**Mini Project 3: Cloud Computing Labs: Using AWS and Docker**

**Part 1: Setup AWS and Docker**

**1.1** Review the lecture about using AWS. Create your own AWS account. Note that a credit card will be used for creating the account. New users will get the AWS Free Tier for one year, which will be sufficient for this lab.

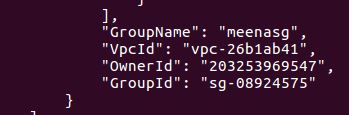
* Setup AWS command line tool. Use the following commands in your ubuntu box.
* Read the Python boto3 library documentation and understand how to use boto3 to access EC2 and S3

**1.2** Setup Docker in your linux system.

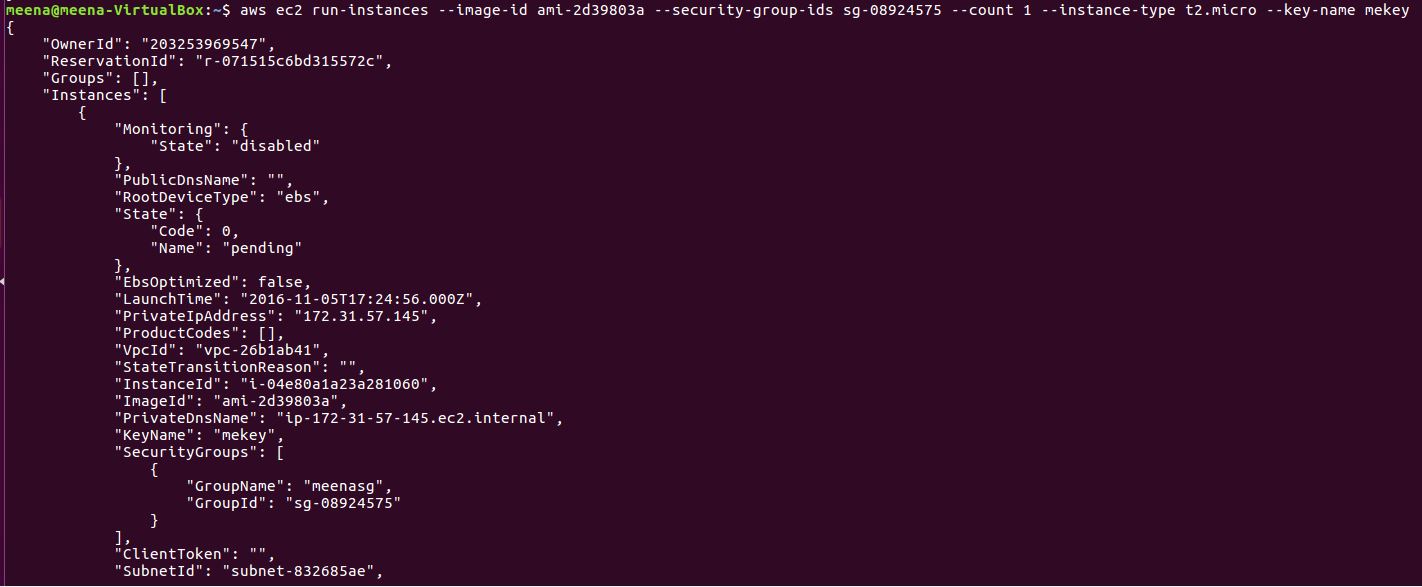
Now, answer the following questions:

**Question 1.1** create an instance with the commands discussed in the lecture note. Use the ubuntu image ami2d39803a. Remember that for free tier use, you need to use "instancetype t2.micro".Finally login the instance with ssh. Save your screen shots to showthat you have successfully done.

