

## 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

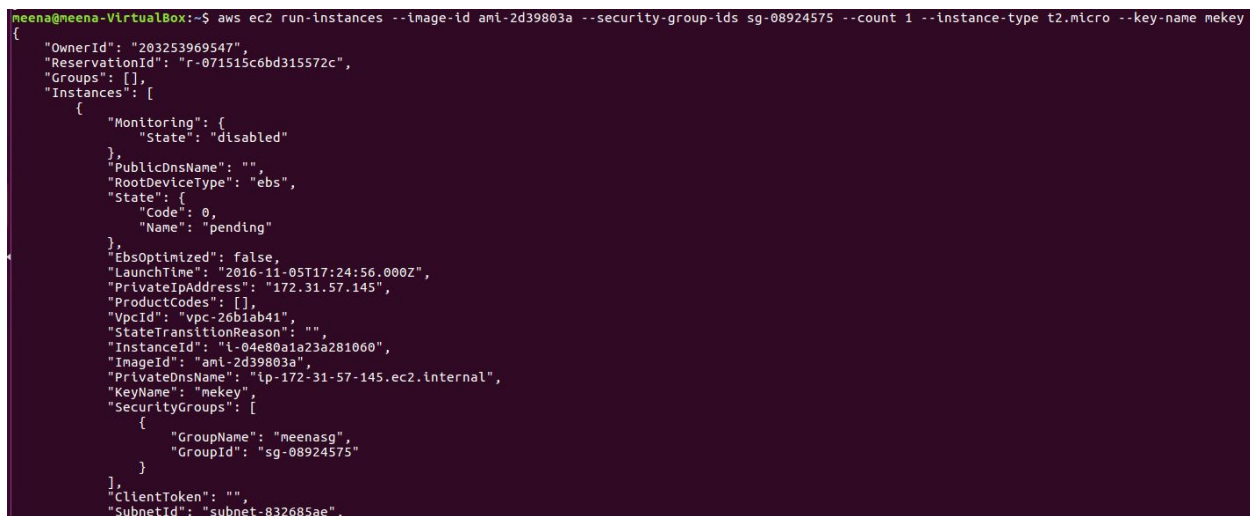


Fig. 2 Creating an instance

```
meena@meena-VirtualBox:~$ aws ec2 describe-instances --instance-id i-04e80a1a23a281060 --output text
RESERVATIONS 203253969547 r-071515c6bd315572c
INSTANCES 0 x86_64 False xen ami-2d39803a i-04e80a1a23a281060 t2.micro mekey 2016-11-05T17:24:56.000Z ip-172-31-57-145.ec2.internal 172.31.57.145 ec2-
52-201-234-133.compute-1.amazonaws.com 52.201.234.133 /dev/sda1 ebs True subnet-832685ae hvm vpc-26b1ab41
BLOCKDEVICESMAPPINGS /dev/sda1
EBS 2016-11-05T17:24:57.000Z True attached vol-033eab29cc173bac8
MONITORING disabled
NETWORKINTERFACES 12:93:96:aa:0d:24 eni-bb5f5247 203253969547 ip-172-31-57-145.ec2.internal 172.31.57.145 True in-use subnet-832685ae vpc-26b1ab41
ASSOCIATION amazon ec2-52-201-234-133.compute-1.amazonaws.com 52.201.234.133
ATTACHMENT 2016-11-05T17:24:56.000Z eni-attach-d2d64a62 True 0 attached
GROUPS sg-08924575 meenasg
PRIVATEIPADDRESSES True ip-172-31-57-145.ec2.internal 172.31.57.145
ASSOCIATION amazon ec2-52-201-234-133.compute-1.amazonaws.com 52.201.234.133
PLACEMENT us-east-1a default
SECURITYGROUPS sg-08924575 meenasg
STATE 16 running
```

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

```
meena@meena-VirtualBox:~$ ssh -i mekey.pem ubuntu@public_ip_address
ssh: Could not resolve hostname public_ip_address: Name or service not known
meena@meena-VirtualBox:~$ ssh -i mekey.pem ubuntu@52.201.234.133
The authenticity of host '52.201.234.133 (52.201.234.133)' can't be established.
ECDSA key fingerprint is SHA256:x2CfHaRJG0Q5m7295Q7ectMeBq1o1eH/pxVC9513enw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '52.201.234.133' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-91-generic x86_64)

* Documentation:  https://help.ubuntu.com/

System information as of Sat Nov  5 17:25:36 UTC 2016

System load: 0.53           Memory usage: 5%    Processes:      82
Usage of /: 10.0% of 7.74GB  Swap usage:  0%    Users logged in: 0

Graph this data and manage this system at:
https://landscape.canonical.com/

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

ubuntu@ip-172-31-57-145:~$
```

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.

