Confluent Kafka Installation

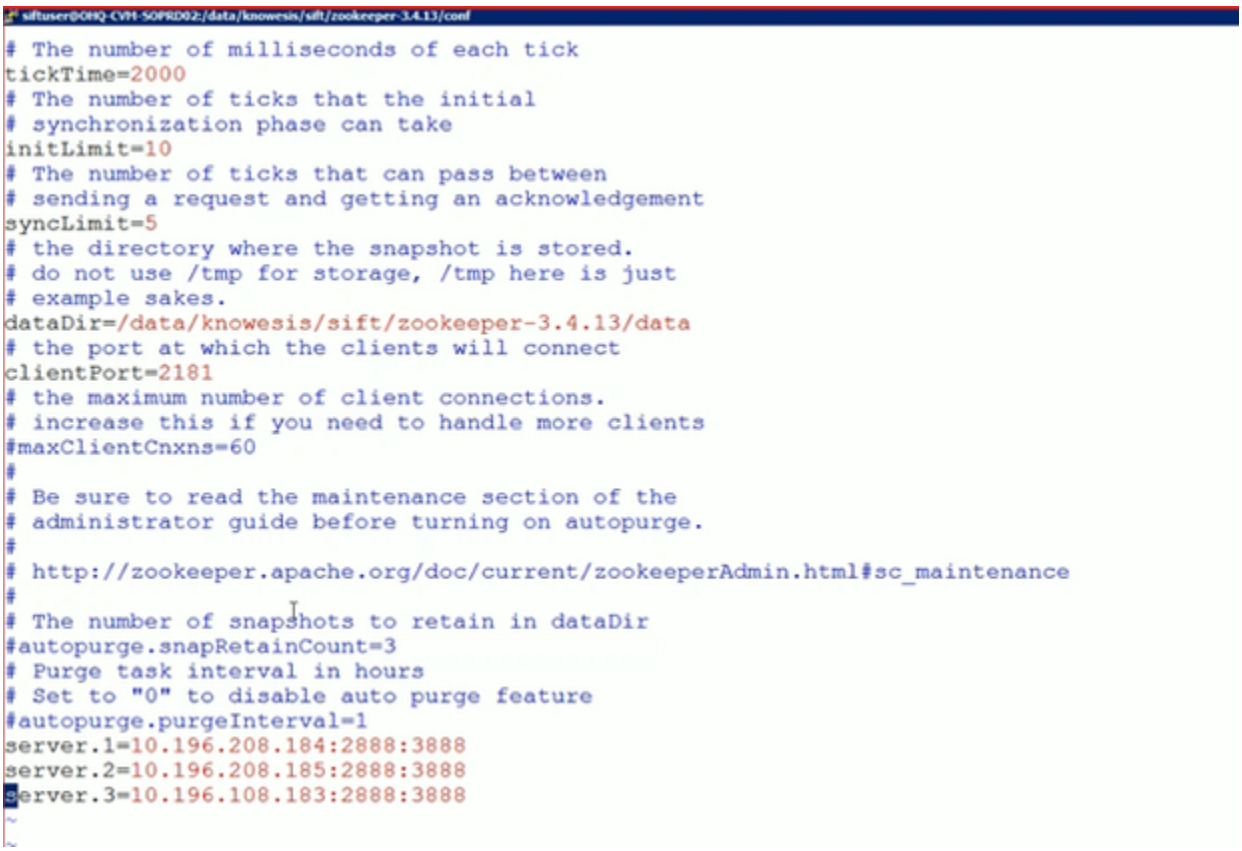
Confluent Kafka Installation

First, we need to install zookeeper in each server.

1) Change the configuration of the zookeeper.

