DEADLOCK AVOIDANCE THROUGH EFFICIENT LOAD BALANCING IN CLOUD ENVIRONMENT

By

Jananya Sivakumar (19BCE1602)

Tejas V (19BCE1607)

Sandhya S (19BCE1614)

Shravan Ganapathiraman (19BCE1703)

M.A. RAM(19BCE1162)

Submitted to

Prof.Subbulakshmi P

Overview:

- Using cloud computing implementation of detection and avoidance algorithm of deadlock to prevent cloud disaster in cloud computing since many users use the same platform concurrently so there Is a huge demand for resources which sometimes cause deadlock conditions.
- Deadlock should be minimized to get better multiprocessing efficiency. The aim of this paper is therefore to analyze different simulation tools for various disaster recovery in Cloud computing.
- This examination empowers one to appreciate the selection of instruments for their applications. This learning further gives data about parameters that are bolstered by the devices and thus opens up research roads towards those parameters which are not yet esteemed in the devices.
- The condition of- specialty of the innovation centers around information handling to manage enormous measures of information.
- Distributed computing is a rising innovation, which empathizes one to achieve the previously mentioned goal, driving towards improved business execution

OBJECTIVES:

- The objective of this review paper is to summarize the powerful data backup recovery techniques that are used in the cloud computing domain.
- Further, this paper likewise furnishes the foreseen outcomes with the execution of the proposed calculation.
- The halt evasion upgrades the quantity of occupations to be adjusted by cloud specialist co-op and along these lines improving working execution and the matter of the cloud specialist organization.

DESCRIPTION:

- Deadlock is defined as the permanent blocking of a set of processes that compete for system resources, including database records or communication lines.
- Unlike other problems in multiprogramming systems, there is no efficient solution to the deadlock problem in the general case.
- The main applicability of deadlock avoidance is that it is not necessary to preempt and roll back mechanisms, as in deadlock mechanisms.
- Also one of the other applications of deadlock avoidance is decreased restriction then deadlock prevention which makes it easier to implement it.
- The approach to application of deadlock avoidance is to manipulate to find at least one safe path but knowing that future resource requirements must be known and processes might be waiting for a long time.
- One of the applications of deadlock avoidance is to allocate all required resources to the process before the start of its execution, this way hold and wait conditions are eliminated but it will lead to low device utilization.

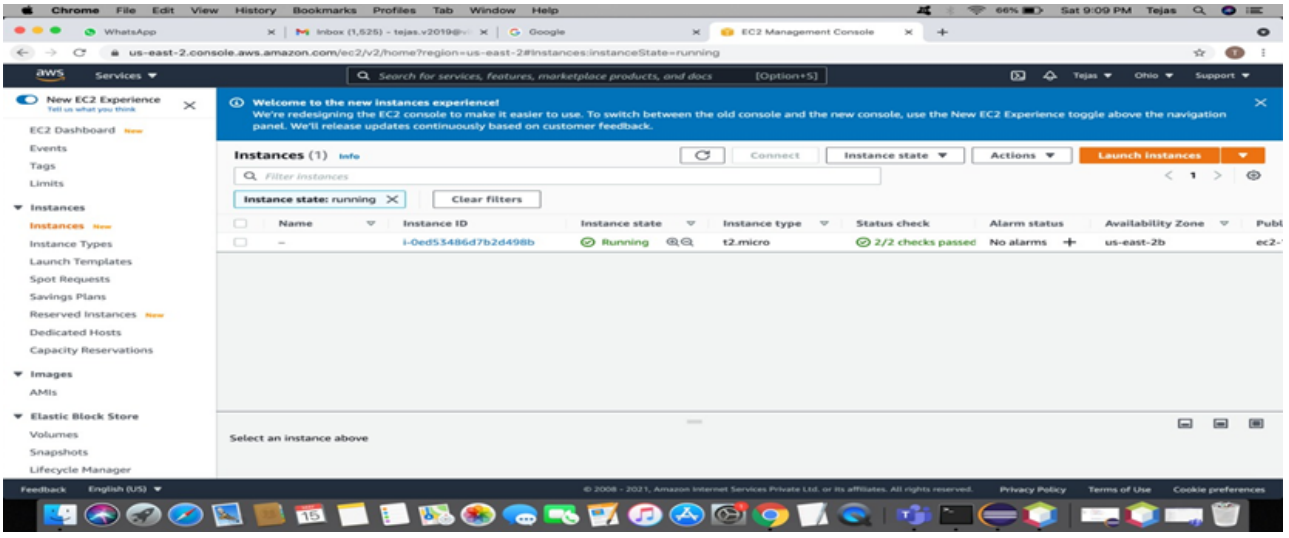
MOTIVATION:

- The performance of this model is influenced by various factors like Scalability, Reliability, Security, Disaster Management .
- The advancement in the Cloud computing model has led to an increase in demand for its services.
- An efficient and effective performance of this model is essential to have better business returns.
- The performance of this model is influenced by various factors like Scalability, Reliability, Security, Disaster Management etc.
- Disaster is one of the factors which deteriorate Cloud performance.

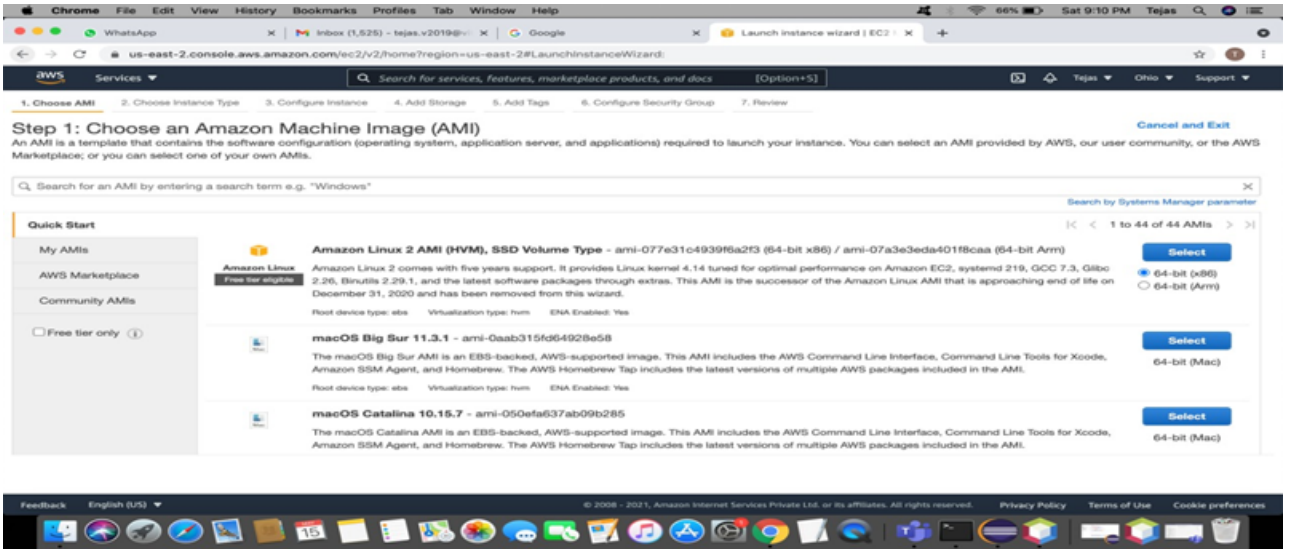
- Though it is an unpredictable event it can be prevented if the system is designed with efficient Scheduling, Resource allocation, Load Balancing etc. Disaster management is a challenging task, it provides a platform for a Cloud Service provider to exhibit the best service.

Implementation details:

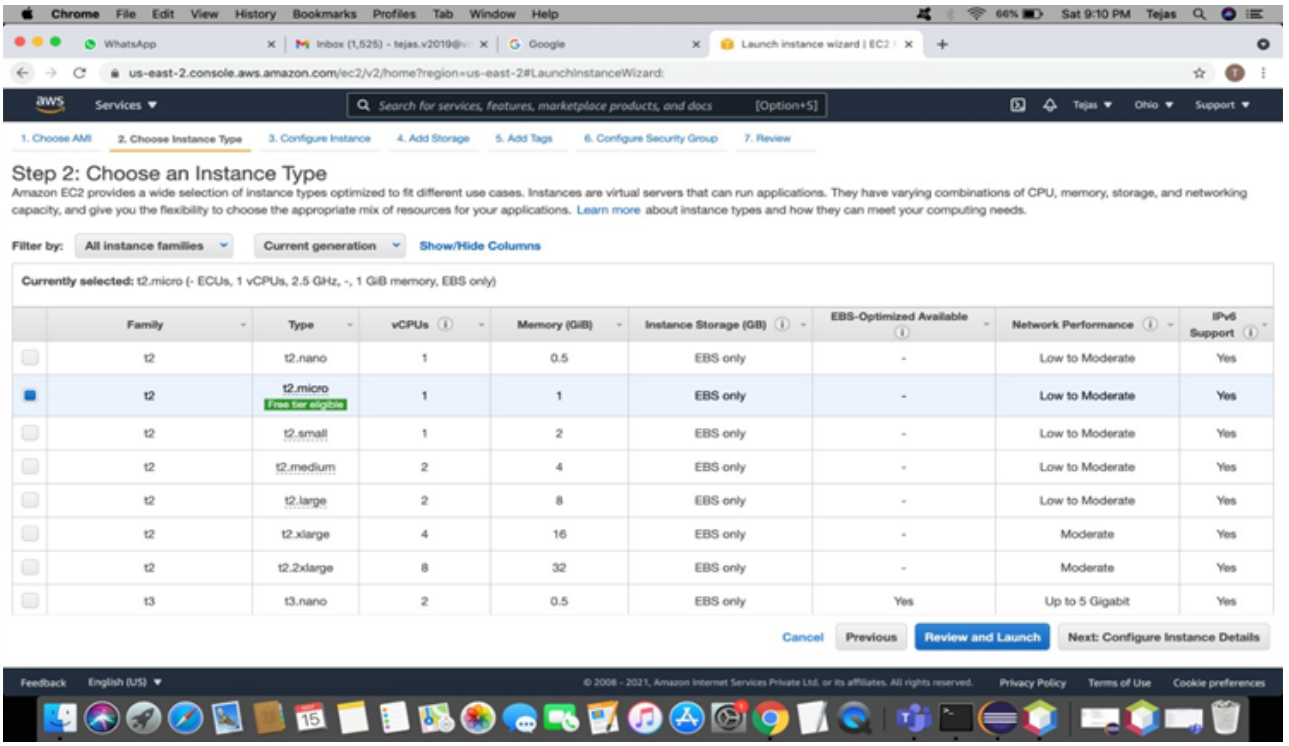
How to Make Amazon EC2 Instance



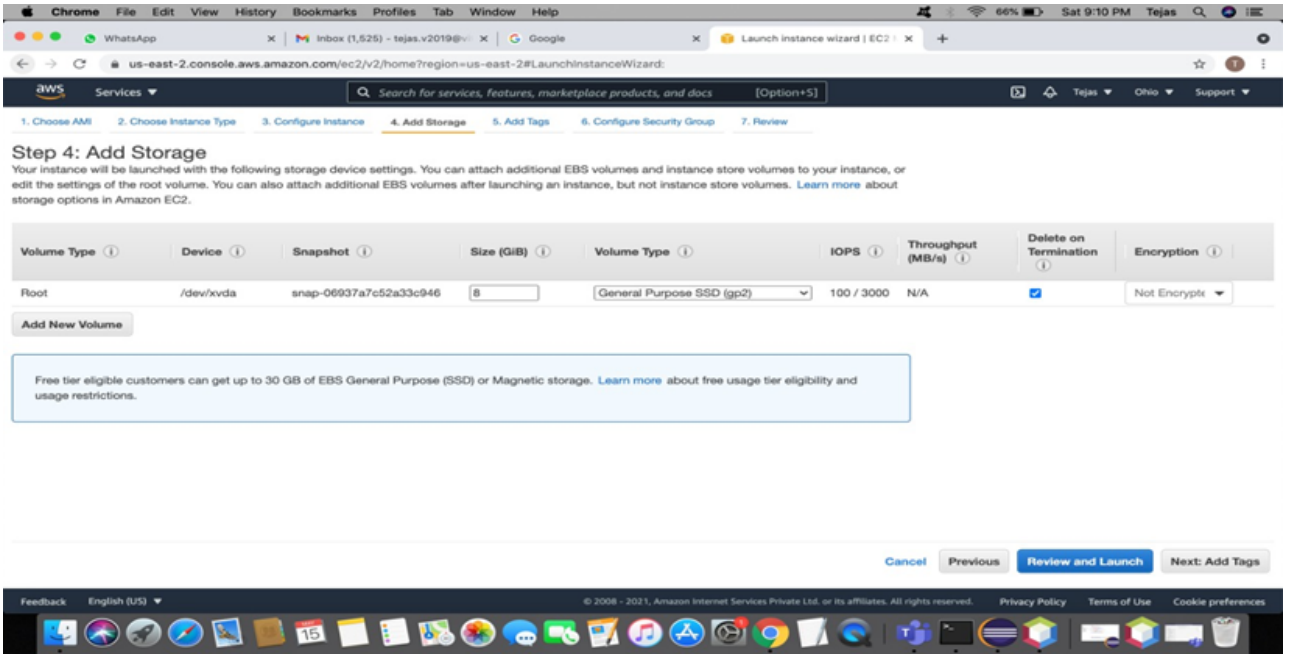
Log in to AWS and go to services where we have to go to launch instances



