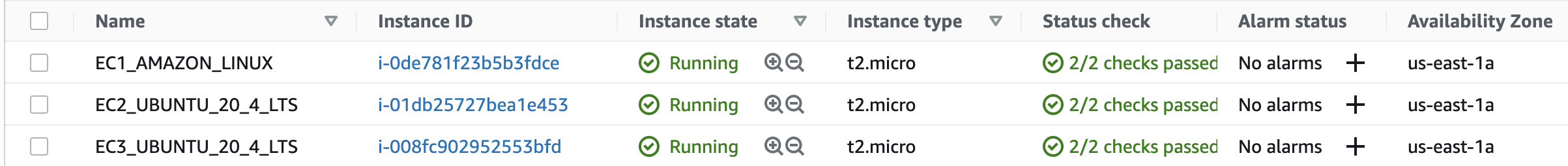
**Assignment -1**

**Configuration and Setup**

****

**Instance Details**

**Instance -EC1**

**Name** = EC1\_AMAZON\_LINUX

**OSType** = Amazon Linux

**Software Installed** = Java, Python

**OPEN SSH Connect command**

ssh -i "EC1\_AMAZON\_LINUX\_KEY.pem" [ec2-user@ec2-54-242-76-157.compute-1.amazonaws.com](mailto:ec2-user@ec2-54-242-76-157.compute-1.amazonaws.com)

**Java install Command**

sudo yum install java-1.8.0-openjdk

**Python Install Command**

sudo yum install python3

sudo yum install python3-pip

**Instance -2**

**Name** = EC2\_UBUNTU\_20\_4\_LTS, EC3\_UBUNTU\_20\_4\_LTS

**OSType** = Ubuntu 20\_4

**Software Installed** = Java, Python

**OPEN SSH Connect**

ssh -i "EC1\_AMAZON\_LINUX\_KEY.pem" [ec2-user@ec2-54-242-76-157.compute-1.amazonaws.com](mailto:ec2-user@ec2-54-242-76-157.compute-1.amazonaws.com)

**Java install :**

sudo apt install openjdk-8-jre-headless

**Python Install :**

sudo apt install python3

sudo apt install python3-pip

**Instance -3**

**Name** = EC3\_UBUNTU\_20\_4\_LTS, EC3\_UBUNTU\_20\_4\_LTS

**OSType** = Ubuntu 20\_4

**Software Installed** = Java, Python

**OPEN SSH Connect**

ssh -i "EC1\_AMAZON\_LINUX\_KEY.pem" [ec2-user@ec2-54-242-76-157.compute-1.amazonaws.com](mailto:ec2-user@ec2-54-242-76-157.compute-1.amazonaws.com)

**Java install :**

sudo apt install openjdk-8-jre-headless

**Python Install :**

sudo apt install python3

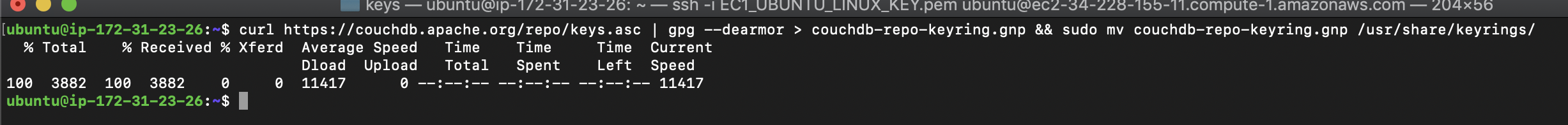
sudo apt install python3-pip

**CouchDB Install :**

**Step 1: Add the CouchDB GPG key**

First, let's add the CouchDB GPG key. To fetch the official CouchDB repo key, type:

curl https://couchdb.apache.org/repo/keys.asc | gpg --dearmor > couchdb-repo-keyring.gnp && sudo mv couchdb-repo-keyring.gnp /usr/share/keyrings/



**Step 2: Enable the CouchDB repository**

To enable the CouchDB repository, run the following command:

echo "deb [signed-by=/usr/share/keyrings/couchdb-repo-keyring.gnp] https://apache.jfrog.io/artifactory/couchdb-deb focal main" > couchdb.list && sudo mv couchdb.list /etc/apt/sources.list.d/