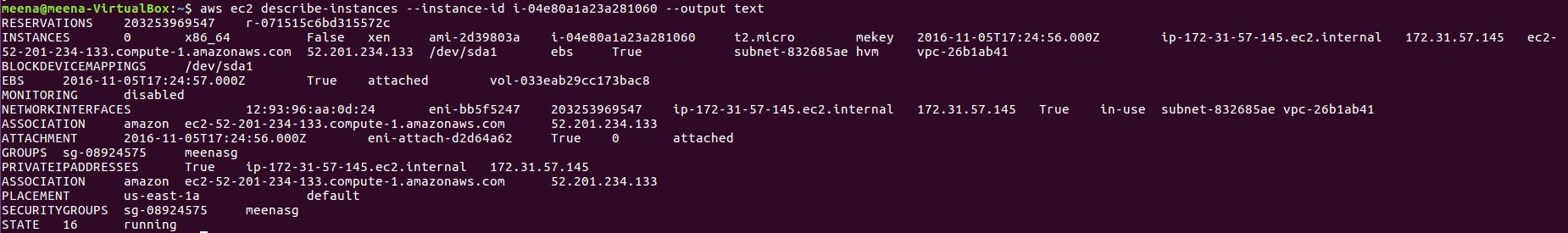
The initials steps configuring aws tool, creating a security group and a key pair has been successfully done. Below are the screenshots that shows creating an instance and login the instance with ssh.

**

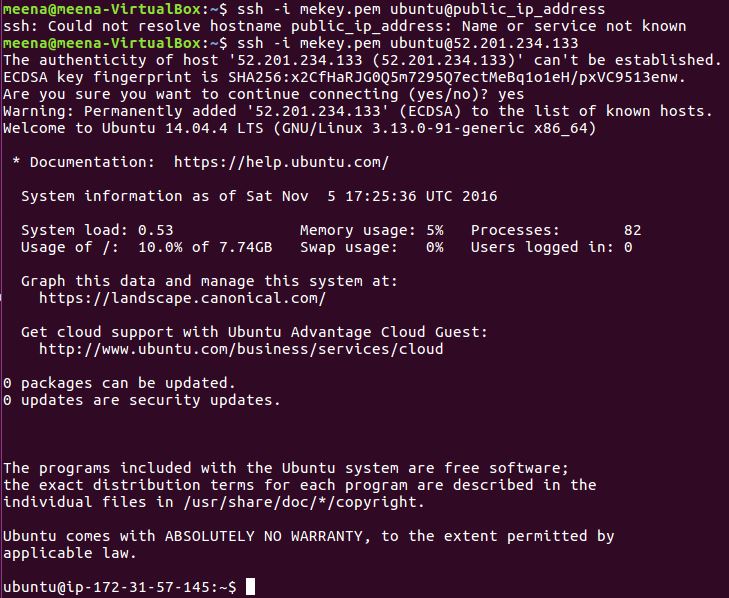
*Fig.1 Security group details*

**

*Fig. 2 Creating an instance*

**

*Fig 3. Describe instances to fetch public\_ip\_address*

**

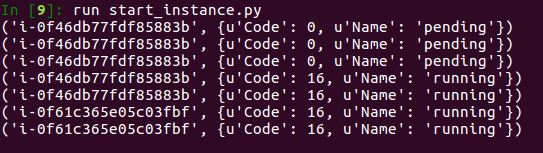
*Fig. 4 Login the instance with ssh*

**Question 1.2**Use the boto APIs to implement a python function start\_instances(num\_instances), where the parameter num\_instances is the number of instances you will be creating. This function will create a number of t2.micro instances with the AMI ami2d39803a. It should wait until the state of the instances become "running", and then return the list of instances.