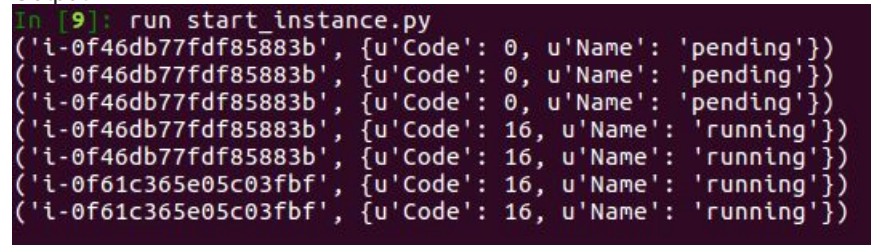
The python function is given below with appropriate comments and screen shot of the output has been given.

```
import boto3
import time

#create ec2 resource
ec2=boto3.resource('ec2')

#function definition
def start_instance(num_instances):
    #create instance
    ins=ec2.create_instances(ImgId='ami-2d39803a', InstanceType='t2.micro', MinCount=1,
MaxCount=num_instances, KeyName='mekey')
    #check status
    for i in ins:
        while i.state['Name']!='running':
            #suspends execution for 5 seconds
            time.sleep(5)
            #instance reload
            i.reload()
            print(i.id, i.state)
        if i.state['Name']=='running':
            print(i.id, i.state)
    #function calling
    start_instance(2)
```

Output:



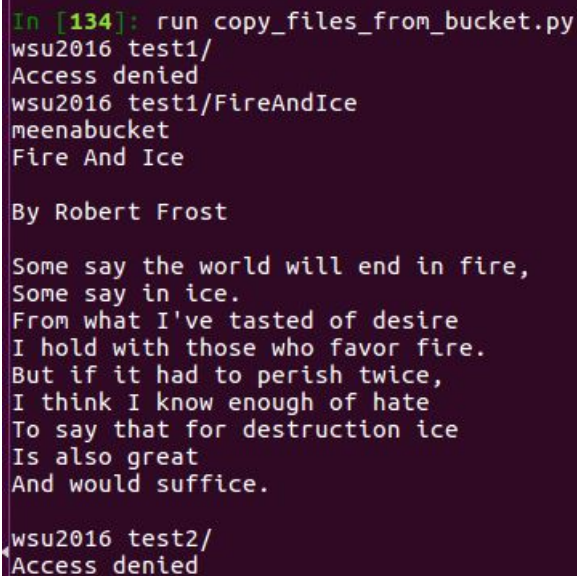
```
In [9]: run start_instance.py
('i-0f46db77fdf85883b', {'u'Code': 0, u'Name': 'pending'})
('i-0f46db77fdf85883b', {'u'Code': 0, u'Name': 'pending'})
('i-0f46db77fdf85883b', {'u'Code': 0, u'Name': 'pending'})
('i-0f46db77fdf85883b', {'u'Code': 16, u'Name': 'running'})
('i-0f46db77fdf85883b', {'u'Code': 16, u'Name': 'running'})
('i-0f61c365e05c03fbf', {'u'Code': 16, u'Name': 'running'})
('i-0f61c365e05c03fbf', {'u'Code': 16, u'Name': 'running'})
```

**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,key.key
            obj=s3.Object(bucket_name=bucket.name, key=key.key)
            response = obj.get()
            #read contents of file
            data=response['Body'].read()
            print mybucket.name
            #copy files to mybucket from wsu2016
            s3.Object(mybucket.name,key.key).copy_from(CopySource=bucket.name+'/'+key.key)
            print data
        except Exception:
            print("Access denied")
```

Output:



```
In [134]: run copy_files_from_bucket.py
wsu2016 test1/
Access denied
wsu2016 test1/FireAndIce
meenabucket
Fire And Ice

By Robert Frost

Some say the world will end in fire,
Some say in ice.
From what I've tasted of desire
I hold with those who favor fire.
But if it had to perish twice,
I think I know enough of hate
To say that for destruction ice
Is also great
And would suffice.

wsu2016 test2/
Access denied
```

**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