**Step 3: Update the Ubuntu system**

sudo apt update

**Step 4: Install CouchDB on Ubuntu**

$ sudo apt-get install -y couchdb

## CouchDB interface bind address: 127.0.0.1

**Step 6: Verify CouchDB installation**

curl http://127.0.0.1:5984

## 

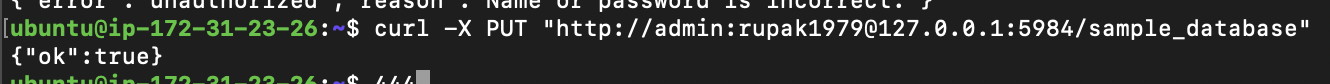
**Step 7: Create a database**

To create a database, the syntax is "http://{username}:{password}@127.0.0.1:5984/{your new database}".

**For example:**

$ curl -X PUT "http://admin:sample\_password@127.0.0.1:5984/sample\_database"

curl -X PUT <http://admin:rupak1979@127.0.0.1:5984/sample_database>



To list all CouchDB databases, type:

$ curl -X GET <http://admin:sample_password@127.0.0.1:5984/_all_dbs>

$ curl -X GET http://admin:rupak1979@127.0.0.1:5984/\_all\_dbs

Important Commands :

sudo find / -name \*.ini

DB Commands :

Create a database

curl -X PUT <http://admin:rupak1979@127.0.0.1:5984/sample_database>

Put value in a database

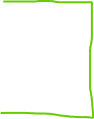
sudo curl -X PUT http://admin:rupak1979@127.0.0.1:5984/sample\_database/"001" -d '{ " Name " : " Raju " , " age " :" 23 " , " Designation " : " Designer " }'

Get value in a database

sudo curl -X GET <http://admin:rupak1979@127.0.0.1:5984/sample_database/001>



**AWS Assignment -1**







**Description :**

Publishing message from client instances deployed in an EC2 instance in AWS to a Kafka cluster ( AWS MSK Cluster) and consuming message from another client instance deployed within same or different EC2 instance.

**What is Amazon MSK :**

Amazon MSK is a new AWS streaming data service that manage kafka infrastructure and operation.Basically it is an managed kafka cluster

**Steps for the demo :**

1. Create Kafka Cluster using AWS MSK.

<https://docs.aws.amazon.com/msk/latest/developerguide/msk-create-cluster.html>

Graphical user interface, text, application, email

Description automatically generated

1. Install AWS CLI

<https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html>

1. Create a EC2 instance to host the producer and consumer.
2. Install java and Kafka on EC2 instance ( Considering Amazon Linux EC2 instance)

* Open a terminal and SSH to EC2 instance
* install Java: sudo yum install java-1.8.0
* Get Kafka:wget <https://archive.apache.org/dist/kafka/2.2.1/kafka_2.12-2.2.1.tgz>
* Extract Kafka: tar -xzf kafka\_2.12-2.2.1.tgz

1. Get the zoo keeper connection string and other information about your cluster by entering the following code

* Open a terminal
* Make sure AWS CLI is already installed
* Execute the below command
* aws kafka describe-cluster --region <region-name> --cluster-arn <MSK Cluster arn>
* Example is given below
* aws kafka describe-cluster --region us-east-1 --cluster-arn arn:aws:kafka:us-east-1:246653441795:cluster/demo-cluster-1/ee4ad7f;2C1-3b88-4e00-8ae4-460e6082b8ca-19.
* Copy the zookeeper string which will be used to create the topic

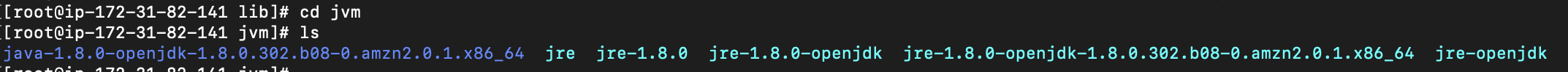
1. Create a kafka topic

* Open up a terminal
* SSH to EC2 instance
* cd kafka\_2.12-2.2.1/
* bin/kafka-topics.sh --create --zookeeper "ZookeeperConnectString" --replication-factor 2 --partitions 1 --topic <topicname>
* replace the ZookeeperConnectString with the value generated in step-5
* example command is given below
* bin/kafka-topics.sh --create --zookeeper "z-1.demo-cluster-1.usw9js.c19.kafka.us-east-1.amazonaws.com:2181,z-2.demo-cluster-1.usw9js.c19.kafka.us-east-1.amazonaws.com:2181,z-3.demo-cluster-1.usw9js.c19.kafka.us-east-1.amazonaws.com:2181" --replication-factor 2 --partitions 1 --topic AWSKafkaTutorialTopic