The python function is given below with appropriate comments and screen shot of the output has been given.

import boto3  
import time  
  
#create ec2 resource  
ec2=boto3.resource('ec2')  
  
#function definition  
def start\_instance(num\_instances):  
#create instance  
    ins=ec2.create\_instances(ImageId='ami-2d39803a', InstanceType='t2.micro', MinCount=1, MaxCount=num\_instances, KeyName='mekey')  
    #check status  
    for i in ins:  
        while i.state['Name']=='pending':  
            #suspends execution for 5 seconds  
            time.sleep(5)  
            #instance reload  
            i.reload()  
            print([i.id](http://i.id/), i.state)  
        if i.state['Name']=='running':  
            print([i.id](http://i.id/), i.state)  
#function calling  
start\_instance(2)

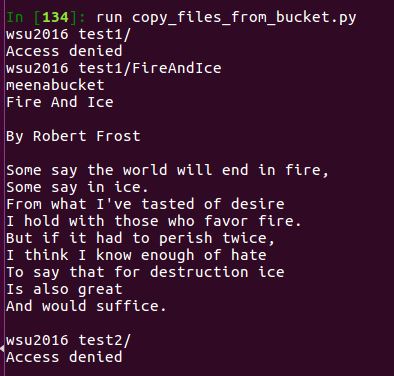
Output:



**Question 1.3** Write a python script that uses the boto APIs to find out all the files in the bucket "wsu2016", print out the contents in the files, and copy the files to your own bucket. Keep your bucket undeleted until we finish grading!

import boto3,botocore  
#get resource  
s3=boto3.resource('s3')  
#create bucket  
s3.create\_bucket(Bucket='meenabucket')  
bucket=s3.Bucket('wsu2016')  
mybucket=s3.Bucket('meenabucket')  
exists=True  
#validating bucket  
try:  
    s3.meta.client.head\_bucket(Bucket="wsu2016")  
except botocore.exceptions.ClientError as e:  
    error\_code=int(e.response['Error']['Code'])  
        if error\_code==404:  
            exists=False  
if exists==True:  
    for key in bucket.objects.all():  
        #validating files in bucket  
            try:  
                print [bucket.name](http://bucket.name/" \t "_blank),key.key  
                    obj=s3.Object(bucket\_name=[bucket.name](http://bucket.name/), key=key.key)  
                    response = obj.get()  
            #read contents of file  
                    data=response['Body'].read()  
                       print [mybucket.name](http://mybucket.name/)  
            #copy files to mybucket from wsu2016  
                    s3.Object([mybucket.name](http://mybucket.name/),key.key).copy\_from(CopySource=[bucket.name](http://bucket.name/)+'/'+key.key)  
                    print data  
            except Exception:  
                    print("Access denied")

Output:

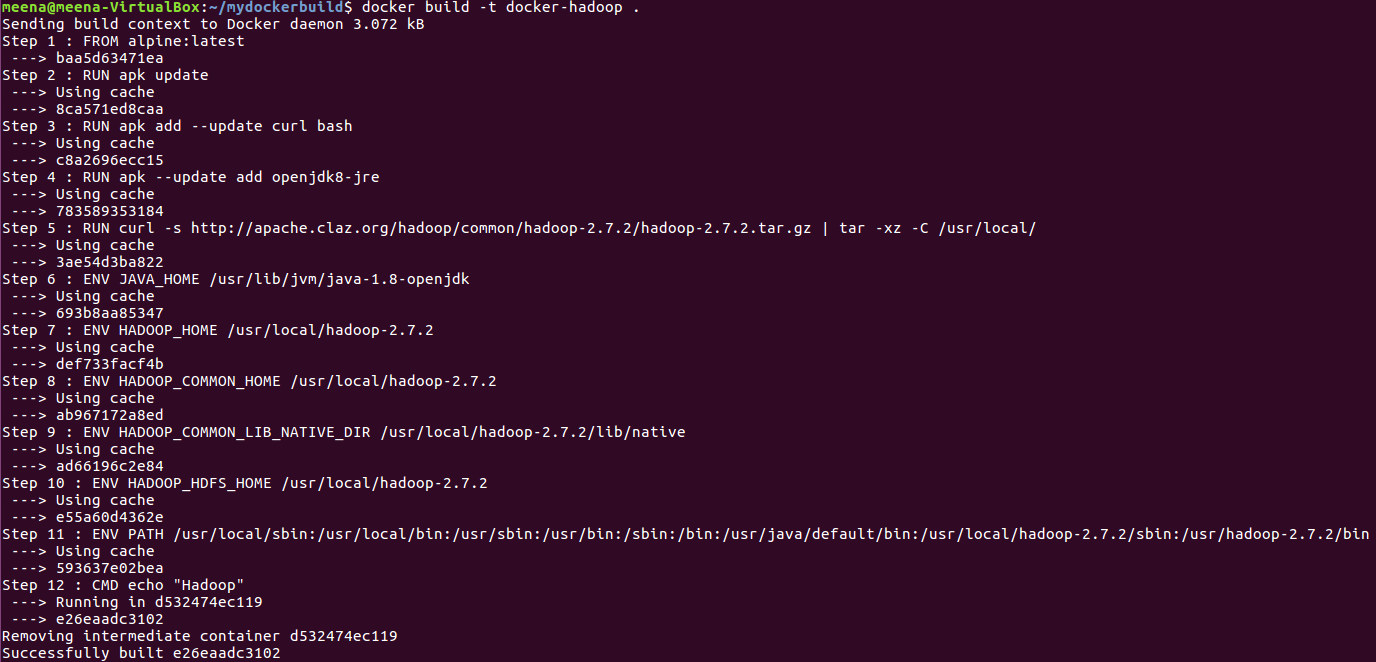


**Question 1.4** Create a Docker image that is based on Alpine linux and has the latest version of Hadoop. Post your dockerfile and your Docker hub link.

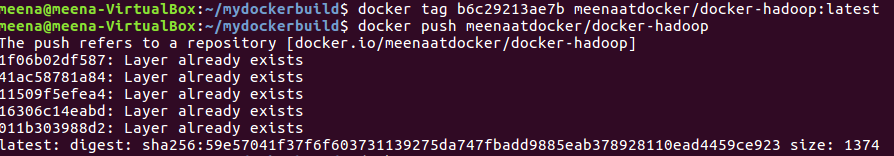
DockerFile:

#image based on alpine linux  
FROM alpine:latest  
RUN apk update  
#install curl  
RUN apk add --update curl bash  
#install java  
RUN apk --update add openjdk8-jre  
#download hadoop   
RUN curl -s <http://apache.claz.org/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz> | tar -xz -C /usr/local/  
#setting env variables  
ENV JAVA\_HOME /usr/lib/jvm/java-1.8-openjdk  
ENV HADOOP\_HOME /usr/local/hadoop-2.7.2  
ENV HADOOP\_COMMON\_HOME /usr/local/hadoop-2.7.2  
ENV HADOOP\_COMMON\_LIB\_NATIVE\_DIR /usr/local/hadoop-2.7.2/lib/native  
ENV HADOOP\_HDFS\_HOME /usr/local/hadoop-2.7.2  
ENV PATH /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/java/default/bin:/usr/local/hadoop-2.7.2/sbin:/usr/hadoop-2.7.2/bin  
CMD echo "Hadoop installed"

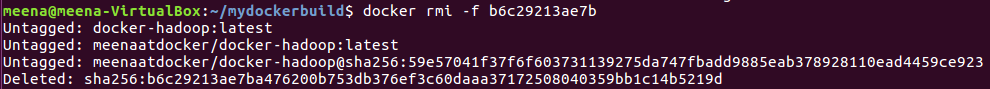
Build the docker image



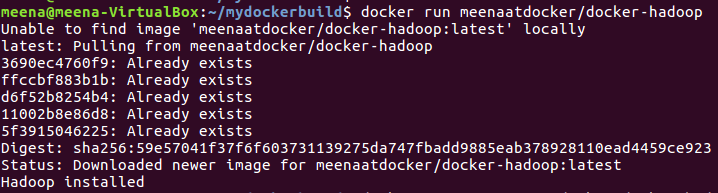
Tag docker-hadoop image as latest and push the image to the repository