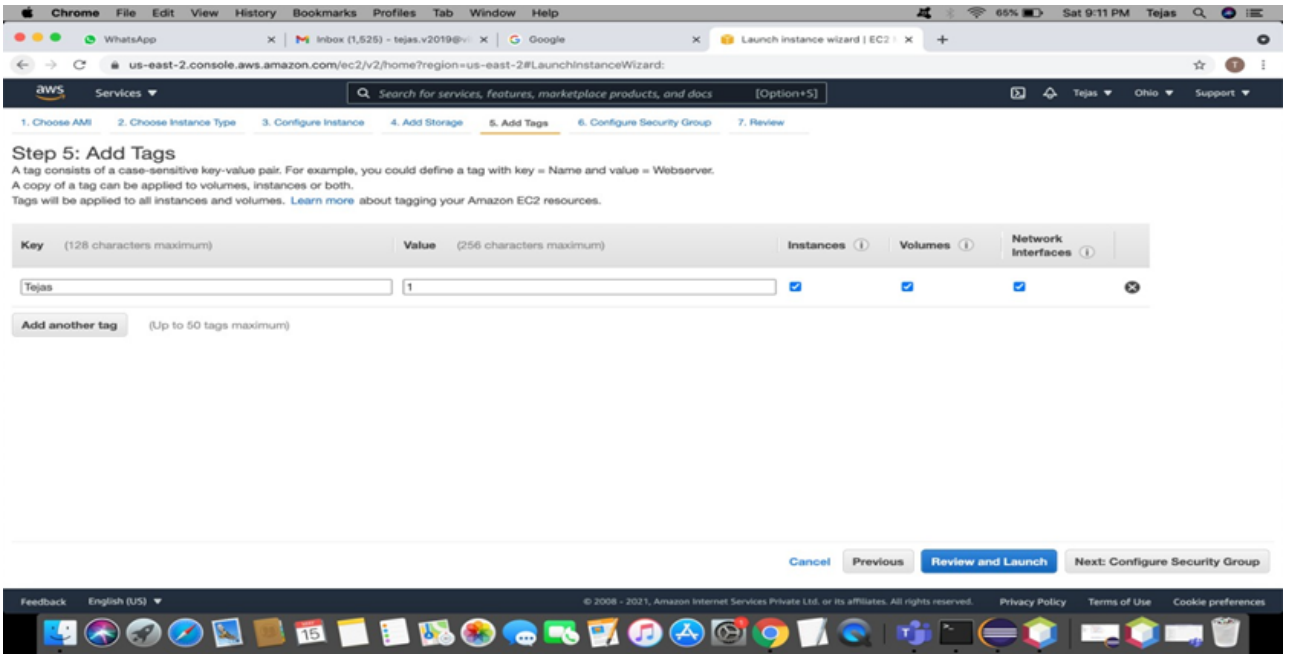
Choose amazon linux 2 AMI



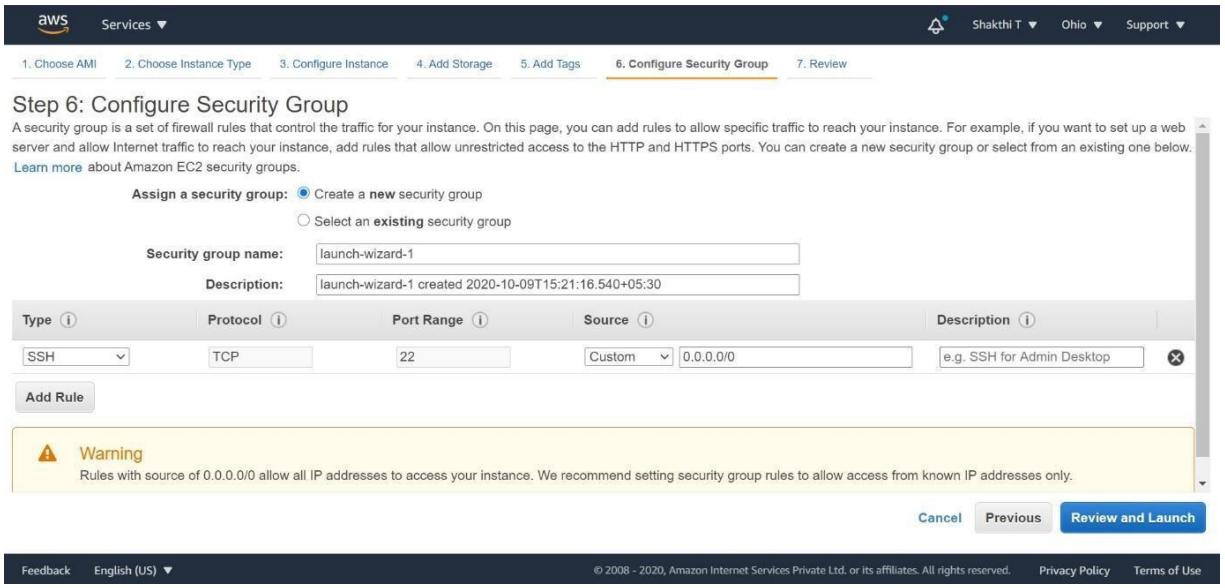
Configure instance details



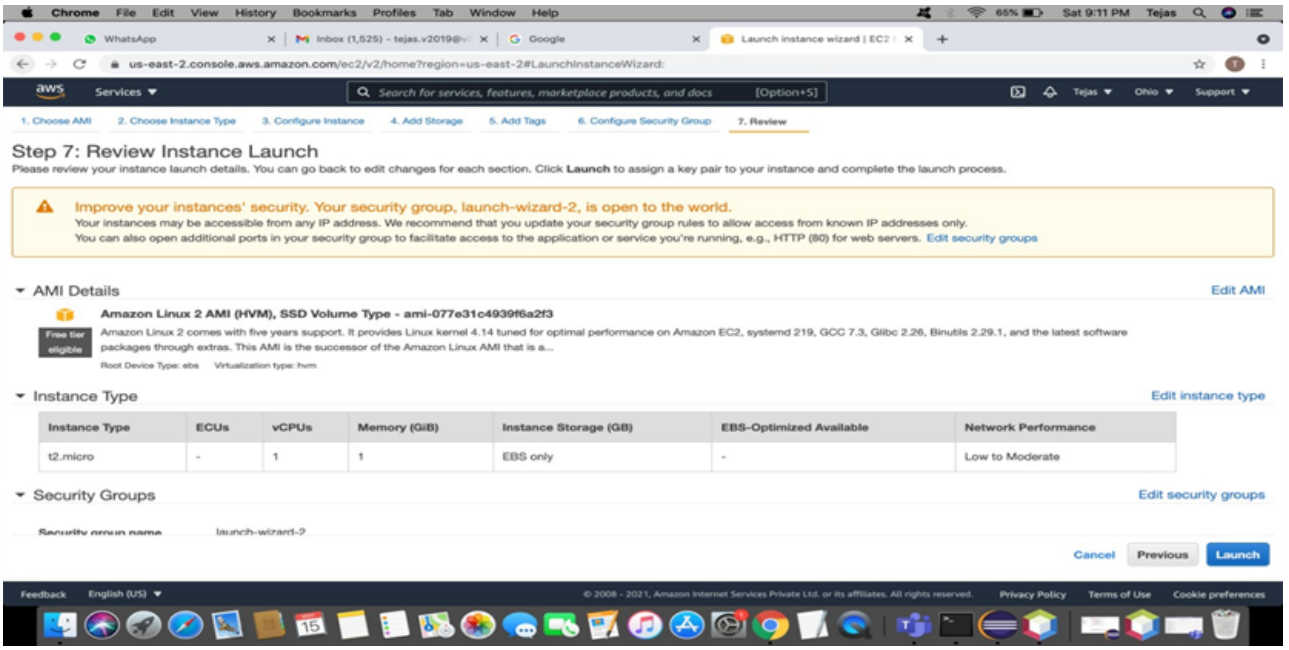
Add storage of your instance according to your need



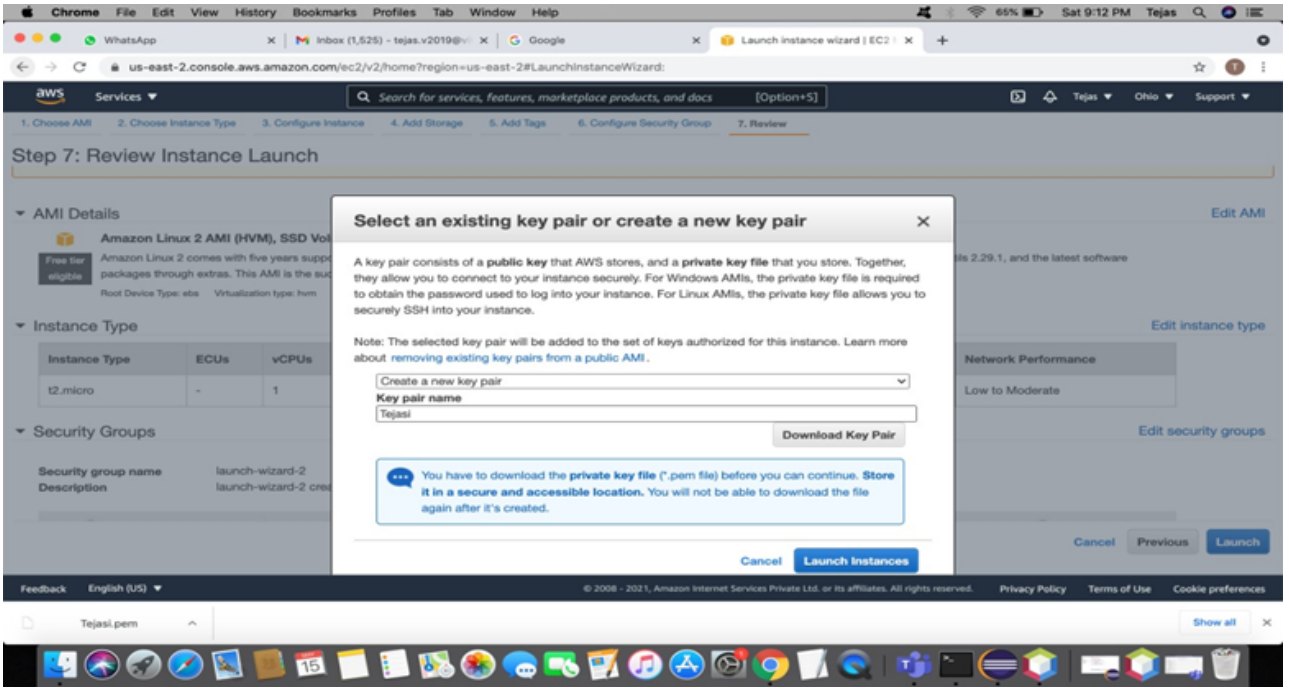
Add tags to make your instance descriptive



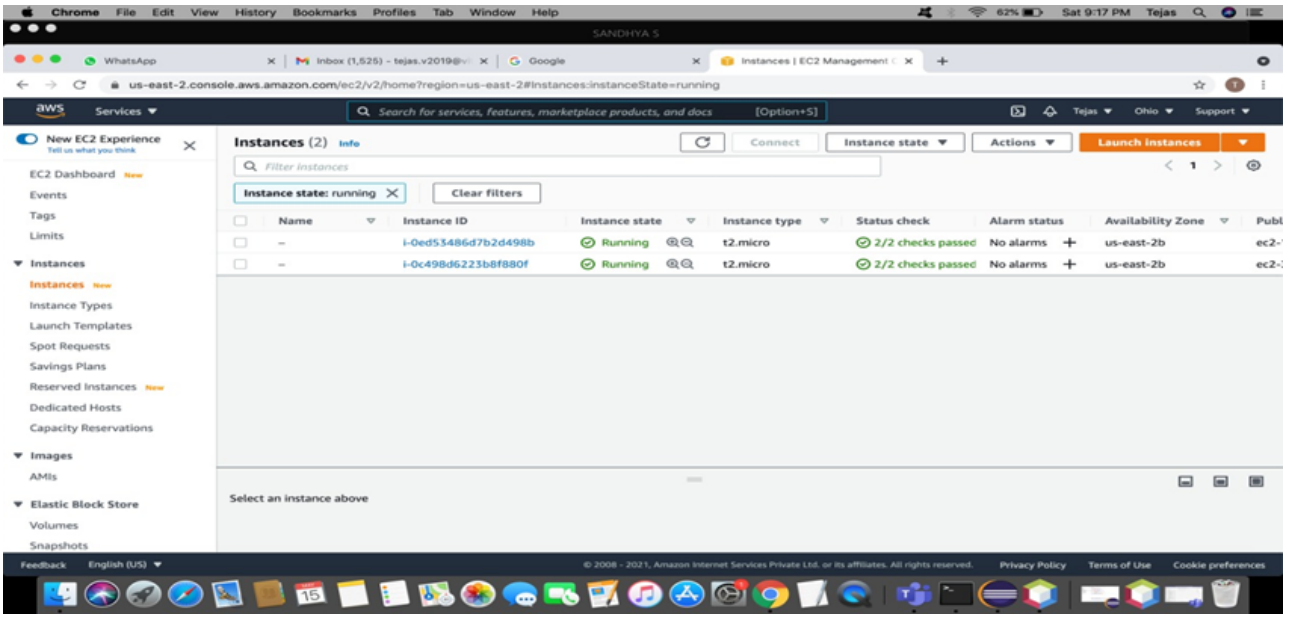
Confirm security group to allow public or private access to your instance



