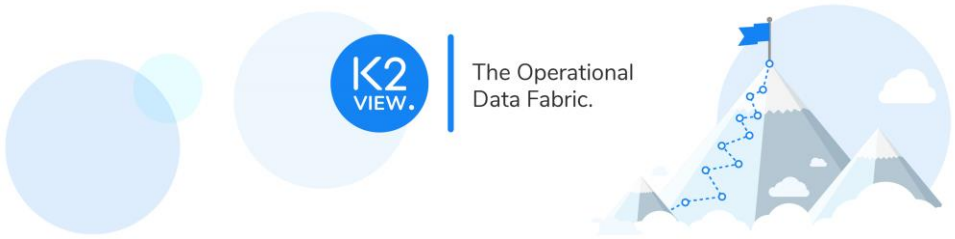
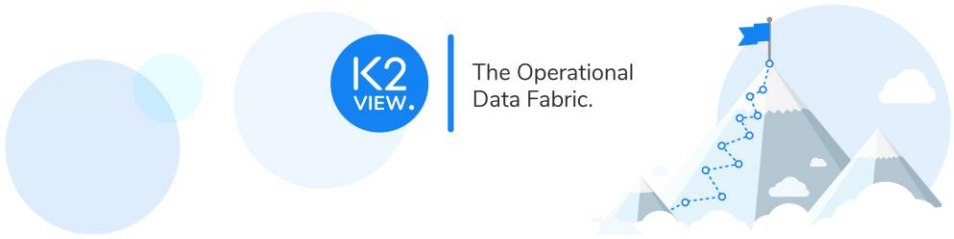


K2VIEW

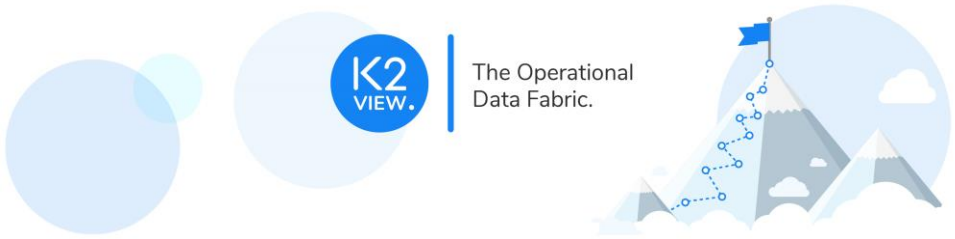
Fabric Installation

Fabric 6.xx

July, 2021



CONTENTS



1 Introduction

Follow the instructions in this document to install Fabric in a production environment.

2 Prerequisites

Minimum hardware for each Linux execution server is as follows:

- Modern Xeon Processor
- 16 Physical Cores
- RAM:
  - For servers that run all components (Fabric, Cassandra & Kafka) - 64GB RAM
  - For servers that run single component (preferred method) - 32GB RAM
- Network: Minimum 1G per sec between the nodes and source databases
- Storage: The preferred storage is attached local SSD's in a non-RAID configuration

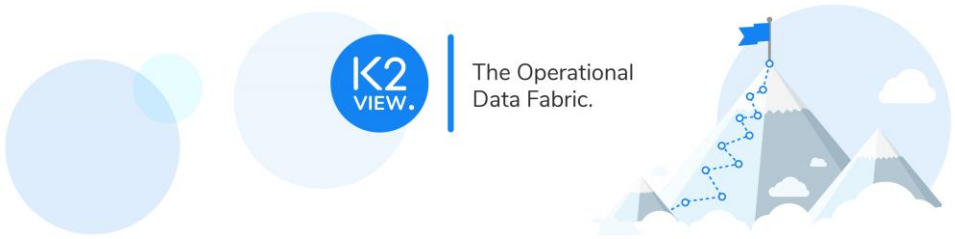
  Note: If you must use SAN, it must be flash and in RAID-0.
  NAS are not certified.

Operating System: Redhat/CentOS latest Operating System and above, with latest patches.

The following File Server volumes must be made available:

- Volume of 50G /opt/apps/fabric/ will be use also as the home directory for k2view user
- Volume of 100G* /opt/apps/fabric/storage
- Volume of 50G /opt/apps/cassandra/
- Volume of 2T* /opt/apps/cassandra/storage/data
- Volume of 10% of data volume - /opt/apps/cassandra/storage/hints
- Volume of 25% of data volume - /opt/apps/cassandra/storage/commitlog
- Volume of 100G* /opt/apps/kafka/

  Note: The file server must provide IOPS of at least 30K read & 10K write.
  The number of servers should be increased based on project scope and data retention requirements.