Location :-

```
$vim zookeeper/conf/zoo.cfg
```



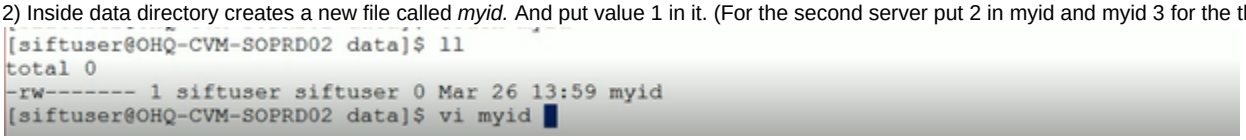
1.a) change datadir= <location to store data>

1.b) give the servers list to be cluster
Eg: server.1=<ip>:2888:3888

server.2=<ip>:2888:3888

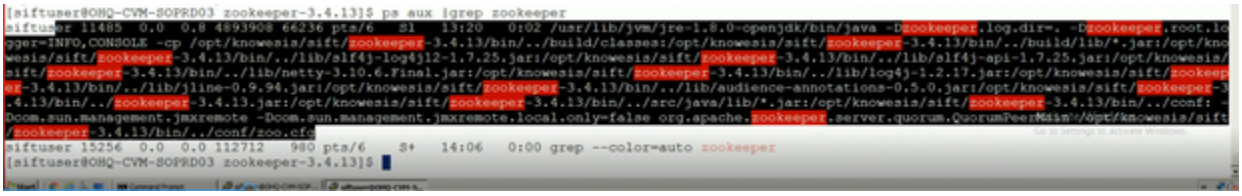
server.3=<ip>:2888:3888

2) Inside data directory creates a new file called *myid.* And put value 1 in it. (For the second server put 2 in myid and myid 3 for the third server)



3) Start zookeeper in each server.

```
[siftuser@OHQ-CVM-SOPRD03 bin]$ ./zkServer.sh start ../conf/zoo.cfg
ZooKeeper JMX enabled by default
Using config: ../conf/zoo.cfg
Starting zookeeper ... STARTED
[siftuser@OHQ-CVM-SOPRD03 bin]$
```



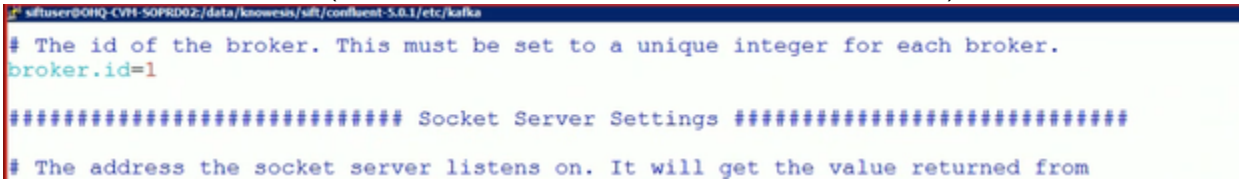
Now we want to start Kafka, before that, change the configuration of Kafka

Location : confluent/etc/kafka/server.properties

Note: Before changing the configuration take a backup of the configuration file.

Changes in server.properties

Put broker.id=1 for the first server. (broker.id=2 for the second server, broker.id=3 for the third server)



```
# The id of the broker. This must be set to a unique integer for each broker.
broker.id=1

############################# Socket Server Settings #############################

# The address the socket server listens on. It will get the value returned from
```

b) Change the listeners=PLAINTEXT://<serverip>:8021(For the first server)

```
############################# Socket Server Settings #############################

# The address the socket server listens on. It will get the value returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://10.196.208.184:8021

# Hostname and port the broker will advertise to producers and consumers. If not set,
# it uses the value for "listeners" if configured.  Otherwise, it will use the value
# returned from java.net.InetAddress.getCanonicalHostName().
#advertised.listeners=PLAINTEXT://10.196.208.184:8021
```

c) change the log.dirs=<location to store data of kafka>

```
############################# Log Basics #############################

# A comma separated list of directories under which to store log files
log.dirs=/data/knowesis/sift/kafka_data/node1
```

Note: node1 for the first server, for the second server node2, for the third server node3.

d) Change num.partitions=2

```
# The default number of log partitions per topic. More partitions allow greater
# parallelism for consumption, but this will also result in more files across
# the brokers.
num.partitions=1
```

e) Change zookeeper.connect=<server1>:2181, <server2>:2181,<server3>:2181

```
############################# Zookeeper #############################

# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify the
# root directory for all kafka znodes.
zookeeper.connect=10.196.208.184:2181,10.196.208.185:2181
```

e) In the Config file at the end put a new parameter
default.replication.factor=2

