**Kafka Setup**

Create Kafka user

I recommend to create a separate kafka user for operating the Kafka node.

[user@kafka1 ~]$ sudo adduser kafka

[user@kafka1 ~]$ sudo passwd kafka

#######

After creating the user we can switch to the new user and can perform the remaining configuration.

[user@kafka1 ~]$ su kafka

Install ZooKeeper and Kafka

Now we can download Apache Kafka

[kafka@kafka1 ~]$ wget http://apache.forsale.plus/kafka/1.0.0/kafka\_2.11-1.0.0.tgz

Let's decompress the tar.gz file

[kafka@kafka1 ~]$ tar -xzf kafka\_2.11-1.0.0.tgz

That's it! As the Kafka archive already includes ZooKeeper we already have a fully working Kafka installation on our system. To operate a cluster we have to adjust the configuration.

######

Create directories

We have to create data and log directories for ZooKeeper and Kafka. To simplify this process we can add the directories within the user home directory. In a production environment, we would use different locations, e.g. separate mount points or physical disks for data and log directories.

[kafka@kafka1 ~]$ mkdir -p /home/kafka/zookeeper/data

[kafka@kafka1 ~]$ mkdir -p /home/kafka/kafka/kafka-logs

#####

ZooKeeper configuration

The configuration file of the embedded ZooKeeper instance is located at kafka\_2.11-1.0.0/config/zookeeper.properties.

[kafka@kafka1 ~]$ vi kafka\_2.11-1.0.0/config/zookeeper.properties

Within this file, we have to locate the dataDir property and set the value to point to the new ZooKeeper directory we created above.

dataDir=/home/kafka/zookeeper/data

At the end of this file, we have to add all available ZooKeeper servers. We also add the initLimit and syncLimit properties. Additional information about these properties can be found here.

server.1=kafka1:2888:3888

initLimit=5

syncLimit=2

Each of our cluster nodes needs a unique server id. ZooKeeper looks up this information from the following file: /home/kafka/zookeeper/data/myid. We have to execute a command like this on each server - using a different value for each instance. For instance 1 on server kafka1 we use the value "1".

[kafka@kafka1 ~]$ echo "1" > /home/kafka/zookeeper/data/myid

**#########**

Apache Kafka configuration

Now we can adjust the Kafka configuration files stored here: kafka\_2.11-1.0.0/config/server.properties.

[kafka@kafka1 ~]$ vi kafka\_2.11-1.0.0/config/server.properties

Similar to the ZooKeeper configuration, each Kafka cluster node needs a unique id. We have to find the broker.id property in the configuration file and change the id for each server. I recommend to use 1, 2 and 3.

broker.id=1

We also have to change the log directory location specified in the log.dirs parameter.

log.dirs=/home/kafka/kafka/kafka-logs

Additionally, we have to update the listeners and advertised.listeners properties with the current Kafka node hostname.

• listeners: the address/server name and protocol kafka is listening to (internal traffic between Kafka nodes)

• advertised.listener: the address/server name and protocol clients can use to connect to the Kafka cluster (external traffic). Only need to be specified if different from above setting.

listeners=PLAINTEXT://kafka1:9092

advertised.listeners=PLAINTEXT://kafka1:9092

The next step is to tell Kafka which ZooKeeper nodes can be used to connect to. Especially when operating a big cluster with hundreds of nodes, not all available server nodes have to be added here. It is sufficient to add a couple of seed nodes. Kafka will identify all available nodes and updates the available nodes if new nodes join or leave the cluster.

zookeeper.connect=kafka1:2181

In a development environment, I usually add the property delete.topic.enable. Setting this property to true allows us to easily delete topics at runtime. If this property is not being set, Kafka will only mark topics as deleted.

delete.topic.enable=true

That's it, we have configured our node as Kafka node!

Start and test the cluster setup

We finally can startup ZooKeeper and Kafka and perform a quick test.

#########

Start ZooKeeper

To start ZooKeeper execute the following command on each node:

[kafka@kafka1 ~]$ cd kafka\_2.11-1.0.0

[kafka@kafka1 ~]$ nohup bin/zookeeper-server-start.sh config/zookeeper.properties &

############

Start Apache Kafka

Now we can start Kafka:

[kafka@kafka1 ~]$ nohup bin/kafka-server-start.sh config/server.properties &

Create a new topic

To test the setup we have to create a topic.

[kafka@kafka1 ~]$ bin/kafka-topics.sh --create --zookeeper kafka1:2181 --replication-factor 1 --partitions 6 --topic topic1 --config cleanup.policy=delete --config delete.retention.ms=60000

The command above creates a new topic named topic1 with 6 partitions. The replication-factor has been set to 1, which means data is not being replicated and the data for a particular partition will only be stored on one server.

We can also get a list of all existing topics

[kafka@kafka1 ~]$ bin/kafka-topics.sh --list --zookeeper kafka1:2181

And we can get a detailed description of our topic.

[kafka@kafka1 ~]$ bin/kafka-topics.sh --describe --zookeeper kafka1:2181 --topic topic1

In my case the command above prints out:

Topic:topic1 PartitionCount:6 ReplicationFactor:1 Configs:delete.retention.ms=60000,cleanup.policy=delete

Topic: topic1 Partition: 0 Leader: 2 Replicas: 2 Isr: 2

Topic: topic1 Partition: 1 Leader: 3 Replicas: 3 Isr: 3

Topic: topic1 Partition: 2 Leader: 1 Replicas: 1 Isr: 1

Topic: topic1 Partition: 3 Leader: 2 Replicas: 2 Isr: 2

Topic: topic1 Partition: 4 Leader: 3 Replicas: 3 Isr: 3

Topic: topic1 Partition: 5 Leader: 1 Replicas: 1 Isr: 1

The command shows which server is responsible for which partition and which server replicates the data.

**Test the cluster**

The Kafka package already includes two command line tools to create a producer and a consumer that can be used to check if the cluster works.

We can start the producer on one of our servers. The command opens a prompt and anything we enter here will be sent to the topic.

[kafka@kafka1 ~]$ bin/kafka-console-producer.sh --broker-list kafka1:9092 --topic topic1

Now we can start a consumer on one of our servers.

[kafka@kafka1 ~]$ bin/kafka-console-consumer.sh --bootstrap-server kafka1:9092 --topic topic1

Whenever we enter something in the producer prompt it will be printed out in our consumer terminal.

This means our test was successful, the Kafka cluster has been set up. In the next couple of weeks, I will add more Kafka examples and use cases, including