Add the following users:

```
mkdir -p /opt/apps
chmod 755 /opt/apps
useradd -m -d /opt/apps/fabric fabric
useradd -m -d /opt/apps/cassandra cassandra
useradd -m -d /opt/apps/kafka kafka
```

Update the OS limits as follows:

```
echo "root soft    nproc     unlimited" >> /etc/security/limits.conf
echo "cassandra - nofile 100000" >> /etc/security/limits.conf
echo "cassandra - nproc 50000" >> /etc/security/limits.conf
echo "fabric - nofile 100000" >> /etc/security/limits.conf
echo "fabric - nproc 50000" >> /etc/security/limits.conf
echo "kafka hard nofile 100000" >> /etc/security/limits.conf
echo "kafka soft nofile 100000" >> /etc/security/limits.conf
echo "kafka - nproc 50000" >> /etc/security/limits.conf
## update /etc/sysctl.conf ##
echo "## Added by K2view - GabiO" >> /etc/sysctl.conf
echo "vm.max_map_count = 1048575" >> /etc/sysctl.conf
echo "fs.file-max = 500000" >> /etc/sysctl.conf
```

3 Setup for Cassandra

3.1 Load The Package

1. Retrieve the latest Cassandra package (located at: latest version).
2. Connect to the Linux execution server as “cassandra” user and copy the package to the home directory.
3. Untar the package (the package name varies based on the version) as follows:

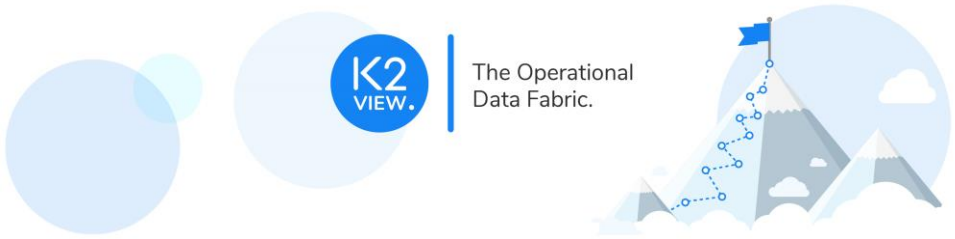
```
tar -zxvf k2v_cassandra-3.11.xxx.tar.gz
```

4. Updated the `.bash_propile` to use python 2.7

```
sed -i '11i\alias python='/usr/bin/python2.7'\' ~/.bash_profile
```

5. If you are doing a standalone configuration, run the `init.sh` and start Cassandra as follows:

```
./init.sh && cassandra
```



3.2 Setup: Single Data Center, Multiple Nodes

Run the commands as shown below for each node in turn. When doing so, update the parameters that are marked in yellow.

1. Run the pre setup commands,
2. Start Cassandra, then
3. Run the post setup commands.

Pre setup run on one node (called DC1 in code):

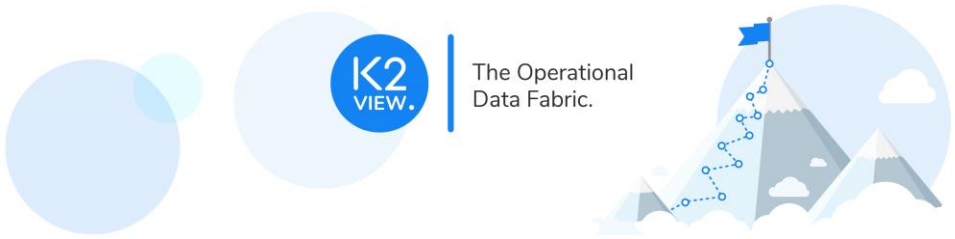
```
sed -i 's@dc=.*@dc=DC1@'  $INSLATT_DIR/cassandra/conf/cassandra-rackdc.properties
sed -i 's@cluster_name: .*@cluster_name: 'PreProd1'@'  $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i s/seeds:.*/"seeds: \"192.168.168.212\""/g $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i s/listen_address:.*/"listen_address: $(hostname -I |awk {'print $1'})"/g
     $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i s/broadcast_rpc_address:.*/"broadcast_rpc_address: $(hostname -I |awk {'print $1'})"/g
     $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i 's@endpoint_snitch:.*@endpoint_snitch: GossipingPropertyFileSnitch@'
     $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i 's@LOCAL_JMX=.*@LOCAL_JMX='no'@'  $INSLATT_DIR/cassandra/conf/cassandra-env.sh
sed -i "s@-Djava.rmi.server.hostname=.*@-Djava.rmi.server.hostname=$(hostname -I |awk {'print
     $1'})\"@"  $INSLATT_DIR/cassandra/conf/cassandra-env.sh
sed -i "s@-Dcom.sun.management.jmxremote.password.file=.*@-
     Dcom.sun.management.jmxremote.password.file=$INSLATT_DIR/cassandra/conf/.jmxremote.password\"@"
     $INSLATT_DIR/cassandra/conf/cassandra-env.sh
sed -i "s@num_tokens:.*@num_tokens: 16@" $INSLATT_DIR/cassandra/conf/cassandra.yaml
```

Start Cassandra:

```
cassandra
```

Post setup run on one node

```
echo "create user k2admin with password 'Q1w2e3r4t5' superuser;" |cqlsh -u cassandra -p cassandra
echo "ALTER KEYSPACE system_auth WITH REPLICATION = {'class' : 'NetworkTopologyStrategy', 'DC1':
     '3'};" | cqlsh -ucassandra -pcassandra
echo "ALTER KEYSPACE system_distributed WITH REPLICATION = {'class' : 'NetworkTopologyStrategy',
     'DC1': '3'};" | cqlsh -ucassandra -pcassandra
echo "ALTER KEYSPACE system_traces WITH REPLICATION = {'class' : 'NetworkTopologyStrategy', 'DC1':
     '3'};" | cqlsh -ucassandra -pcassandra
echo "CREATE KEYSPACE keyspace_with_replication_factor_3 WITH replication = {'class':
     'NetworkTopologyStrategy', 'DC1': 3} AND durable_writes = true;"|cqlsh -u cassandra -p cassandra

# run repair on all the nodes
nodetool -u k2view -pw Q1w2e3r4t5 repair
```



Note: The 4th node and above should be added only after the first 3 nodes are up and have been configured. The commands below must be run on the 4th node and above:

```
sed -i 's@dc=.*@dc=DC1@'  $INSLATT_DIR/cassandra/conf/cassandra-rackdc.properties
sed -i 's@cluster_name: .*@cluster_name: 'PreProd1'@'  $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i s/seeds:.*/"seeds: \"192.168.168.212\""/g $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i s/listen_address:.*/"listen_address: $(hostname -I |awk {'print $1'})"/g
     $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i s/broadcast_rpc_address:.*/"broadcast_rpc_address: $(hostname -I |awk {'print $1'})"/g
     $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i 's@endpoint_snitch:.*@endpoint_snitch: GossipingPropertyFileSnitch@'
     $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i 's@LOCAL_JMX=.*@LOCAL_JMX='no'@'  $INSLATT_DIR/cassandra/conf/cassandra-env.sh
sed -i "s@-Djava.rmi.server.hostname=.*@-Djava.rmi.server.hostname=$(hostname -I |awk {'print
     $1'})\"@"  $INSLATT_DIR/cassandra/conf/cassandra-env.sh
sed -i "s@-Dcom.sun.management.jmxremote.password.file=.*@-
     Dcom.sun.management.jmxremote.password.file=$INSLATT_DIR/cassandra/conf/.jmxremote.password\"@"
     $INSLATT_DIR/cassandra/conf/cassandra-env.sh
sed -i "s@num_tokens:.*@num_tokens: 16@" $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i "s@# allocate_tokens_for_keyspace:.*@allocate_tokens_for_keyspace:
     keyspace_with_replication_factor_3@" $INSLATT_DIR/cassandra/conf/cassandra.yaml
```

3.3 Setup: Multiple Data Centers, Multiple Nodes

Run the commands for each data center. When doing so, update the parameters that are marked in yellow. After these commands are carried out, run Cassandra (as shown below).

For DC1 (run on all nodes**):**

```
sed -i 's@dc=.*@dc=DC1@'  $INSLATT_DIR/cassandra/conf/cassandra-rackdc.properties
sed -i 's@cluster_name: .*@cluster_name: 'cassandra'@'  $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i s/seeds:.*/"seeds: \"192.168.168.201\""/g $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i s/listen_address:.*/"listen_address: $(hostname -I |awk {'print $1'})"/g
     $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i s/broadcast_rpc_address:.*/"broadcast_rpc_address: $(hostname -I |awk {'print $1'})"/g
     $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i 's@endpoint_snitch:.*@endpoint_snitch: GossipingPropertyFileSnitch@'
     $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i 's@LOCAL_JMX=.*@LOCAL_JMX='no'@'  $INSLATT_DIR/cassandra/conf/cassandra-env.sh
sed -i "s@-Djava.rmi.server.hostname=.*@-Djava.rmi.server.hostname=$(hostname -I |awk {'print
     $1'})\"@"  $INSLATT_DIR/cassandra/conf/cassandra-env.sh
sed -i "s@-Dcom.sun.management.jmxremote.password.file=.*@-
     Dcom.sun.management.jmxremote.password.file=$INSLATT_DIR/cassandra/conf/.jmxremote.password\"@"
     $INSLATT_DIR/cassandra/conf/cassandra-env.sh
```

Start Cassandra:

```
cassandra
```

For DC2 (run on all nodes**):**
The parameters marked in green must be the same as used in DC1.

```
sed -i 's@dc=.*@dc=DC2@'  $INSLATT_DIR/cassandra/conf/cassandra-rackdc.properties
sed -i 's@cluster_name: .*@cluster_name: 'cassandra'@'  $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i s/seeds:.*/"seeds: \"192.168.168.201\""/g $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i s/listen_address:.*/"listen_address: $(hostname -I |awk {'print $1'})"/g
     $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i s/broadcast_rpc_address:.*/"broadcast_rpc_address: $(hostname -I |awk {'print $1'})"/g
     $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i 's@endpoint_snitch:.*@endpoint_snitch: GossipingPropertyFileSnitch@'
     $INSLATT_DIR/cassandra/conf/cassandra.yaml
sed -i 's@LOCAL_JMX=.*@LOCAL_JMX='no'@'  $INSLATT_DIR/cassandra/conf/cassandra-env.sh
sed -i "s@-Djava.rmi.server.hostname=.*@-Djava.rmi.server.hostname=$(hostname -I |awk {'print
     $1'})\"@"  $INSLATT_DIR/cassandra/conf/cassandra-env.sh
sed -i "s@-Dcom.sun.management.jmxremote.password.file=.*@-
     Dcom.sun.management.jmxremote.password.file=$INSLATT_DIR/cassandra/conf/.jmxremote.password\"@"
     $INSLATT_DIR/cassandra/conf/cassandra-env.sh
```

Start Cassandra:

```
cassandra
```

3.4 Post First Start of the Cassandra Cluster

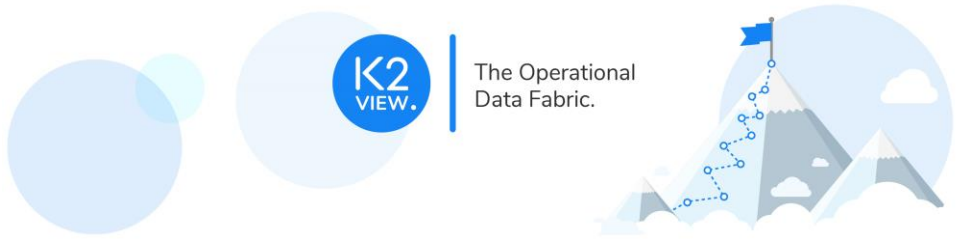
After the Cassandra cluster has been started for the first time, run the commands as shown below.

- Run the commands from only one of the nodes.
- Before running the commands, update the **system_auth**, **system_distributed**, and **system_traces** parameters with the relevant DCs and RFs (marked in yellow).
- If you have more the one DC add the other DCs and RFs.

```
echo "create user k2admin with password 'Q1w2e3r4t5' superuser;" |cqlsh -u cassandra -p cassandra
echo "ALTER KEYSPACE system_auth WITH REPLICATION = {'class' : 'NetworkTopologyStrategy', 'DC1': '3',
     'DC2': '3};" | cqlsh -ucassandra -pcassandra
echo "ALTER KEYSPACE system_distributed WITH REPLICATION = {'class' : 'NetworkTopologyStrategy',
     'DC1': '3', 'DC2': '3};" | cqlsh -ucassandra -pcassandra
echo "ALTER KEYSPACE system_traces WITH REPLICATION = {'class' : 'NetworkTopologyStrategy', 'DC1':
     '3', 'DC2': '3};" | cqlsh -ucassandra -pcassandra
```

After you have run the above commands, run the command below on each node in turn:

```
nodetool -u k2view -pw Q1w2e3r4t5 rebuild
```



4 Setup for Kafka

4.1 Load the Package

1. Download the package from: latest version.
2. Connect to Linux as "kafka" user and copy the package to the home directory.
3. Untar the package (the package name varies based on the version) as follows:

```
tar -zxvf k2view_Confluent_5.xxx.tar.gz
```

4.2 Setup: For a Single Node

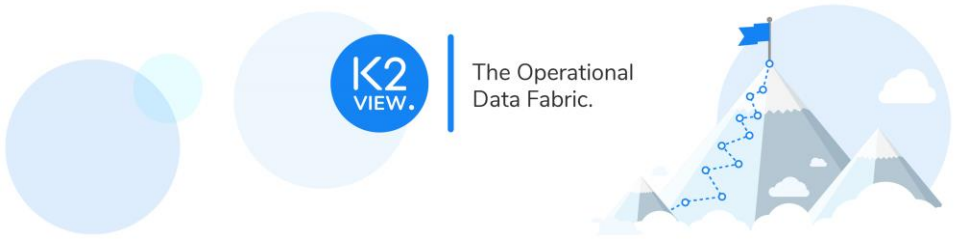
Run the commands as shown below. When doing so, update the parameters that are marked in yellow.

Pre setup:

```
wget https://owncloud-bkp2.s3.amazonaws.com/adminoc/fabricint/kafka/5.5.1/confluent-community-5.5.1-
     2.12.tar.gz
tar -zxvf k2view_Confluent_5.5.1_CE_Package_01.tar.gz && bash -l

## update the IP bellow and run the following on all the nodes
export kserver1=172.31.11.198

if [ "$(hostname -I |awk {'print $1'})" == "$kserver1" ]; then echo 1 > $K2_HOME/zk_data/myid; fi
if [ "$(hostname -I |awk {'print $1'})" == "$kserver1" ]; then sed -i "s@broker.id=.@broker.id=1@"
     $CONFLUENT_HOME/server.properties ; fi

sed -i "s@log.retention.minutes=.*@log.retention.hours=48@" $CONFLUENT_HOME/server.properties
sed -i "s@advertised.listeners=.*@advertised.listeners=PLAINTEXT:\/\/$(hostname -I |awk {'print
     $1'}):9093@" $CONFLUENT_HOME/server.properties
sed -i "s@advertised.host.name=.*@advertised.host.name=PLAINTEXT:\/\/$(hostname -I |awk {'print
     $1'}):9093@" $CONFLUENT_HOME/server.properties
sed -i "s@listeners=PLAINTEXT:\/\/.*@listeners=PLAINTEXT:\/\/$(hostname -I |awk {'print $1'}):9093@"
     $CONFLUENT_HOME/server.properties
sed -i "s@zookeeper.connect=.*.@zookeeper.connect=$kserver1:2181@" $CONFLUENT_HOME/server.properties
echo "default.replication.factor=3" >> $CONFLUENT_HOME/server.properties
sed -i "s@log.dirs=.*@log.dirs=$K2_HOME/zk_data@" $CONFLUENT_HOME/server.properties
sed -i "s@dataDir=.*@dataDir=$K2_HOME/zk_data@" $CONFLUENT_HOME/zookeeper.properties
sed -i "s@server.1=.*@#server.1=$(hostname -I |awk {'print $1'}):2888:3888@"
     $CONFLUENT_HOME/zookeeper.properties
echo "server.1=$kserver1:2888:3888" >> $CONFLUENT_HOME/zookeeper.properties
```

Start Kafka and Zookeeper:

```
$K2_HOME/kafka/bin/zookeeper-server-start -daemon $K2_HOME/kafka/zookeeper.properties
sleep 10
$K2_HOME/kafka/bin/kafka-server-start -daemon $K2_HOME/kafka/server.properties
```



Verify result:
You should see the exact number of nodes that you have, and their unique ID's:

```
~/kafka/bin/zookeeper-shell localhost:2181 <<< "ls /brokers/ids"
```

4.3 Setup: For a Cluster

Run the commands as shown below for each node in turn. When doing so, update the parameters that are marked in yellow.

The example shown is for three Kafka & Zookeeper clusters.

Pre setup:

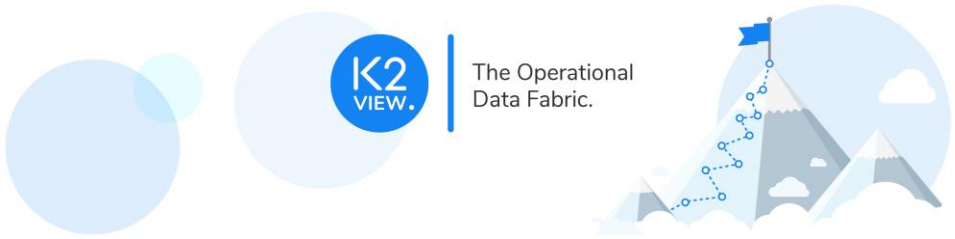
```
export kserver1=192.168.168.215
export kserver2=192.168.168.216
export kserver3=192.168.168.217

if [ "$(hostname -I |awk {'print $1'})" == "$kserver1" ]; then echo 1 > $K2_HOME/zk_data/myid; fi
if [ "$(hostname -I |awk {'print $1'})" == "$kserver2" ]; then echo 2 > $K2_HOME/zk_data/myid; fi
if [ "$(hostname -I |awk {'print $1'})" == "$kserver3" ]; then echo 3 > $K2_HOME/zk_data/myid; fi
if [ "$(hostname -I |awk {'print $1'})" == "$kserver1" ]; then sed -i "s@broker.id=.@broker.id=1@"
     ~/kafka/server.properties ; fi
if [ "$(hostname -I |awk {'print $1'})" == "$kserver2" ]; then sed -i "s@broker.id=.@broker.id=2@"
     ~/kafka/server.properties ; fi
if [ "$(hostname -I |awk {'print $1'})" == "$kserver3" ]; then sed -i "s@broker.id=.@broker.id=3@"
     ~/kafka/server.properties ; fi
sed -i "s@log.retention.minutes=.*@log.retention.hours=48@" $CONFLUENT_HOME/server.properties
sed -i "s@advertised.listeners=.*@advertised.listeners=PLAINTEXT:\/\/$(hostname -I |awk {'print
     $1'}):9093@" $CONFLUENT_HOME/server.properties
sed -i "s@advertised.host.name=.*@advertised.host.name=PLAINTEXT:\/\/$(hostname -I |awk {'print
     $1'}):9093@" $CONFLUENT_HOME/server.properties
sed -i "s@listeners=PLAINTEXT:\/\/.*@listeners=PLAINTEXT:\/\/$(hostname -I |awk {'print $1'}):9093@"
     $CONFLUENT_HOME/server.properties
sed -i "s@zookeeper.connect=.*@zookeeper.connect=$kserver1:2181,$kserver2:2181,$kserver3:2181@"
     ~/kafka/server.properties
echo "default.replication.factor=3" >> $CONFLUENT_HOME/server.properties
sed -i "s@log.dirs=.*@log.dirs=$K2_HOME/zk_data@" $CONFLUENT_HOME/server.properties
sed -i "s@dataDir=.*@dataDir=$K2_HOME/zk_data@" $CONFLUENT_HOME/zookeeper.properties
sed -i "s@server.1=.*@#server.1=$(hostname -I |awk {'print $1'}):2888:3888@"
     $CONFLUENT_HOME/zookeeper.properties
echo "server.1=$kserver1:2888:3888" >> $CONFLUENT_HOME/zookeeper.properties
echo "server.2=$kserver2:2888:3888" >> $CONFLUENT_HOME/zookeeper.properties
echo "server.3=$kserver3:2888:3888" >> $CONFLUENT_HOME/zookeeper.properties
```

Start Kafka and Zookeeper:

```
~/kafka/bin/zookeeper-server-start -daemon ~/kafka/zookeeper.properties
sleep 10
~/kafka/bin/kafka-server-start -daemon ~/kafka/server.properties
```

Verify result:
You should see the exact number of nodes that you have, and their unique ID's:

```
~/kafka/bin/zookeeper-shell localhost:2181 <<< "ls /brokers/ids"
```



5 Setup for Fabric

5.1 Load the Package

1. Download the package from the links that were provided to you.
2. Untar the package:

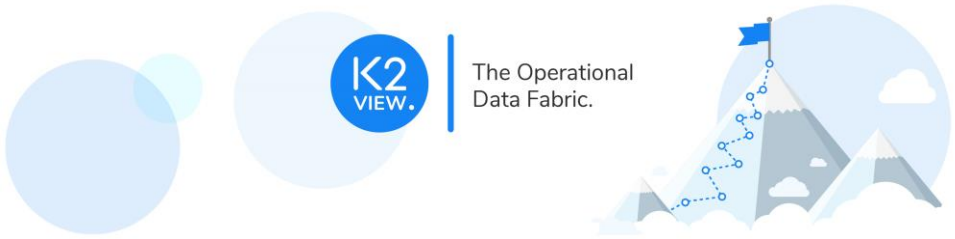
```
tar -zxvf k2fabric-server-*
```

5.2 Setup: For a Single Node

Run the commands as shown below. When doing so, update the parameters that are marked in yellow.

Pre setup:

```
sed -i "s@K2_HOME=.*@K2_HOME=$(pwd)@" .bash_profile
bash -l

## update the cassandra IP's
export cserver1=192.168.168.212
## update the fafka IP's
export kserver1=172.31.11.198

cp -r $K2_HOME/fabric/config.template $K2_HOME/config

sed -i 's@-Xmx2G@-Xmx8G@' $INSLATT_DIR/config/jvm.options
sed -i 's@-Xms2G@-Xms8G@' $INSLATT_DIR/config/jvm.options

cp config/adminInitialCredentials.template config/adminInitialCredentials
sed -i 's@user.*@k2consoleadmin/KW4RVG98RR9xcrTv@' config/adminInitialCredentials

sed -i 's@#REPLICATION_OPTIONS=.*@REPLICATION_OPTIONS={ '"'"'class'"'"' :
     '"'"'NetworkTopologyStrategy'"'"', '"'"DC1"'"' : 3}@' $K2_HOME/config/config.ini
sed -i "s@#HOSTS=.*@HOSTS=$cserver1,$cserver2,$cserver3@" $K2_HOME/config/config.ini
sed -i "s@#USER=.*@USER=k2admin@" $K2_HOME/config/config.ini
sed -i "s@#PASSWORD=.*@PASSWORD=Q1w2e3r4t5@" $K2_HOME/config/config.ini
sed -i "s@#MESSAGES_BROKER_TYPE=.*@MESSAGES_BROKER_TYPE=KAFKA@" $K2_HOME/config/config.ini
sed -i "s@#BOOTSTRAP_SERVERS=.*@BOOTSTRAP_SERVERS=$kserver1:9093@" $K2_HOME/config/config.ini
sed -i "s@HOSTS=.*@HOSTS=$cserver1,$cserver2,$cserver3@" $K2_HOME/config/iifConfig.ini
sed -i "s@#USER=.*@USER=k2admin@" $K2_HOME/config/iifConfig.ini
sed -i "s@#PASSWORD=.*@PASSWORD=Q1w2e3r4t5@" $K2_HOME/config/iifConfig.ini
sed -i "s@#KAFKA_BOOTSTRAP_SERVERS=.*@KAFKA_BOOTSTRAP_SERVERS=$kserver1:9093@"
     $K2_HOME/config/iifConfig.ini
sed -i
     "s@#ZOOKEEPER_BOOTSTRAP_SERVERS=.*@ZOOKEEPER_BOOTSTRAP_SERVERS=$kserver1:2181,$kserver2:2181,$ks
     erver3:2181@" $K2_HOME/config/iifConfig.ini
sed -i 's@#IIF_REPLICATION_OPTIONS=.*@IIF_REPLICATION_OPTIONS={ '"'"'class'"'"' :
     '"'"'NetworkTopologyStrategy'"'"', '"'"DC1"'"' : 3}@' $K2_HOME/config/iifConfig.ini
sed -i "s@#BOOTSTRAP_SERVERS=.*@BOOTSTRAP_SERVERS=$kserver1:9093@" $K2_HOME/config/iifConfig.ini

# start fabric
```




```
k2fabric start && k2fabric status
```

Start Fabric:

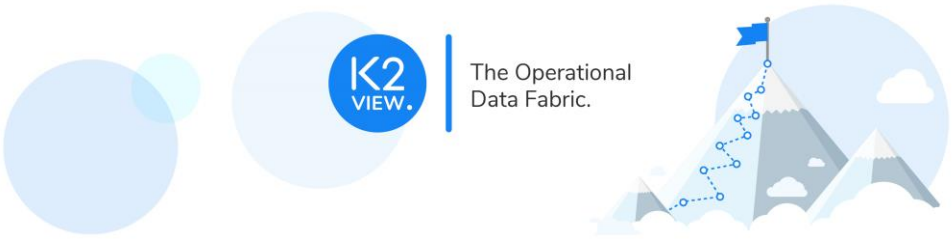
```
k2fabric start
```

5.3 Setup: Fabric with Multiple Nodes

Run the commands as shown below. When doing so, update the parameters that are marked in yellow.

Pre setup:

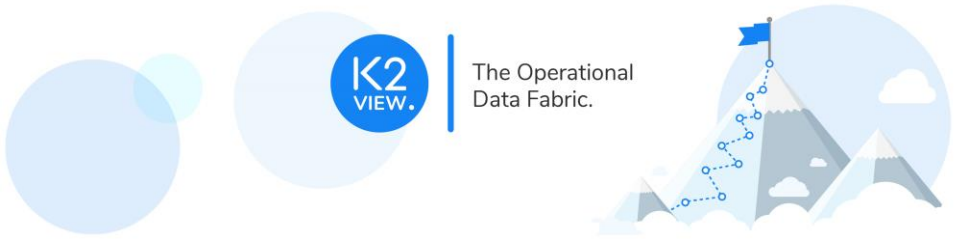
```
sed -i "s@K2_HOME=.*@K2_HOME=$(pwd)@" .bash_profile
bash -l

## update the cassandra IP's
export cserver1=192.168.168.212
export cserver2=192.168.168.213
export cserver3=192.168.168.214

## update the kafka IP's
export kserver1=172.31.11.198
export kserver2=172.31.35.204
export kserver3=172.31.31.69

cp -r $K2_HOME/fabric/config.template $K2_HOME/config

sed -i 's@-Xmx2G@-Xmx8G@' $INSLATT_DIR/config/jvm.options
sed -i 's@-Xms2G@-Xms8G@' $INSLATT_DIR/config/jvm.options

cp config/adminInitialCredentials.template config/adminInitialCredentials
sed -i 's@user.*@k2consoleadmin/KW4RVG98RR9xcrTv@' config/adminInitialCredentials

sed -i 's@#REPLICATION_OPTIONS=.*@REPLICATION_OPTIONS={ '"'"'class'"'"' :
     '"'"'NetworkTopologyStrategy'"'"', '"'"'DC1'"'"' : 3}@' $K2_HOME/config/config.ini
sed -i "s@#HOSTS=.*@HOSTS=$cserver1,$cserver2,$cserver3@" $K2_HOME/config/config.ini
sed -i "s@#USER=.*@USER=k2admin@" $K2_HOME/config/config.ini
sed -i "s@#PASSWORD=.*@PASSWORD=Q1w2e3r4t5@" $K2_HOME/config/config.ini
sed -i "s@#MESSAGES_BROKER_TYPE=.*@MESSAGES_BROKER_TYPE=KAFKA@" $K2_HOME/config/config.ini
sed -i "s@#BOOTSTRAP_SERVERS=.*@BOOTSTRAP_SERVERS=$kserver1:9093,$kserver2:9093,$kserver3:9093@"
     $K2_HOME/config/config.ini
sed -i "s@HOSTS=.*@HOSTS=$cserver1,$cserver2,$cserver3@" $K2_HOME/config/iifConfig.ini
sed -i "s@#USER=.*@USER=k2admin@" $K2_HOME/config/iifConfig.ini
sed -i "s@#PASSWORD=.*@PASSWORD=Q1w2e3r4t5@" $K2_HOME/config/iifConfig.ini
sed -i
     "s@#KAFKA_BOOTSTRAP_SERVERS=.*@KAFKA_BOOTSTRAP_SERVERS=$kserver1:9093,$kserver2:9093,$kserver3:9
     093@" $K2_HOME/config/iifConfig.ini
sed -i
     "s@#ZOOKEEPER_BOOTSTRAP_SERVERS=.*@ZOOKEEPER_BOOTSTRAP_SERVERS=$kserver1:2181,$kserver2:2181,$ks
     erver3:2181@" $K2_HOME/config/iifConfig.ini
sed -i 's@#IIF_REPLICATION_OPTIONS=.*@IIF_REPLICATION_OPTIONS={ '"'"'class'"'"' :
     '"'"'NetworkTopologyStrategy'"'"', '"'"'DC1'"'"' : 3}@' $K2_HOME/config/iifConfig.ini
sed -i "s@#BOOTSTRAP_SERVERS=.*@BOOTSTRAP_SERVERS=$kserver1:9093,$kserver2:9093,$kserver3:9093@"
     $K2_HOME/config/iifConfig.ini
```



Start Fabric:

Start the first fabric node, and wait until it is ready.
Once it is, you can start all other Fabric nodes:

```
k2fabric start && k2fabric status
```



6 External Jars Installation

6.1 Jar File Location

This description is for Fabric 5.1 and above.

There are two options for loading the Jar-compressed files: dynamically (default) or statically (if jars are used directly in the implementation).

6.1.1 Loading Jars Dynamically

Studio:

- Implementation external jars are located on the same folder as on previous releases <project folder>/lib/.
- Move database external jars (JDBC drivers) to <project folder>/lib/<database type>/ for each database type.

Server:

- Move external jars to /home/k2view/ExternalJars (part of classpath).
- Move database external jars (JDBC drivers) to /home/k2view/ExternalJars/<database type>/ for each database type that is not included as a part of fabric package.

6.1.2 Loading Jars Statically

Studio:

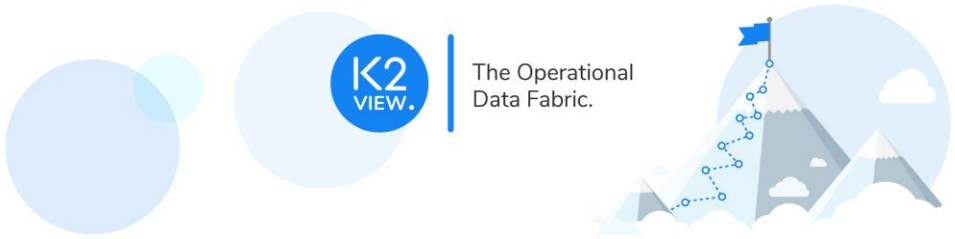
- Create a new database type in the studio project tree based on the existing database type (New Database Type From Template).
- Change the Dynamic Jars property to **false**.
- Save it with the same name.
- Locate the database type jars under <project folder>/lib (part of classpath).

Server:

- Move external jars to /home/k2view/ExternalJars (part of classpath).
- Move database external jars (JDBC drivers) to /home/k2view/ExternalJars/.

Notes:

1. Fabric and Cassandra Jars are loaded statically and are part of the classpath.
2. Fabric/DynamicJars folder has been removed.
3. Loading jars Dynamically for Oracle/SqlServer/PostgreSql is supported out of the box and do not require any special treatment.
4. Out of the box supported database types that require only creation of Database type folder are as follows:
   DB2, MySQL, Salesforce, Netezza, Vertica, Teradata, Denodo, CiscoInfoServer, Informix, Impala, DVM, Hadoop_Hive, Sybase_ASE, Sybase_IQ, Sybase_ASA, Cache.
5. Jars should be copied manually as JDBC drivers are not provided as a part of the standard Fabric package.



7 Stopping Services

7.1 Cassandra

From Cassandra 3.11.x, execute the following commands once to enable the stop command for the service:

```
su – cassandra
sed -i 's/Qw1w2e3r4t5/Q1w2e3r4t5/g' $CASSANDRA_HOME/bin/stop-server
stop-server
```

NOTE: In higher versions of Cassandra, the fix is already placed and the stop-server command can be used as-is.

7.2 Kafka and Zookeeper

Stop Kafka and Zookeeper by running the following commands:

```
su - kafka
kafka-server-stop
zookeeper-server-stop
```

7.3 Fabric

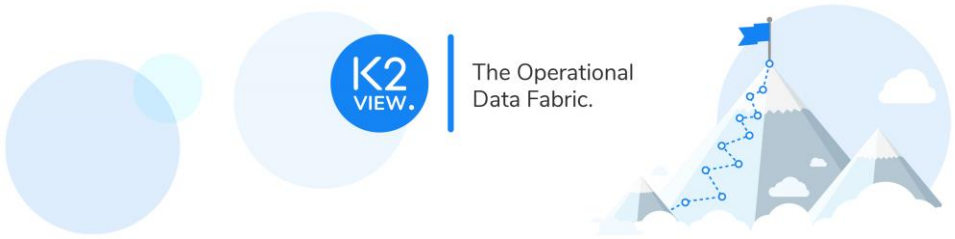
Stop Fabric by running the following commands:

```
su – k2view
k2fabric stop
```

8 Change Cassandra password

To change the Cassandra password, run the following command using a different DBA user than “Cassandra”:

```
echo "ALTER ROLE cassandra WITH SUPERUSER = true AND LOGIN = true AND password='q1w2e3r4t5';"
     | cqlsh -uadmin -padmin localhost
```



9 Legal Notices