This is to create topics with replication factor 2.

f) Start kafka

```
[siftuser@OHQ-CVM-SOPRD02 bin]$
[siftuser@OHQ-CVM-SOPRD02 bin]$ ./kafka-server-start ../etc/kafka/server.properties &
```

For testing Kafka is working:

Need to create a topic

```
# ./kafka-topics.sh --create --topic test1 --zookeeper 127.0.0.1:2181 --
replication-factor 2 --partitions 1
```

(inside kafka bin dir)

```
ubuntu@ip-172-31-17-225:~/kafka/config$ kafka-topics.sh --create --topic test1 --zookeeper 127.0.0.1
:2181 --replication-factor2 --partitions1
Exception in thread "main" joptsimple.UnrecognizedOptionException: replication-factor2 is not a reco
gnized option
        at joptsimple.OptionException.unrecognizedOption(OptionException.java:108)
        at joptsimple.OptionParser.handleLongOptionToken(OptionParser.java:510)
        at joptsimple.OptionParserState$2.handleArgument(OptionParserState.java:56)
        at joptsimple.OptionParser.parse(OptionParser.java:396)
        at kafka.admin.TopicCommand$TopicCommandOptions.<init>(TopicCommand.scala:669)
        at kafka.admin.TopicCommand$.main(TopicCommand.scala:51)
        at kafka.admin.TopicCommand.main(TopicCommand.scala)
```

Here we can type messages for testing

```
# kafka-console-producer.sh --broker-list 127.0.0.1:8021 --topic test1
```

```
ubuntu@ip-172-31-17-225:~/kafka/config$ kafka-console-producer.sh --broker-list 127.0.0.1:8021 --top
ic test1
>[2021-05-14 05:49:48,298] WARN [Producer clientId=console-producer] Error while fetching metadata w
ith correlation id 1 : {test1=LEADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
>

>Hello
>^Cubuntu@ip-172-31-17-225:~/kafka/config$ kafka-console-producer.sh --broker-list 127.0.0.1:8021 --topic test1
>hi
>kafka
```

If everything working fine, then the messages we can see when applying below command:

```
 # kafka-console-consumer.sh --bootstrap-server 127.0.0.1:8021 --topic
test1 --from-beginning
```

```
ubuntu@ip-172-31-17-225:~/kafka/config$ kafka-console-consumer.sh --bootstrap-server 127.0.0.1:8021 --topic test1 --from-beginning


hi
kafka
Hello
hi
kafka
^CProcessed a total of 7 messages
```

Step 2:

Now the zookeeper and kafka broker are configured. Then we need to set up SSL cert on both zookeeper clients and kafka brokers.

SETTING UP SSL :

We need to generate SSL cer for Kafka servers and clients client.properties (This file we need to create manually.) by using below command:

create CA => result: file ca-cert and the priv.key ca-key

```
$ mkdir ssl
$ cd ssl
$ openssl req -new -newkey rsa:4096 -days 365 -x509 -subj "/CN=Kafka-
Security-CA" -keyout ca-key -out ca-cert -nodes
$cat ca-cert
$cat ca-key
$keytool -printcert -v -file ca-cert
```

Add-On: public certificates check:

```
$ echo |
 openssl s_client -connect www.google.com:443 2>/dev/null |
 openssl x509 -noout -text -certopt no_header,no_version,no_serial,
no_signame,no_pubkey,no_sigdump,no_aux -subject -nameopt multiline -
issuer
```

Next we need to create Kafka server certificate:

Note: create a server certificate !! put your public EC2-DNS here, after "CN="

```
$ export SRVPASS=serversecret
$ cd ssl
$ keytool -genkey -keystore kafka.server.keystore.jks -validity 365 -
storepass $SRVPASS -keypass $SRVPASS  -dname "CN=<<Public EC2-DNS here
>>" -storetype pkcs12
$ ll
$ keytool -list -v -keystore kafka.server.keystore.jks
```

create a certification request file, to be signed by the CA:

```
$ keytool -keystore kafka.server.keystore.jks -certreq -file cert-file -
storepass $SRVPASS -keypass $SRVPASS
$ ll
```

sign the server certificate => output: file "cert-signed"

```
$ openssl x509 -req -CA ca-cert -CAkey ca-key -in cert-file -out cert-
signed -days 365 -CAcreateserial -passin pass:$SRVPASS
$ ll
```

Trust the CA by creating a truststore and importing the ca-cert

```
$ keytool -keystore kafka.server.truststore.jks -alias CARoot -import -
file ca-cert -storepass $SRVPASS -keypass $SRVPASS -noprompt
```

Import CA and the signed server certificate into the keystore

```
$ keytool -keystore kafka.server.keystore.jks -alias CARoot -import -
file ca-cert -storepass $SRVPASS -keypass $SRVPASS -noprompt
$ keytool -keystore kafka.server.keystore.jks -import -file cert-signed
-storepass $SRVPASS -keypass $SRVPASS -noprompt
```

After creating certificates, we need edit "**server.properties**" (Note: *Ensure that you set your public-DNS* !!) file as below:

```
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version
2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# see kafka.server.KafkaConfig for additional details and defaults

############################## Server Basics
##############################

# The id of the broker. This must be set to a unique integer for each
broker.
broker.id=0

############################## Socket Server Settings
##############################

# The address the socket server listens on. It will get the value
returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
```

```
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://0.0.0.0:9092,SSL://0.0.0.0:9093
advertised.listeners=PLAINTEXT://##your-public-DNS##:9092,SSL://##your-
public-DNS##:9093
zookeeper.connect=##your-public-DNS##:2181

ssl.keystore.location=/home/ubuntu/ssl/kafka.server.keystore.jks
ssl.keystore.password=serversecret
ssl.key.password=serversecret
ssl.truststore.location=/home/ubuntu/ssl/kafka.server.truststore.jks
ssl.truststore.password=serversecret


# Maps listener names to security protocols, the default is for them to
be the same. See the config documentation for more details
#listener.security.protocol.map=PLAINTEXT:PLAINTEXT,SSL:SSL,
SASL_PLAINTEXT:SASL_PLAINTEXT,SASL_SSL:SASL_SSL

# The number of threads that the server uses for receiving requests
from the network and sending responses to the network
num.network.threads=3

# The number of threads that the server uses for processing requests,
which may include disk I/O
num.io.threads=8

# The send buffer (SO_SNDBUF) used by the socket server
socket.send.buffer.bytes=102400

# The receive buffer (SO_RCVBUF) used by the socket server
socket.receive.buffer.bytes=102400

# The maximum size of a request that the socket server will accept
(protection against OOM)
socket.request.max.bytes=104857600


auto.create.topics.enable=false

############################# Log Basics #############################

# A comma seperated list of directories under which to store log files
log.dirs=/home/ubuntu/kafka-logs

# The default number of log partitions per topic. More partitions allow
greater
# parallelism for consumption, but this will also result in more files
```

```
across
# the brokers.
num.partitions=1

# The number of threads per data directory to be used for log recovery
at startup and flushing at shutdown.
# This value is recommended to be increased for installations with data
dirs located in RAID array.
num.recovery.threads.per.data.dir=1

############################## Internal Topic Settings
#############################
# The replication factor for the group metadata internal topics
"__consumer_offsets" and "__transaction_state"
# For anything other than development testing, a value greater than 1
is recommended for to ensure availability such as 3.
offsets.topic.replication.factor=1
transaction.state.log.replication.factor=1
transaction.state.log.min.isr=1

############################## Log Flush Policy
#############################

# Messages are immediately written to the filesystem but by default we
only fsync() to sync
# the OS cache lazily. The following configurations control the flush
of data to disk.
# There are a few important trade-offs here:
#    1. Durability: Unflushed data may be lost if you are not using
replication.
#    2. Latency: Very large flush intervals may lead to latency spikes
when the flush does occur as there will be a lot of data to flush.
#    3. Throughput: The flush is generally the most expensive
operation, and a small flush interval may lead to exceessive seeks.
# The settings below allow one to configure the flush policy to flush
data after a period of time or
# every N messages (or both). This can be done globally and overridden
on a per-topic basis.

# The number of messages to accept before forcing a flush of data to
disk
#log.flush.interval.messages=10000

# The maximum amount of time a message can sit in a log before we force
a flush
#log.flush.interval.ms=1000

############################## Log Retention Policy
#############################
```

```
# The following configurations control the disposal of log segments.
The policy can
# be set to delete segments after a period of time, or after a given
size has accumulated.
# A segment will be deleted whenever *either* of these criteria are
met. Deletion always happens
# from the end of the log.

# The minimum age of a log file to be eligible for deletion due to age
log.retention.hours=168

# A size-based retention policy for logs. Segments are pruned from the
log unless the remaining
# segments drop below log.retention.bytes. Functions independently of
log.retention.hours.
#log.retention.bytes=1073741824

# The maximum size of a log segment file. When this size is reached a
new log segment will be created.
log.segment.bytes=1073741824

# The interval at which log segments are checked to see if they can be
deleted according
# to the retention policies
log.retention.check.interval.ms=300000

############################## Zookeeper ##############################

# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify
the
# root directory for all kafka znodes.

# Timeout in ms for connecting to zookeeper
zookeeper.connection.timeout.ms=6000


############################## Group Coordinator Settings
##############################

# The following configuration specifies the time, in milliseconds, that
the GroupCoordinator will delay the initial consumer rebalance.
# The rebalance will be further delayed by the value of group.initial.
rebalance.delay.ms as new members join the group, up to a maximum of
max.poll.interval.ms.
# The default value for this is 3 seconds.
# We override this to 0 here as it makes for a better out-of-the-box
experience for development and testing.
```

```
# However, in production environments the default value of 3 seconds is
more suitable as this will help to avoid unnecessary, and potentially
expensive, rebalances during application startup.
```

Client configuration for using SSL

1. grab CA certificate from remote server and add it to local CLIENT truststore:

```
$ export CLIPASS=clientpass
$ cd ~
$ mkdir ssl
$ cd ssl
$ scp -i ~/kafka-security.pem ubuntu@<<your-public-DNS>>:/home
/ubuntu/ssl/ca-cert .
$ keytool -keystore kafka.client.truststore.jks -alias CARoot -
import -file ca-cert  -storepass $CLIPASS -keypass $CLIPASS -
noprompt
$ keytool -list -v -keystore kafka.client.truststore.jks
```

2. create client.properties and configure SSL parameters:

```
security.protocol=SSL
ssl.truststore.location=/home/gerd/ssl/kafka.client.truststore.jks
ssl.truststore.password=clientpass
```

TEST

test using the console-consumer/-producer and the client.properties

Producer

```
~/kafka/bin/kafka-console-producer.sh --broker-list <<your-public-DNS>>:
9093 --topic <<topic name>> --producer.config ~/ssl/client.properties
```

```
ubuntu@ip-172-31-43-59:~/ssl/ssl-clients$ ~/kafka/bin/kafka-console-producer.sh --broker-list ec2-18-133-234-236.eu-west-2.compute.amazonaws.com:9093
 --topic test2  --producer.config ~/ssl/ssl-clients/client.properties
>Hi All
>How are you?
>All good here!
>
```

Consumer

```
~/kafka/bin/kafka-console-consumer.sh --bootstrap-server <<your-public-
DNS>>:9093 --topic <<topic name>> --consumer.config ~/ssl/client.
properties
```

```
ubuntu@ip-172-31-43-59:~$ ~/kafka/bin/kafka-console-consumer.sh --bootstrap-server ec2-18-133-234-23
6.eu-west-2.compute.amazonaws.com:9093 --topic test2 --consumer.config ~/ssl/ssl-clients/client.prop
erties


Hi All
How are you?
All good here!
```