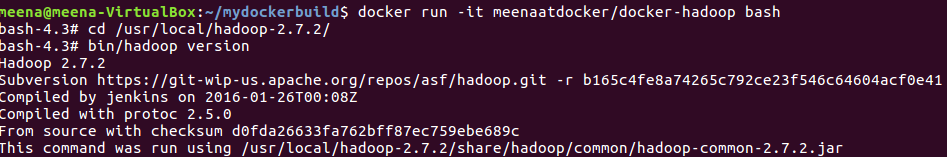
Remove the docker-hadoop and meenaatdockee/docker-hadoop image



Pull image from the repository

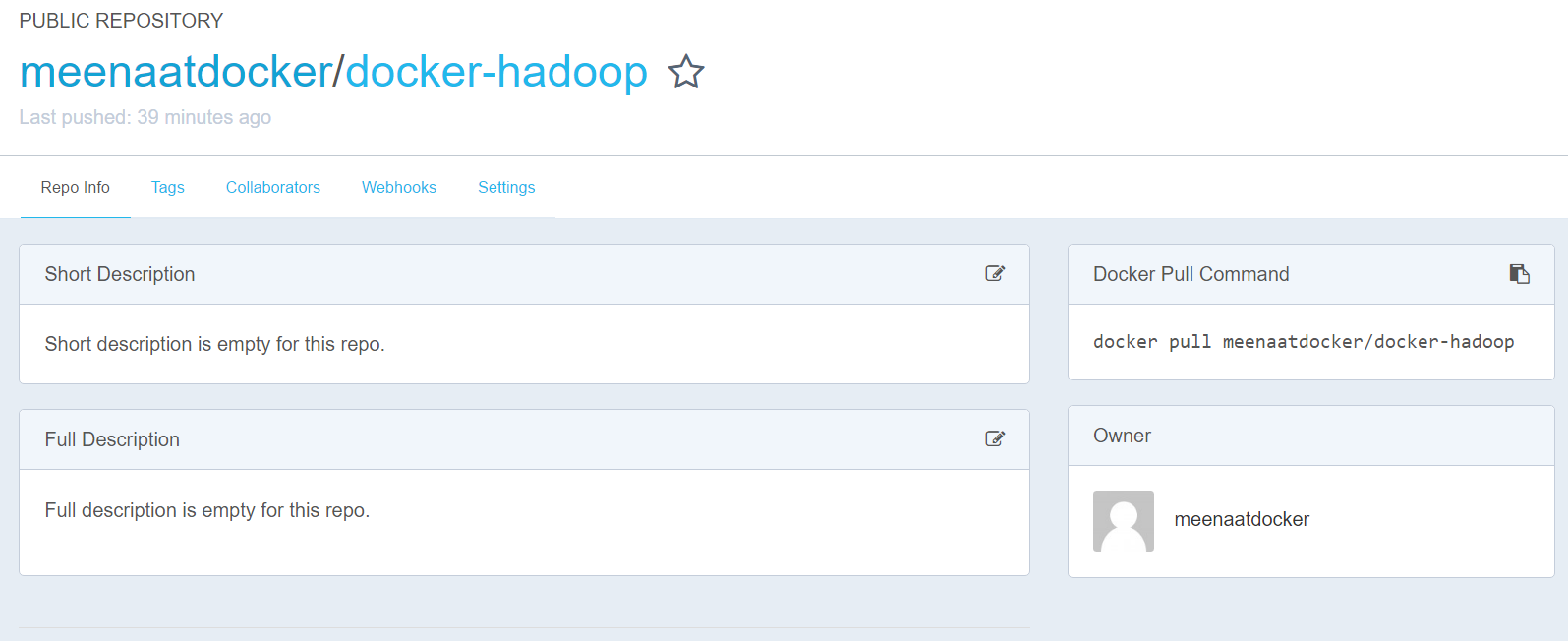


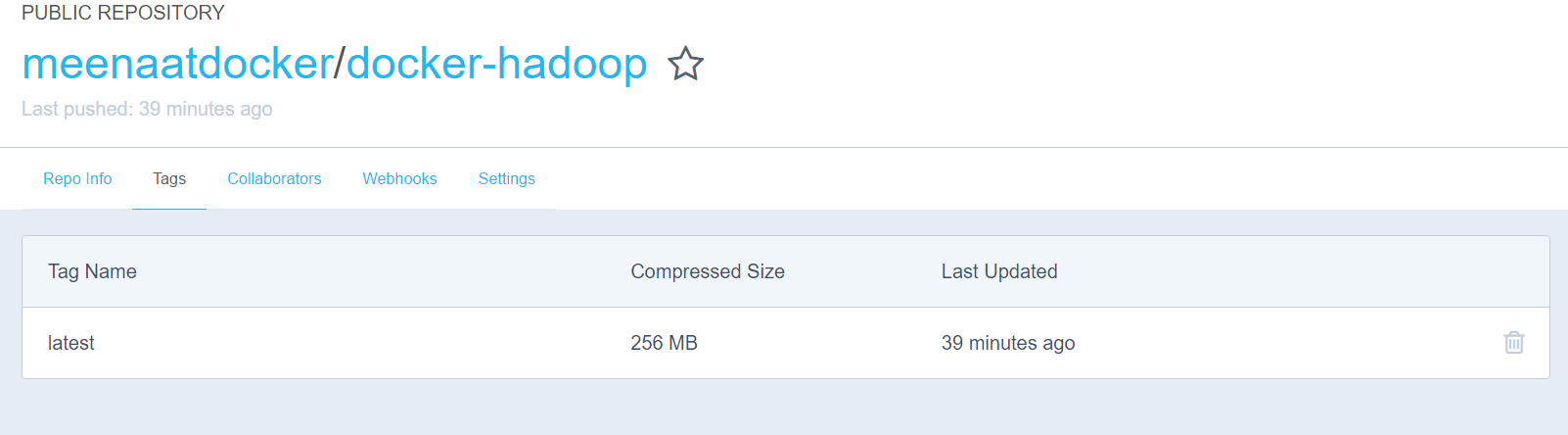
Starting a container and checking the version of Hadoop



Docker hub link:

<https://hub.docker.com/r/meenaatdocker/docker-hadoop/>





**Part 2: Monitoring VM instances and Docker containers**

**Question 2.1** In this