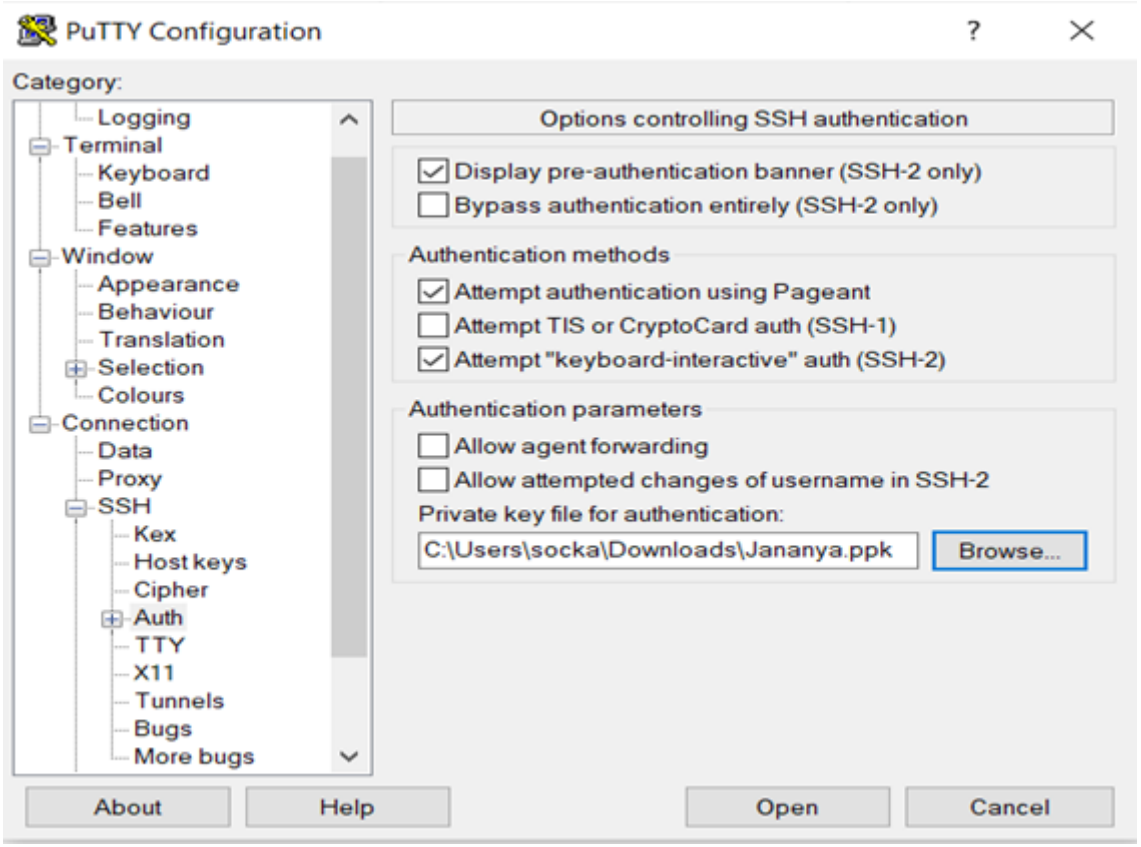
Review of the instance



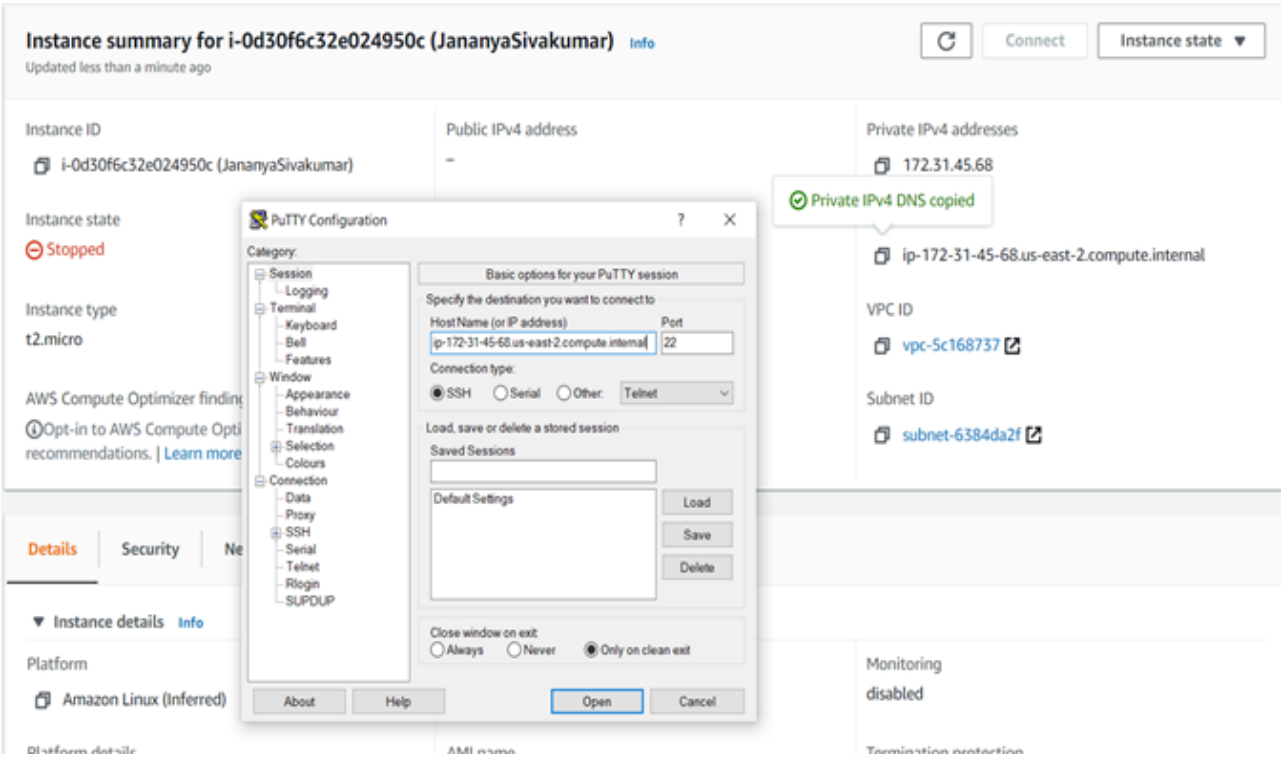
Create a key pair, download it and then convert it to ppk form using putty



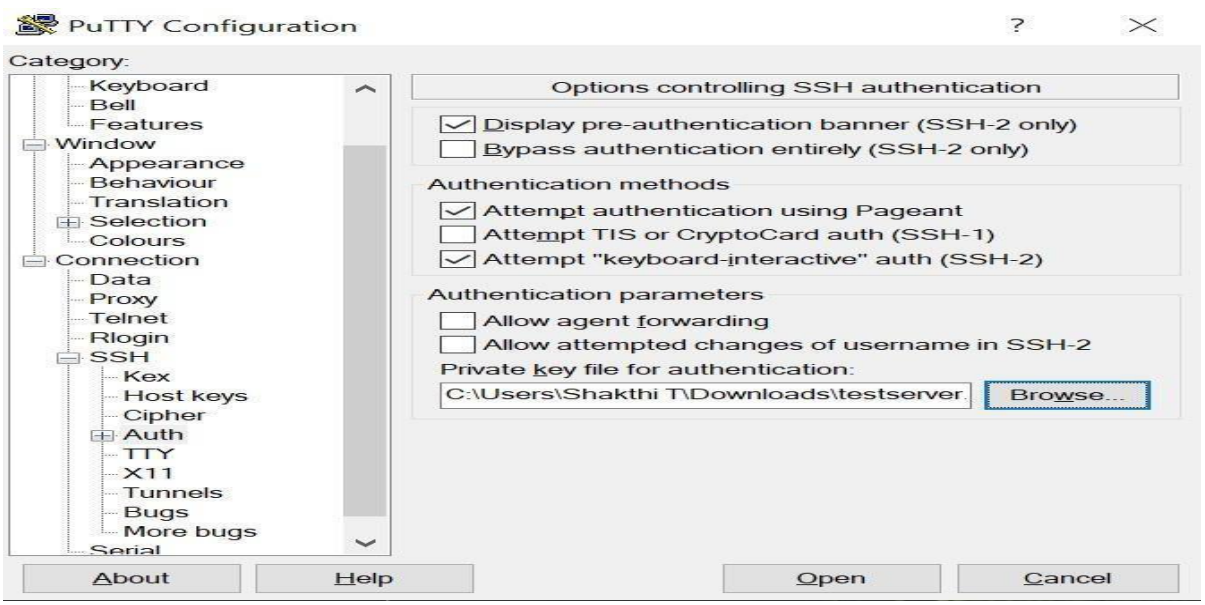
Copy the DNS name from the instance created



Using putty gen generate the key



The key will be generated



Activate the server from AWS via putty by entering the DNS and then going auth and select the ppk file generated

- Banker's Algorithm :

If Requesti <= Needi

Goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.

2) If Requesti <= Available

Goto step (3); otherwise, Pi must wait, since the resources are not available.

3) Have the system pretend to have allocated the requested resources to process Pi by modifying the state as follows:

Available = Available –

Requesti Allocationi =

Allocationi + Requesti Needi =

Needi– Requesti

1) Let Work and Finish be vectors of length 'm' and 'n'

respectively.

2) Initialize: Work = Available

3) Finish[i] = false; for i=1, 2, 3, 4….n

2) Find an i such that both

a) Finish[i] = false

b) Needi <= Work if no such i exists goto step (4)

3) Work = Work + Allocation[i] Finish[i] = true goto step (2)

4) if Finish [i] = true for all i then the system is in a safe state

Code:

```
#include<stdio.h>

int current[5][5], maximum_claim[5][5],

available[5]; int allocation[5] = {0, 0, 0, 0, 0};
```

```
int maxres[5], running[5], safe = 0;
 int counter = 0, i, j, exec, resources, processes,
 k = 1; int main() {
printf("\nEnter number of processes:
 "); scanf("%d", &processes);
for (i = 0; i < processes; i++) {
running[i] = 1; counter++;
 } printf("\nEnter number of resources:
"); scanf("%d", &resources);
 printf("\nEnter Claim
Vector:"); for (i = 0; i <
resources; i++) { scanf("%d",
&maxres[i]); }
printf("\nEnter Allocated Resource
 Table:\n"); for (i = 0; i < processes; i++) {
 for(j = 0; j < resources; j++) { scanf("%d",
&current[i][j]); } } printf("\nEnter Maximum Claim
Table:\n");
for (i = 0; i < processes;
 i++) { for(j = 0; j <
 resources; j++) {
scanf("%d", &maximum_claim[i][j]);
 } } printf("\nThe Claim Vector is: ");
for (i = 0; i < resources;
i++) { printf("\t%d",
maxres[i]); }
```

```
printf("\nThe Allocated Resource
 Table:\n"); for (i = 0; i < processes; i++) {
```

```
for (j = 0; j < resources;
j++)      {      printf("\t%d",
current[i][j]); } printf("\n"); }
printf("\nThe Maximum Claim
 Table:\n"); for (i = 0; i < processes;
 i++) {
for (j = 0; j < resources; j++) {
printf("\t%d", maximum_claim[i][j]);
} printf("\n"); }
for (i = 0; i < processes;
i++) { for (j = 0; j <
resources;          j++)          {
allocation[j] += current[i][j]; }
}
printf("\nAllocated
resources:"); for (i = 0; i <
resources; i++)
{ printf("\t%d", allocation[i]);
 } for (i = 0; i < resources;
 i++)
{ available[i] = maxres[i] - allocation[i];
 } printf("\nAvailable resources:");
 for (i = 0; i < resources; i++) { printf("\t%d",
available[i]); } printf("\n");
while (counter != 0) {
safe = 0; for (i = 0; i < processes; i++)
```

```
 { if (running[i])
{ exec = 1;
for (j = 0; j < resources; j++) {
```

```
if (maximum_claim[i][j] - current[i][j] >
available[j]) { exec = 0; break;
} }
if (exec) {
printf("\nProcess%d is executing\n", i +
1); running[i] = 0;
counter--
; safe =
1;
for (j = 0; j < resources;
j++) { available[j] +=
current[i][j]; } break;
} } }

if (!safe) { printf("\nThe processes are in unsafe
state.\n"); break; }
else { printf("\nThe process is in safe
state"); printf("\nAvailable vector:");
for (i = 0; i < resources; i++) { printf("\t%d",
available[i]); } printf("\n");
} }
return 0;
}
```