```
meena@meena-VirtualBox:~/mydockerbuild$ docker build -t docker-hadoop .
Sending build context to Docker daemon 3.072 kB
Step 1 : FROM alpine:latest
--> baa5d63471ea
Step 2 : RUN apk update
--> Using cache
--> 8ca571ed8caa
Step 3 : RUN apk add --update curl bash
--> Using cache
--> c8a2696ecc15
Step 4 : RUN apk --update add openjdk8-jre
--> Using cache
--> 783589353184
Step 5 : RUN curl -s http://apache.claz.org/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz | tar -xz -C /usr/local/
--> Using cache
--> 3ae54d3ba822
Step 6 : ENV JAVA_HOME /usr/lib/jvm/java-1.8-openjdk
--> Using cache
--> 693b8aa85347
Step 7 : ENV HADOOP_HOME /usr/local/hadoop-2.7.2
--> Using cache
--> def733facf4b
Step 8 : ENV HADOOP_COMMON_HOME /usr/local/hadoop-2.7.2
--> Using cache
--> ab967172a8ed
Step 9 : ENV HADOOP_COMMON_LIB_NATIVE_DIR /usr/local/hadoop-2.7.2/lib/native
--> Using cache
--> ad66196c2e84
Step 10 : ENV HADOOP_HDFS_HOME /usr/local/hadoop-2.7.2
--> Using cache
--> e55a60d4362e
Step 11 : 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
--> Using cache
--> 593637e02bea
Step 12 : CMD echo "Hadoop"
--> Running in d532474ec119
--> e26eaadc3102
Removing intermediate container d532474ec119
Successfully built e26eaadc3102
```

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

```
meena@meena-VirtualBox:~/mydockerbuild$ docker tag b6c29213ae7b meenaatdocker/docker-hadoop:latest
meena@meena-VirtualBox:~/mydockerbuild$ docker push meenaatdocker/docker-hadoop
The push refers to a repository [docker.io/meenaatdocker/docker-hadoop]
1f06b02df587: Layer already exists
41ac58781a84: Layer already exists
11509f5efea4: Layer already exists
16306c14eabd: Layer already exists
911b303988d2: Layer already exists
latest: digest: sha256:59e57041f37f6f603731139275da747fbadd9885eab378928110ead4459ce923 size: 1374
```

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

```
meena@meena-VirtualBox:~/mydockerbuild$ docker rmi -f b6c29213ae7b
Untagged: docker-hadoop:latest
Untagged: meenaatdocker/docker-hadoop:latest
Untagged: meenaatdocker/docker-hadoop@sha256:59e57041f37f6f603731139275da747fbadd9885eab378928110ead4459ce923
Deleted: sha256:b6c29213ae7ba476200b753db376ef3c60daaa37172508040359bb1c14b5219d
```



Pull image from the repository

```
meena@meena-VirtualBox:~/mydockerbuild$ docker run meenaatdocker/docker-hadoop
Unable to find image 'meenaatdocker/docker-hadoop:latest' locally
latest: Pulling from meenaatdocker/docker-hadoop
3690ec4760f9: Already exists
ffccbf883b1b: Already exists
d6f52b8254b4: Already exists
11002b8e86d8: Already exists
5f3915046225: Already exists
Digest: sha256:59e57041f37f6f603731139275da747fbadd9885eab378928110ead4459ce923
Status: Downloaded newer image for meenaatdocker/docker-hadoop:latest
Hadoop installed
```

Starting a container and checking the version of Hadoop

```
meena@meena-VirtualBox:~/mydockerbuild$ docker run -it meenaatdocker/docker-hadoop bash
bash-4.3# cd /usr/local/hadoop-2.7.2/
bash-4.3# bin/hadoop version
Hadoop 2.7.2
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r b165c4fe8a74265c792ce23f546c64604acf0e41
Compiled by jenkins on 2016-01-26T00:08Z
Compiled with protoc 2.5.0
From source with checksum d0fda26633fa762bff87ec759ebe689c
This command was run using /usr/local/hadoop-2.7.2/share/hadoop/common/hadoop-common-2.7.2.jar
```

Docker hub link:

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

PUBLIC REPOSITORY

meenaatdocker/docker-hadoop ☆

Last pushed: 39 minutes ago

Repo Info Tags Collaborators Webhooks Settings

Short Description

Short description is empty for this repo.

Full Description

Full description is empty for this repo.

Docker Pull Command

docker pull meenaatdocker/docker-hadoop

Owner




meenaatdocker

PUBLIC REPOSITORY

meenaatdocker/docker-hadoop ☆

Last pushed: 39 minutes ago

[Repo Info](#)
[Tags](#)
[Collaborators](#)
[Webhooks](#)
[Settings](#)

Tag Name	Compressed Size	Last Updated	
latest	256 MB	39 minutes ago	

## 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(Imgeld='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)
#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 + "\t" + "Cpu usage:" + stdout.read())
c.close()

```

Output:

```

In [7]: run project321.py
i-0236f1e146c9f19db
i-0939383da2503b540
Instance id:i-0236f1e146c9f19db Cpu usage:%Cpu(s):  7.9 us, 10.2 sy,  4.2 ni,  7.2 id, 52.3 wa,  0.0 hi,  0.0 si, 18.2 st
Instance id:i-0939383da2503b540 Cpu usage:%Cpu(s):  4.7 us,  6.3 sy,  3.2 ni, 73.6 id,  1.2 wa,  0.0 hi,  0.1 si, 11.0 st
Instance id:i-0236f1e146c9f19db Cpu usage:%Cpu(s):  7.1 us,  8.9 sy,  3.6 ni, 19.5 id, 45.2 wa,  0.0 hi,  0.0 si, 15.6 st
Instance id:i-0939383da2503b540 Cpu usage:%Cpu(s):  4.5 us,  5.8 sy,  2.9 ni, 75.7 id,  1.1 wa,  0.0 hi,  0.1 si, 10.0 st
Instance id:i-0236f1e146c9f19db Cpu usage:%Cpu(s):  6.6 us,  8.2 sy,  3.3 ni, 26.5 id, 41.1 wa,  0.0 hi,  0.0 si, 14.2 st
Instance id:i-0939383da2503b540 Cpu usage:%Cpu(s):  4.3 us,  5.4 sy,  2.7 ni, 77.4 id,  1.0 wa,  0.0 hi,  0.0 si,  9.2 st
Instance id:i-0236f1e146c9f19db Cpu usage:%Cpu(s):  6.2 us,  7.5 sy,  3.0 ni, 32.7 id, 37.5 wa,  0.0 hi,  0.0 si, 13.0 st
Instance id:i-0939383da2503b540 Cpu usage:%Cpu(s):  4.2 us,  5.1 sy,  2.5 ni, 78.9 id,  0.9 wa,  0.0 hi,  0.0 si,  8.5 st
Instance id:i-0236f1e146c9f19db Cpu usage:%Cpu(s):  5.9 us,  7.0 sy,  2.8 ni, 37.6 id, 34.6 wa,  0.0 hi,  0.0 si, 12.0 st
Instance id:i-0939383da2503b540 Cpu usage:%Cpu(s):  4.0 us,  4.8 sy,  2.3 ni, 80.1 id,  0.9 wa,  0.0 hi,  0.0 si,  7.9 st

```

**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+"\\t"+"Container id:" +a[0]+"\\t"+"Cpu usage:" +stdout.read())
        c.close()
```

Output:

```
In [115]: run project322.py
Instance id:i-04e27718ac8e53a6f Container id:3073d7c0b864 Cpu usage:CPU: 0% usr 0% sys 0% nic 100% idle 0% io 0% irq 0% irq
PID PPID USER STAT VSZ %VSZ CPU %CPU COMMAND
Instance id:i-04e27718ac8e53a6f Container id:1efd154775cf Cpu usage:CPU: 0% usr 0% sys 0% nic 100% idle 0% io 0% irq 0% irq
PID PPID USER STAT VSZ %VSZ CPU %CPU COMMAND
Instance id:i-0c3ae1465d2fe2c3d Container id:2e1a3a6f19b3 Cpu usage:CPU: 0% usr 0% sys 0% nic 100% idle 0% io 0% irq 0% irq
PID PPID USER STAT VSZ %VSZ CPU %CPU COMMAND
Instance id:i-0c3ae1465d2fe2c3d Container id:de67a7a96e0b Cpu usage:CPU: 0% usr 0% sys 0% nic 100% idle 0% io 0% irq 0% irq
PID PPID USER STAT VSZ %VSZ CPU %CPU COMMAND
Instance id:i-04e27718ac8e53a6f Container id:3073d7c0b864 Cpu usage:CPU: 0% usr 0% sys 0% nic 100% idle 0% io 0% irq 0% irq
PID PPID USER STAT VSZ %VSZ CPU %CPU COMMAND
Instance id:i-04e27718ac8e53a6f Container id:1efd154775cf Cpu usage:CPU: 0% usr 0% sys 0% nic 100% idle 0% io 0% irq 0% irq
PID PPID USER STAT VSZ %VSZ CPU %CPU COMMAND
Instance id:i-0c3ae1465d2fe2c3d Container id:2e1a3a6f19b3 Cpu usage:CPU: 0% usr 9% sys 0% nic 90% idle 0% io 0% irq 0% irq
PID PPID USER STAT VSZ %VSZ CPU %CPU COMMAND
Instance id:i-0c3ae1465d2fe2c3d Container id:de67a7a96e0b Cpu usage:CPU: 0% usr 0% sys 0% nic 100% idle 0% io 0% irq 0% irq
PID PPID USER STAT VSZ %VSZ CPU %CPU COMMAND
Instance id:i-04e27718ac8e53a6f Container id:3073d7c0b864 Cpu usage:CPU: 0% usr 0% sys 0% nic 100% idle 0% io 0% irq 0% irq
PID PPID USER STAT VSZ %VSZ CPU %CPU COMMAND
Instance id:i-04e27718ac8e53a6f Container id:1efd154775cf Cpu usage:CPU: 0% usr 0% sys 0% nic 100% idle 0% io 0% irq 0% irq
PID PPID USER STAT VSZ %VSZ CPU %CPU COMMAND
```