task, you will implement a tool with Python Boto3 library and the Paramiko Python SSH library to monitor the status of the instances you created. This monitoring tool will constantly (e.g., every 5 seconds) print out the CPU usage of each instance. Note that you can execute commands in instances remotely via ssh, like

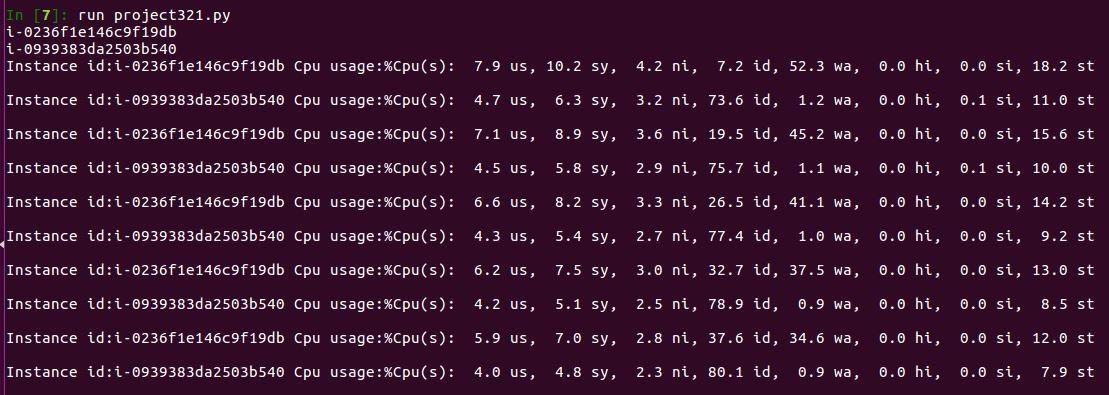
ssh ‐i your\_private\_key.pem ubuntu@EC2\_instance\_Public\_DNS "top ‐bn1 | grep Cpu

