****

****

****

**"DevOps Practice: Effective Data Distribution on Kafka Cluster to attain Low latency and High throughput in Data streaming Use cases."**

**RUN BOOK**

**"DevOps Practice: Effective Data Distribution on Kafka Cluster to attain Low latency and High throughput in Data streaming Use cases."**

**ABSTRACT**

Kafka is a cluster in nature which is used as an enterprise messaging system to decouple source and target systems to exchange data. An instance in a Kafka cluster is called a broker. As the Kafka cluster may have n number of brokers, it’s still bottlenecked to maintain the cluster “Data Balanced” across all the disks in the broker. Whenever a Kafka server is down, the load of that server is distributed to the other nodes in the cluster and this distribution is not even because of Topics and its partition counts, i.e the load is not distributed evenly across all nodes/disks in the cluster(brokers). There are two things that Kafka is providing usually when we talk about rebalancing. One is leader re-election, or preferred replica election and the other one is partition rebalancing. In this session, we have enhanced Effective Kafka Rebalancing methodology and how we can regulate this when the cluster exceeds the unbalanced threshold. Kafka has gained popularity with application developers and data management experts because it greatly simplifies working with data streams. But Kafka can get complex at scale. We have tried to orchestrate Kafka Infrastructure to reducing the complexity.

* **Key focus of the session:**

1. How effectively we can rebalance the cluster.
2. Best practices low latency and high throughput.

* **Tools/services:**

Here we are going to have 3 node Kafka Cluster for Demo purpose. The cluster will be built in AWS Cloud.

1. AWS Cloud - Platform
2. Kafka
3. Ansible (Configuration Management Tool)
4. **AWS Instance Creation**

Please find the attachment for creating AWS instances (CentOS7) and connecting it through SSH client.



1. **Zookeeper Installation**
2. Install some tools like ‘wget’ ‘nc’ which will be required to download packages and checking connectivity.

$yum -y install wget zip net-tools vim nano tar netcat nc telnet

1. Install JAVA as it is mandatory to run Kafka.

$yum -y install java-1.8.0-openjdk

1. Make the vm.swappiness to 0 or 1 as it is recommended for good performance. By default, it will be 30.

$echo 'vm.swappiness=1' | tee --append /etc/sysctl.conf

$cat /etc/sysctl.conf

1. Download Zookeeper and Kafka. Recommended is latest Kafka.

$wget <http://apachemirror.wuchna.com/kafka/2.5.0/kafka_2.12-2.5.0.tgz>

$tar -xvzf kafka\_2.12-2.5.0.tgz

$rm kafka\_2.12-2.5.0.tgz

$mv kafka\_2.12-2.5.0/ kafka

1. Start the Zookeeper

$cd kafka/

$cat config/zookeeper.properties --> We will start the zookeeper with this default property.

$bin/zookeeper-server-start.sh config/zookeeper.properties

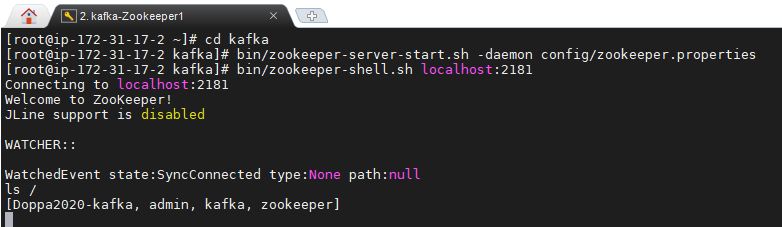
# Start Zookeeper in the background

$bin/zookeeper-server-start.sh -daemon config/zookeeper.properties

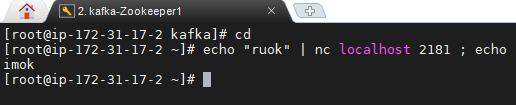
$bin/zookeeper-shell.sh localhost:2181

Try list out the znodes

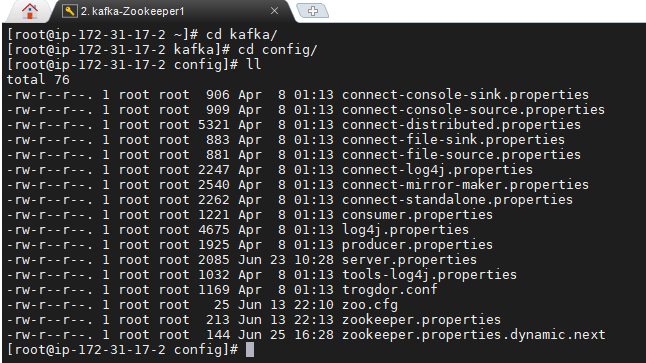
ls /



$ echo "ruok" | nc localhost 2181 ; echo



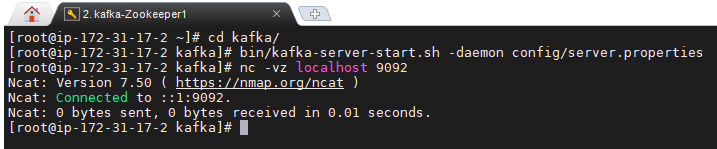
1. **Kafka Installation**
2. Kafka Properties will be in /kafka/config/ -->server.properties



1. Start the kafka with default property.

$cd kafka/

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



1. **Kafka Multi-Node cluster setup.**

We have installed Zookeeper and Kafka with its default properties in a single AWS instances. We created a image of this Instance and launched 2 more instances with using this Image, so that we don’t need to install all the packages instead we can configure the properties required for multi-node cluster.

1. Configure the /etc/hosts on all the 3 servers

$vi /etc/hosts

*172.31.17.2 zookeeper1*

*172.31.17.2 kafka1*

*172.31.1.2 zookeeper2*

*172.31.1.2 kafka2*

*172.31.18.2 zookeeper3*

*172.31.18.2 kafka3*

1. Configure the zookeeper.properties on all the 3 servers for setting Zookeeper Quorum.

$vi kafka/config/zookeeper.properties

Remove the default and paste the below configuration

*dataDir=/data/zookeeper*

*clientPort=2181*

*maxClientCnxns=0*

*tickTime=2000*

*initLimit=10*

*syncLimit=5*

*server.1=zookeeper1:2888:3888*

*server.2=zookeeper2:2888:3888*

*server.3=zookeeper3:2888:3888*

*4lw.commands. whitelist=\* #this is for allowing ZK four letter commands*

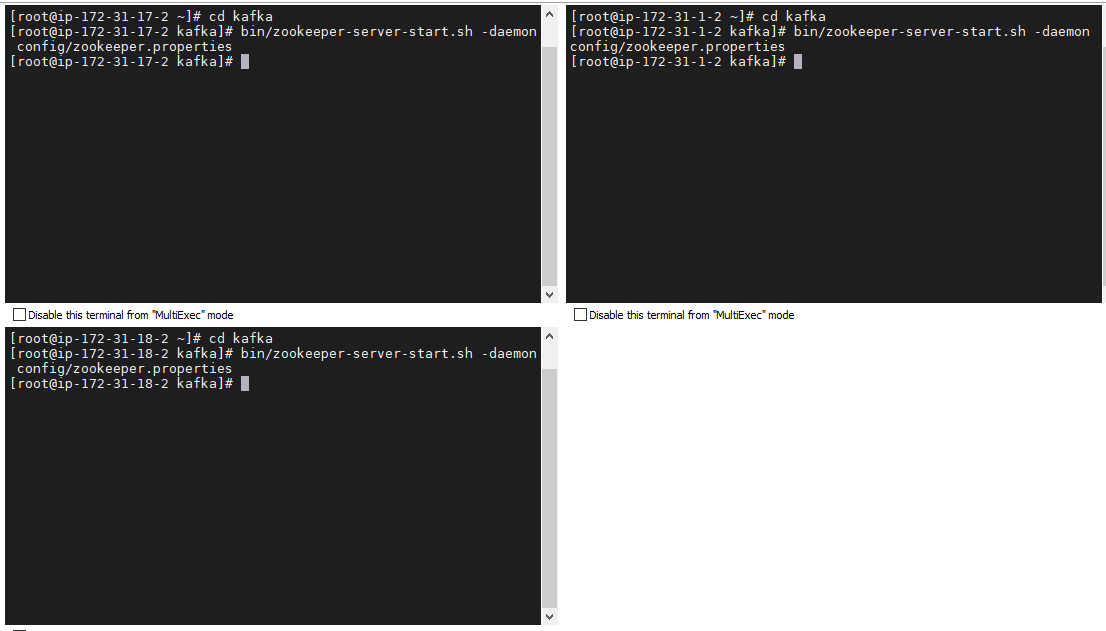
1. Configure the server.properties on all the 3 servers