1. Generate the client and server certificate by copying the certificate from JRE’s certificate

Form the same terminal connecting to EC2 instance use the following command

* cd /home/ec2-user
* cd /
* ls
* cd /usr
* cd /lib
* cd jvm
* copy the java runtime



java-1.8.0-openjdk-1.8.0.302.b08-0.amzn2.0.1.x86\_64

1. copy the certificate to truststore

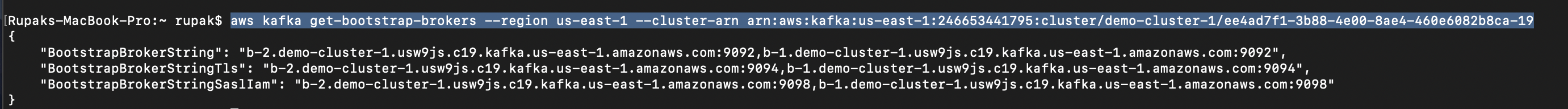
* cp /usr/lib/jvm/JDKFolder/jre/lib/security/cacerts /tmp/kafka.client.truststore.jks
* Replace the JDKfolder to that of the jdk runtime . The final command is as below
* cp /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.302.b08-0.amzn2.0.1.x86\_64/jre/lib/security/cacerts /tmp/kafka.client.truststore.jks

1. Creating Client Properties File

cd kafka\_2.12-2.2.1/

* ls
* cd bin
* vim client.properties
* [add following entries and save]
* security.protocol=SSL
* ssl.truststore.location=/tmp/kafka.client.truststore.jks

1. Get BootstrapBrokerString to create some message

* Open a terminal without SSH
* Make sure AWS CLI is installed and run below command.regionanme is the aws region anme and arn is the arn of the MSK kafka cluster created in step-1
* aws kafka get-bootstrap-brokers --region <regionname> —cluster-arn <arn name>
* aws kafka get-bootstrap-brokers --region us-east-1 --cluster-arn arn:aws:kafka:us-east-1:246653441795:cluster/demo-cluster-1/ee4ad7f1-3b88-4e00-8ae4-460e6082b8ca-19
* copy the BootstrapBrokerStringTls which will be used to create producer and consumer
* ****

1. Creating producer

* Open up a terminal to be used as a client machine which is act as a producer.
* SSH to EC2 instance
* cd /home/ec2-user/kafka\_2.12-2.2.1/bin
* run the following command by replacing the BootstrapBrokerStringTls with the value generated in step 10
* ./kafka-console-producer.sh --broker-list BootstrapBrokerStringTls --producer.config client.properties --topic AWSKafkaTutorialTopic
* Example is as below
* ./kafka-console-producer.sh --broker-list "b-2.demo-cluster-1.usw9js.c19.kafka.us-east-1.amazonaws.com:9094,b-1.demo-cluster-1.usw9js.c19.kafka.us-east-1.amazonaws.com:9094" --producer.config client.properties --topic AWSKafkaTutorialTopic
* The prompt will ask you to enter message that you need to publish

1. Creating Consumer

* Open up a terminal to be used as a client machine which is act as a producer.
* SSH to EC2 instance
* cd /home/ec2-user/kafka\_2.12-2.2.1/bin
* run the following command by replacing the BootstrapBrokerStringTls with the value generated in step 10
* ./kafka-console-consumer.sh --bootstrap-server BootstrapBrokerStringTls --consumer.config client.properties --topic AWSKafkaTutorialTopic --from-beginning
* Example is as below
* ./kafka-console-consumer.sh --bootstrap-server “b-2.demo-cluster-1.usw9js.c19.kafka.us-east-1.amazonaws.com:9094,b-1.demo-cluster-1.usw9js.c19.kafka.us-east-1.amazonaws.com:9094” --consumer.config client.properties --topic AWSKafkaTutorialTopic --from-beginning

The prompt will display the message consumed by producer

Text

Description automatically generated