The command "top bn1 | grep Cpu" will get the the line of the command "top" output that contains Cpu information. The output of the remote command execution will be sent to you.

In your python code, you will need to create 2 instances using the EC2 functions and then in a loop every 5 seconds the command is executed remotely in the instances by using the ssh functions provided by the Paramiko library, and print out the information "instance\_ id \t Cpu usage".

import boto3  
import paramiko  
import threading  
import time  
#create ec2 resource  
ec2=boto3.resource('ec2')  
#create 2 ec2 instances MaxCount=2  
ins=ec2.create\_instances(ImageId='ami-2d39803a',InstanceType='t2.micro',MinCount=1,MaxCount=2, KeyName='mekey',SecurityGroupIds=('sg-08924575','sg-08924575'))  
#check status  
for i in ins:  
    while i.state['Name']=='pending':  
        time.sleep(5)  
        i.reload()  
    if i.state['Name']=='running':  
        print([i.id](http://i.id/))  
#create a key object  
k = paramiko.RSAKey.from\_private\_key\_file("mekey.pem")  
while True:  
    #suspend execution for 5 seconds  
    time.sleep(5)  
    for i in ins:  
        #connect to client with SSH agent  
        c=paramiko.SSHClient()  
        c.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())  
        #connect to instance  
        c.connect(hostname=i.public\_dns\_name, username='ubuntu', pkey=k)  
        #execute command in instance  
        stdin, stdout, stderr = c.exec\_command('top -bn1|grep Cpu')  
        print("Instance id:" +[i.id](http://i.id/)+"\t"+"Cpu usage:" +stdout.read())  
        c.close()

Output:



**Question 2.2** Extend your tool to monitor Docker containers in VM instances. Assume you have started 2 EC2 instances. For simplicity, for each instance, you can manually install Docker and start 2 Docker container daemons as follows. Note that the d option is used to run the container as a daemon.

docker run ‐d ‐t alpine sh

You can retrieve the container IDs (similar to VM instance IDs) using the following command.

docker ps | grep alpine

To execute a command in the container, for instance, getting the CPU usage, you can use

docker exec container\_ID top ‐bn1 | grep CPU

Now you implement your python program to monitor the CPU usage of each container in each instance every 5 seconds and print out "instance\_ID \t container\_ID \t CPU usage".

import boto3  
import paramiko  
import time  
#create ec2 resource  
ec2=boto3.resource('ec2')  
#create a key object  
k = paramiko.RSAKey.from\_private\_key\_file("mekey.pem")  
while True:  
    for i in ec2.instances.all():  
        time.sleep(5)  
        c=paramiko.SSHClient()  
        c.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())  
        #connect to instance  
        c.connect(hostname=i.public\_dns\_name, username='ubuntu', pkey=k)  
        stdin, stdout, stderr = c.exec\_command('docker ps | grep alpine')  
        for j in stdout.readlines():  
            #getting container id  
            a=j.strip().split()  
            #saving the command to be executed in a variable  
            command='docker exec ' + a[0] + ' top -bn1 | grep CPU'  
            #command execution  
            stdin, stdout, stderr = c.exec\_command(command)  
            print("Instance id:" +[i.id](http://i.id/)+"\t"+"Container id:" +a[0]+"\t"+"Cpu usage:" +stdout.read())  
        c.close()

Output:

