2023-05-22 05:05:20,165]
May 22 05:05:20 Zookeeper-1 zookeeper-server-start.sh[1920]: [2023-05-22 05:05:20,183]
lines 1-20/20 (END)
```

Output of "sudo systemctl status zookeeper" command

Go to Zookeeper Node 2 using ssh into Zookeeper Node 2 Instance and use below command.

- **sudo systemctl start zookeeper**
- **sudo systemctl status zookeeper**

Go to Zookeeper Node 3 using ssh into Zookeeper Node 3 Instance and use below command.

- **sudo systemctl start zookeeper**
- **sudo systemctl status zookeeper**

Start Kafka Brokers

Go to Kafka Node 1 using ssh into Kafka Node 1 Instance and use below command.

- **sudo systemctl start kafka**
- **sudo systemctl status kafka**


```
[ec2-user@Kafka-2 ~]$ sudo systemctl status kafka
● kafka.service - Kafka
   Loaded: loaded (/etc/systemd/system/kafka.service; enabled; preset: disabled)
   Active: active (running) since Mon 2023-05-22 05:04:38 UTC; 5h 49min ago
 Main PID: 1893 (java)
    Tasks: 86 (limit: 1108)
   Memory: 483.3M
      CPU: 25min 37.044s
   CGroup: /system.slice/kafka.service
           └─1893 java -Xmx256M -Xms128M -server -XX:+UseG1GC -XX:MaxGCPauseMillis=20

May 22 09:34:29 Kafka-2 kafka-server-start.sh[1893]: org.apache.kafka.common.network.In
May 22 09:34:29 Kafka-2 kafka-server-start.sh[1893]: at org.apache.kafka.common
May 22 09:34:29 Kafka-2 kafka-server-start.sh[1893]: at org.apache.kafka.common
May 22 09:34:29 Kafka-2 kafka-server-start.sh[1893]: at org.apache.kafka.common
May 22 09:34:29 Kafka-2 kafka-server-start.sh[1893]: at org.apache.kafka.common
May 22 09:34:29 Kafka-2 kafka-server-start.sh[1893]: at org.apache.kafka.common
May 22 09:34:29 Kafka-2 kafka-server-start.sh[1893]: at org.apache.kafka.common
May 22 09:34:29 Kafka-2 kafka-server-start.sh[1893]: at kafka.network.Processor
May 22 09:34:29 Kafka-2 kafka-server-start.sh[1893]: at kafka.network.Processor
May 22 09:34:29 Kafka-2 kafka-server-start.sh[1893]: at java.base/java.lang.Thr
lines 1-20/20 (END)
```

Output of "sudo systemctl status kafka" command

Go to Kafka Node 2 using ssh into Kafka Node 2 Instance and use below command.

- **sudo systemctl start kafka**
- **sudo systemctl status kafka**

Go to Kafka Node 3 using ssh into Kafka Node 3 Instance and use below command.

- **sudo systemctl start kafka**
- **sudo systemctl status kafka**

Now, Zookeeper Server 1, 2 and 3 are up and running. Kafka Broker 1, 2 and 3 are up and running.

Setup Monitoring/Admin Machine

Introduction of Monitoring/Admin Machine

Using this machine we will install and configure servers for Prometheus, Grafana for monitor Kafka Cluster. We will also Install and configure tools like - Zoonavigator, Kafka Manager for manage Kafka Cluster.

Creating and Launch Monitoring/Admin Machine

Launch EC2 Instance with t2.micro (Instance Type) with Amazon Linux 2 AMI. Details below:-

- AMI - Amazon Linux 2 (Search in AWS Marketplace for OS)
- Instance Type - t2.micro
- EBS - 25 GB
- Region - Europe (London) eu-west-2
- Zone - eu-west-2a
- Security Group Allow Port - 22, 80, 7070, 9090, 9000, 3000

Security Group Port Explained Below:-

- SSH - 22
- HTTP - 80
- Zoonavigator - 7070
- Prometheus - 9090
- Kafka Manager - 9000
- Grafana - 3000

Setup Monitoring/Admin Machine

Let's setup Admin/Monitoring Machine. We will start Zoonavigator, Kafka Manager, Prometheus Server, Rolling Restart Brokers with Jolokia Agent, Node Exporter in Prometheus, Grafana Server for monitor all things on a dashboard.

We have to configure agents on Zookeeper Nodes, Kafka Nodes during configuration and setup. When creating service file for zookeeper and kafka I have commented using (#) sign in some lines. We will uncomment during setup and configuration.

First of all ssh into machine using below command.

- `ssh -i <key pair name> ec2-user@<Public IP>`
- Now you are in Admin/Monitor Machine

Prerequisites

We have to download and install some prerequisites for deploy lots of things that we have discuss before. So, Follow instructions and commands to do this.

Download/Install Docker and Docker-Compose With Service File

- `sudo yum update -y`
- `sudo yum install -y docker`
- `sudo systemctl start docker.service`
- `sudo usermod -a -G docker ec2-user`

Now, Exit from terminal and relogin using ssh for reflect things.

- `ssh -i <key pair name> ec2-user@<Public IP>`

Now, Continue to command for do further things

- `sudo curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose`
- `sudo chmod +x /usr/local/bin/docker-compose`

Creating Docker-Compose Service File

Using Docker-Compose service file we will start and stop services for Zoonavigator and Kafka Manager. Because we will use .yml file and create container in docker. So, We need to create.

- `sudo nano /etc/systemd/system/docker-compose@.service`

Copy and paste below code for setup docker-compose service in nano editor.

```
# /etc/systemd/system/docker-compose@.service
# From: https://github.com/docker/compose/issues/4266#issuecomment-302813256
[Unit]
Description=%i service with docker compose
Requires=docker.service
After=docker.service

[Service]
Restart=always

WorkingDirectory=/etc/docker/compose/%i

# Remove old containers, images and volumes
ExecStartPre=/usr/local/bin/docker-compose down -v
ExecStartPre=/usr/local/bin/docker-compose rm -fv
ExecStartPre=-/bin/bash -c 'docker volume ls -qf "name=%i_" | xargs docker volume rm'
ExecStartPre=-/bin/bash -c 'docker network ls -qf "name=%i_" | xargs docker network rm'
ExecStartPre=-/bin/bash -c 'docker ps -aqf "name=%i_*" | xargs docker rm'

# Compose up
ExecStart=/usr/local/bin/docker-compose up

# Compose down, remove containers and volumes
ExecStop=/usr/local/bin/docker-compose down -v

[Install]
WantedBy=multi-user.target



- Ctrl + o
- Ctrl + x

```

Install Zoonavigator as SystemD

ZooNavigator is a **web-based ZooKeeper UI and editor/browser with many features.**

- **sudo mkdir -p /etc/docker/compose/zoonavigator/**
- **sudo nano /etc/docker/compose/zoonavigator/docker-compose.yml**
- Copy below code and paste it in nano editor after run previous command

version: '3.3'

services:

zoonavigator:

ports:

- '7070:7070'

environment:

- HTTP_PORT=7070

container_name: zoonavigator

restart: unless-stopped

image: 'elkozmon/zoonavigator:latest'

- **Ctrl + o** For write changes

- **Ctrl + x** For exit from nano editor
- **sudo systemctl enable docker-compose@zoonavigator**
- **sudo systemctl start docker-compose@zoonavigator**
- **sudo systemctl status docker-compose@zoonavigator**

```
[ec2-user@admin-machine ~]$ sudo systemctl status docker-compose@zoonavigator
● docker-compose@zoonavigator.service - zoonavigator service with docker compose
   Loaded: loaded (/etc/systemd/system/docker-compose@.service; enabled; vendor pre
   Active: active (running) since Mon 2023-05-22 05:04:58 UTC; 7h ago
     Process: 5058 ExecStartPre=/bin/bash -c docker ps -aqf "name=%i_*" | xargs docker
     Process: 5022 ExecStartPre=/bin/bash -c docker network ls -qf "name=%i_" | xargs
     Process: 4991 ExecStartPre=/bin/bash -c docker volume ls -qf "name=%i_" | xargs d
     Process: 4938 ExecStartPre=/usr/local/bin/docker-compose rm -fv (code=exited, sta
     Process: 4859 ExecStartPre=/usr/local/bin/docker-compose down -v (code=exited, st
   Main PID: 5085 (docker-compose)
     CGroup: /system.slice/system-docker\x2dcompose.slice/docker-compose@zoonavigator
            └─5085 /usr/local/bin/docker-compose up
              └─5089 /usr/local/bin/docker-compose up

May 22 05:05:06 admin-machine docker-compose[5085]: [69B blob data]
May 22 05:05:06 admin-machine docker-compose[5085]: zoonavigator      | SLF4J: Class
May 22 05:05:06 admin-machine docker-compose[5085]: zoonavigator      | SLF4J: Found
May 22 05:05:06 admin-machine docker-compose[5085]: zoonavigator      | SLF4J: Found
May 22 05:05:06 admin-machine docker-compose[5085]: zoonavigator      | SLF4J: See ht
May 22 05:05:08 admin-machine docker-compose[5085]: zoonavigator      | SLF4J: Actual
May 22 05:05:15 admin-machine docker-compose[5085]: zoonavigator      | [info] applic
May 22 05:05:15 admin-machine docker-compose[5085]: zoonavigator      | [info] applic
May 22 05:05:15 admin-machine docker-compose[5085]: zoonavigator      | [info] play.a
May 22 05:05:18 admin-machine docker-compose[5085]: zoonavigator      | [info] p.c.s.
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@admin-machine ~]$
```

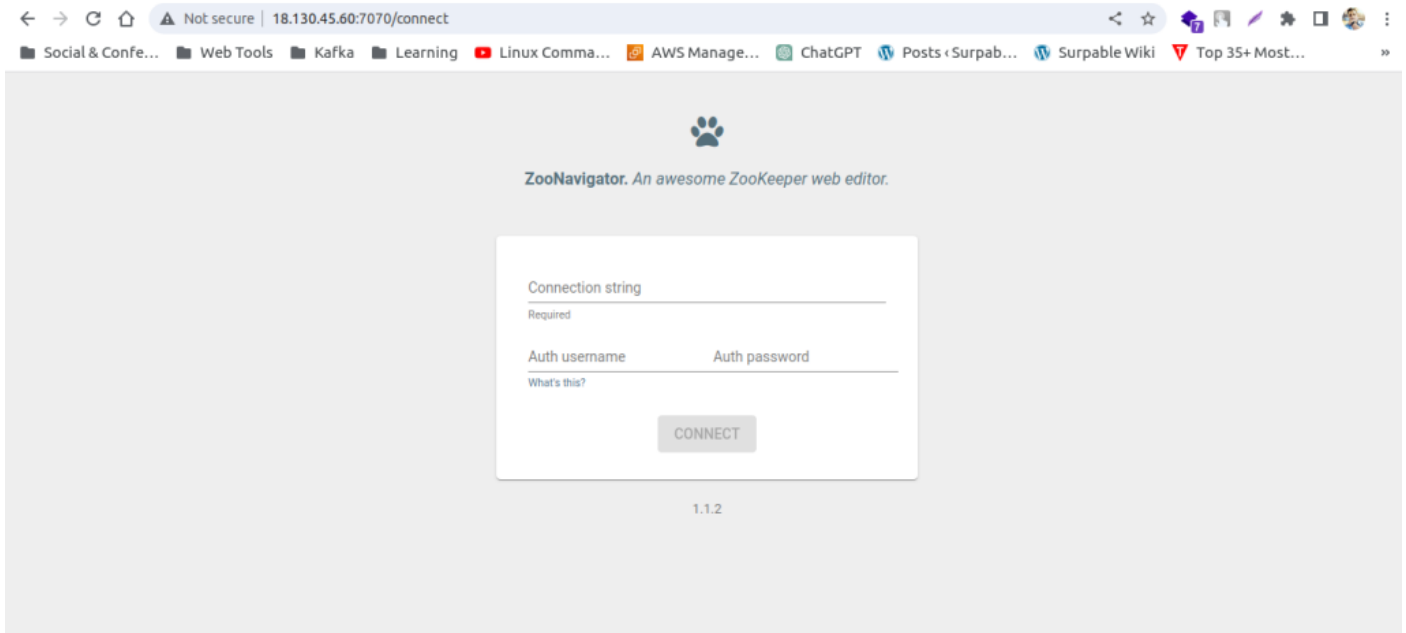
Output After Run Command "sudo systemctl status docker-compose@zoonavigator"

Zoonavigator is running and can be access using web browser.

Access Zoonavigator UI

Go to any web browser and type in address box <Public IP of Admin/Monitoring Machine>:7070

- Example - **18.130.45.60:7070**

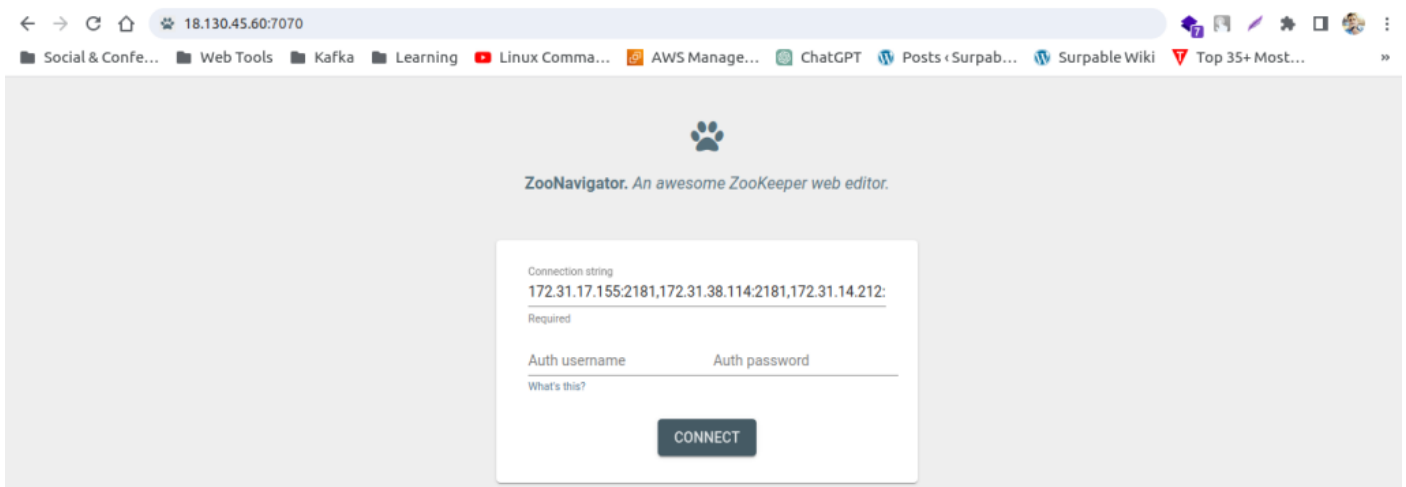


Zoonavigator UI Looks Like This After Enter URL

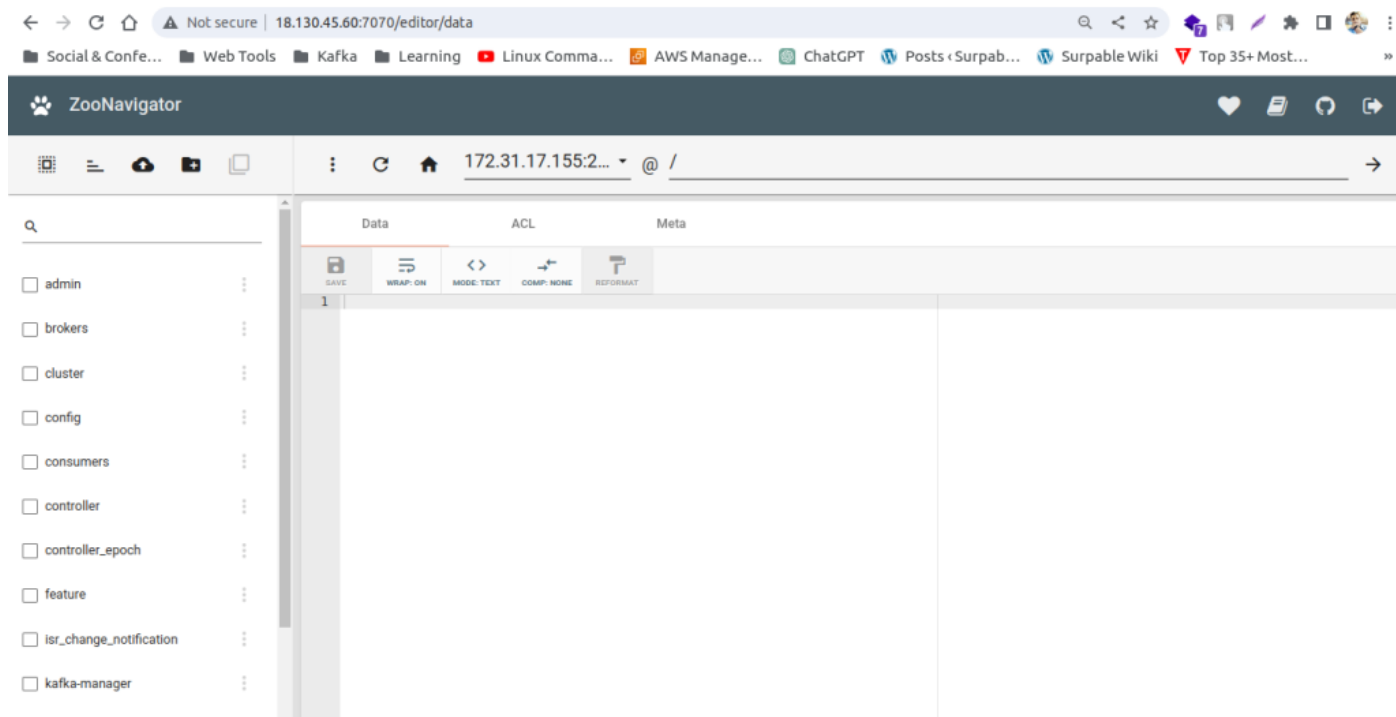
Login Into Zoonavigator

For Login - In connection string enter Zookeeper Node 1, 2 and 3 Instance's Private IP with port number 2181 and press connect. See example below

- 172.31.17.155:2181,172.31.38.114:2181,172.31.14.212:2181



After Enter Private IPs with Port 2181



After Press on Connect. We are in Zoonavigator

Install Kafka Manager as SystemD

Kafka Manager is **an open-source managing tool for Apache Kafka clusters**. With Kafka Manager, you can: Manage multiple clusters. Easy inspection of cluster state (topics, consumers, offsets, brokers, replica distribution, partition distribution) Run preferred replica election.

Follow below commands for setup Kafka Manager

- **sudo mkdir -p /etc/docker/compose/kafka-manager/**
- **sudo nano /etc/docker/compose/kafka-manager/docker-compose.yml**
- Copy below code and paste it in nano editor after run previous command

```
# /etc/docker/compose/kafka-manager/docker-compose.yml
version: '3.6'
services:
  kafka_manager:
    image: hlebalbau/kafka-manager:1.3.3.18
    ports:
      - "9000:9000"
    environment:
      ZK_HOSTS: "172.31.17.155:2181,172.31.38.114:2181,172.31.14.212:2181"
      APPLICATION_SECRET: "random-secret"
    command: -Dpidfile.path=/dev/null
```

- **Ctrl + o** For write changes
- **Ctrl + x** For exit from nano editor

Note: In the line of above script you can see **ZK_HOSTS**: This is Zookeeper Instances Nodes Private IP. Check yours and replace with it.

- **sudo systemctl enable docker-compose@kafka-manager**
- **sudo systemctl start docker-compose@kafka-manager**

- `sudo systemctl status docker-compose@kafka-manager`

```
[ec2-user@admin-machine ~]$ sudo systemctl status docker-compose@kafka-manager
● docker-compose@kafka-manager.service - kafka-manager service with docker compose
   Loaded: loaded (/etc/systemd/system/docker-compose@.service; enabled; vendor preset:
   Active: active (running) since Mon 2023-05-22 05:04:58 UTC; 7h ago
     Process: 5052 ExecStartPre=/bin/bash -c docker ps -aqf "name=%i_*" | xargs docker rm
     Process: 5019 ExecStartPre=/bin/bash -c docker network ls -qf "name=%i_" | xargs dock
     Process: 4995 ExecStartPre=/bin/bash -c docker volume ls -qf "name=%i_" | xargs docke
     Process: 4935 ExecStartPre=/usr/local/bin/docker-compose rm -fv (code=exited, status=
     Process: 4858 ExecStartPre=/usr/local/bin/docker-compose down -v (code=exited, status=
   Main PID: 5082 (docker-compose)
     CGroup: /system.slice/system-docker\x2dcompose.slice/docker-compose@kafka-manager.se
             └─5082 /usr/local/bin/docker-compose up
               └─5088 /usr/local/bin/docker-compose up

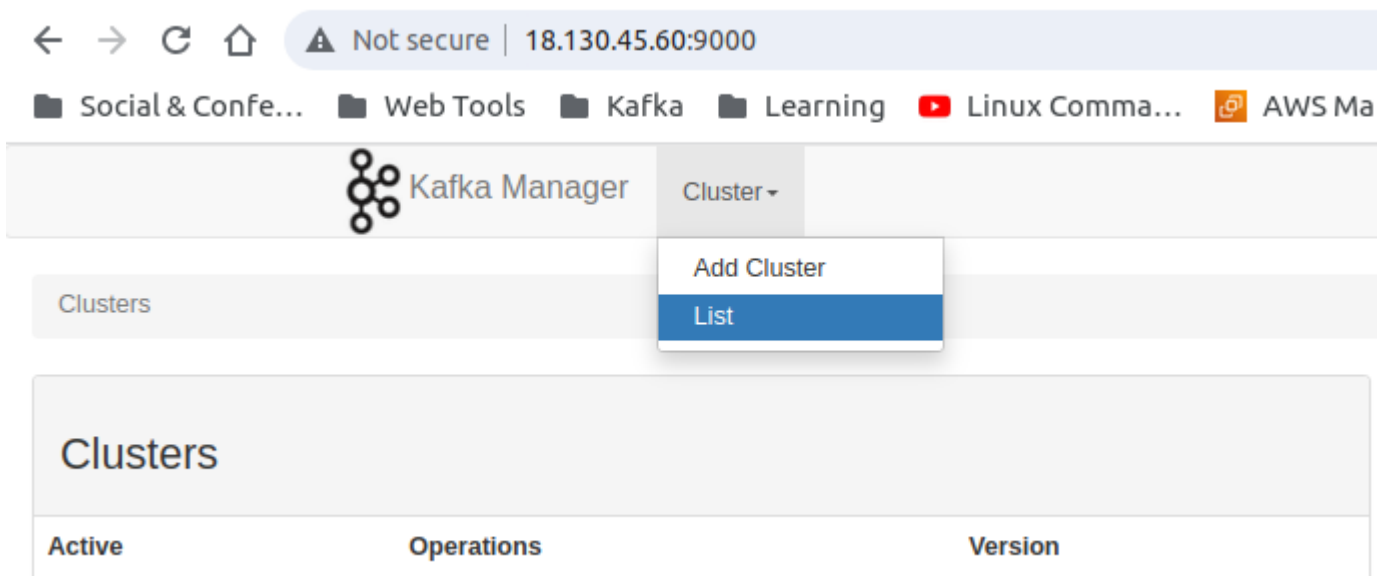
May 22 12:57:32 admin-machine docker-compose[5082]: kafka_manager_1 | [error] k.m.a.c.f
May 22 12:57:32 admin-machine docker-compose[5082]: kafka_manager_1 | org.apache.kafka
May 22 12:57:38 admin-machine docker-compose[5082]: kafka_manager_1 | [info] k.m.a.Kaf
May 22 12:57:48 admin-machine docker-compose[5082]: kafka_manager_1 | [info] k.m.a.Kaf
May 22 12:57:50 admin-machine docker-compose[5082]: kafka_manager_1 | [info] k.m.a.c.B
May 22 12:57:58 admin-machine docker-compose[5082]: kafka_manager_1 | [info] k.m.a.Kaf
May 22 12:58:08 admin-machine docker-compose[5082]: kafka_manager_1 | [info] k.m.a.Kaf
May 22 12:58:18 admin-machine docker-compose[5082]: kafka_manager_1 | [info] k.m.a.Kaf
May 22 12:58:20 admin-machine docker-compose[5082]: kafka_manager_1 | [info] k.m.a.c.B
May 22 12:58:28 admin-machine docker-compose[5082]: kafka_manager_1 | [info] k.m.a.Kaf
Hint: Some lines were ellipsized, use -l to show in full.
```

Output of After Run "sudo systemctl status docker-compose@kafka-manager" Command

Access Kafka Manager UI and Setup

Go to any web browser and type in address box <Public IP of Admin/Monitoring Machine>:9000

- Example - **18.130.45.60:9000**



UI of Kafka Manger After Enter URL

For configure Kafka Manger Click on "Cluster" and then "Add Cluster".

←

→

↻

🏠

⚠ Not secure | 18.130.45.60:9000/addCluster

📁 Social & Confe...


📁 Web Tools

📁 Kafka

📁 Learning

📺 Linux Comma...

📺 AWS M

Kafka Manager

Cluster ▾

Clusters / Add Cluster

← Add Cluster

Cluster Name

Kafka

Cluster Zookeeper Hosts

172.31.17.155:2181,172.31.38.114:2181,172.31.14.212:2181

Kafka Version

1.1.0 ▾

☐ Enable JMX Polling (Set JMX_PORT env variable before starting kafka server)

JMX Auth Username

JMX Auth Password

Cluster Configuration Image - 1/4

In Cluster Zookeeper Hosts, Replace Private IPs of your Zookeeper Nodes Private IPs. As we have done before many times. Other configurations will be same as shown in images.

- ☐ JMX with SSL
- ☐ Enable Logkafka
- ☒ Poll consumer information (Not recommended for large # of consumers)
- ☐ Filter out inactive consumers
- ☐ Enable Active OffsetCache (Not recommended for large # of consumers)
- ☐ Display Broker and Topic Size (only works after applying [this patch](#))

brokerViewUpdatePeriodSeconds

30

clusterManagerThreadPoolSize

2

clusterManagerThreadPoolQueueSize

100

kafkaCommandThreadPoolSize

2

kafkaCommandThreadPoolQueueSize

100

logkafkaCommandThreadPoolSize

2

kafkaCommandThreadPoolQueueSize

100

logkafkaCommandThreadPoolSize

2

logkafkaCommandThreadPoolQueueSize

100

logkafkaUpdatePeriodSeconds

30

partitionOffsetCacheTimeoutSecs

5

brokerViewThreadPoolSize

2

brokerViewThreadPoolQueueSize

1000

offsetCacheThreadPoolSize

2

offsetCacheThreadPoolQueueSize

1000

offsetCacheThreadPoolQueueSize

1000

kafkaAdminClientThreadPoolSize

2

kafkaAdminClientThreadPoolQueueSize

1000

kafkaManagedOffsetMetadataCheckMillis

30000

kafkaManagedOffsetGroupCacheSize

1000000

kafkaManagedOffsetGroupExpireDays

7

Security Protocol

PLAINTEXT

Save

Cancel

References

1. [Kafka Quickstart](#)
2. [LogKafka](#)

Cluster Configuration Image - 4/4

Now, Click on "Save" then Click on "Go To Cluster View".

The screenshot shows the Kafka Manager web interface in a browser. The address bar indicates the URL is `18.130.45.60:9000/clusters/Kafka`. The interface has a top navigation bar with links for Clusters, Kafka, Summary, Brokers, Topic, Preferred Replica Election, Reassign Partitions, and Consumers. Below this, there's a breadcrumb trail: Clusters / Kafka / Summary. The main content area is divided into two sections: 'Cluster Information' and 'Cluster Summary'. The 'Cluster Information' section shows a table with 'Zookeepers' (three IP addresses) and 'Version' (1.1.0). The 'Cluster Summary' section shows a table with 'Topics' (4) and 'Brokers' (3).

Cluster Information	
Zookeepers	172.31.17.155:2181 172.31.38.114:2181 172.31.14.212:2181
Version	1.1.0

Cluster Summary	
Topics	4
Brokers	3

UI After Configured and Save of Kafka Manager

We have successfully Install, Create and Configured Kafka Manager. You can access anytime by Public IP:9000 anytime and manage your Kafka Cluster i.e. topics, consumers, offsets, brokers, replica distribution, partition distribution.

Setup Prometheus Server and Agent

Prometheus is a free software application used for event monitoring and alerting. It records metrics in a time series database built using an HTTP pull model, with flexible queries and real-time alerting.

Using Prometheus we can fetch data from Zookeeper Node 1, 2 and 3, Kafka Node/Broker 1, 2 and 3 related to cluster.

First we will setup Prometheus JMX Agent on Zookeeper and Kafka Nodes then we will setup server on Admin/Monitoring machine.

Setup Prometheus JMX Java Agent on Zookeeper Node 1

Follow below instructions and commands to download Prometheus JMX agent on Zookeeper Node.

- ssh into Zookeeper Node 1
- `cd /home/ec2-user`
- `mkdir prometheus`
- `cd prometheus`
- `wget https://repo1.maven.org/maven2/io/prometheus/jmx/jmx_prometheus_javaagent/0.3.1/jmx_prometheus_javaagent-0.3.1.jar`
- `nano zookeeper.yml`
- Copy below code and paste into zookeeper.yml file. When you use "nano zookeeper.yml" then nano editor will be open.

rules:

```
# replicated Zookeeper
- pattern: "org.apache.ZooKeeperService<name0=ReplicatedServer_id(\\d+)><>(\\w+)"
  name: "zookeeper_$2"
```

```

type: GAUGE
- pattern: "org.apache.ZooKeeperService<name0=ReplicatedServer_id(\\d+),
name1=replica.(\\d+)><>(\\w+)"
name: "zookeeper_$3"
type: GAUGE
labels:
  replicald: "$2"
- pattern: "org.apache.ZooKeeperService<name0=ReplicatedServer_id(\\d+), name1=replica.(\\d+),
name2=(\\w+)><>(Packets\\w+)"
name: "zookeeper_$4"
type: COUNTER
labels:
  replicald: "$2"
  memberType: "$3"
- pattern: "org.apache.ZooKeeperService<name0=ReplicatedServer_id(\\d+), name1=replica.(\\d+),
name2=(\\w+)><>(\\w+)"
name: "zookeeper_$4"
type: GAUGE
labels:
  replicald: "$2"
  memberType: "$3"
- pattern: "org.apache.ZooKeeperService<name0=ReplicatedServer_id(\\d+), name1=replica.(\\d+),
name2=(\\w+), name3=(\\w+)><>(\\w+)"
name: "zookeeper_$4_$5"
type: GAUGE
labels:
  replicald: "$2"
  memberType: "$3"
# standalone Zookeeper
- pattern: "org.apache.ZooKeeperService<name0=StandaloneServer_port(\\d+)><>(\\w+)"
type: GAUGE
name: "zookeeper_$2"
- pattern: "org.apache.ZooKeeperService<name0=StandaloneServer_port(\\d+),
name1=InMemoryDataTree><>(\\w+)"
type: GAUGE
name: "zookeeper_$2"

```

- **Ctrl + o** For Write Out
- **Ctrl + x** For Exit From Nano Editor

Now we have downloaded Prometheus Java/JMX Agent and configured zookeeper.yml file for fetch metrics for zookeeper node. Now we need to start this agent. So, We have to make an entry in zookeeper.service file for do this.

As you remember we have configured zookeeper.service and comment some lines in file using (#) sign. So, We will uncomment it and restart zookeeper.service for work properly. To do this follow below commands and instructions.

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

```

[Unit]
Description=zookeeper
After=network.target

```

```
[Service]
#User=ec2-user
#Group=ec2-user
Environment="KAFKA_HEAP_OPTS=-Xmx256M -Xms128M"
Environment="KAFKA_OPTS=-javaagent:/home/ec2-user/prometheus/jmx_prometheus_javaagent-0.3.1.jar=8080:/home/ec2-user/prometheus/zookeeper.yml"
ExecStart=/home/ec2-user/kafka/bin/zookeeper-server-start.sh
/home/ec2-user/kafka/config/zookeeper.properties
SuccessExitStatus=143
```

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

- Copy above script and Paste into zookeeper.service file.
- **Ctrl + o**
- **Ctrl + x**

Now, You can see in this script we have uncomment

Environment="KAFKA_OPTS=-javaagent:/home/ec2-user/prometheus/jmx_prometheus_javaagent-0.3.1.jar=8080:/home/ec2-user/prometheus/zookeeper.yml" if you compare to old one.

In this line we have give path to Prometheus JMX Java Agent File and zookeeper.yml file.

- **sudo systemctl daemon-reload**
- **sudo systemctl restart zookeeper**

Above command will restart zookeeper and restart Prometheus JMX Java Agent on port 8080. You can see metrics on terminal for verification. Is it working or not using below command.

- **curl localhost:8080**

```
# HELP jvm_memory_pool_bytes_init Initial bytes of a given JVM memory pool.
# TYPE jvm_memory_pool_bytes_init gauge
jvm_memory_pool_bytes_init{pool="CodeHeap 'non-nmethods'",} 2555904.0
jvm_memory_pool_bytes_init{pool="Metaspace",} 0.0
jvm_memory_pool_bytes_init{pool="CodeHeap 'profiled nmethods'",} 2555904.0
jvm_memory_pool_bytes_init{pool="Compressed Class Space",} 0.0
jvm_memory_pool_bytes_init{pool="G1 Eden Space",} 7340032.0
jvm_memory_pool_bytes_init{pool="G1 Old Gen",} 1.26877696E8
jvm_memory_pool_bytes_init{pool="G1 Survivor Space",} 0.0
jvm_memory_pool_bytes_init{pool="CodeHeap 'non-profiled nmethods'",} 2555904.0
# HELP jvm_classes_loaded The number of classes that are currently loaded in the JVM
# TYPE jvm_classes_loaded gauge
jvm_classes_loaded 3923.0
# HELP jvm_classes_loaded_total The total number of classes that have been loaded since the JVM has started execution
# TYPE jvm_classes_loaded_total counter
jvm_classes_loaded_total 3923.0
# HELP jvm_classes_unloaded_total The total number of classes that have been unloaded since the JVM has started execution
# TYPE jvm_classes_unloaded_total counter
jvm_classes_unloaded_total 0.0
# HELP jvm_gc_collection_seconds Time spent in a given JVM garbage collector in seconds.
# TYPE jvm_gc_collection_seconds summary
jvm_gc_collection_seconds_count{gc="G1 Young Generation",} 90.0
jvm_gc_collection_seconds_sum{gc="G1 Young Generation",} 0.314
jvm_gc_collection_seconds_count{gc="G1 Old Generation",} 0.0
jvm_gc_collection_seconds_sum{gc="G1 Old Generation",} 0.0
```

Output After Use curl localhost:8080 command

Setup Prometheus JMX Java Agent on Zookeeper Node 2 and 3

We have completed setup Prometheus JMX Java Agent in Zookeeper Node 1. Same process will be done on Zookeeper Node 2 and 3 with same Agent file and same zokeeper.yml file.

- ssh into Zookeeper Node 2
- Follow Zookeeper Node 1 Setup Configuration for Configure Zookeeper Node 2

After complete setup of Zookeeper Node 2

- ssh into Zookeeper Node 3
- Follow Zookeeper Node 1 Setup Configuration for Configure Zookeeper Node 3

Note: Don't forget to restart zookeeper.service using **sudo systemctl restart zookeeper** command on each Node.

Setup Prometheus JMX Java Agent on Kafka Node/Broker 1

Follow below instructions and commands to download Prometheus JMX Java Agent on Kafka Node/Broker.

- **cd /home/ec2-user**
- **mkdir prometheus**
- **cd prometheus**
- **wget**
https://repo1.maven.org/maven2/io/prometheus/jmx/jmx_prometheus_javaagent/0.3.1/jmx_prometheus_javaagent-0.3.1.jar
- **wget**
https://raw.githubusercontent.com/prometheus/jmx_exporter/master/example_configs/kafka-0-8-2.yml

This time we have downloaded Prometheus JMX Java agent and kafka.yml script. So, We no need to configure kafka.yml manually.

Now, Let's modify service file for kafka.service like we have done for zookeeper.

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

Copy below code and Replace/Paste into kafka.service file or uncomment Environment line.

[Unit]

Description=Kafka

After=network.target

[Service]

#User=ec2-user

#Group=ec2-user

Environment="KAFKA_HEAP_OPTS=-Xmx256M -Xms128M"

Environment="KAFKA_OPTS=-javaagent:/home/ec2-user/prometheus/jmx_prometheus_javaagent-0.3.1.jar=8080:/home/ec2-user/prometheus/kafka-0-8-2.yml"

ExecStart=/home/ec2-user/kafka/bin/kafka-server-start.sh /home/ec2-user/kafka/config/server.properties

SuccessExitStatus=143

[Install]

WantedBy=multi-user.target

- **Ctrl + o**
- **Ctrl + x**

Now, You can see in this script we have uncomment

Environment="KAFKA_OPTS=-javaagent:/home/ec2-user/prometheus/jmx_prometheus_javaagent-0.3.1.jar=8080:/home/ec2-user/prometheus/zookeeper.yml" if you compare to old one.

In this line we have give path to Prometheus JMX Java Agent File and kafka.yml file.

- **sudo systemctl daemon-reload**
- **sudo systemctl restart kafka**

Above command will restart zookeeper and restart Prometheus JMX Java Agent on port 8080. You can see metrics on terminal for verification. Is it working or not using below command.

- **curl localhost:8080**

```
kafka_server_delayedoperationpurgatory_numdelayedoperations_delayedoperation_rebalance 0.0
# HELP kafka_controller_controllerchannelmanager_totalqueuesize Attribute exposed for management (kafka.controller<type=ControllerChannelManager>TotalQueueSize<><Value>)
# TYPE kafka_controller_controllerchannelmanager_totalqueuesize untyped
kafka_controller_controllerchannelmanager_totalqueuesize 0.0
# HELP kafka_controller_kafkacontroller_topicsineligibletodeletecount Attribute exposed for management (kafka.controller<type=KafkaController>TopicsIneligibleToDeleteCount<><Value>)
# TYPE kafka_controller_kafkacontroller_topicsineligibletodeletecount untyped
kafka_controller_kafkacontroller_topicsineligibletodeletecount 0.0
# HELP kafka_server_brokertopicmetrics_fetchmessageconversions_total Attribute exposed for management (kafka.server<type=Broker>FetchMessageConversionsPerSec<><Count>)
# TYPE kafka_server_brokertopicmetrics_fetchmessageconversions_total counter
kafka_server_brokertopicmetrics_fetchmessageconversions_total 0.0
# HELP jmx_scrape_duration_seconds Time this JMX scrape took, in seconds.
# TYPE jmx_scrape_duration_seconds gauge
jmx_scrape_duration_seconds 0.580612105
# HELP jmx_scrape_error Non-zero if this scrape failed.
# TYPE jmx_scrape_error gauge
jmx_scrape_error 0.0
# HELP jvm_gc_collection_seconds Time spent in a given JVM garbage collector in seconds.
# TYPE jvm_gc_collection_seconds summary
jvm_gc_collection_seconds_count{gc="G1 Young Generation",} 3929.0
jvm_gc_collection_seconds_sum{gc="G1 Young Generation",} 12.272
jvm_gc_collection_seconds_count{gc="G1 Old Generation",} 2.0
jvm_gc_collection_seconds_sum{gc="G1 Old Generation",} 0.145
```

Output After Use "curl localhost:8080" command

Setup Prometheus JMX Java Agent on Kafka Node/Broker 2 and 3

We have completed setup Prometheus JMX Java Agent in Kafka Node/Broker 1. Same process will be done on Kafka Node/Broker 2 and 3.

- ssh into Kafka Node/Broker 2
- Follow Kafka Node/Broker 1 Setup Configuration for Configure Kafka Node/Broker 2

After complete setup of Kafka Node/Broker 2

- ssh into Kafka Node/Broker 3
- Follow Kafka Node/Broker 1 Setup Configuration for Configure Kafka Node/Broker 3

Note: Don't forget to restart **kafka.service** using **sudo systemctl restart kafka** command on each Node/Broker.

Setup Prometheus Server on Admin/Monitoring Machine

Now, Prometheus JMX Java Agent deployed on Zookeeper Node 1, 2 and 3. Kafka Node/Broker 1, 2 and 3. All metrics is ready to be use. So, We will configure Prometheus Server. Prometheus server will get all metrics form JMX Java agent and show us on Webpage UI.

Follow below instructions and commands for configure and setup:-

- ssh into Admin/Monitoring Machine
- **cd /home/ec2-user**
- **wget**
<https://github.com/prometheus/prometheus/releases/download/v2.44.0/prometheus-2.44.0.linux-amd64.tar.gz>
- **tar -xvf prometheus-2.44.0.linux-amd64.tar.gz**
- **rm prometheus-2.44.0.linux-amd64.tar.gz**
- **mv prometheus-2.44.0.linux-amd64/ prometheus**
- **cd prometheus**
- **nano prometheus.yml**

Copy below code and Paste into prometheus.yml file which has opened in nano editor after use last command.

```
global:
  scrape_interval: 10s
  evaluation_interval: 10s
scrape_configs:
  - job_name: 'kafka'
    static_configs:
      - targets:
        - 172.31.19.35:8080 # Kafka 1 - change IP for your use case
        - 172.31.32.42:8080 # Kafka 2
        - 172.31.5.194:8080 # Kafka 3
  - job_name: 'zookeeper'
    static_configs:
      - targets:
        - 172.31.17.155:8080 # zookeeer IP and Assign port
        - 172.31.38.114:8080 # zookeeper 2 IP and assign port
        - 172.31.14.212:8080 # zookeeper 3 IP and assign port
```

Given above configuration need to be modify. i.e. Kafka Private IPs and Zookeeper Private IPs of all Instances. Copy own and replace here.

- **Ctrl + o**
- **Ctrl + x**

Note: **.yml** file is very sensitive for format, design. So, use in given format. Do not modify design or format. Just replace your IPs here.

Create SystemD or Service File For Prometheus Server

Now, Prometheus binaries and configuration file downloaded and configured properly. Let's create systemd or service file for Start/Run Prometheus Server. Follow below commands and instructions to do this.

- **sudo nano /etc/systemd/system/prometheus.service**

Copy below code and Paste into prometheus.service file

```
# /etc/systemd/system/prometheus.service
[Unit]
Description=Prometheus Server
Documentation=https://prometheus.io/docs/introduction/overview/
After=network-online.target
```

[Service]

User=ec2-user

ExecStart=/home/ec2-user/prometheus/prometheus

--config.file=/home/ec2-user/prometheus/prometheus.yml

--storage.tsdb.path=/home/ec2-user/prometheus/data

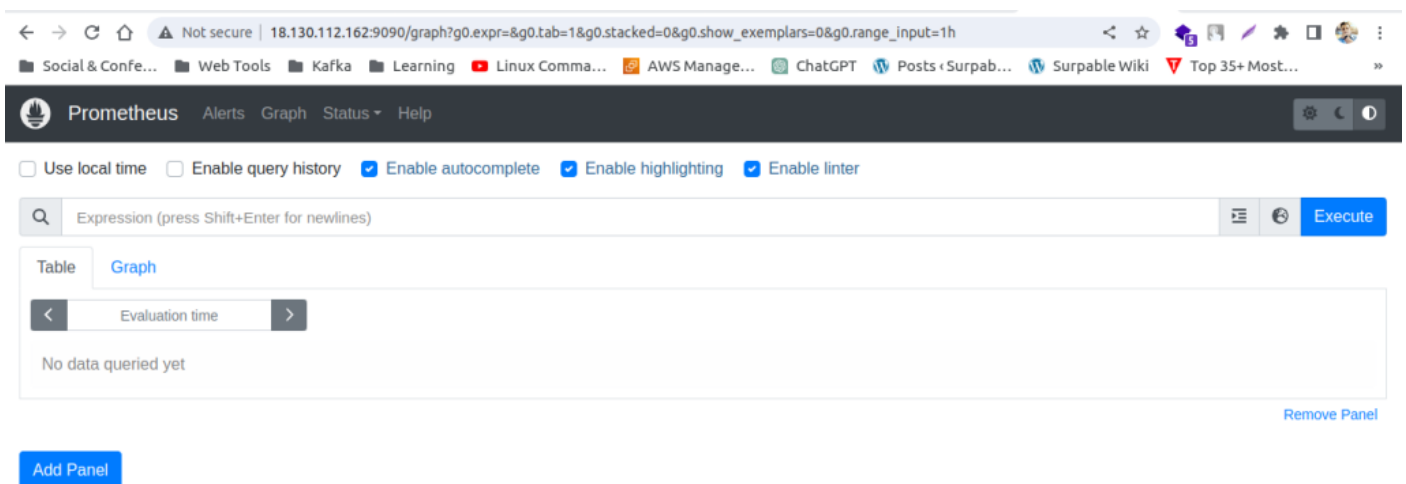
[Install]

WantedBy=multi-user.target

- **Ctrl + o**
- **Ctrl +x**
- **sudo systemctl daemon-reload**
- **sudo systemctl start prometheus**
- **sudo systemctl enable prometheus**
- **sudo systemctl status prometheus**

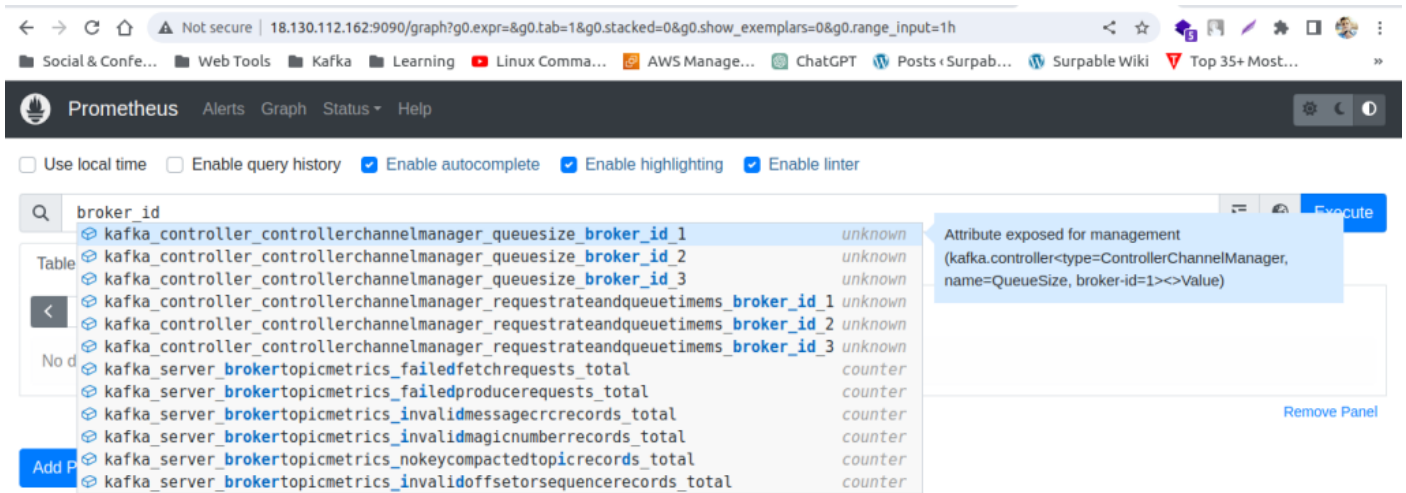
After use last command you can see prometheus is running. Now It's time to access metrics on Prometheus Server UI on Webpage.

- Open Web Browser and Enter below URL
- <Public IP of Admin/Monitor Machine>:9090
- Example - **18.130.112.162:9090**



Prometheus Server UI

Search in Search Box for get metrics of Zookeeper and Kafka. See below:-



Searching Metrics in Prometheus

Go to Status >> Targets To See Kafka and Zookeeper Metrics UP/Down.

Setup and Access Grafana Server and Dashboard

Grafana is a multi-platform open source analytics and interactive visualization web application. It provides charts, graphs, and alerts for the web when connected to supported data sources. It can be easily installed using Docker or Docker Compose.

Today, We will setup for Prometheus Metrics to Monitor Kafka Cluster.

- SSH into Admin/Monitor Machine
- **cd /home/ec2-user**
- **wget**
<https://dl.grafana.com/enterprise/release/grafana-enterprise-9.3.0-beta1.linux-amd64.tar.gz>
- **tar -zxvf grafana-enterprise-9.3.0-beta1.linux-amd64.tar.gz**
- **rm grafana-enterprise-9.3.0-beta1.linux-amd64.tar.gz**
- **mv grafana-9.3.0-beta1/ grafana**
- **cd grafana**
- **nano conf/defaults.ini**
- See Image below and find the section as modify as given image
- For Find Press "**Ctrl + w**" and type "**Anonymous Auth**" and Press Enter

```
##### Anonymous Auth #####
[auth.anonymous]
# enable anonymous access
enabled = true

# specify organization name that should be used for unauthenticated users
org_name = Main Org.

# specify role for unauthenticated users
org_role = Admin

# mask the Grafana version number for unauthenticated users
hide_version = false
```

Find this section and modify like this on your defaults.ini file

- **Ctrl + o**
- **Ctrl + x**

This configuration help us to authenticate direct into Grafana without login. This is good for learning purpose but not for production purpose.

Setup Grafana SystemD or Service File

Now, Grafana Server has been successfully configured and it's time to start but we have to create service file for Start/Stop Grafana like Kafka and Zookeeper. To do this follow below instructions and commands.

- **sudo nano /etc/systemd/system/grafana.service**

Copy below code and paste into grafana.service file in nano editor. Nano editor will be open after use below command.

```
# /etc/systemd/system/grafana.service
[Unit]
Description=Grafana Server
After=network-online.target

[Service]
User=ec2-user
WorkingDirectory=/home/ec2-user/grafana
ExecStart=/home/ec2-user/grafana/bin/grafana-server

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

- **Ctrl + o**
- **Ctrl + x**

Service file of Grafana has been configured. Let's Start it.

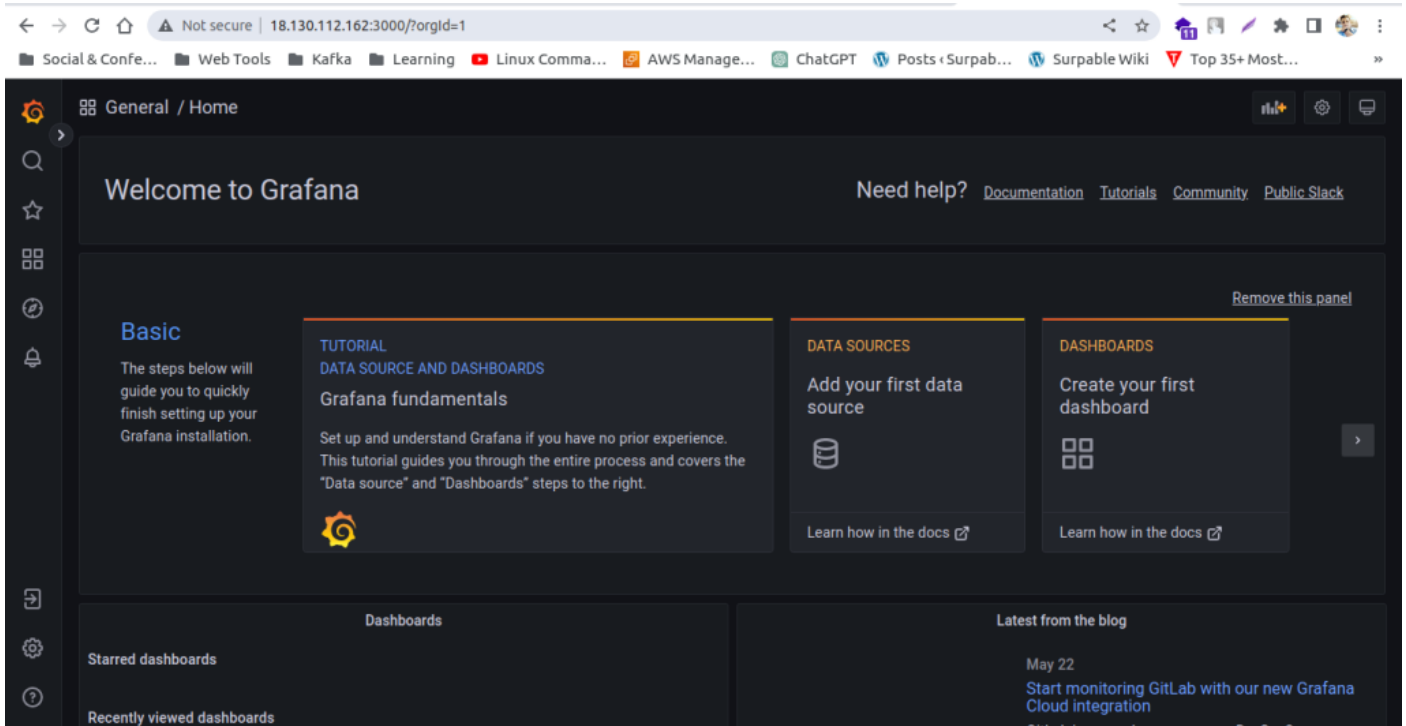
- **sudo systemctl daemon-reload**
- **sudo systemctl enable grafana**
- **sudo systemctl start grafana**

Access Grafana UI on Web Browser For Configure Graph and Data Source

First we have to add data source for graph in Grafana. In this section we will add Prometheus's data source. Grafana will fetch data from Prometheus and Display on Graph.

Access Grafana UI

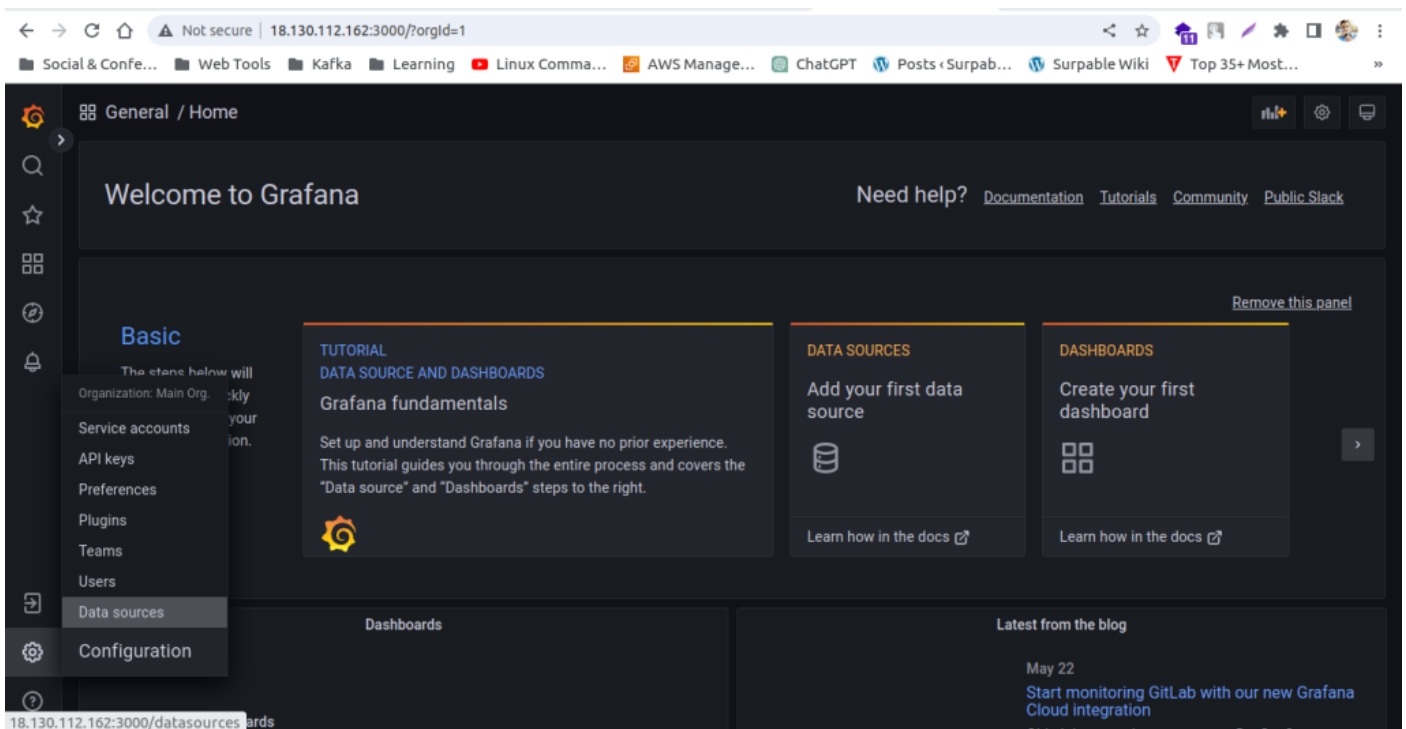
- Go to Web Browser and type URL
- **<Public IP of Admin/Monitoring Instance>:3000**
- Example - **18.130.112.162:3000**



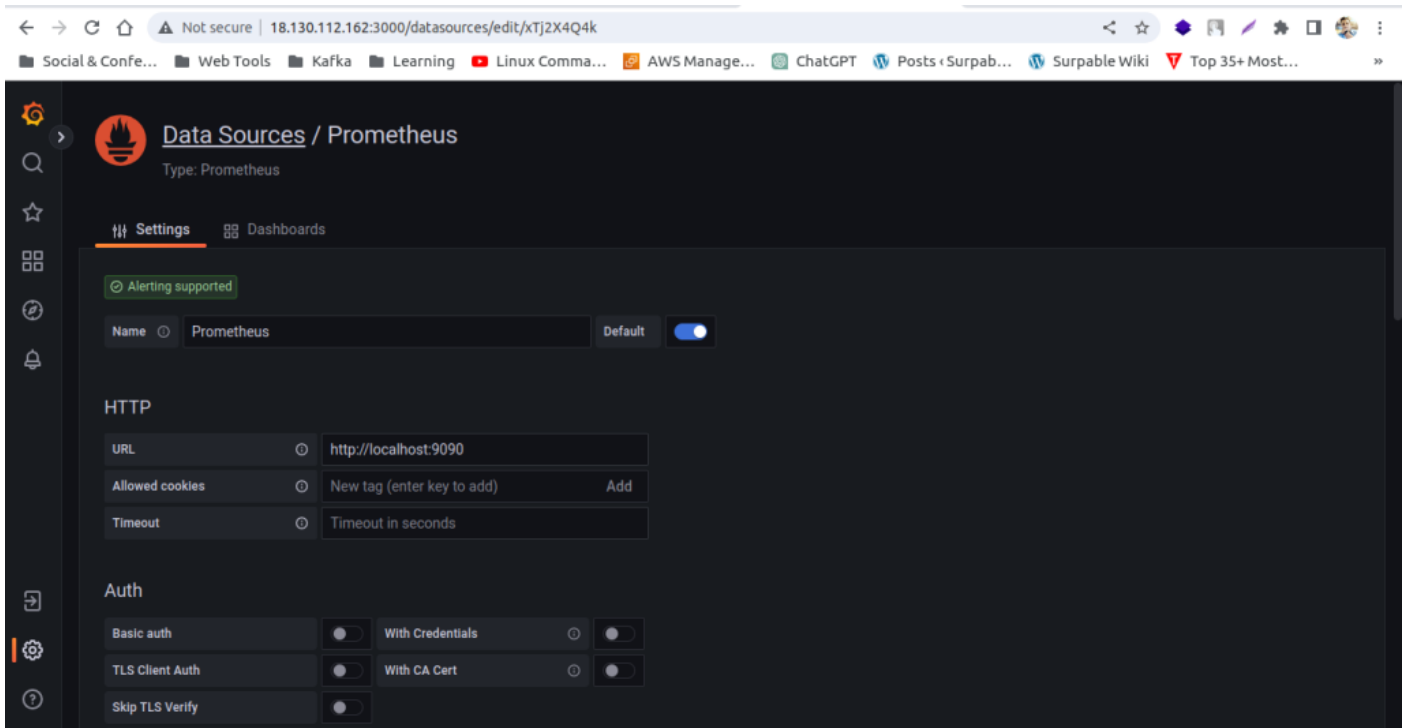
Grafna First UI Before Configure Data Source and Graph

Setup and Configure Data Source

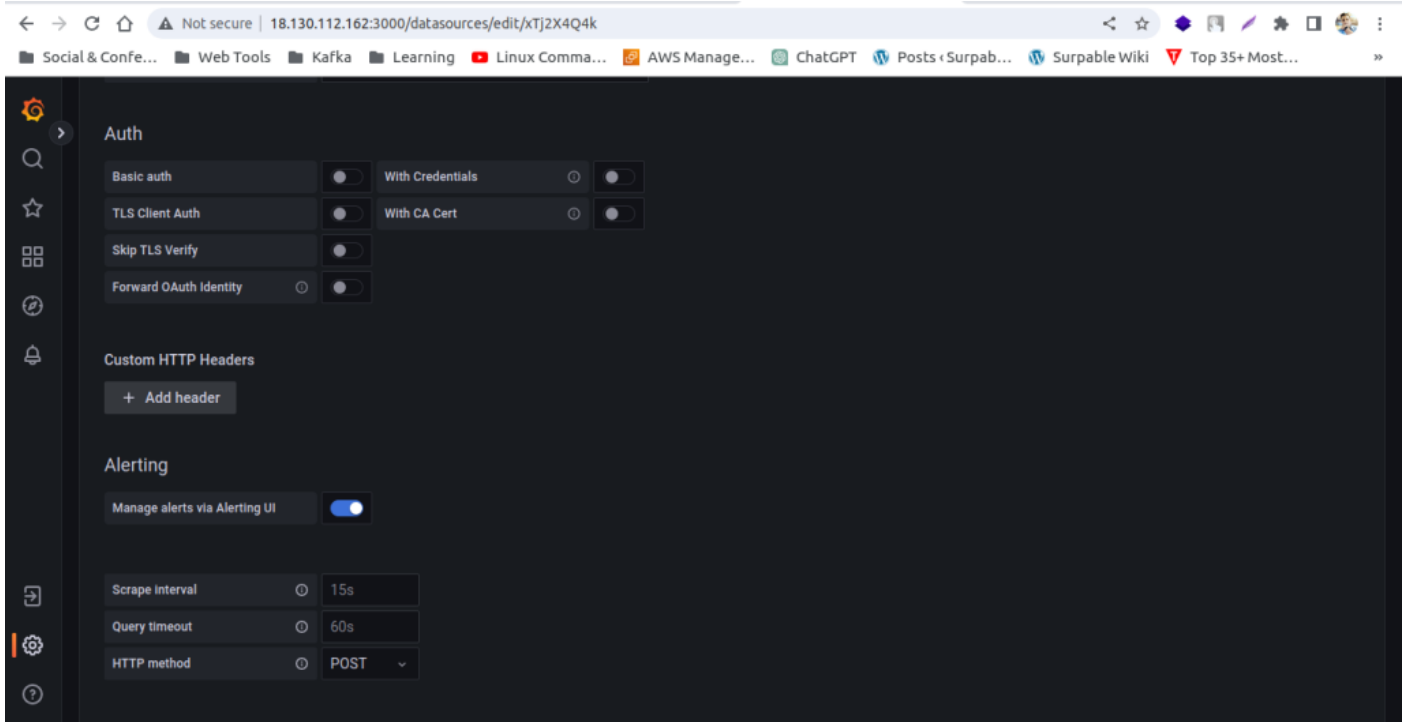
- Click on Setting Icon >> Data Sources
- See Image Below



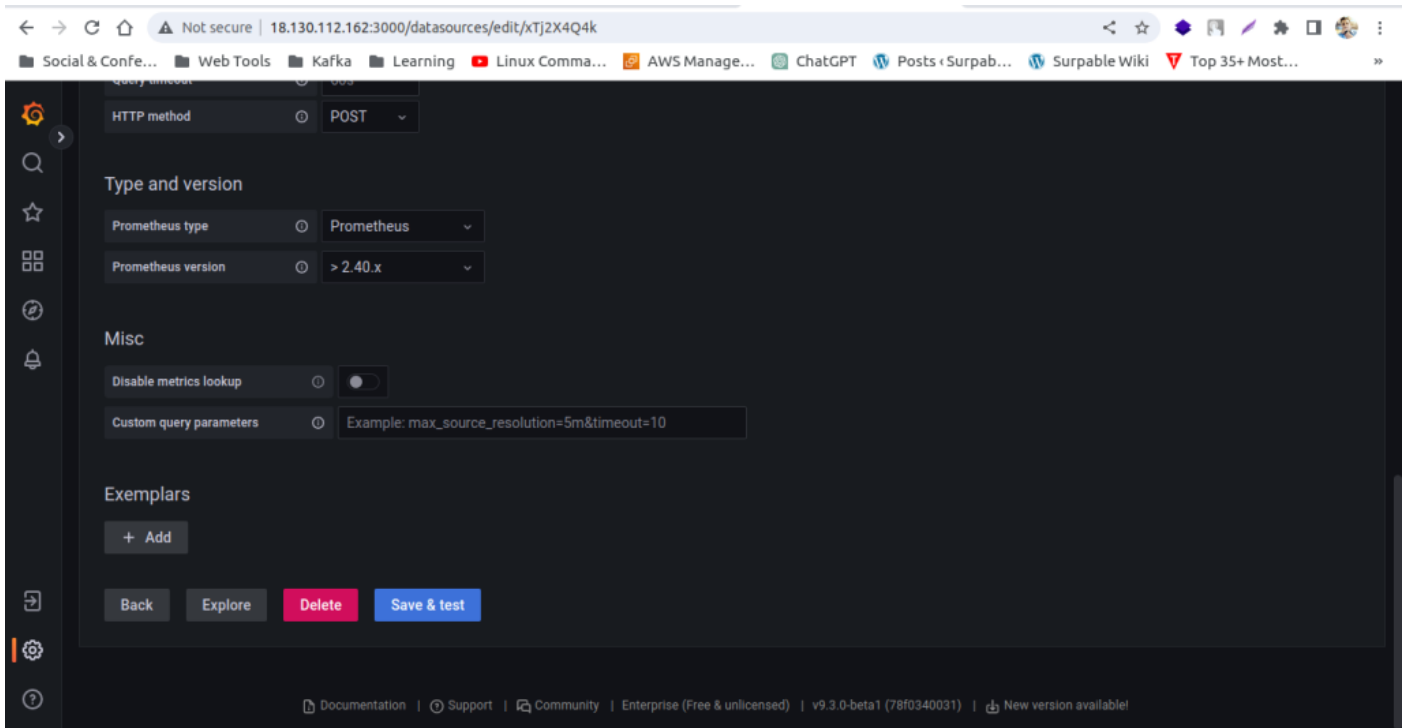
- Click on "Add Data Source" >> Click on "Prometheus"
- Then you will see configuration Page for Configure Data Source For Prometheus
- Follow below images for configuration



Prometheus Data Source Configuration - 1/3



Prometheus Data Source Configuration - 2/3

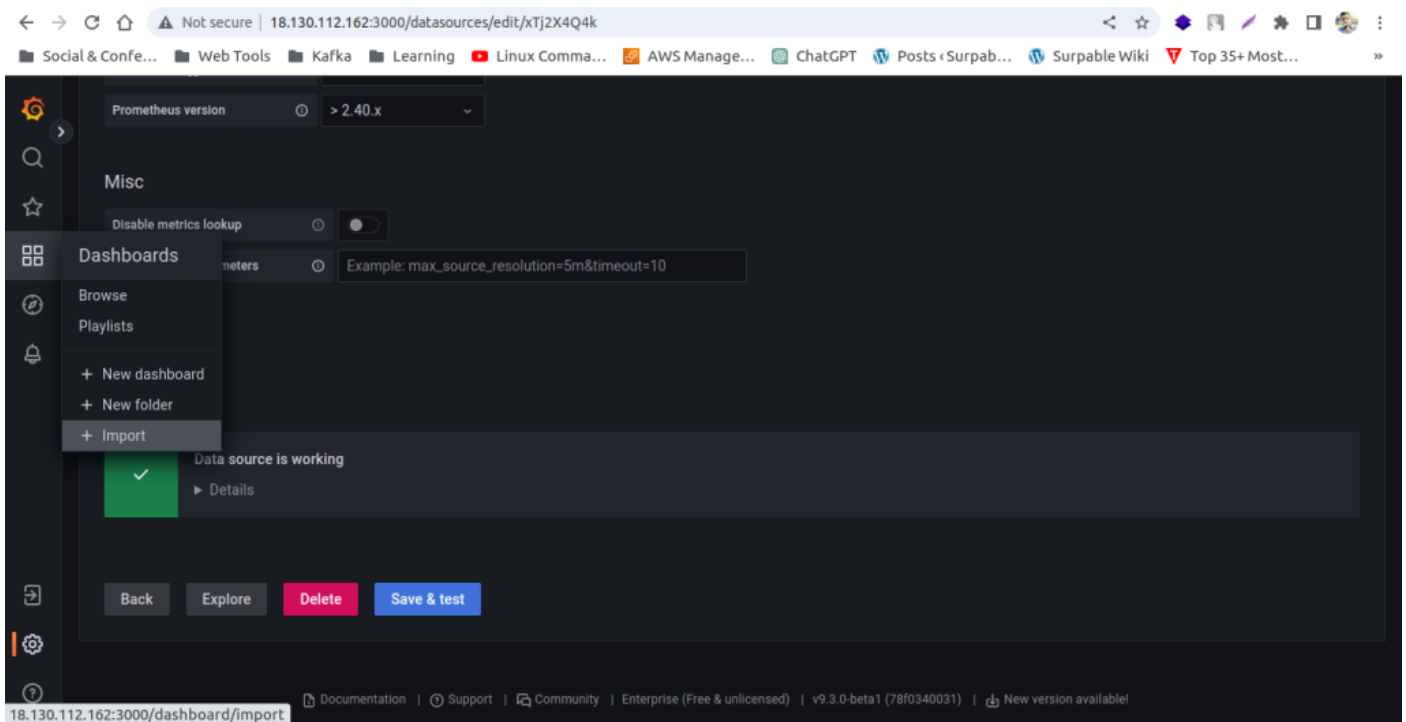


Prometheus Data Source Configuration - 3/3

- Click on **Save & test**. If you get green message "Data Source is Working" then your Data source is configure properly and working.

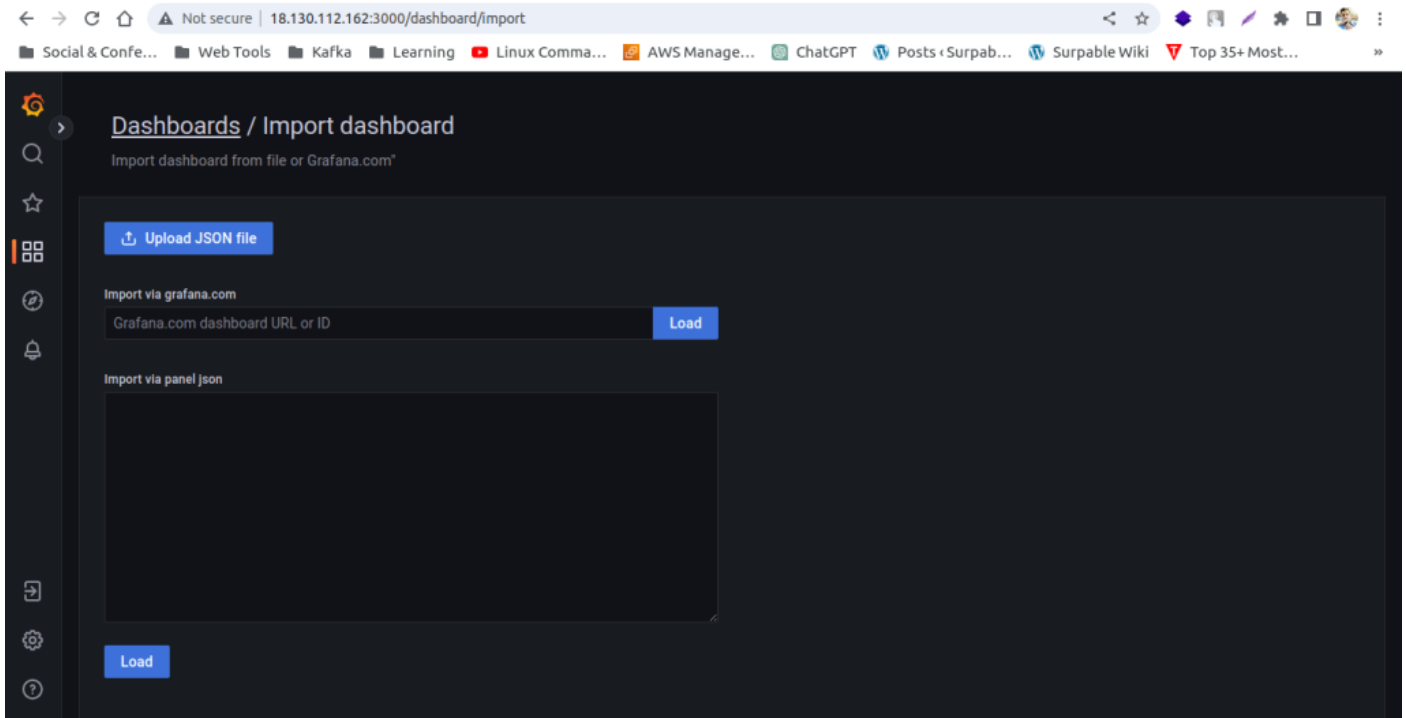
Add Graph in Grafana

- Move your mouse cursor over Dashboards icon then click on Import



Grafana Dashboard Import Step - 1/4

- Now, You will see Import Dashboard Section. Here, We can Enter URL of Grafana dashboard or Enter "json" code of dashboard.
- We will create using "json" code. See image below



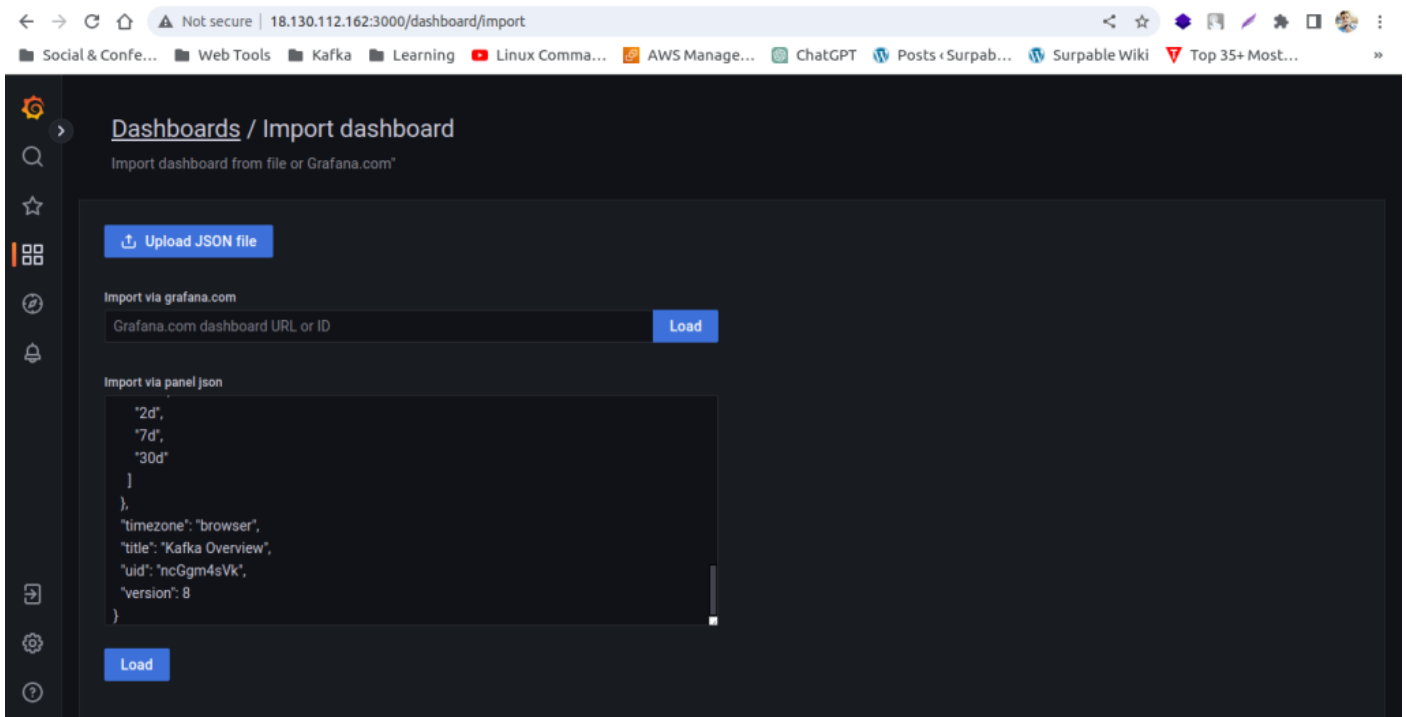
Grafana Dashboard Import Step - 2/4

Kafka Overview Grafana Dashboard Code

- Click below link or open in new tab and copy code and paste into "Import via panel json" box.

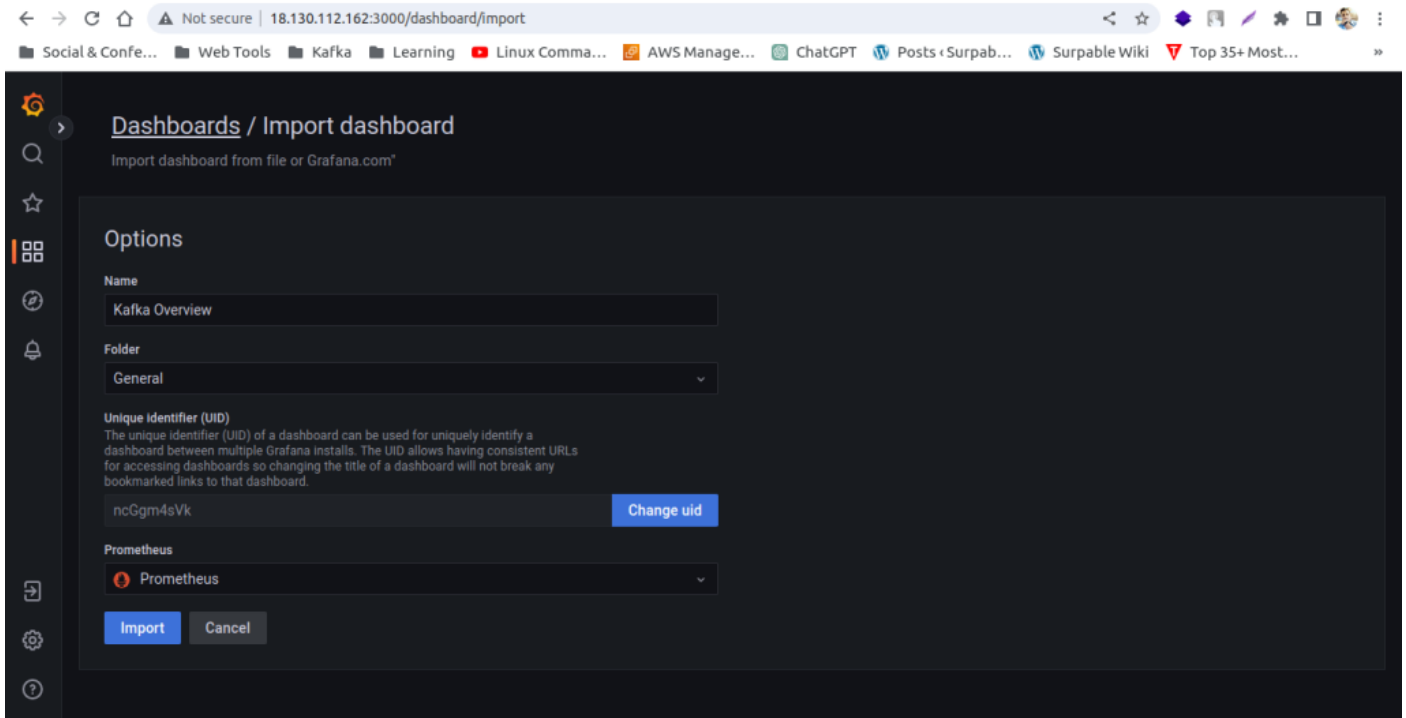
[Kafka-Overview-Grafana-DashboardDownload](#)

- After Paste Click on Load. See Image Below



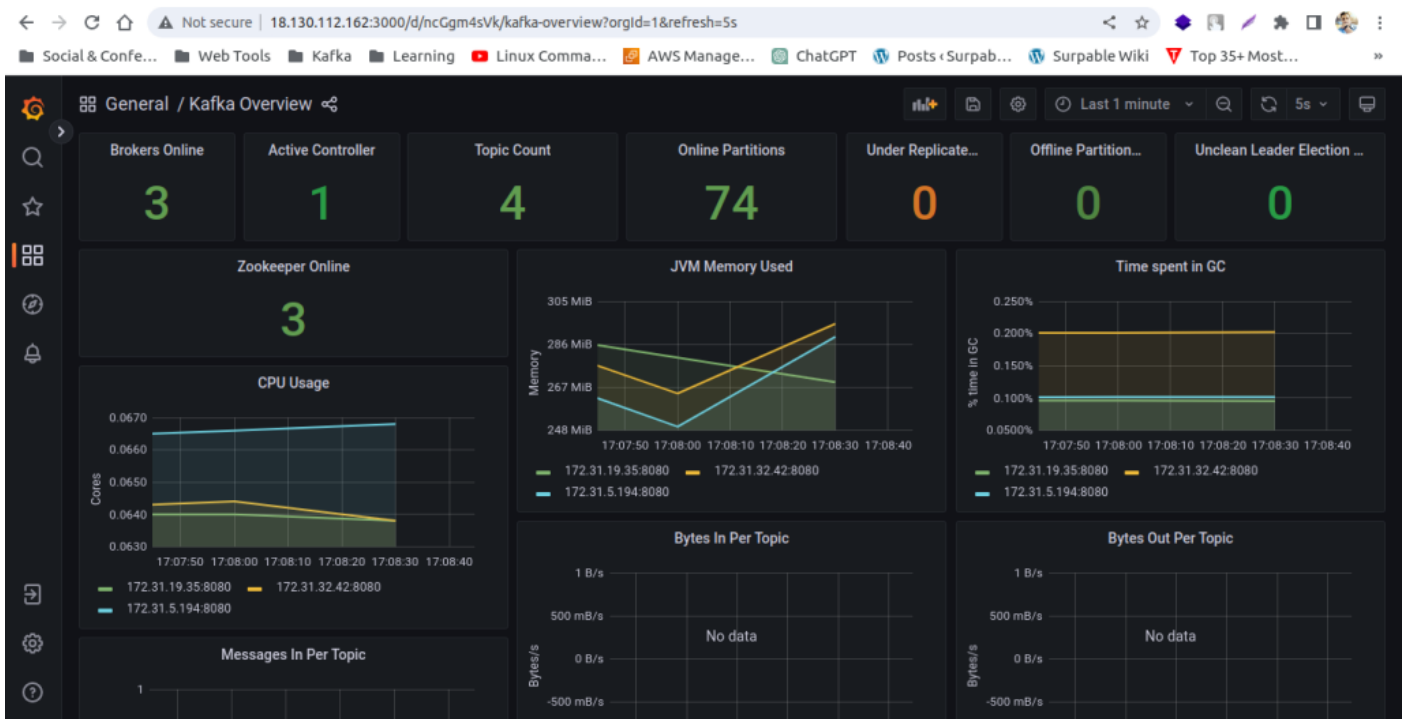
Grafana Dashboard Import Step - 3/4

- After Click on Load. Click on Prometheus and Select Prometheus (Default) >> Click on Import
- See Image Below



Grafana Dashboard Import Step - 4/4

- Grana Graph will be loaded and you can see information Like - Broker Online, Zookeeper Online etc.
- Press **Ctrl + s** and save the dashboard
- See Image below



Grafana Basic Dashboard of Kafka Cluster

Setup Prometheus Node Exporter For Zookeeper and Kafka

Node Exporter is a **Prometheus exporter for server level and OS level metrics**, and measures various server resources such as RAM, disk space, and CPU utilization.

We can use Grafana for monitor Zookeeper and Kafka Instances metrics on Graph. We will setup Graph after configure Node Exporter.

First of all we will setup Node Exporter on Zookeeper Instances 1, 2 and 3 then setup service file. Then setup for Kafka Instances 1, 2 and 3 and also setup service file for start and stop.

Then, Modify prometheus.yml file in Admin/Monitor machine for get those metrics to prometheus after that once prometheus server will get those metrics then we can view on Grafana using dashboard.

Setup Node Exporter For Zookeeper Instances 1, 2 and 3

We will download Prometheus Node Exporter, Configure and create systemd or service file for start metrics. So, Follow given below instructions and commands.

Node Exporter For Zookeeper Instance 1

- SSH into Zookeeper Instance 1
- **cd /home/ec2-user**
- **wget**
https://github.com/prometheus/node_exporter/releases/download/v1.5.0/node_exporter-1.5.0.linux-amd64.tar.gz
- **tar -xvf node_exporter-1.5.0.linux-amd64.tar.gz**
- **rm node_exporter-1.5.0.linux-amd64.tar.gz**

Node Exporter downloaded. Let's create service file for node exporter for run and start metrics.

- **sudo nano /etc/systemd/system/node-exporter.service**
- Copy below code and Paste into **node-exporter.service** file in nano editor that opened after use previous command

[Unit]

Description=Node Exporter

After=network.target

[Service]

#User=node_exporter

#Group=node_exporter

#Type=simple

Type=simple

ExecStart=/home/ec2-user/node_exporter-1.5.0.linux-amd64/node_exporter

[Install]

WantedBy=multi-user.target

- **Ctrl + o**
- **Ctrl + x**

Node Exporter service file has configured. Now do follow commands to start.

- **sudo systemctl daemon-reload**
- **sudo systemctl enable node-exporter**
- **sudo systemctl start node-exporter**
- **sudo systemctl status node-exporter**

Node Exporter service has been started. You can verify using below command.

- **curl localhost:9100/metrics**
- If you see lots of metrics on terminal then it means your Node Exporter working properly

Node Exporter will expose metrics on port 9100. Zookeeper Instance 1 has been configured for Node Exporter and exposing metrics.

Node Exporter For Zookeeper Instance 2 and 3

We have configured properly Node Exporter for Zookeeper Node 1. You can now easily configure for Zookeeper Instance 2 and 3 using following instructions **Node Exporter For Zookeeper Instance 1**.

- SSH into Zookeeper Instance 2 and follow **Node Exporter For Zookeeper Instance 1** process.
- SSH into Zookeeper Instance 3 and follow **Node Exporter For Zookeeper Instance 1** process.

Setup Node Exporter For Kafka Instances 1, 2 and 3

Now, We will configure Node Exporter for Kafka Instance 1, 2 and 3

Node Exporter For Kafka Instance 1

- SSH into Kafka Instance 1
- **cd /home/ec2-user**
- **wget**
https://github.com/prometheus/node_exporter/releases/download/v1.5.0/node_exporter-1.5.0.linux-amd64.tar.gz
- **tar -xvf node_exporter-1.5.0.linux-amd64.tar.gz**
- **rm node_exporter-1.5.0.linux-amd64.tar.gz**

Node Exporter downloaded. Let's create a service file for node exporter for run and start metrics.

- **sudo nano /etc/systemd/system/node-exporter.service**
- Copy below code and Paste into node-exporter.service file in nano editor that opened after use previous command

[Unit]

Description=Node Exporter

After=network.target

[Service]

#User=node_exporter

#Group=node_exporter

#Type=simple

Type=simple

ExecStart=/home/ec2-user/node_exporter-1.5.0.linux-amd64/node_exporter

[Install]

WantedBy=multi-user.target

- **Ctrl + o**
- **Ctrl + x**

Node Exporter service file has configured. Now do follow commands to start.

- **sudo systemctl daemon-reload**
- **sudo systemctl enable node-exporter**
- **sudo systemctl start node-exporter**
- **sudo systemctl status node-exporter**

Node Exporter service has been started. You can verify using below command.

- **curl localhost:9100/metrics**
- If you see lots of metrics on terminal then it means your Node Exporter working properly

Node Exporter will expose metrics on port 9100. Kafka Instance 1 has been configured for Node Exporter and exposing metrics.

Node Exporter For Kafka Instance 2 and 3

We have configured properly Node Exporter for Kafka Node 1. You can now easily configure for Kafka Instance 2 and 3 using following instructions **Node Exporter For Kafka Instance 1**.

- SSH into Kafka Instance 2 and follow **Node Exporter For Kafka Instance 1** process.
- SSH into Kafka Instance 3 and follow **Node Exporter For Kafka Instance 1** process.

We have configured and start Node Exporter on Zookeeper Instances and Kafka Instances. Both are exposing metrics over port 9100. Now We will modify **prometheus.yml** file on **Admin/Monitoring Machine**. After modify Prometheus server will able to get metrics.

Modify Prometheus Server Configuration For Node Exporter

Follow the process and commands for modify prometheus.yml file on Monitoring/Admin machine.

- SSH into Admin/Monitoring Instance
- **cd /home/ec2-user**
- **nano prometheus/prometheus.yml**
- Copy and Paste or modify code given below as per your convenient.

global:

scrape_interval: 10s

evaluation_interval: 10s

scrape_configs:

- job_name: 'kafka'

static_configs:

- targets:

- 172.31.19.35:8080 # Kafka 1 - change IP for your use case

- 172.31.32.42:8080 # Kafka 2

- 172.31.5.194:8080 # Kafka 3

- job_name: 'zookeeper'

static_configs:

- targets:

- 172.31.17.155:8080 # zookeeper IP and Assign port on environment section of systemd file

- 172.31.38.114:8080

- 172.31.14.212:8080

- job_name: node

static_configs:

- targets:

- 172.31.17.155:9100 # zookeeper-1 prometheus node exporter

- 172.31.38.114:9100 # zookeeper-2 prometheus node exporter
- 172.31.14.212:9100 # zookeeper-3 prometheus node exporter
- 172.31.19.35:9100 # kafka-1 prometheus node exporter
- 172.31.32.42:9100 # kafka-2 prometheus node exporter
- 172.31.5.194:9100 # kafka-3 prometheus node exporter

- New line need to be added. See the image below

- job_name: node

static_configs:

- targets:

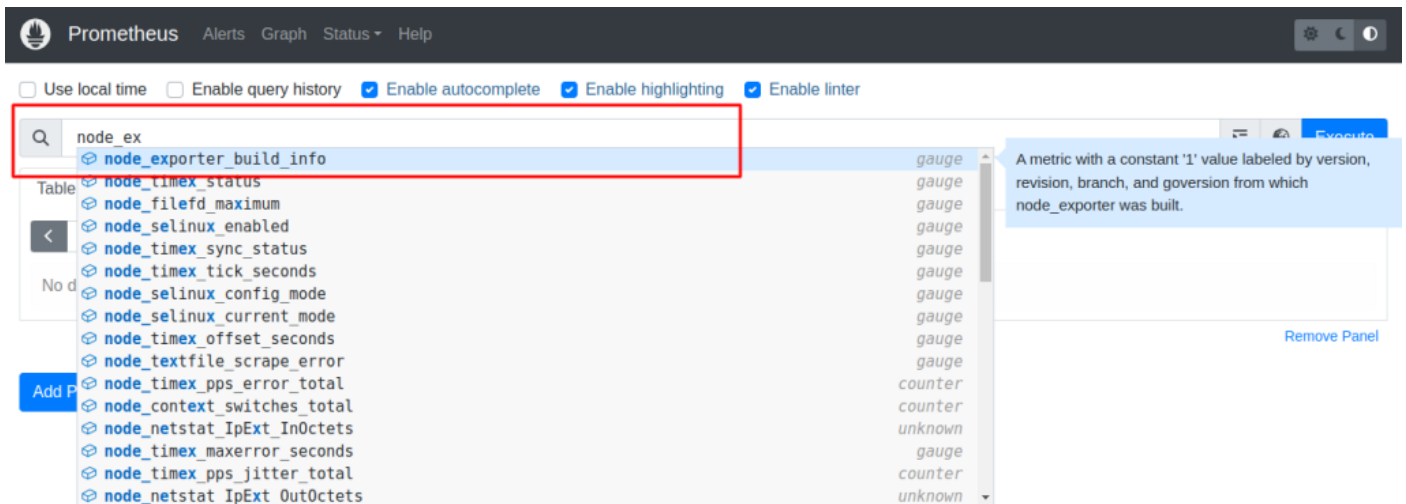
- 172.31.17.155:9100 # zookeeper-1 prometheus node exporter
- 172.31.38.114:9100 # zookeeper-2 prometheus node exporter
- 172.31.14.212:9100 # zookeeper-3 prometheus node exporter
- 172.31.19.35:9100 # kafka-1 prometheus node exporter
- 172.31.32.42:9100 # kafka-2 prometheus node exporter
- 172.31.5.194:9100 # kafka-3 prometheus node exporter

- Modify **Private IPs** with your own. Remain will be same. Then save and exit nano editor using below command
- **Ctrl + o**
- **Ctrl + x**

Node Exporter Configured Properly in Zookeeper Instances, Kafka Instances and also configured in Prometheus server configuration file. Now restart prometheus to start gathering Node Exporter metrics.

- **sudo systemctl restart prometheus**

Now, Access Prometheus server using **<Public IP of your Admin/Monitoring Machine>:9090** for see metrics of Node Exporter. See Image below



Searching Metrics of Node Exporter on Prometheus

You can also see Nodes Up/Down.

- Click on **Status >> Targets**
- See Image Below

Prometheus Alerts Graph Status ▾ Help						
Endpoint	State	Labels	Last Scrape	Duration	Error	
http://172.31.32.42:8080/metrics	UP	instance="172.31.32.42:8080" job="kafka"	3.491s ago	636.525ms		
http://172.31.5.194:8080/metrics	UP	instance="172.31.5.194:8080" job="kafka"	10.838s ago	595.459ms		
http://172.31.19.35:8080/metrics	UP	instance="172.31.19.35:8080" job="kafka"	5.194s ago	625.582ms		

node (6/6 up) show less						
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error	
http://172.31.38.114:9100/metrics	UP	instance="172.31.38.114:9100" job="node"	6.111s ago	14.284ms		
http://172.31.14.212:9100/metrics	UP	instance="172.31.14.212:9100" job="node"	2.346s ago	13.557ms		
http://172.31.19.35:9100/metrics	UP	instance="172.31.19.35:9100" job="node"	4.236s ago	14.067ms		
http://172.31.32.42:9100/metrics	UP	instance="172.31.32.42:9100" job="node"	5.453s ago	13.298ms		
http://172.31.5.194:9100/metrics	UP	instance="172.31.5.194:9100" job="node"	8.345s ago	13.413ms		
http://172.31.17.155:9100/metrics	UP	instance="172.31.17.155:9100" job="node"	4.482s ago	14.822ms		

zookeeper (3/3 up) show less						
--	--	--	--	--	--	--

Nodes Targets Status on Prometheus For Check Up/Down

We have configured and see things working for Node Exporter in Prometheus. Now Let's Import Grafana dashboard for see metrics for Node Exporter.

Setup Grafana Dashboard For Node Exporter

Now, We will setup Grafana dashboard for monitor Node Exporter Metrics. We know how to setup grafana dashboard using json code. So, Copy below json code using given link and import as we do before.

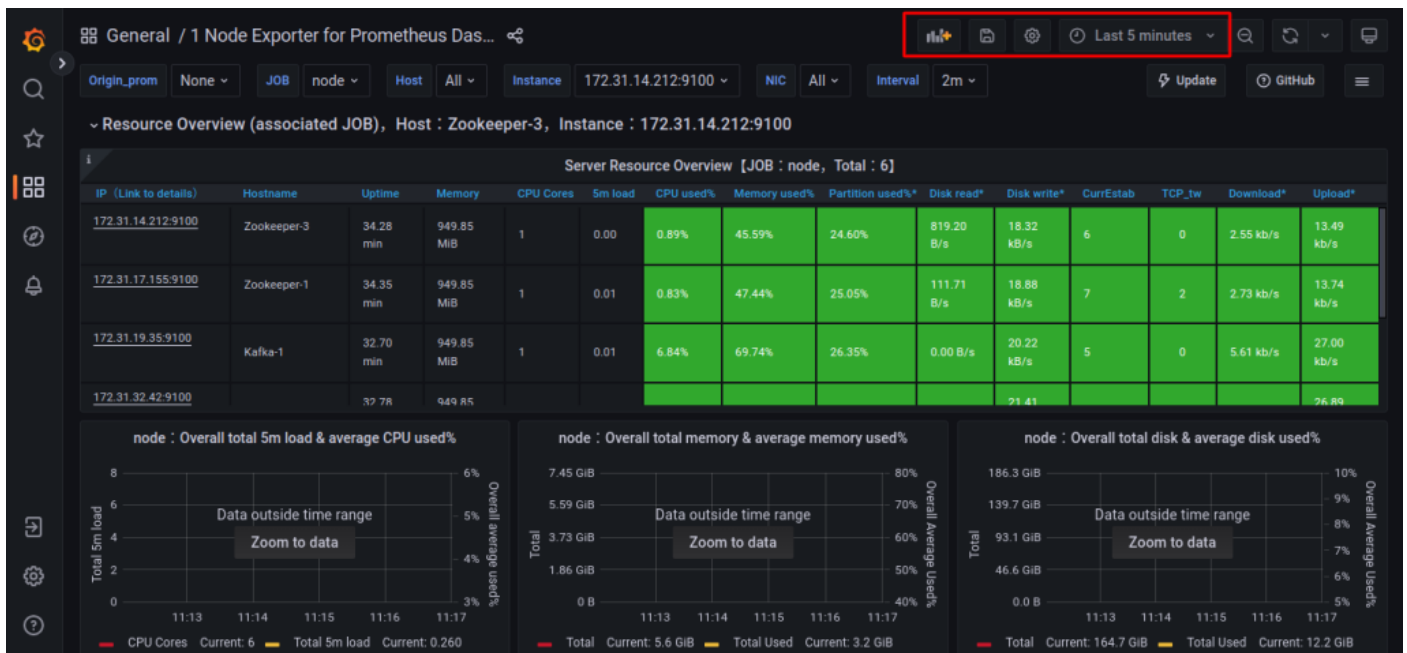
Remember this time we no need to configure Data Source. Because we have already did it. Just copy json code and import in Grafana for dashboard. You need to only select **Prometheus (Default)** on Data Source section.

Node Exporter Grafana Dashboard Code

- Click below link or open in new tab and copy code and paste into "Import via panel json" box.

[node-exporter-for-prometheus-dashboard-en-v20201010_rev9Download](#)

- After Paste Click on Load. See Image Below



Node Exporter Metrics/OS Metrics For Monitor Server Resources

You can change time for accurate information. See highlighted section in image above.

Setup Jolokia Java Agent

Jolokia is **plugin agent that can be used to enable remote access to JMX**. Jolokia exposes JMX metrics for Java applications which can then be queried by third-party agent over HTTP via POST or GET requests.

In our scenario we will setup Jolokia Java agent on Kafka Broker 1, 2 and 3 Instances for rolling restart feature. We will discuss later about Rolling Restart feature. Jolokia Java Agent will exposed on port number 8778.

Setup Jolokia Java Agent on Kafka Brokers

Let's setup Jolokia Java Agent on Brokers.

Setup on Kafka Broker 1

Follow instructions and commands for setup Jolokia Java Agent and test that agent working or not

- SSH into Kafka Broker 1 Instance
- `cd /home/ec2-user`
- `mkdir jolokia`
- `cd jolokia`
- `wget http://search.maven.org/remotecontent?filepath=org/jolokia/jolokia-jvm/1.6.0/jolokia-jvm-1.6.0-agent.jar -O jolokia-agent.jar`
- Jolokia Agent File will be downloaded and renamed to **jolokia-agent.jar** in jolokia directory

Modify kafka.service File

Now, We need to modify kafka.service file for start jolokia agent. Follow the instructions and commands below

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

Copy below script and paste into kafka.service file.

[Unit]

Description=Kafka

After=network.target

[Service]

#User=ec2-user

#Group=ec2-user

Environment="KAFKA_HEAP_OPTS=-Xmx256M -Xms128M"

Environment="KAFKA_OPTS=-javaagent:/home/ec2-user/prometheus/jmx_prometheus_javaagent-0.3.1.jar=8080:/home/ec2-user/prometheus/kafka-0-8-2.yml

-javaagent:/home/ec2-user/jolokia/jolokia-agent.jar=host=*

ExecStart=/home/ec2-user/kafka/bin/kafka-server-start.sh /home/ec2-user/kafka/config/server.properties

SuccessExitStatus=143

[Install]

WantedBy=multi-user.target

We have added a line in 2nd Environment line that is

"-javaagent:/home/ec2-user/jolokia/jolokia-agent.jar=host=*" add this line or copy given script above and replace with previous one. Then, Save and exit from nano editor.

- **Ctrl + o**
- **Ctrl + x**
- **sudo systemctl daemon-reload**
- **sudo systemctl restart kafka**

Jolokia Agent successfully configured on Kafka Broker-1 and we have restarted daemon services and also kafka. Jolokia will start exposing metrics on port 8778.

Setup on Kafka Broker 2 and 3

We have setup and configured Jolokia Java Agent on Broker-1 previously. Follow **Setup on Kafka Broker 1** instructions and configure and setup Jolokia Java Agent on Broker-2 and 3.

This is very simple just SSH into each broker and setup Jolokia, Modify kafka.service, Reload daemon services and restart broker.

Testing Jolokia Java Agent

Now, We will test our Jolokia Java Agent working or not.

- SSH Into Admin/Monitor Machine
- **sudo yum install -y jq**
- **curl <Private Ip of Kafka Broker 1>:8778/jolokia/read/kafka.server:name=UnderReplicatedPartitions,type=ReplicaManager/Value | jq**
- Example - **curl 172.31.19.35:8778/jolokia/read/kafka.server:name=UnderReplicatedPartitions,type=ReplicaManager/Value | jq**


```
[ec2-user@ip-172-31-24-158 ~]$ curl 172.31.19.35:8778/jolokia/read/kafka.server:name=UnderReplicatedPartitions,type=ReplicaManager/Value | jq
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload  Total   Spent    Left     Speed
100    167    0    167    0    0    24429    0 --:--:-- --:--:-- --:--:-- 27833
{
  "request": {
    "mbean": "kafka.server:name=UnderReplicatedPartitions,type=ReplicaManager",
    "attribute": "Value",
    "type": "read"
  },
  "value": 0,
  "timestamp": 1684923156,
  "status": 200
}
[ec2-user@ip-172-31-24-158 ~]$
```

Output of json query of Jolokia Agent on Admin/Monitor Machine

If you see output like above image or similar then your Jolokia Java Agent is working perfectly. You can test Broker 2 and 3 by modifying Private IP of Broker Instance 2 and 3.

Rolling Restart Kafka Brokers

Using Rolling Restart feature we can restart our brokers one by one by ssh into every Broker Instance then restart broker then wait for URP (Under Replication Partitions) goes to zero then ssh into Second Broker;s Instance and do the same for others.

This is safest method to restart brokers. When we modify Kafka configuration or upgrade Kafka Broker to new version then we need to do this kind of operation.

So, Rolling Restart feature is life saver for us. We will download and configure **Kafka-Utills** to configure rolling restart feature.

Download and Setup Kafka-Utills

Kafka-Utills is a library containing tools to interact with kafka clusters and manage them. Follow below instructions for download and configure. This tool will be install on Admin/Monitor machine.

Install Dependencies

- SSH into Admin/Monitor Machine
- `cd /home/ec2-user`
- `sudo yum update -y`
- `sudo yum -y install gcc openssl-devel`
- `curl -O https://bootstrap.pypa.io/get-pip.py`
- `python3 get-pip.py --user`
- `pip install urllib3==1.26.6`
- `pip install --user kafka-utils`
- `kafka-utils -h` (For Check kafka-utils installed or not)
- `sudo mkdir -p /etc/kafka_discovery`
- `sudo nano /etc/kafka_discovery/kafka.yaml`

After last command you will be in nano editor. Copy below code and paste into nano editor in kafka.yaml file. Don't forget to modify Zookeeper Instances Private IP with your.

```
---
clusters:
  cluster-1:
    broker_list:
```

```
- "rolling-restart:9092"
zookeeper: "172.31.17.155:2181,172.31.38.114:2181,172.31.14.212:2181"
local_config:
cluster: cluster-1
```

- **Ctrl + o**
- **Ctrl + x**

List all clusters using below command. But we have configured only 1 Kafka cluster. So, It will show only one on terminal.

- **kafka-utils**

```
[ec2-user@ip-172-31-24-158 ~]$ kafka-utils
cluster-type kafka:
  cluster-name: cluster-1
  broker-list: rolling-restart:9092
  zookeeper: 172.31.17.155:2181,172.31.38.114:2181,172.31.14.212:2181
```

Output of "kafka-utils" command

Use Below command for see brokers will detected or not.

- **kafka-rolling-restart --cluster-type kafka**

```
[ec2-user@ip-172-31-24-158 ~]$ kafka-rolling-restart --cluster-type kafka
Will restart the following brokers in cluster-1:
  1: 172.31.19.35
  2: 172.31.32.42
  3: 172.31.5.194
Do you want to restart these brokers? 
```

Output of "kafka-rolling-restart --cluster-type kafka" command

If you see output that shown in above image then kafka-utils has been configured successfully.

Configure SSH in Admin/Monitor Machine

We will login into our brokers instances using monitoring machine and kafka rolling restart feature will do the same for us. So, We have to configure some SSH in Admin/Monitor machine. So, Follow below step and commands.

Disable Strict Host Checking

- **nano /home/ec2-user/.ssh/config**

Copy below code and paste into opened nano editor

```
Host *
StrictHostKeyChecking no
```

- **Ctrl + o**
- **Ctrl + x**

Give read only permission to config file. Use below command

- `sudo chmod 400 ~/.ssh/config`

Generate Public and Private Key Key For Admin/Monitor Box

- `ssh-keygen`

After execute above command press Enter/Return key untill exit from keygen.

```
[ec2-user@ip-172-31-24-158 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ec2-user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ec2-user/.ssh/id_rsa.
Your public key has been saved in /home/ec2-user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:f0To0yskve+Jq+Rp6xrZVkfkbTXkv2VcpxVe45rRzBA ec2-user@ip-172-31-24-158
The key's randomart image is:
+---[RSA 2048]---+
|      .Eo=o |
|      + .O.= |
|      . +.oB+ |
|      o+ .+++ |
|      S * +O. = |
|      o = = . .O |
|      o + + o . |
|      =.. = . |
|      .+*ooo+ |
+---[SHA256]-----+
[ec2-user@ip-172-31-24-158 ~]$
```

Output of "ssh-keygen" command

Now there are two keys generated into `.ssh` directory. One is private key named `id_rsa` another is Public key named `id_rsa.pub`. We will copy `id_rsa.pub` content and paste into Kafka Brokers Instance's file named `authorized_keys` which is located on `.ssh/authorized_keys`.

For copy `id_rsa.pub` key content follow below commands

- `cd .ssh`
- `cat id_rsa.pub`

```
[ec2-user@ip-172-31-24-158 .ssh]$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCA8Dx98rY3+/FvyrVbrFCPLuQ3AteLletuTKG09Bmf4GstQeUCFMgre+20ondyEF/7nURZtjW5nW+pzILUxc5V5ljG02QULYRzkHV0cSxWJoSpB7
MlpPr8nSnenqGcJKAcNgUIduIVUfYm9bPuLOXMPb8pTHhpGHFTkdL9Dp+NVhLCabGJVf0xweNa273N/v7EhCFxS/+sNXNQLVQx7RucyYdAL5V1bXH0hzQwzgaLuMAjK6+c+J/XYjQxoWlq0W3gPuu
dqepyaJ6/11g2hNg+A0XPn4ouc/uratdGEHBvdw0stVIouwNrbl8UGdZIU1TB9NsUYKGI0Kb8fxVKx ec2-user@ip-172-31-24-158.eu-west-2.compute.internal
```

Output of "cat id_rsa.pub" command

Select text from `ssh-rsa` till end `compute.internal` and copy the key.

Copy Public Key of Admin/Monitor Machine To Brokers 1

Public key has been copied and need to be paste in Broker Instances

- SSH into Broker Instance 1
- `cd /home/ec2-user/.ssh`
- `nano authorized_keys`
- Paste copied key in new line

- | GNU nano 5.8 | authorized keys | Modified |
|--|-----------------|----------|
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCbZit9RC5Hzrt4pARrpjdz9x0tZvibLRzoTJAEg8ARUNJ6kKqkj/WPTypjAvK8ie1Wey7nglZmNVU6sm+Vq5mKGtNQ2PPBAZrVS5AK54ALF1 | | |
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDPQbNg15RFgyTnKRYv6z4q5jBmtFvcXF1v0f1VAYj7F2LkVzbUx4C06kLmqTgwbwP9TeN8djcVLgKiIg47+soby+H0hngzQ5f9ZDfhsiCazZbY | | |
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCB8DxX98rY3+/FvyrVbrFCPLuQ3AtelTetuTKG09Bmf4GStQeUCFMqre+20ondyEF/7nURZtjW5nW+pzlUxc5VsijG02QuiYRzkHVOcSxWJoSpB | | |

Copy Public Key of Admin/Monitor Machine To Brokers 2 and 3

Just SSH into Broker 2 Instance and Paste the public key in **authorized_keys** file which have we copied from **Admin/Monitor machine** and do the same process for **Broker Instance 3**.

Now, After Configure we need to test that our SSH configuration working or not. This is very easy step follow the instructions and commands below.

- ```
[ec2-user@ip-172-31-24-158 ~]$ ssh ec2-user@172.31.19.35

~_#####_ Amazon Linux 2023
~~\#####\
~~ \###|
~~ \#/ https://aws.amazon.com/linux/amazon-linux-2023
~~ V~' '->
~~~~  
~~._.  
_/_/_/  
_/m/' -
```
- Last login: Wed May 24 08:55:44 2023 from 172.31.24.158  
[ec2-user@Kafka-1 ~]\$ █

# Kafka Rolling Restart Brokers

- SSH into Admin/Monitor Machine
- Execute Below Command

- **kafka-rolling-restart --cluster-type kafka --start-command "systemctl start kafka" --stop-command "systemctl stop kafka" --check-count 3**
- Type "Yes" after execute above command and hit Enter/Return key.

```
[ec2-user@ip-172-31-24-158 ~]$ kafka-rolling-restart --cluster-type kafka --start-command "systemctl start kafka" --stop-command "systemctl stop kafka" --check-count 3
Will restart the following brokers in cluster-1:
 1: 172.31.19.35
 2: 172.31.32.42
 3: 172.31.5.194
Do you want to restart these brokers? yes
```

Output of "kafka-rolling-restart" command

```
 3: 172.31.5.194
Do you want to restart these brokers? yes
Execute restart
Under replicated partitions: 0, missing brokers: 0 (1/1)
The cluster is stable
Stopping 172.31.19.35 (1/3)
Starting 172.31.19.35 (1/3)
Broker 172.31.19.35 is down: HTTPConnectionPool(host='172.31.19.35', port=8778): Max retries exceeded with url: /jolokia/read/kafka.server.name=Under
ReplicatedPartitions,type=ReplicaManager/Value (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f27854a0c90>: Failed to
establish a new connection: [Errno 111] Connection refused')).This maybe because it is starting up
Under replicated partitions: 6, missing brokers: 1 (0/3)
Under replicated partitions: 0, missing brokers: 0 (1/3)
Under replicated partitions: 0, missing brokers: 0 (2/3)
Under replicated partitions: 0, missing brokers: 0 (3/3)
The cluster is stable
Stopping 172.31.32.42 (2/3)
Starting 172.31.32.42 (2/3)
Broker 172.31.32.42 is down: HTTPConnectionPool(host='172.31.32.42', port=8778): Max retries exceeded with url: /jolokia/read/kafka.server.name=Under
ReplicatedPartitions,type=ReplicaManager/Value (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f2784a2bd50>: Failed to
establish a new connection: [Errno 111] Connection refused')).This maybe because it is starting up
Under replicated partitions: 6, missing brokers: 1 (0/3)
Under replicated partitions: 0, missing brokers: 0 (1/3)
Under replicated partitions: 0, missing brokers: 0 (2/3)
Under replicated partitions: 0, missing brokers: 0 (3/3)
The cluster is stable
Stopping 172.31.5.194 (3/3)
Starting 172.31.5.194 (3/3)
Broker 172.31.5.194 is down: HTTPConnectionPool(host='172.31.5.194', port=8778): Max retries exceeded with url: /jolokia/read/kafka.server.name=Under
ReplicatedPartitions,type=ReplicaManager/Value (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f2784a2f4d0>: Failed to
establish a new connection: [Errno 111] Connection refused')).This maybe because it is starting up
Under replicated partitions: 6, missing brokers: 1 (0/3)
Under replicated partitions: 0, missing brokers: 0 (1/3)
Under replicated partitions: 0, missing brokers: 0 (2/3)
Under replicated partitions: 0, missing brokers: 0 (3/3)
The cluster is stable
[ec2-user@ip-172-31-24-158 ~]$
```

Output of "kafka-rolling-restart" command Execution

Kafka Rolling Restart command will restart broker, Then wait for cluster will stable then go to another broker and so on. You can see in terminal "The Cluster is stable"

-----End-----