$vi kafka/config/server.properties

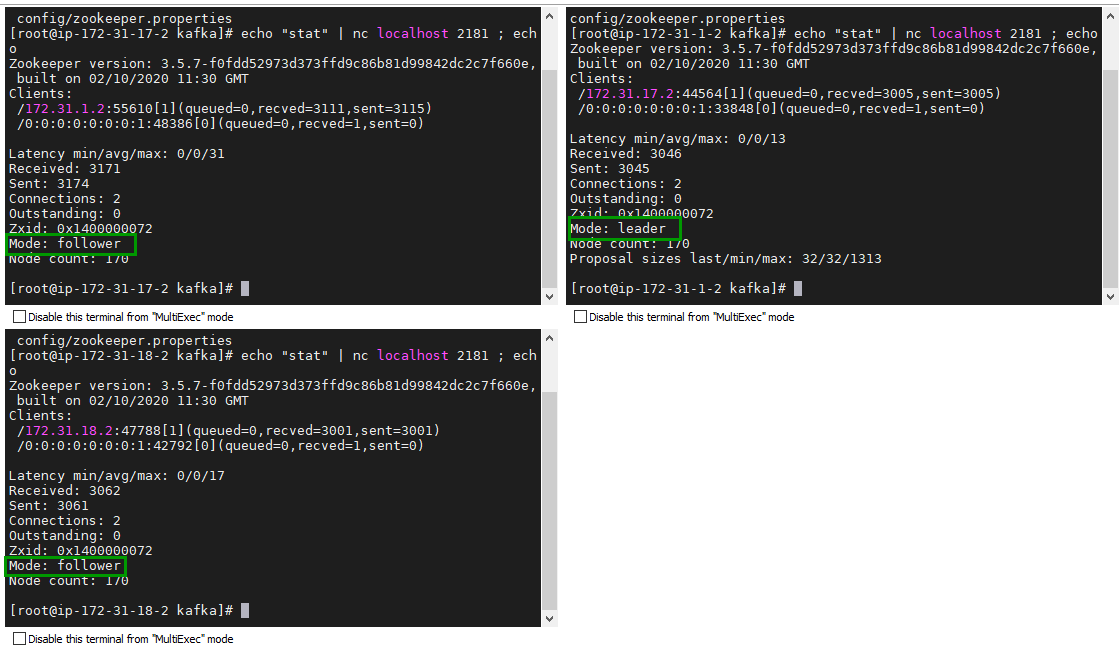
Remove the default broker ID and configure 1, 2 and 3 for 3 servers respectively

1. Start the Zookeeper in all the nodes

$ bin/zookeeper-server-start.sh -daemon config/zookeeper.properties

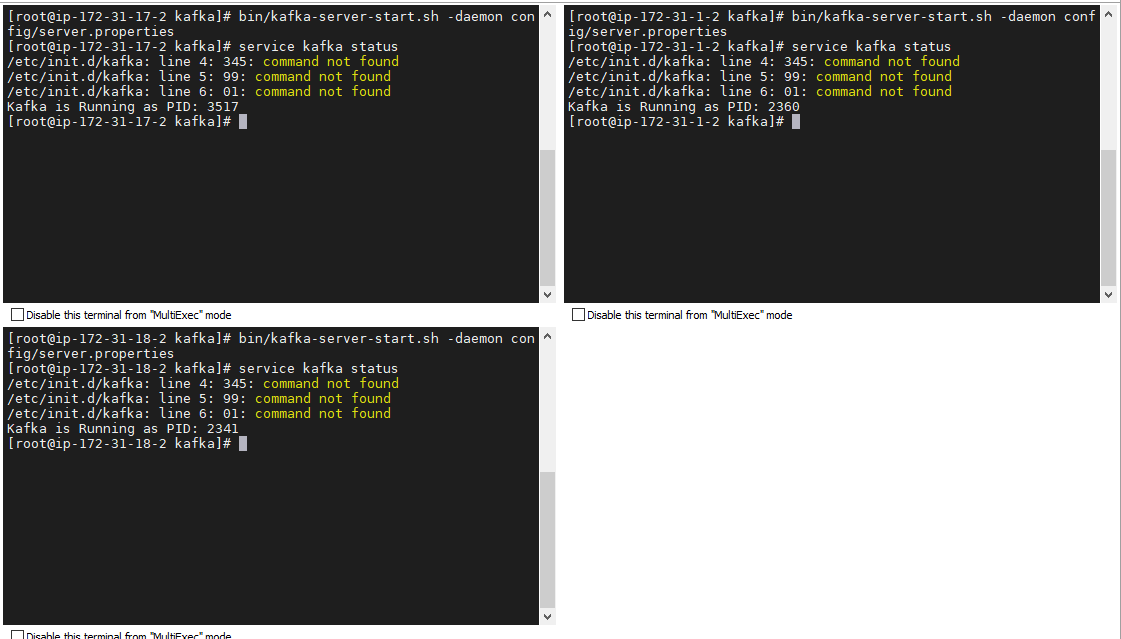


$ echo "stat" | nc localhost 2181 ; echo --> This command will check the status of all the zookeeper



1. Start the Kafka in all the servers

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



We have configured Kafka as a service in the OS, so we can use “$service kafka start” command to start the Kafka.

1. **Kafka – Partition Reassignment.**

Please find the attached document which describes the step by step process of Effective Partition Reassignment.



1. **Reference**

* <https://docs.cloudera.com/runtime/7.0.3/kafka-managing/topics/kafka-manage-cli-reassign-overview.html>
* <https://www.datadoghq.com/blog/engineering/introducing-kafka-kit-tools-for-scaling-kafka/>
* <https://www.confluent.io/blog/how-choose-number-topics-partitions-kafka-cluster/>
* <https://docs.cloudera.com/runtime/7.1.0/kafka-performance-tuning/topics/kafka-tune-sizing-partition-number.html>
* <https://blog.newrelic.com/engineering/kafka-best-practices/>