INSTALLING HADOOP 2.7.3

#Download the latest stable Hadoop using wget from one of the Apache mirrors.

```
root@ip-172-31-23-142:/home/ubuntu# wget
https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz
--2020-04-05 13:02:47--  https://archive.apache.org/dist/hadoop/core/hadoop-
2.7.3/hadoop-2.7.3.tar.gz
Resolving archive.apache.org (archive.apache.org)... 163.172.17.199
Connecting to archive.apache.org (archive.apache.org)|163.172.17.199|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 214092195 (204M) [application/x-gzip]
Saving to: 'hadoop-2.7.3.tar.gz'

hadoop-2.7.3.tar.gz
100%[=================================================================
================>] 204.17M  15.4MB/s    in 19s

2020-04-05 13:03:07 (10.9 MB/s) - 'hadoop-2.7.3.tar.gz' saved [214092195/214092195]

root@ip-172-31-23-142:/home/ubuntu# tar -xvf hadoop-2.7.3.tar.gz
```

Create a directory where the hadoop will store its data. We will set this directory path in hdfs-site.

```
root@ip-172-31-23-142:/home/ubuntu# Java -version

#Add the Hadoop related environment variables in your bash file.
root@ip-172-31-23-142:/home/ubuntu#vi ~/.bashrc

#Copy and paste these environment variables.
#java variables
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/
export JRE_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre
export PATH:$JAVA_HOME:$JRE_HOME/bin
```

#Hadoop variables
export HADOOP_HOME=/home/ubuntu/hadoop-2.7.3
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

#Save and exit and use this command to refresh the bash settings.
root@ip-172-31-23-142:/home/ubuntu# source ~/.bashrc

Setting hadoop environment for password less ssh access. Password less SSH Configuration is a mandatory installation requirement. However it is more useful in a distributed environment.
root@ip-172-31-23-142:/home/ubuntu# ssh-keygen -t rsa -P "
root@ip-172-31-23-142:/home/ubuntu# cat $HOME/.ssh/id_rsa.pub >> root@ip-172-31-23-142:/home/ubuntu# $HOME/.ssh/authorized_keys
check password less ssh access to localhost
root@ip-172-31-23-142:/home/ubuntu# ssh localhost
#exit from inner localhost shell
root@ip-172-31-23-142:/home/ubuntu# exit

Set the hadoop config files. We need to set the below files in order for hadoop to function properly.

- core-site.xml
- hadoop-env.sh
- yarn-site.xml
- hdfs-site.xml
- mapred-site.xml

go to directory where all the config files are present (cd /home/ubuntu/hadoop-2.6.0/etc/Hadoop)

- Copy and paste the below configurations in core-site.xml

##Add the following text between the configuration tabs.
<property>
<name>hadoop.tmp.dir</name>

<value>/home/ubuntu/hadoop tmp/hadoop-${user.name}</value>
<description>A base for other temporary directories.</description>
</property>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>

- Copy and paste the below configurations in hadoop-env.sh

get the java home directory using:
readlink -f `which java`
Example output: /usr/lib/jvm/java-8-oracle/jre/bin/java (NOTE THE JAVA_HOME PATH. JUST GIVE THE BASE DIRECTORY PATH)
##Need to set JAVA_HOME in hadoop-env.sh
export JAVA_HOME=/usr/lib/jvm/java-8-oracle

- Copy and paste the below configurations in mapred-site.xml

#copy mapred-site.xml from mapred-site.xml.template
cp mapred-site.xml.template mapred-site.xml
vi mapred-site.xml
#Add the following text between the configuration tabs.
<property>
<name>mapred.job.tracker</name>
<value>localhost:9001</value>
</property>

- Copy and paste the below configurations in yarn-site.xml

##Add the following text between the configuration tabs.
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>

- Copy and paste the below configurations in hdfs-site.xml

##Add the following text between the configuration tabs.
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property><name>dfs.name.dir</name>

```
<value>file:///home/ubuntu/hadoopdata/hdfs/namenode</value>
</property>
<property>
<name>dfs.data.dir</name>
<value>file:///home/ubuntu/hadoopdata/hdfs/datanode</value>
</property>
```

Formatting the HDFS file system via NameNode (after installing hadoop, for the first time we have to format the HDFS file system to make it work)
hdfs namenode -format

Issue the following commands to start hadoop
cd sbin/
./start-dfs.sh
./start-yarn.sh
#If you have properly done step 5, you can start Hadoop from any directory. (Note the user should be the one where you installed Hadoop)
start-all.sh
OR you can separately start required services as below:
Name node:
hadoop-daemon.sh start namenode
Data node:
hadoop-daemon.sh start datanode
Resource Manager:
yarn-daemon.sh start resourcemanager
Node Manager:
yarn-daemon.sh start nodemanager
Job History Server:
mr-jobhistory-daemon.sh start historyserver
Once hadoop has started point your browser to http://localhost:50070/

Check for hadoop processes /daemons running on hadoop with Java Virtual Machine Process Status Tool.
root@ip-172-31-23-142:/home/ubuntu# jps

OR you can check TCP and port details by using
root@ip-172-31-23-142:/home/ubuntu# sudo netstat -plten | grep java

INSTALLING ZOOKEEPER

root@ip-172-31-23-142:/home/ubuntu# sudo apt-get install zookeeperd

INSTALLING KAFKA 2.4.0

Download Kafka

To install Kafka on your machine, click on the below link –

wget https://downloads.apache.org/kafka/2.4.0/kafka_2.11-2.4.0.tgz
Extract the tar file
root@ip-172-31-23-142:/home/ubuntu# tar -xvf kafka_2.11-2.4.0.tgz

Start Server

You can start the server by giving the following command –

root@ip-172-31-23-142:/home/ubuntu/kafka/ bin/kafka-server-start.sh
config/server.properties
To run in the background
$KAFKA_HOME/bin/zookeeper-server-start.sh -daemon config/zookeeper.properties
$KAFKA_HOME/bin/kafka-server-start.sh -daemon config/server.properties

Note before going to start the kafka server the Please specify the kafka_home path in the bashrc file.

INSTALLING HIVE 2.1.0

root@ip-172-31-23-142:/home/ubuntu# wget https://archive.apache.org/dist/hive/hive-2.1.0/apache-hive-2.1.0-bin.tar.gz
root@ip-172-31-23-142:/home/ubuntu# tar -xvf apache-hive-2.1.0-bin.tar.gz
root@ip-172-31-23-142:/home/ubuntu# sudo vi ~/.bashrc

#Add the following lines at the end of the file
HIVE Paths
export HIVE_HOME=/home/ubuntu/apache-hive-2.1.0-bin
export PATH=$PATH:$HIVE_HOME/bin
export CLASSPATH=$CLASSPATH:/home/ubuntu/apache-hive-2.1.0-bin/Hadoop/lib/*:.
export CLASSPATH=$CLASSPATH:/home/ubuntu/apache-hive-2.1.0-bin/lib/*:.

#The following command is used to execute ~/.bashrc file.
root@ip-172-31-23-142:/home/ubuntu# source ~/.bashrc

#Configure Hive to work with Hadoop
Move to conf directory under $HIVE_HOME and execute the following commands
root@ip-172-31-23-142:/home/ubuntu/apache-hive-2.1.0/conf # cp hive-env.sh.template hive-env.sh

#Edit hive-env.sh and add the below line
export HADOOP_HOME=/home/ubuntu/hadoop/
Hive installation is now complete. We will need an external database server to configure Metastore. We will use Apache Derby for this.

#Download and untar Apache Derby
root@ip-172-31-23-142:/home/ubuntu# wget http://archive.apache.org/dist/db/derby/db-derby-10.13.1.1/db-derby-10.13.1.1-bin.tar.gz

root@ip-172-31-23-142:/home/ubuntu# tar xvzf db-derby-10.13.1.1-bin.tar.gz

#Update bashrc
root@ip-172-31-23-142:/home/ubuntu# vi ~/.bashrc

Add the following lines at the end of the file

#Derby Paths

export DERBY_HOME=/usr/local/derby

export PATH=$PATH:$DERBY_HOME/bin

Export CLASSPATH=$CLASSPATH:$DERBY_HOME/lib/derby.jar:$DERBY_HOME/lib/derbytools.jar

root@ip-172-31-23-142:/home/ubuntu# source .bashrc

#Create a directory to store metastore
root@ip-172-31-23-142:/home/ubuntu# sudo mkdir $DERBY_HOME/data
Configure Metastore of Hive

#Move to conf directory of Hive

root@ip-172-31-23-142:/home/ubuntu/apache-hive-2.1.0/conf/cp hive-default.xml.template hive-site.xml
#Edit hive-site.xml

#Add the below lines

```
<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:derby://localhost:1527/metastore_db;create=true </value>

<description>JDBC connect string for a JDBC metastore </description>

</property>
```

Run hive

root@ip-172-31-23-142:/home/ubuntu/apache-hive-2.1.0/bin/hive

#Troubleshooting for Hive installation

Error 1

Sometimes, you might get the below error when you run hive.

Logging initialized using configuration in jar:file:/home/hadoopuser1/Projects/hive/lib/hive-common-2.1.0.jar!/hive-log4j2.properties Async: true

Exception in thread “main” java.lang.RuntimeException: org.apache.hadoop.hive.ql.metadata.HiveException: org.apache.hadoop.hive.ql.metadata.HiveException: MetaException(message:Hive metastore database is not initialized. Please use schematool (e.g. ./schematool -initSchema -dbType …) to create the schema. If needed, don’t forget to include the option to auto-create the underlying database in your JDBC connection string (e.g. ?createDatabaseIfNotExist=true for mysql))

If you encounter this error, the execute the below command

root@ip-172-31-23-142:/home/ubuntu/apache-hive-2.1.0/bin/schematool -initSchema -dbType derby

If all goes well, you can now execute hive

If not, there is a chance that you might see this error

Error 2

```
Starting metastore schema initialization to 2.1.0

Initialization script hive-schema-2.1.0.derby.sql

Error: FUNCTION ‘NUCLEUS_ASCII’ already exists. (state=X0Y68,code=30000)

org.apache.hadoop.hive.metastore.HiveMetaException: Schema initialization FAILED! Metastore state would be inconsistent !!

Underlying cause: java.io.IOException : Schema script failed, errorcode 2
```

```
Use –verbose for detailed stacktrace.

*** schemaTool failed ***
```

Running hive, even though it fails, creates a metastore_db directory in the directory from which you ran hive.

You will need to delete this directory

root@ip-172-31-23-142:/home/ubuntu/apache-hive-2.1.0/bin/ mv metastore_db metastore_db.tmp

Once it's deleted, again run the below command

root@ip-172-31-23-142:/home/ubuntu/apache-hive-2.1.0/bin/schematool -initSchema -dbType derby
You should see the below output

```
SLF4J: Class path contains multiple SLF4J bindings.

SLF4J: Found binding in [jar:file:/home/hadoopuser1/Projects/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: Found binding in [jar:file:/home/hadoopuser1/Projects/hadoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.

SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Metastore connection URL: jdbc:derby:;databaseName=metastore_db;create=true

Metastore Connection Driver : org.apache.derby.jdbc.EmbeddedDriver

Metastore connection User: APP

Starting metastore schema initialization to 2.1.0
```

```
Initialization script hive-schema-2.1.0.derby.sql

Initialization script completed

schemaTool completed

root@ip-172-31-23-142:/home/ubuntu/hive/conf$
```

While trying to run hive, you might encounter the below error:

Error 3

Logging initialized using configuration in jar:file:/home/hadoopuser1/Projects/hive/lib/hive-common-2.1.0.jar!/hive-log4j2.properties Async: true

Exception in thread “main” java.lang.IllegalArgumentException: java.net.URISyntaxException: Relative path in absolute URI: ${system:java.io.tmpdir%7D/$%7Bsystem:user.name%7D

If you do, open hive-site.xml

root@ip-172-31-23-142:/home/ubuntu/apache-hive-2.1.0/conf/ sudo vi hive-site.xml
Edit the following lines

```
<property>

<name>hive.exec.scratchdir</name>

<value>/tmp/hive</value>

<description>HDFS root scratch dir for Hive jobs which gets created with write all (733) permission. For each connecting user, an HDFS scratch dir:
```

```
${hive.exec.scratchdir}/&lt;username&gt; is created, with
${hive.scratch.dir.permission}.</description>

</property>

<property>

<name>hive.exec.local.scratchdir</name>

<value>/tmp/hadoopuser1</value>

<description>Local scratch space for Hive jobs</description>

</property>

<property>

<name>hive.downloaded.resources.dir</name>

<value>/tmp/hadoopuser1_resources</value>

<description>Temporary local directory for added resources in the remote file
system.</description>

</property>

<property>

<name>hive.scratch.dir.permission</name>

<value>733</value>

<description>The permission for the user specific scratch directories that get
created.</description>

</property>
```

Create a directory for the hive warehouse into hdfs. This directory will be used by Hive to store all the data into HDFS-

cmd: hadoop dfs -mkdir -p /user/hive/warehouse

Configuring Metastore of Hive

Configuring Metastore means specifying to Hive where the database is stored. You can do this by editing the hive-site.xml file, which is in the $HIVE_HOME/conf directory. First of all, copy the template file using the following command:

$ cd $HIVE_HOME/conf

$ cp hive-default.xml.template hive-site.xml

 # Edit hive-site.xml and append the following lines between the <configuration> and </configuration> tags:

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:derby://localhost:1527/metastore_db;create=true </value>
  <description>JDBC connect string for a JDBC metastore </description>
</property>
```

 # Create a file named jpox.properties and add the following lines into it:

```
javax.jdo.PersistenceManagerFactoryClass =
org.jpox.PersistenceManagerFactoryImpl
org.jpox.autoCreateSchema = false
org.jpox.validateTables = false
org.jpox.validateColumns = false
org.jpox.validateConstraints = false
org.jpox.storeManagerType = rdbms
org.jpox.autoCreateSchema = true
org.jpox.autoStartMechanismMode = checked
org.jpox.transactionIsolation = read_committed
javax.jdo.option.DetachAllOnCommit = true
```

javax.jdo.option.NontransactionalRead = true

javax.jdo.option.ConnectionDriverName = org.apache.derby.jdbc.ClientDriver

javax.jdo.option.ConnectionURL = jdbc:derby://hadoop1:1527/metastore_db;create = true

javax.jdo.option.ConnectionUserName = APP

javax.jdo.option.ConnectionPassword = mine

 # Verifying Hive Installation

Before running Hive, you need to create the /tmp folder and a separate Hive folder in HDFS. Here, we use the /user/hive/warehouse folder. You need to set write permission for these newly created folders as shown below:

chmod g+w

Now set them in HDFS before verifying Hive. Use the following commands:

$HADOOP_HOME/bin/hadoop fs -mkdir /tmp

$HADOOP_HOME/bin/hadoop fs -mkdir /user/hive/warehouse

$HADOOP_HOME/bin/hadoop fs -chmod g+w /tmp

$HADOOP_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse

The following commands are used to verify Hive installation:

$ cd $HIVE_HOME

$ bin/hive

 # On successful installation of Hive, you get to see the following response:

Logging initialized using configuration in jar:file:/home/hadoop/hive-0.9.0/lib/hive-common-0.9.0.jar!/hive-log4j.properties

Hive history file=/tmp/hadoop/hive_job_log_hadoop_201312121621_1494929084.txt

…….

hive>

The following sample command is executed to display all the tables:

hive> show tables;

OK

Time taken: 2.798 seconds

hive>

INSTALLING NIFI 1.10.0

1.Download the source file from the apache website

```
root@ip-172-31-23-142:/home/ubuntu# wget
https://archive.apache.org/dist/nifi/1.10.0/nifi-1.10.0-bin.tar.gz
--2020-04-06 12:41:43--  https://archive.apache.org/dist/nifi/1.10.0/nifi-1.10.0-bin.tar.gz
Resolving archive.apache.org (archive.apache.org)... 163.172.17.199
Connecting to archive.apache.org (archive.apache.org)|163.172.17.199|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1372451011 (1.3G) [application/x-gzip]
Saving to: 'nifi-1.10.0-bin.tar.gz'

nifi-1.10.0-bin.tar.gz
100%[=================================================================
==============>]   1.28G  14.5MB/s    in 98s
```

2. Extract the file :

```
root@ip-172-31-23-142:/home/ubuntu# tar -xvf nifi-1.10.0-bin.tar.gz
```

3. Configuration

NiFi provides several different configuration options which can be configured on nifi.properties file.

NOTE: Here in nifi.properties to don’t get port conflicts i'm changing the port no :8080 to 9999 and host is localhost.

root@ip-172-31-23-142:/home/ubuntu/nifi-1.10.0/conf/# vi nifi.properties

Step 3: Starting Apache Nifi:
On the terminal window,navigate to the Nifi directory and run the following below commands:

Lauches the applicaion run in the foreground and exit by pressing Ctrl-c.
root@ip-172-31-23-142:/home/ubuntu# bin/nifi.sh run
 Launches the application run the background.
root@ip-172-31-23-142:/home/ubuntu# bin/nifi.sh start
root@ip-172-31-23-142:/home/ubuntu# bin/nifi.sh status - To check the application status
root@ip-172-31-23-142:/home/ubuntu# bin/nifi.sh stop - To shutdown the application

Apache Nifi Web User Interface:

After Apache Nifi Started, Web User Interface (UI) to create and monitor our dataflow.
To use Apache Nifi, open a web browser and navigate to http://localhost:8080/nifi
Note: default port is 8080 in our environment we navigate to the http://localhost:9999/nifi

INSTALLING DRUID 0.18.1

Download Druid
To install Druid on your machine, click on the below link –
https://druid.apache.org/downloads.html

root@ip-172-31-23-142:/home/ubuntu$ sudo wget http://apachemirror.wuchna.com/druid/0.18.1/apache-druid-0.18.1-bin.tar.gz
--2020-07-19 21:44:36-- http://apachemirror.wuchna.com/druid/0.18.1/apache-druid-0.18.1-bin.tar.gz
Resolving apachemirror.wuchna.com (apachemirror.wuchna.com)... 159.65.154.237
Connecting to apachemirror.wuchna.com (apachemirror.wuchna.com)|159.65.154.237|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 264932123 (253M) [application/x-gzip]

Saving to: ‘apache-druid-0.18.1-bin.tar.gz’

apache-druid-0.18.1-bin.tar.gz
100%[==
==>]
252.66M 4.47MB/s in 1m 40s

2020-07-19 21:46:16 (2.53 MB/s) - ‘apache-druid-0.18.1-bin.tar.gz’ saved
[264932123/264932123]

Extract the tar file :
root@ip-172-31-23-142:/home/ubuntu$ sudo tar -xzf apache-druid-0.18.1-bin.tar.gz
root@ip-172-31-23-142:/home/ubuntu$ cd apache-druid-0.18.1/

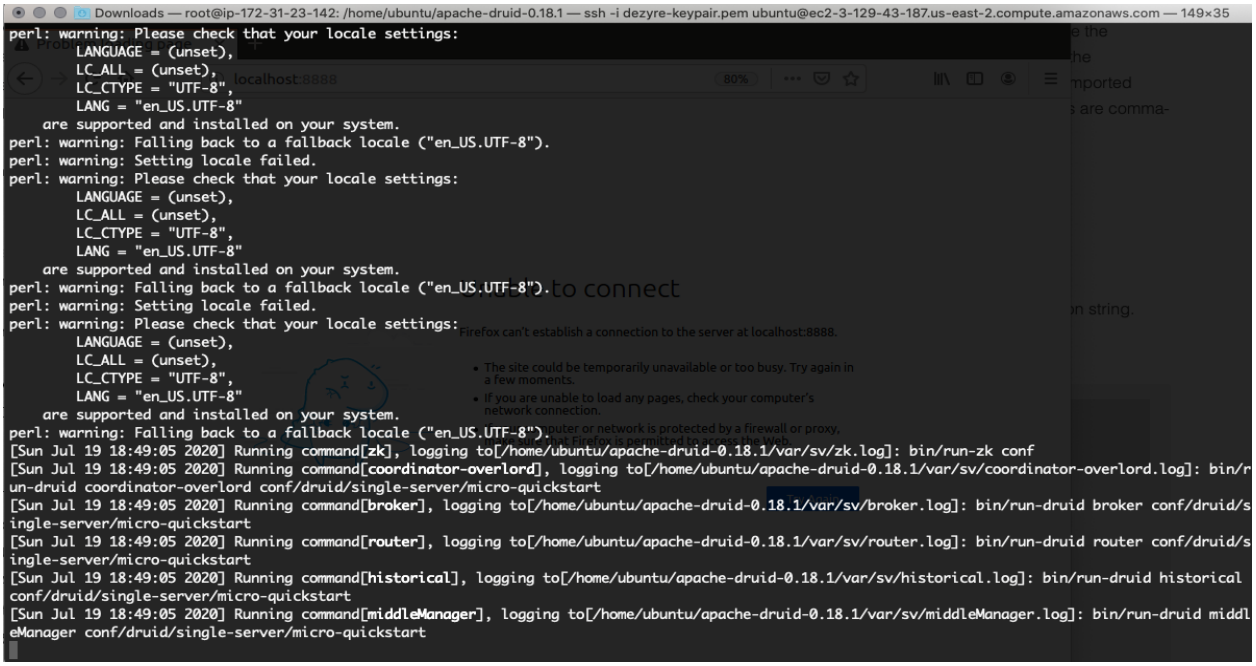
Start the Druid service :

root@ip-172-31-23-142:/home/ubuntu/apache-druid-0.18.1$ sudo ./bin/start-micro-quickstart

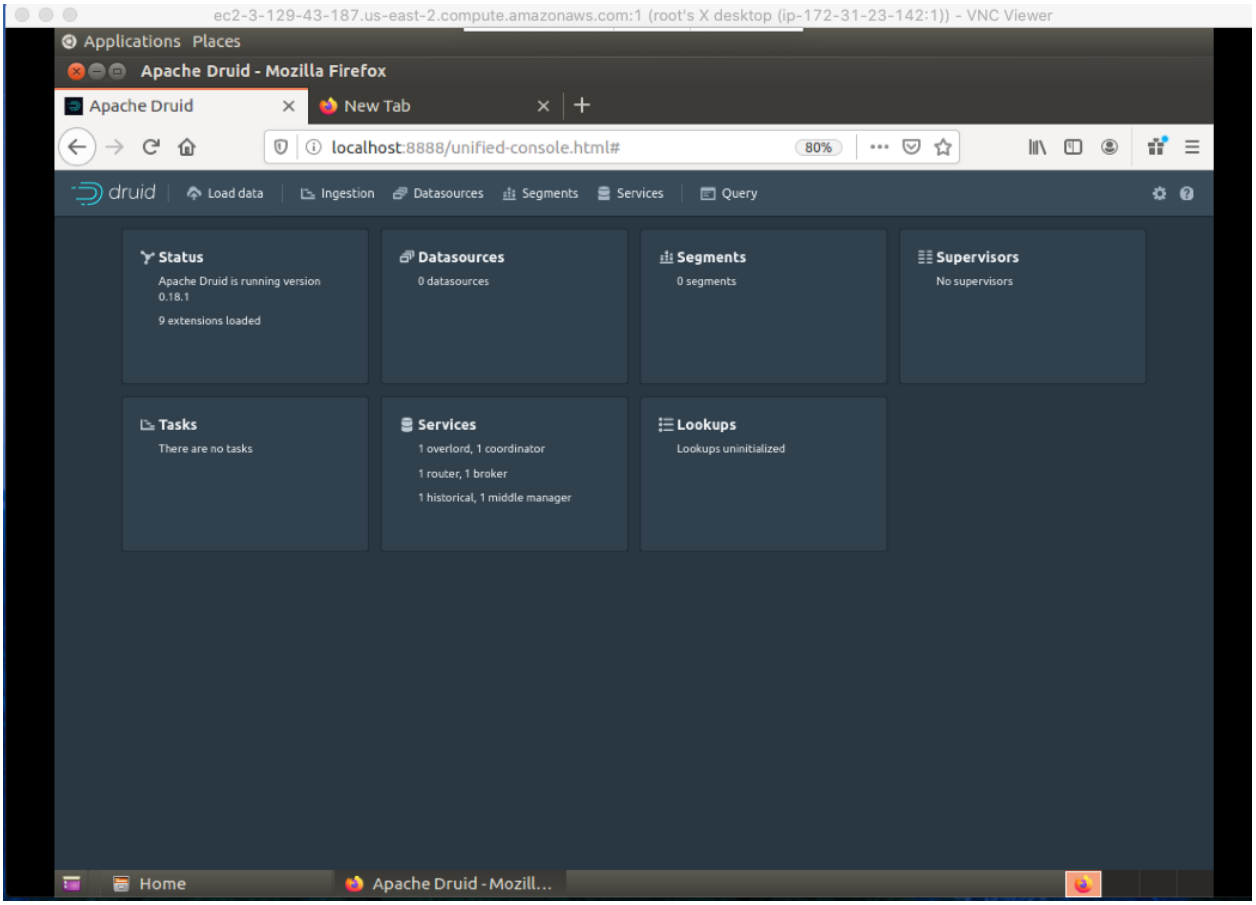
[Sun Jul 19 21:56:57 2020] Running command[zk], logging to[/home/bigdata/apache-druid-0.18.1/var/sv/zk.log]: bin/run-zk conf
[Sun Jul 19 21:56:57 2020] Running command[coordinator-overlord], logging to[/home/bigdata/apache-druid-0.18.1/var/sv/coordinator-overlord.log]: bin/run-druid coordinator-overlord conf/druid/single-server/micro-quickstart
[Sun Jul 19 21:56:57 2020] Running command[broker], logging to[/home/bigdata/apache-druid-0.18.1/var/sv/broker.log]: bin/run-druid broker conf/druid/single-server/micro-quickstart
[Sun Jul 19 21:56:57 2020] Running command[router], logging to[/home/bigdata/apache-druid-0.18.1/var/sv/router.log]: bin/run-druid router conf/druid/single-server/micro-quickstart
[Sun Jul 19 21:56:57 2020] Running command[historical], logging to[/home/bigdata/apache-druid-0.18.1/var/sv/historical.log]: bin/run-druid historical conf/druid/single-server/micro-quickstart
[Sun Jul 19 21:56:57 2020] Running command[middleManager], logging to[/home/bigdata/apache-druid-0.18.1/var/sv/middleManager.log]: bin/run-druid middleManager conf/druid/single-server/micro-quickstart

```
root@ip-172-31-23-142:/home/ubuntu#
root@ip-172-31-23-142:/home/ubuntu# sudo service zookeeper status
● zookeeper.service - LSB: centralized coordination service
   Loaded: loaded (/etc/init.d/zookeeper; bad; vendor preset: enabled)
   Active: inactive (dead) since Sun 2020-07-19 18:48:15 UTC; 4s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 8380 ExecStop=/etc/init.d/zookeeper stop (code=exited, status=0/SUCCESS)

Jul 19 12:01:06 ip-172-31-23-142 systemd[1]: Starting LSB: centralized coordination service...
Jul 19 12:01:06 ip-172-31-23-142 systemd[1]: Started LSB: centralized coordination service.
Jul 19 18:48:14 ip-172-31-23-142 systemd[1]: Stopping LSB: centralized coordination service...
Jul 19 18:48:15 ip-172-31-23-142 systemd[1]: Stopped LSB: centralized coordination service.
root@ip-172-31-23-142:/home/ubuntu# sudo service zookeeper status
● zookeeper.service - LSB: centralized coordination service
   Loaded: loaded (/etc/init.d/zookeeper; bad; vendor preset: enabled)
   Active: inactive (dead) since Sun 2020-07-19 18:48:15 UTC; 16s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 8380 ExecStop=/etc/init.d/zookeeper stop (code=exited, status=0/SUCCESS)

Jul 19 12:01:06 ip-172-31-23-142 systemd[1]: Starting LSB: centralized coordination service...
Jul 19 12:01:06 ip-172-31-23-142 systemd[1]: Started LSB: centralized coordination service.
Jul 19 18:48:14 ip-172-31-23-142 systemd[1]: Stopping LSB: centralized coordination service...
Jul 19 18:48:15 ip-172-31-23-142 systemd[1]: Stopped LSB: centralized coordination service.
root@ip-172-31-23-142:/home/ubuntu# cd apache-druid-0.18.1
root@ip-172-31-23-142:/home/ubuntu/apache-druid-0.18.1# sudo ./bin/start-micro-quickstart
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = (unset),
        LC_ALL = (unset),
        LC_CTYPE = "UTF-8",
        LANG = "en_US.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to a fallback locale ("en_US.UTF-8").
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = (unset),
        LC_ALL = (unset),
        LC_CTYPE = "UTF-8",
        LANG = "en_US.UTF-8"
    are supported and installed on your system.
```

```
perl: warning: Falling back to a fallback locale ("en_US.UTF-8").
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = (unset),
        LC_ALL = (unset),
        LC_CTYPE = "UTF-8",
        LANG = "en_US.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to a fallback locale ("en_US.UTF-8").
[Sun Jul 19 18:49:05 2020] Running command[zk], logging to[/home/ubuntu/apache-druid-0.18.1/var/sv/zk.log]: bin/run-zk conf
[Sun Jul 19 18:49:05 2020] Running command[coordinator-overlord], logging to[/home/ubuntu/apache-druid-0.18.1/var/sv/coordinator-overlord.log]: bin/run-druid coordinator-overlord conf/druid/single-server/micro-quickstart
[Sun Jul 19 18:49:05 2020] Running command[broker], logging to[/home/ubuntu/apache-druid-0.18.1/var/sv/broker.log]: bin/run-druid broker conf/druid/single-server/micro-quickstart
[Sun Jul 19 18:49:05 2020] Running command[router], logging to[/home/ubuntu/apache-druid-0.18.1/var/sv/router.log]: bin/run-druid router conf/druid/single-server/micro-quickstart
[Sun Jul 19 18:49:05 2020] Running command[historical], logging to[/home/ubuntu/apache-druid-0.18.1/var/sv/historical.log]: bin/run-druid historical conf/druid/single-server/micro-quickstart
[Sun Jul 19 18:49:05 2020] Running command[middleManager], logging to[/home/ubuntu/apache-druid-0.18.1/var/sv/middleManager.log]: bin/run-druid middleManager conf/druid/single-server/micro-quickstart
```



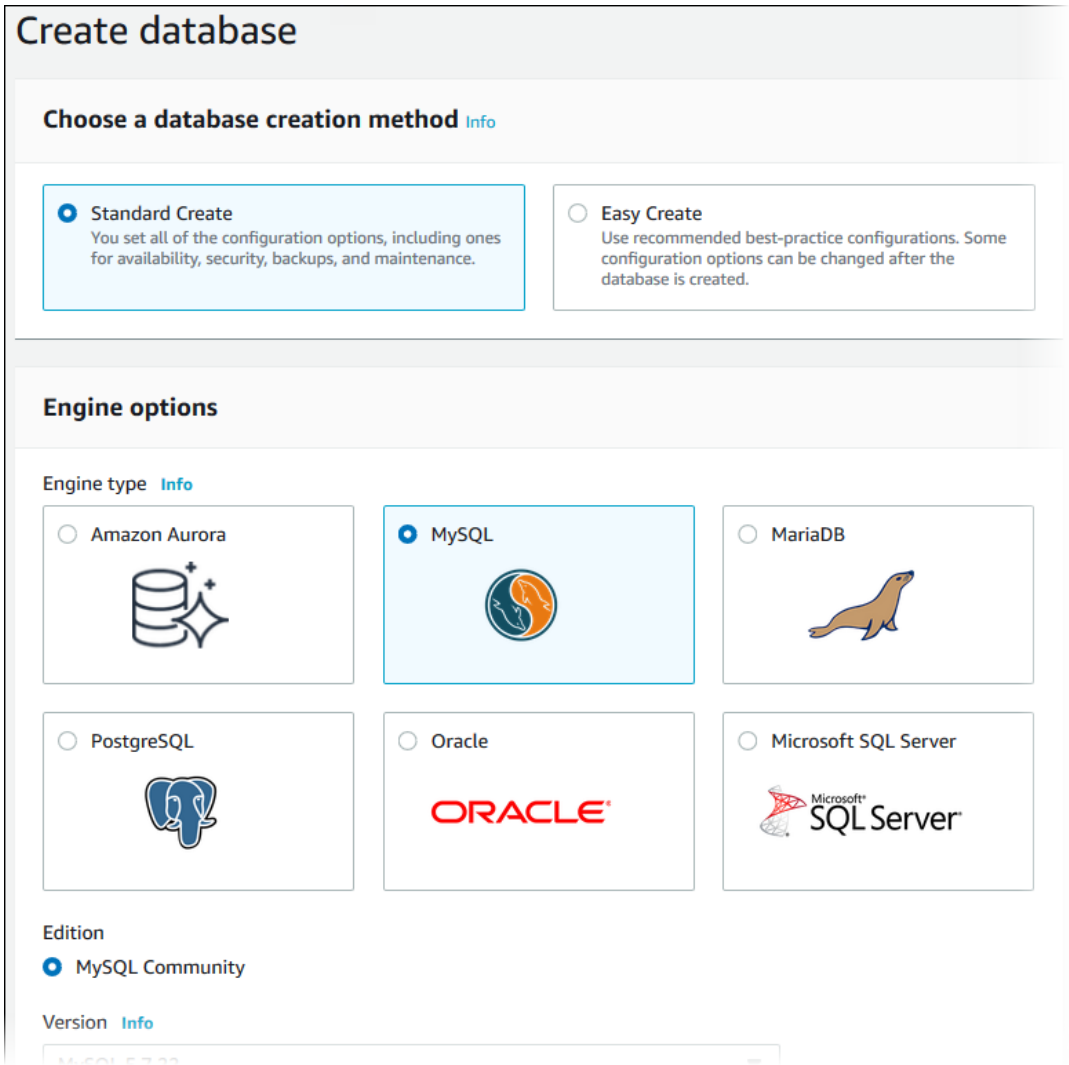
To access the Druid console as below : https://localhost:8888/unified-console.html



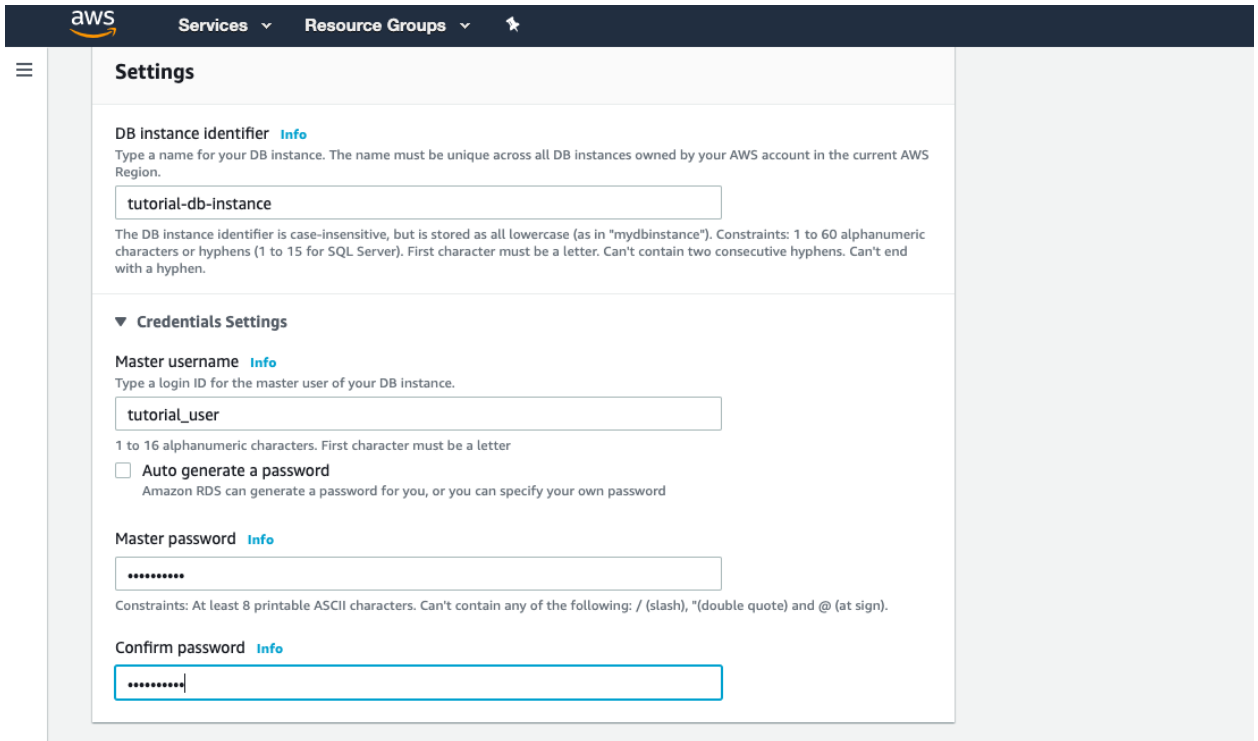
INSTALLING Amazon Relational Database Service(RDS):

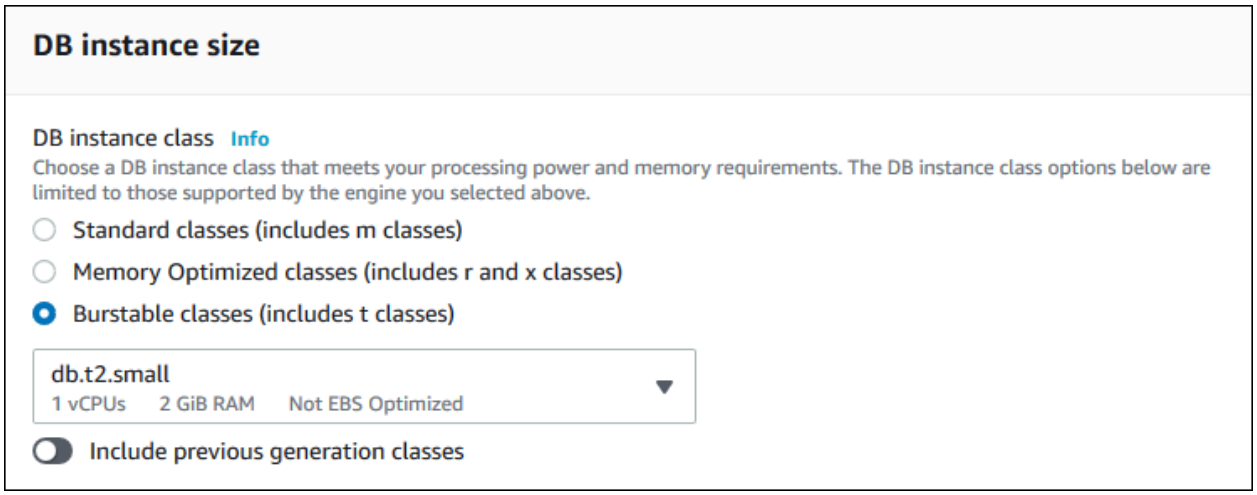
To launch a MySQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the upper-right corner of the AWS Management Console, choose the AWS Region where you want to create the DB instance. This example uses the US West (Oregon) Region.
3. In the navigation pane, choose Databases.
4. Choose Create database.
5. On the Create database page, shown following, make sure that the Standard Create option is chosen, and then choose MySQL.



6. In the **Templates** section, choose **Dev/Test**.
7. In the **Settings** section, set these values:
    a. **DB instance identifier** – **dezyredb**
    b. **Master username** – **dezyre_user**
    c. **Auto generate a password** – Disable the option
    d. **Master password** – Choose a password.
    e. **Confirm password** – Retype the password.



1. In the **DB instance size** section, set these values:
   - **Burstable classes (includes t classes)**
   - **db.t2.small**
2. 

3. In the **Storage** and **Availability & durability** sections, use the default values.
4. In the **Connectivity** section, open **Additional connectivity configuration** and set these values:
   - **Virtual Private Cloud (VPC)** – Choose an existing VPC with both public and private subnets, such as the dezyre-vpc (vpc-*identifier*) created in Create a VPC

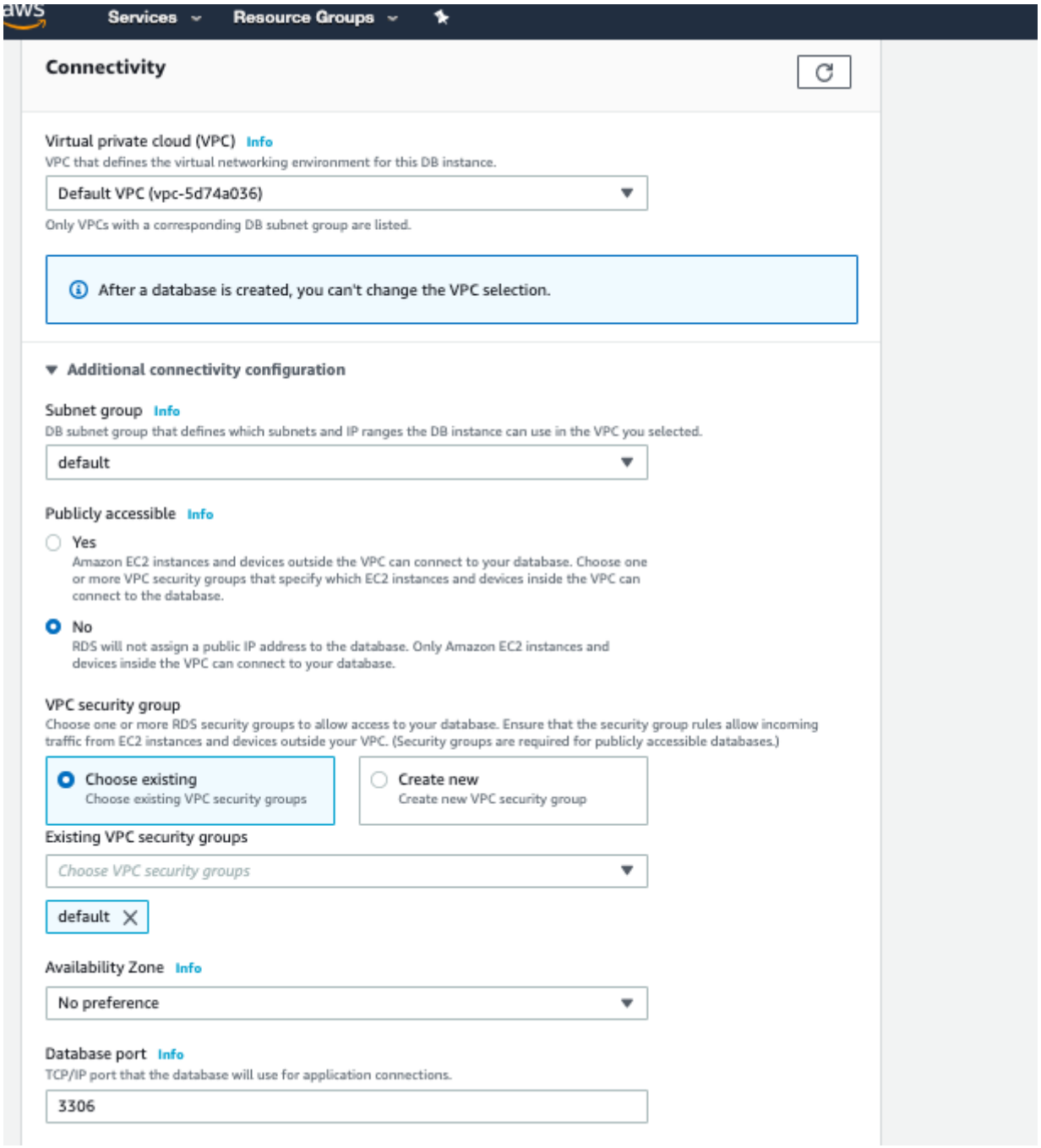
with Private and Public Subnets

Note

The VPC must have subnets in different Availability Zones.

- **Subnet group** – The DB subnet group for the VPC, such as the `tutorial-db-subnet-group` created in Create a DB Subnet Group
- **Publicly accessible – No**
- **VPC security groups** – Choose an existing VPC security group that is configured for private access, such as the `tutorial-db-securitygroup` created in Create a VPC Security Group for a Private DB Instance.

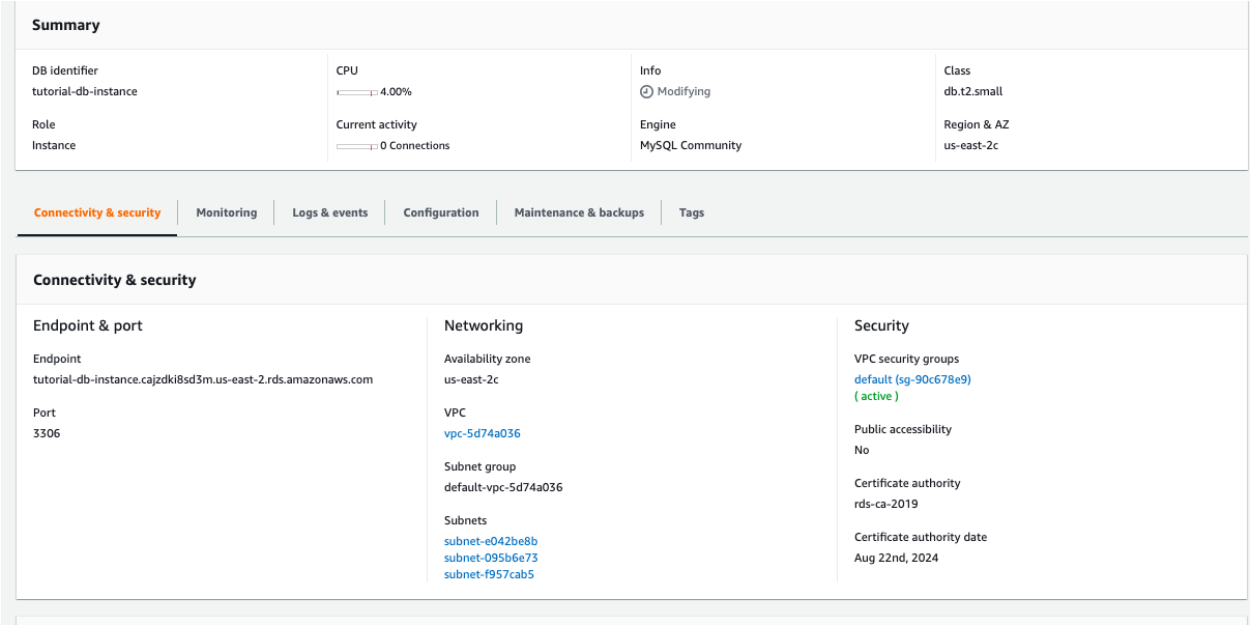
  Remove other security groups, such as the default security group, by choosing the **X** associated with each.
- **Availability zone – No Preference**
- **Database port – 3306**

5. 

6. Open the **Additional configuration** section, and enter **sample** for **Initial database name**. Keep the default settings for the other options.
7. Choose **Create database** to create your RDS MySQL DB instance.

   Your new DB instance appears in the **Databases** list with the status **Creating**.
8. Wait for the **Status** of your new DB instance to show as **Available**. Then choose the DB instance name to show its details.

9. In the **Connectivity & security** section, view the **Endpoint** and **Port** of the DB instance.



Note the endpoint and port for your DB instance. You use this information to connect your web server to your DB instance.

To make sure that your DB instance is as secure as possible, verify that sources outside of the VPC can't connect to your DB instance

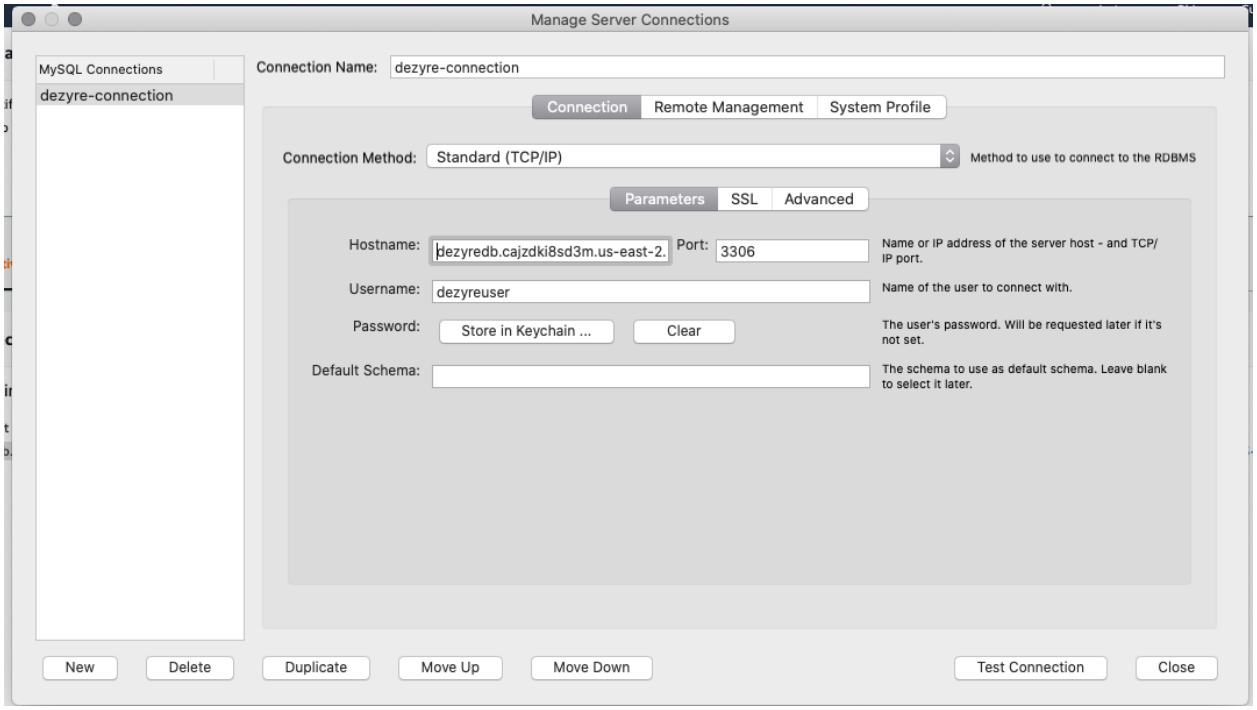
Connecting from MySQL Workbench

To connect from MySQL Workbench

1. Download and install MySQL Workbench
   Link :http://dev.mysql.com/downloads/workbench/
2. Open MySQL Workbench.

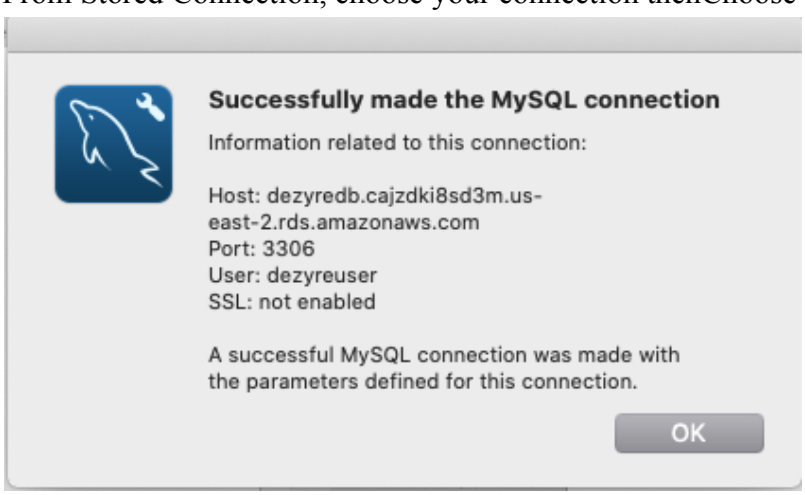
3. From Database, choose Manage Connections.
4. In the Manage Server Connections window, choose New.
5. In the Connect to Database window, enter the following information:
   - Stored Connection – Enter a name for the connection, such as MyDB.
   - Hostname – Enter the DB instance endpoint.
   - Port – Enter the port used by the DB instance.
   - Username – Enter the username of a valid database user, such as the master user.
   - Password – Optionally, choose Store in Vault and then enter and save the password for the user.

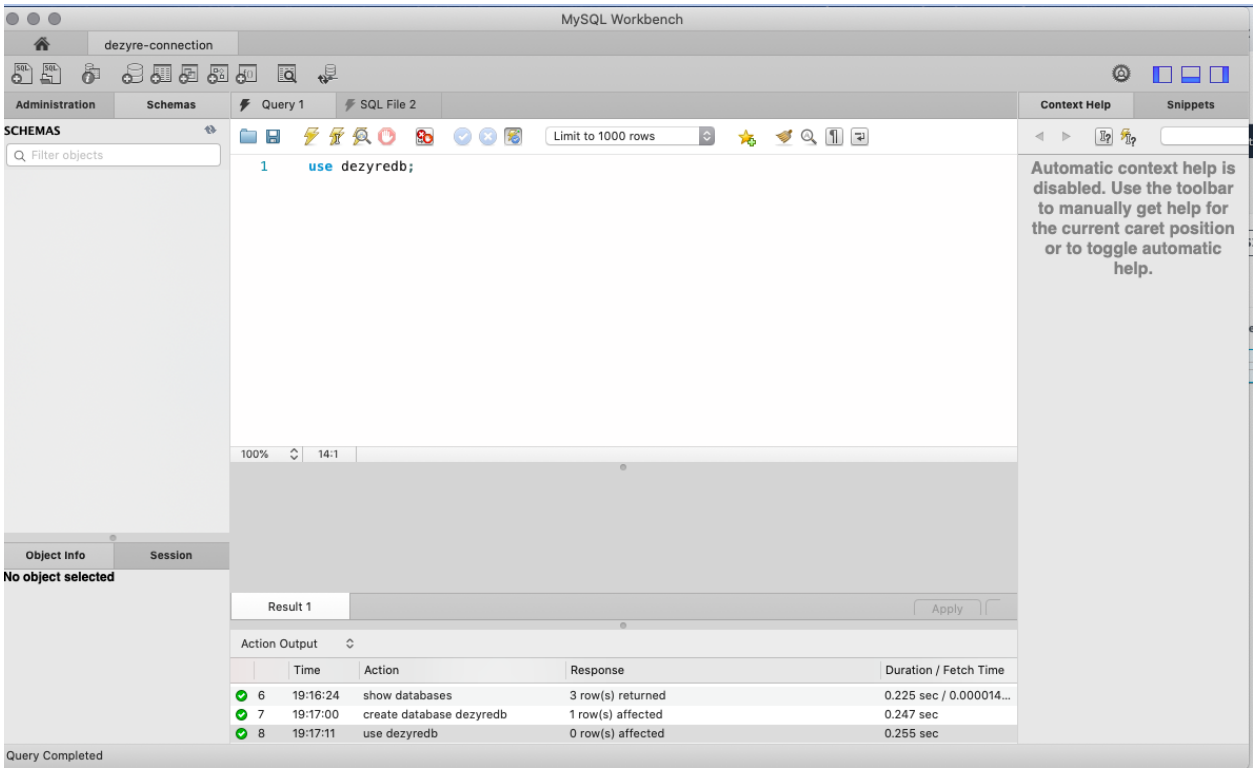
6. The window looks similar to the following:



You can use the features of MySQL Workbench to customize connections. For example, you can use the SSL tab to configure SSL connections. For information about using MySQL Workbench, see the [MySQL Workbench documentation](#).

7. Optionally, choose Test Connection to confirm that the connection to the DB instance is successful.
8. Choose Close.
9. From Database, choose Connect to Database.
10. From Stored Connection, choose your connection thenChoose OK.



11. 
MySQL Workbench
dezyre-connection
Administration
Schemas
Query 1
SQL File 2
Context Help
Snippets
SCHEMAS
Filter objects
Limit to 1000 rows
use dezyredb;
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
100%
14:1
Object Info
Session
No object selected
Result 1
Apply
Action Output
Time
Action
Response
Duration / Fetch Time
6 19:16:24 show databases 3 row(s) returned 0.225 sec / 0.000014...
7 19:17:00 create database dezyredb 1 row(s) affected 0.247 sec
8 19:17:11 use dezyredb 0 row(s) affected 0.255 sec
Query Completed