Output :

```
Enter number of processes: 2

Enter number of resources: 2

Enter Claim Vector:3
2

Enter Allocated Resource Table:
5


4
5
3

Enter Maximum Claim Table:
5
6
7
4

The Claim Vector is:    3       2
The Allocated Resource Table:
        5       4
        5       3

The Maximum Claim Table:
        5       6
        7       4

Allocated resources:    10      7
```

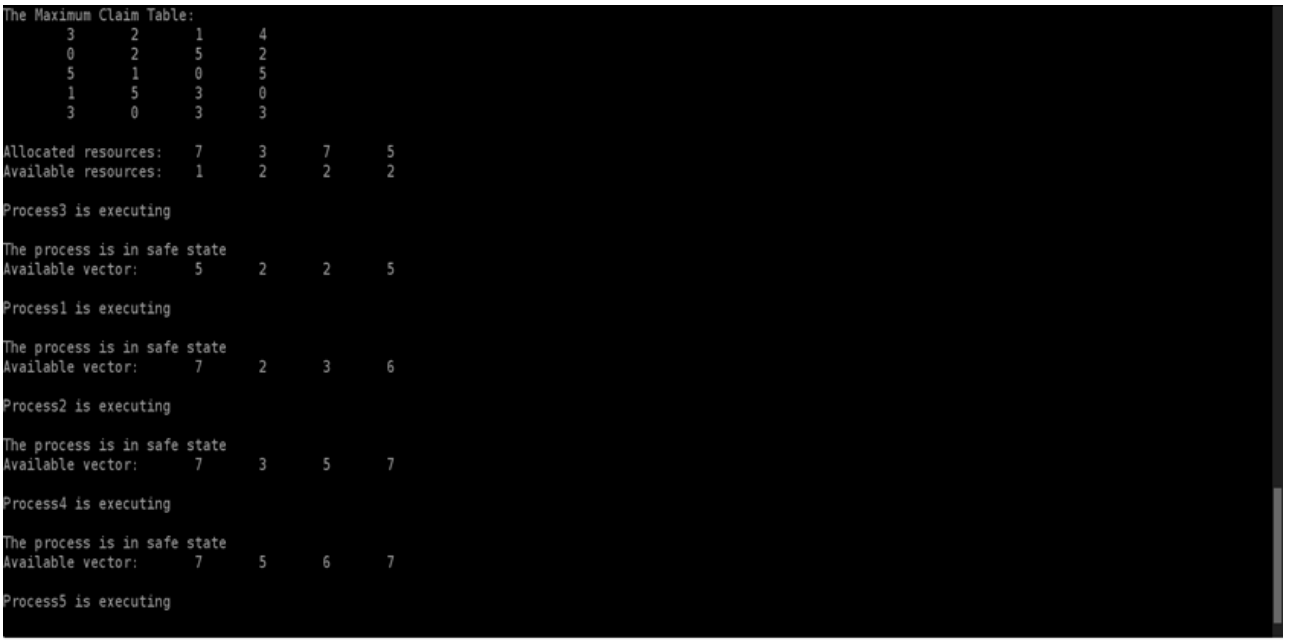
i-0d30f6c32e024950c (JananyaSivakumar)

Public IPs: 3.16.148.50 Private IPs: 172.31.45.68

```
Allocated resources:    10      7
Available resources:    -7      -5

The processes are in unsafe state.
[ec2-user@ip-172-31-45-68 ~]$ gcc jananya.c -o jananya
[ec2-user@ip-172-31-45-68 ~]$ ./jananya

Enter number of processes: 5

Enter number of resources: 4

Enter Claim Vector:8
5
9
7

Enter Allocated Resource Table:
2 0 1 1 0 1 2 1 4 0 0 3 0 2 1 0 1 0 3 0

Enter Maximum Claim Table:
3 2 1 4 0 2 5 2 5 1 0 5 1 5 3 0 3 0 3 3

The Claim Vector is:    8       5       9       7
The Allocated Resource Table:
        2       0       1       1
        0       1       2       1
        4       0       0       3
        0       2       1       0
        1       0       3       0

The Maximum Claim Table:
```

i-0d30f6c32e024950c (JananyaSivakumar)

Public IPs: 3.16.148.50 Private IPs: 172.31.45.68



i-0d30f6c32e024950c (JananyaSivakumar)

Public IPs: 3.16.148.50 Private IPs: 172.31.45.68

i-0d30f6c32e024950c (JananyaSivakumar)

Public IPs: 3.16.148.50 Private IPs: 172.31.45.68

Throttle Algorithm:

```
#include <stdio.h>
#include
<semaphore.h?
#define MAX 50
#include <time.h>
void
inserti();
void
deletei();
```

```
void
display();
int
 queue_array[MAX];
 int rear = - 1;
int fronti = -1;
 int tempresources[MAX];
void swap(int *xp, int *yp)
{ int temp = *xp;
 *xp = *yp;
*yp = temp;
```

```
}
void inserti(x)
 { int add_item=x;
if (rear == MAX -
1)
printf("Queue Overflow \n");
 else { if (fronti == - 1) /*If queue is initially
empty */ fronti = 0;
 printf("Insert the job in queue : ");
scanf("%d", &add_item); rear = rear
+ 1; queue_array[rear] = add_item;
}
} /* End of insert()
*/ void deletei() {
 if (fronti == - 1 || fronti > rear)
 {
printf("Queue Underflow
\n"); return ; } else {
 fronti = fronti + 1; }
} void display()
 {
 int i;
 if (fronti == - 1)
printf("Queue is empty
\n"); else { printf("Queue is
: \n"); for (i = fronti; i <=
rear; i++) printf("%d ",
queue_array[i]);
```

```
printf("\n");
} }
struct semaphore
{ int s;
 };
struct instances {
 int
 instances[MAX];
};
void wait(struct semaphore
s) { s.s=0;
}
void signal(struct semaphore
s) { s.s=1;
 }
int main()
 { int tr,i,j;
 printf("Enter total number of resources
 "); scanf("%d",&tr);
Int
totalresources[tr],priority[10],execution_time[10],sort[10],p[10],arrival_
tim e[10];
for(i=0;i<tr;i++)
{
printf(“Enter resources of instance
%d\n”,i+1); scanf(“%d”.&totalresources[i]);
Tempresources[i]=totalresources[i];
}
```

```
int tj;
 printf("Enter       no       of
jobs\n");
scanf("%d",&tj);//no       of
jobs int a;
for(i=0;i<tr;i++)
{printf(“Enter the arrival time of the job
%d\n”,i+1); Scanf(“%d”,&arrival_time[i]);
}
for(i=0;i<tj;i++)
{printf(“Enter job no %d\n”,i+1);
P[i]=i;
Printf(“Enter priority of job
%d\n”,i+1); scanf(“%d”,&priority[i]);
}
for(i=0;i<tj;i++)
{printf(“Enter execution time of job
%d\n”,i+1); scanf(“%d”,&execution_time[i]);
tempresources[i]=totalresources[i];
}
for (i = 0; i < tj-1; i++) // Last i elements are already in
place for (j = 0; j < tj-i-1; j++)
 if (arrival_time[j] > arrival_time[j+1]) {
swap(&arrival_time[j], &arrival_time[j+1]);
swap(&p[j],&p[j+1]);
swap(&priority[j],&priority[j+1]);
swap(&execution_time[j],&execution_time[j+
1]);
```

```
} if (&arrival_time[j]==&arrival_time[j+1]) {
if(&priority[j]>priority[j+1]){
swap(&arrival_time[j], &arrival_time[j+1]);
swap(&p[j],&p[j+1]);
swap(&priority[j],&priority[j+1]);
swap(&execution_time[j],&execution_time[j+
1]);
 } }
for (i=0;i<tj;i++)
{ //1 2 3 4 5 6
inserti(p[i]);
} struct instances
jobresources[tj]; for
(i=0;i<tj;i++)
{//how much instances
 needed printf("for
 Job%d\n",i+1);
for (j=0;j<tr;j++)
{printf("resources of type %d needed
 \n",j+1);
 scanf("%d",&jobresources[i].instances[j]);
 } }
display();
 struct semaphore
 s; clock_t t;
 t = clock();
 for (i=0;i<tj;i++)
{
```

```
printf("for
 Job%d\n",i+1); int c=0;
```

```
 for (j=0;j<tr;j++)
{ if (totalresources[j]<jobresources[i].instances[j])
{
deletei()
; Break;
}
else
{c++;}}
if (c==tr) {
wait(s); //critical section
printf("executing process%d \n",i+1);
signal(s); }
else { printf("not executed process%d \n",i+1);
 } }
t = clock() - t;
double time_taken = ((double)t)/CLOCKS_PER_SEC;
printf("Job%d took %f seconds to execute
\n",i+1,time_taken);
```

```
Enter the arrival time of the job 3
3
Enter job no 1
Enter priority of job 1
1
Enter job no 2
Enter priority of job 2
2
Enter execution time of job 1
2
Enter execution time of job 2
3
Insert the job in queue : 1
for Job1
resources of type 1 needed
4
resources of type 2 needed
3
resources of type 3 needed
2
for Job2
resources of type 1 needed
3
resources of type 2 needed
1
```

```
Enter execution time of job 2
3
Insert the job in queue : 1
for Job1
resources of type 1 needed
4
resources of type 2 needed
3
resources of type 3 needed
2
for Job2
resources of type 1 needed
3
resources of type 2 needed
2
resources of type 3 needed
1
Queue is :
1
for Job1
executing process1
for Job2
executing process2
Job3 took 0.000008 seconds to execute
[ec2-user@ip-172-31-44-163 ~]$
```

DEADLOCK USING PTHREADS:

```
#include<pthread.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
pthread_mutex_t resource1,resource2;
int test=0;
void *proc1()
{
        printf("\nThis is proc1 using rs1");
        pthread_mutex_lock(&resource1);
        usleep(200);
        printf("\np1 trying to get rs2...");
        pthread_mutex_lock(&resource2);
        test++;
        printf("\nproc1 got rs2!!");
        pthread_mutex_unlock(&resource2);
        pthread_mutex_unlock(&resource1);
return 0;
}

void *proc2()
{
        printf("\nThis is proc2 using rs2");
        pthread_mutex_lock(&resource2);
        usleep(200);
        printf("\np2 trying to get rs1...");
        pthread_mutex_lock(&resource1);
        test--;
        printf("\nproc2 got rs1!!");
        pthread_mutex_unlock(&resource1);
        pthread_mutex_unlock(&resource2);
return 0;
}

int main(){
        pthread_t t1,t2;
        pthread_mutex_init(&resource1, NULL);
        pthread_mutex_init(&resource2, NULL);

        pthread_create(&t1,NULL, proc1 , NULL);
        pthread_create(&t2,NULL, proc2 , NULL);
```

```
        pthread_join(t1,NULL);
        pthread_join(t2,NULL);
// will never arrive here
        pthread_mutex_destroy(&resource1);
        pthread_mutex_destroy(&resource2);
}
```

DEADLOCK AVOIDANCE USING PTHREADS:

```
#include<stdio.h>
#include<pthread.h>
#include<stdlib.h>
pthread_mutex_t
read_mutex;
pthread_mutex_t
write_mutex; void *
write(void *temp) {
char *ret;
    FILE
   *file1;
 char *str;
```

```
pthread_mutex_lock(&write_mute
x); sleep(5);
pthread_mutex_lock(&read_mute
x);
printf("\nFile locked, please enter the message
\n"); str=(char *)malloc(10*sizeof(char));
file1=fopen("temp","w");
scanf("%s",str);
fprintf(file1,"%s",str);
fclose(file1);
pthread_mutex_unlock(&read_mutex);
pthread_mutex_unlock(&write_mutex);
printf("\nUnlocked the file you can read it now
\n"); return ret; }
void * read(void *temp)
{ char *ret; FILE *file1;
char *str;
pthread_mutex_lock(&write_mute
x); sleep(5);
pthread_mutex_lock(&read_mute
x); printf("\n Opening file \n");
file1=fopen("temp","r");
str=(char
 *)malloc(10*sizeof(char));
 fscanf(file1,"%s",str);
 printf("\n Message from file is %s
\n",str); fclose(file1);
pthread_mutex_unlock(&read_mutex);
```

```
pthread_mutex_unlock(&write_mute
x); return ret; }
main() {
pthread_t
thread_id,thread_id1;
pthread_attr_t attr;
int ret;
void
*res;
 ret=pthread_create(&thread_id,NULL,&write,NU
LL);
ret=pthread_create(&thread_id1,NULL,&read,N
ULL);
printf("\n Created thread");
pthread_join(thread_id,&res);
pthread_join(thread_id1,&res)
; }
```

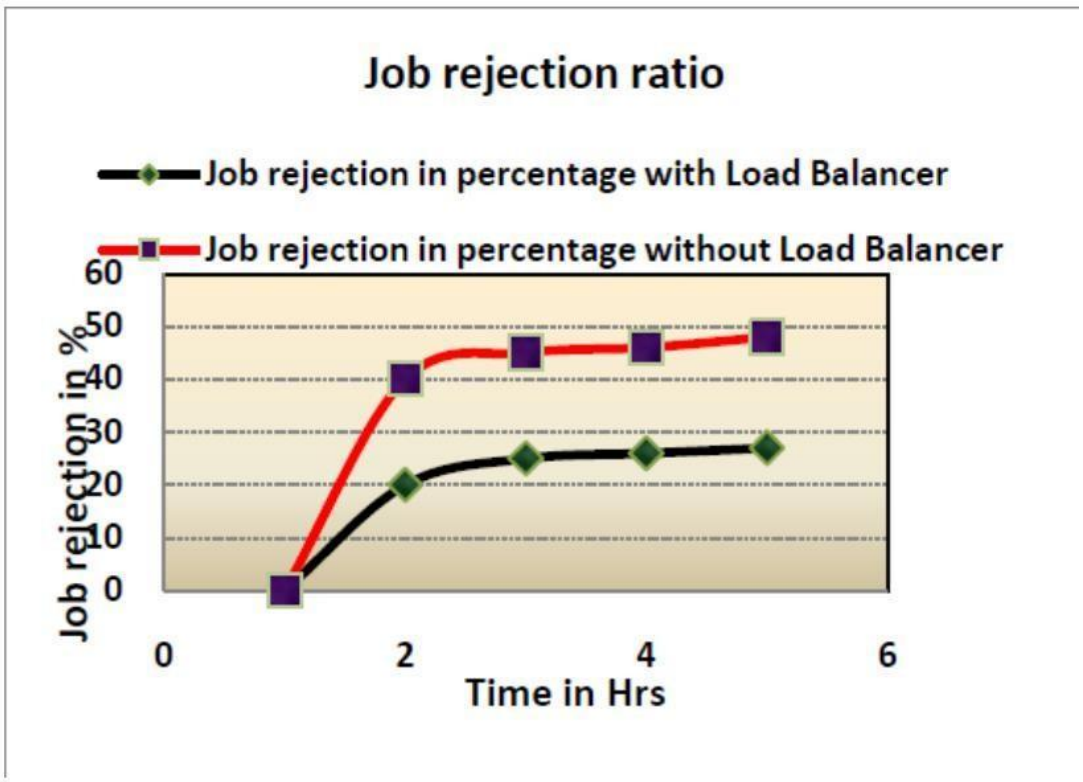
Save the code as mutex_nodeadlock.c, compile it using "-lpthread" flag Execution command- $ cc -lpthread mutex_nodeadlok.c -o create_nodeadlock $ ./create_nodeadlock

Screenshot and Output:

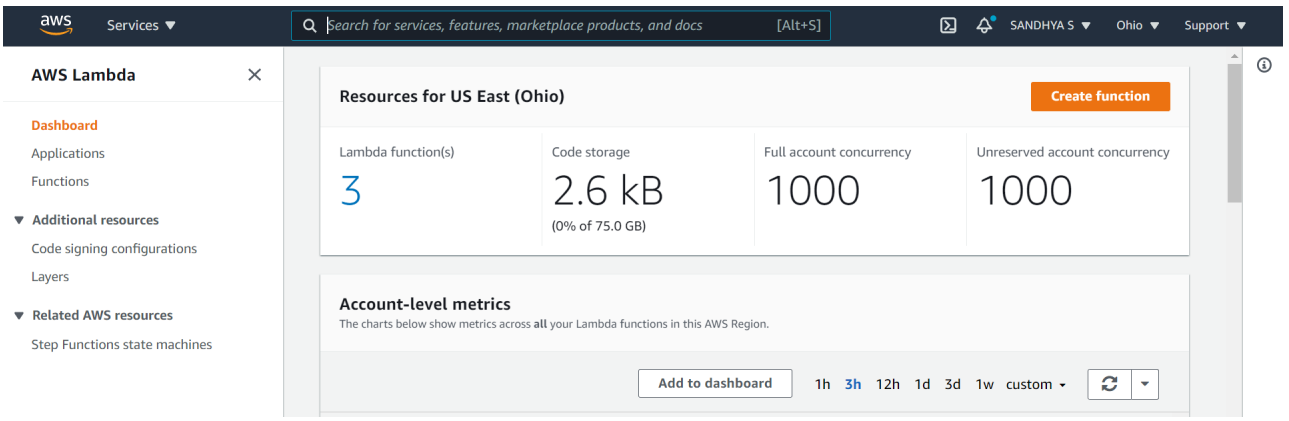
```
pthread_mutex_lock(&read_mutex);
printf("\nFile locked, please enter the message \n");
str=(char *)malloc(10*sizeof(char));
file1=fopen("temp","w");
scanf("%s",str);
fprintf(file1,"%s",str);
fclose(file1);
pthread_mutex_unlock(&read_mutex);
pthread_mutex_unlock(&write_mutex);
printf("\nUnlocked the file you can read it now \n");
return ret;
}


void * read(void *temp) {
char *ret;
FILE *file1;
char *str;
pthread_mutex_lock(&write_mutex);
sleep(5);
pthread_mutex_lock(&read_mutex);
printf("\n Opening file \n");
file1=fopen("temp","r");
str=(char *)malloc(10*sizeof(char));
fscanf(file1,"%s",str);
printf("\n Message from file is %s \n",str);

fclose(file1);

pthread_mutex_unlock(&read_mutex);
pthread_mutex_unlock(&write_mutex);
return ret;
}



main() {

pthread_t thread_id,thread_id1;
pthread_attr_t attr;
int ret;
void *res;
ret=pthread_create(&thread_id,NULL,&write,NULL);
ret=pthread_create(&thread_id1,NULL,&read,NULL);
printf("\n Created thread");
pthread_join(thread_id,&res);
pthread_join(thread_id1,&res);
}
```

```
Created thread
Opening file

File locked, please enter the message

Message from file is ?y
hello
```

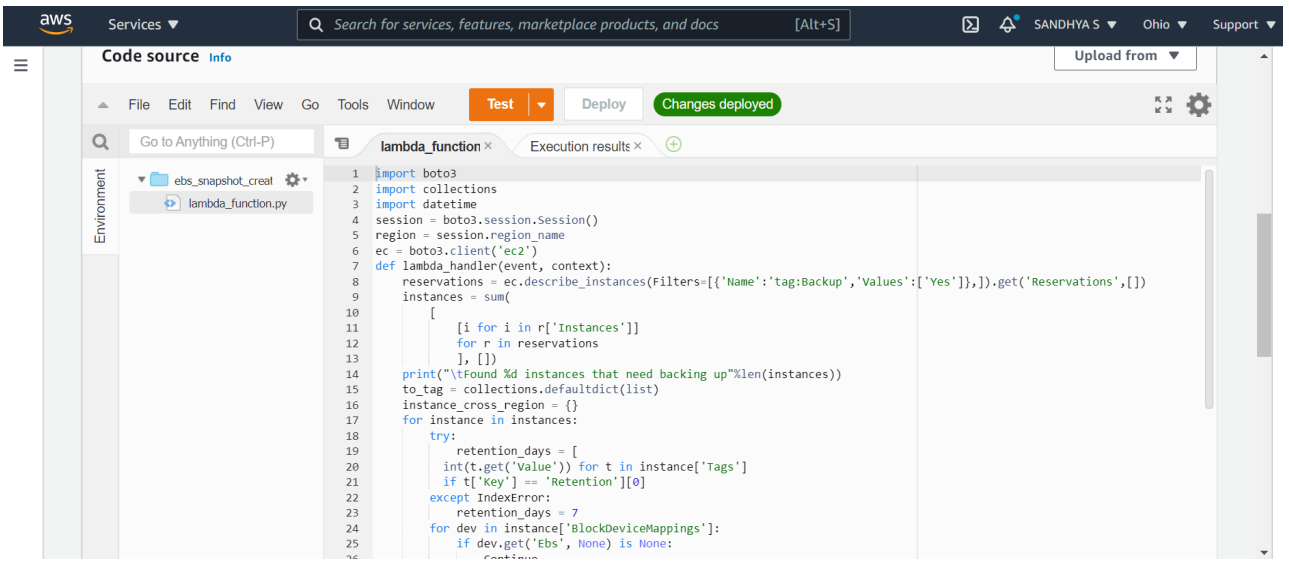
Comparative Analysis of Algorithm:



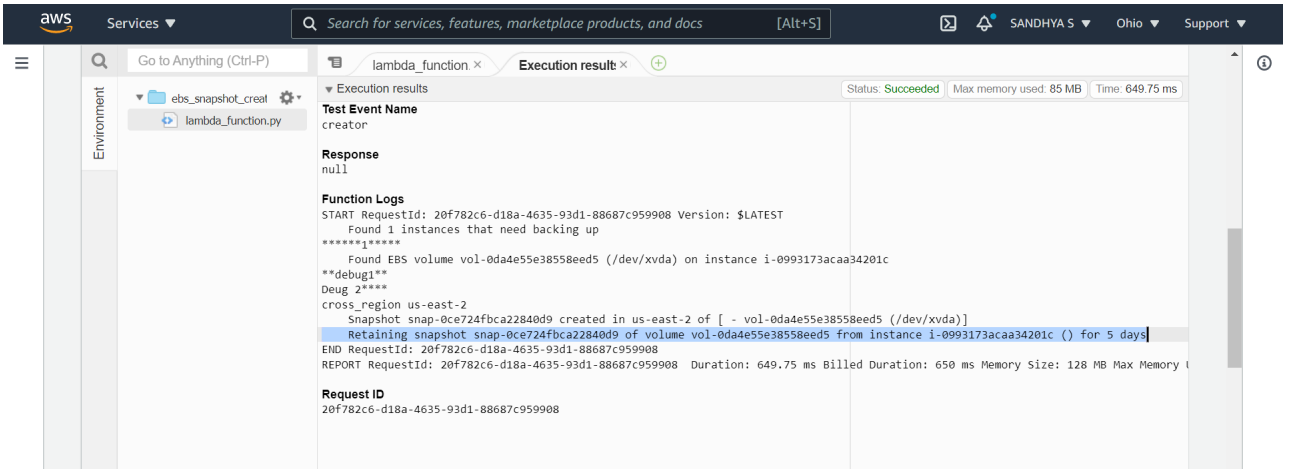
Disaster Recovery execution in AWS Cloud :



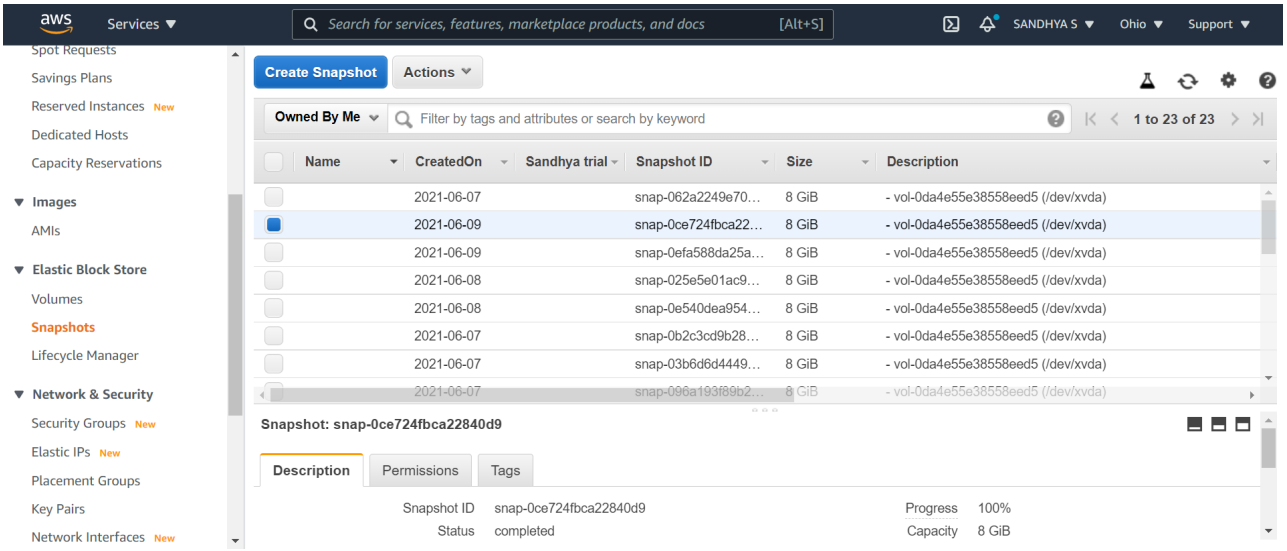
Creating function using AWS lambda through EC2 Server to execute codes



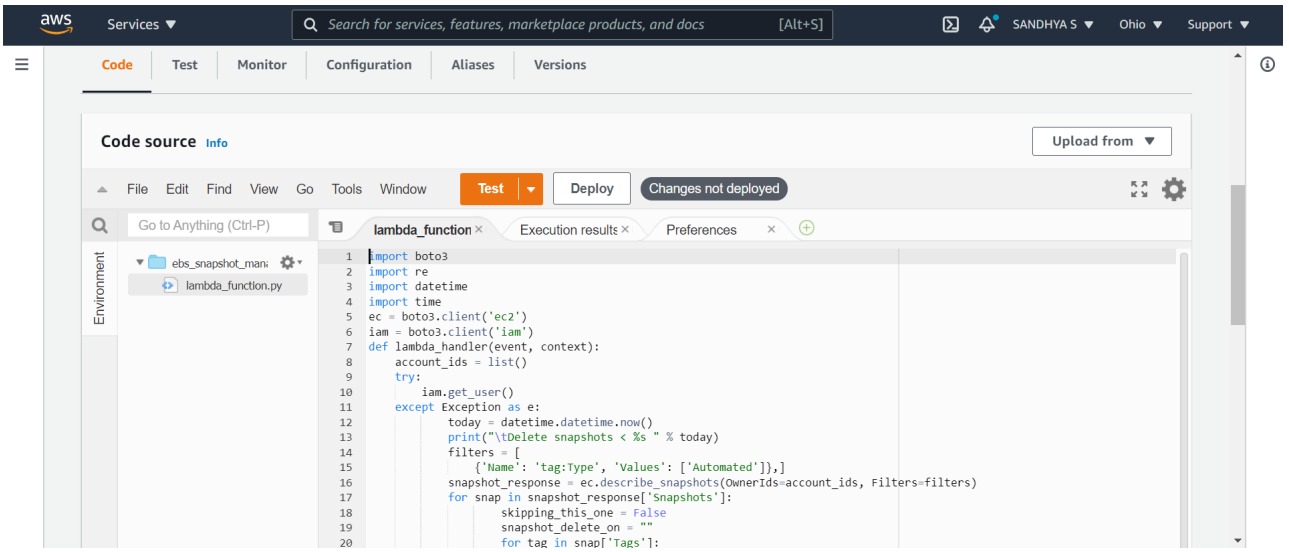
After creation of hosting server with JSON as policy ebs-snapshot-creator as snapshot


aws
Services
Search for services, features, marketplace products, and docs
[Alt+S]
SANDHYA S
Ohio
Support
Go to Anything (Ctrl-P)
Environment
ebs_snapshot_creat
lambda_function.py
lambda_function
Execution result
Execution results
Status: Succeeded
Max memory used: 85 MB
Time: 649.75 ms
Test Event Name
creator
Response
null
Function Logs
START RequestId: 20f782c6-d18a-4635-93d1-88687c959908 Version: $LATEST
Found 1 instances that need backing up
******1*****
Found EBS volume vol-0da4e55e38558eed5 (/dev/xvda) on instance i-0993173acaa34201c
debug1
Deug 2****
cross_region us-east-2
Snapshot snap-0ce724fbca22840d9 created in us-east-2 of [- vol-0da4e55e38558eed5 (/dev/xvda)]
Retaining snapshot snap-0ce724fbca22840d9 of volume vol-0da4e55e38558eed5 from instance i-0993173acaa34201c () for 5 days
END RequestId: 20f782c6-d18a-4635-93d1-88687c959908
REPORT RequestId: 20f782c6-d18a-4635-93d1-88687c959908 Duration: 649.75 ms Billed Duration: 650 ms Memory Size: 128 MB Max Memory
Request ID
20f782c6-d18a-4635-93d1-88687c959908

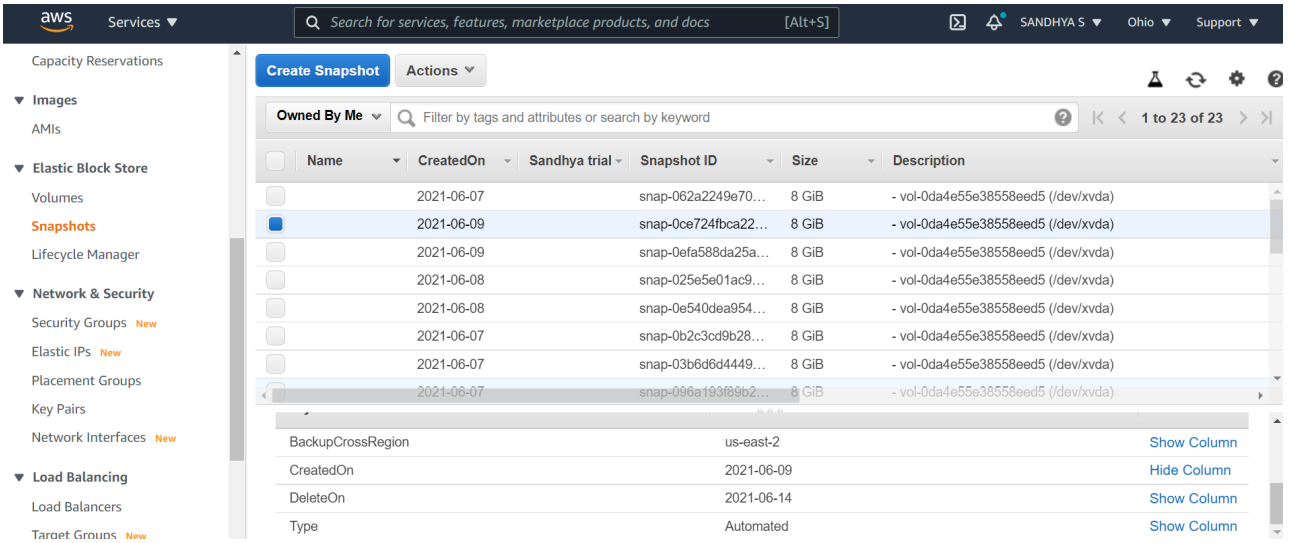
The execution of the code where the backup is being retained



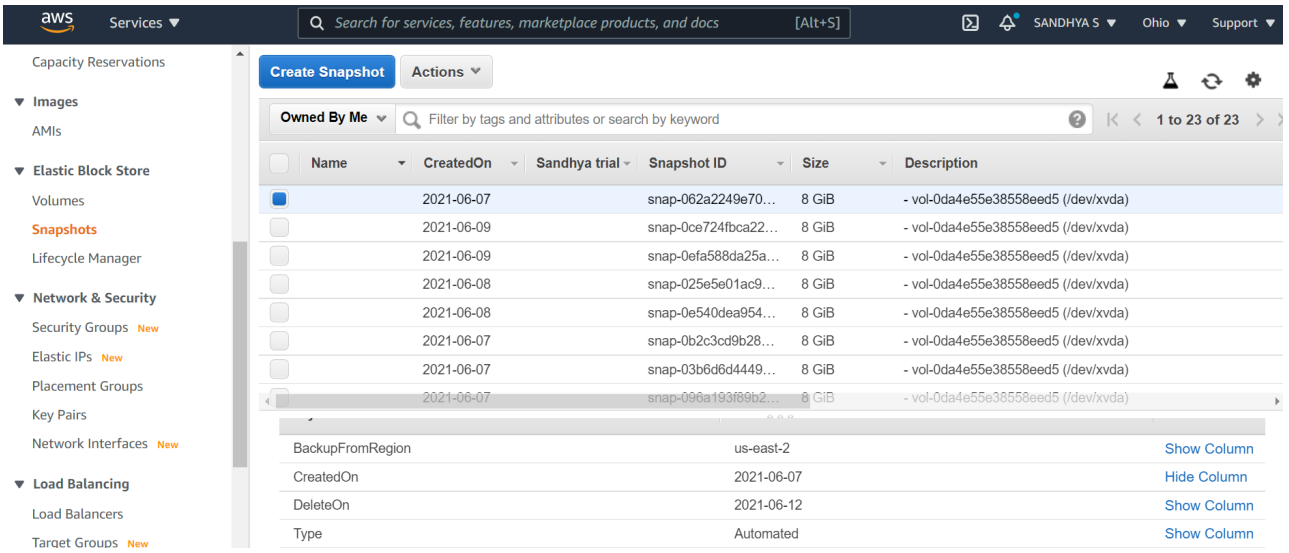
As we can see the status of the snapshot shows completed



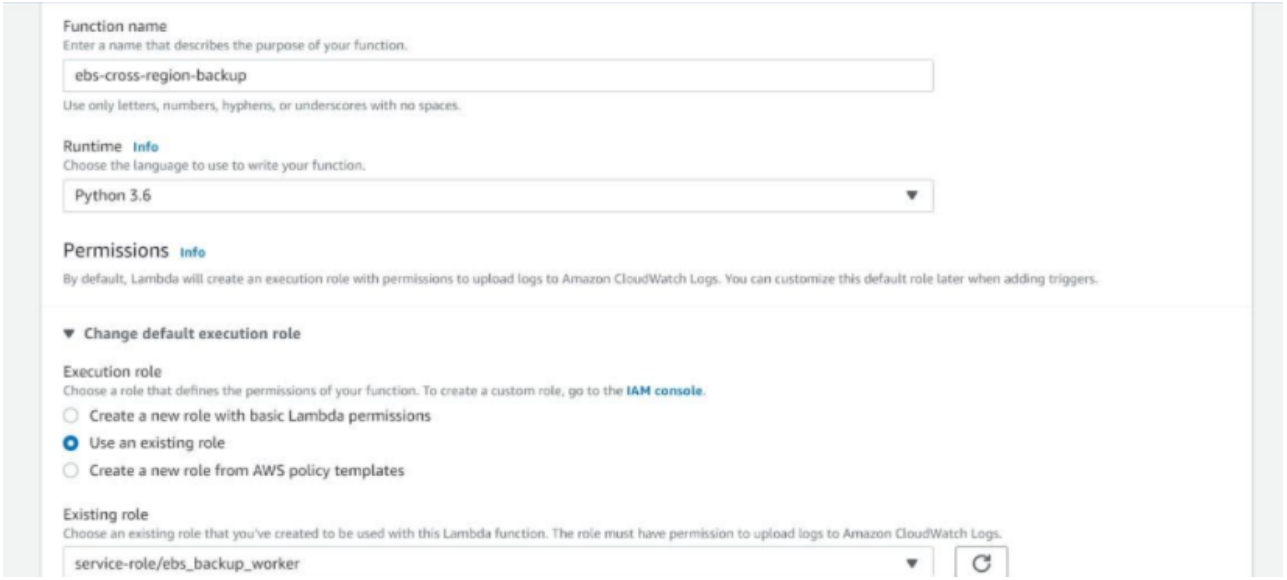
Ebs-snapshot-manager execution code



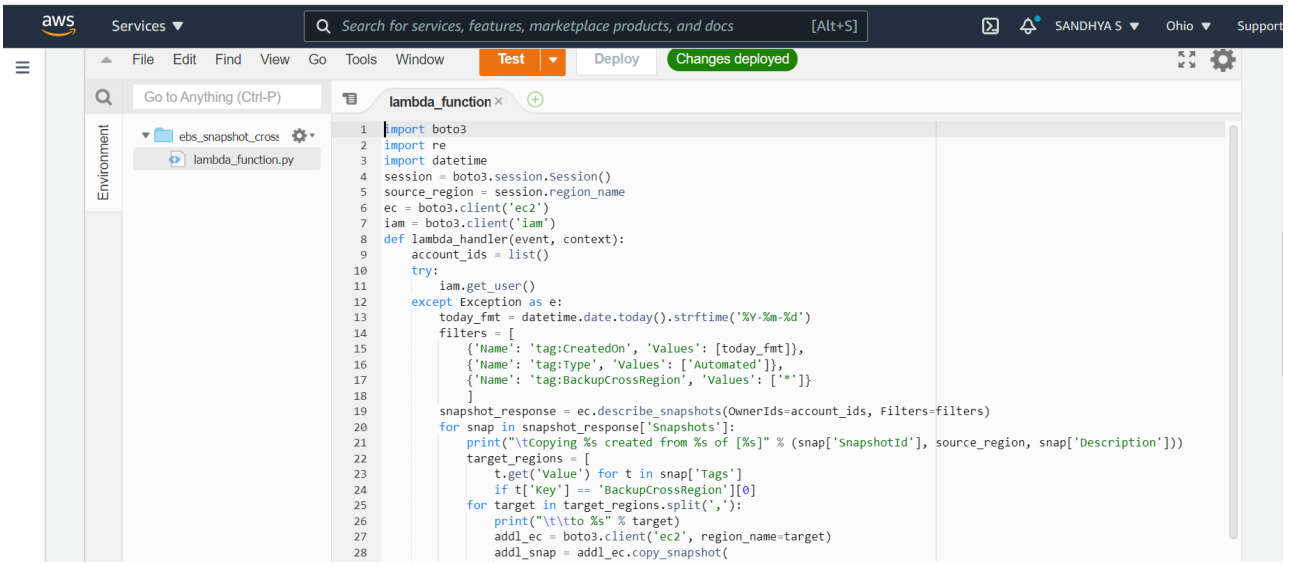
Creation of snapshot after the code and the tags and key being generated

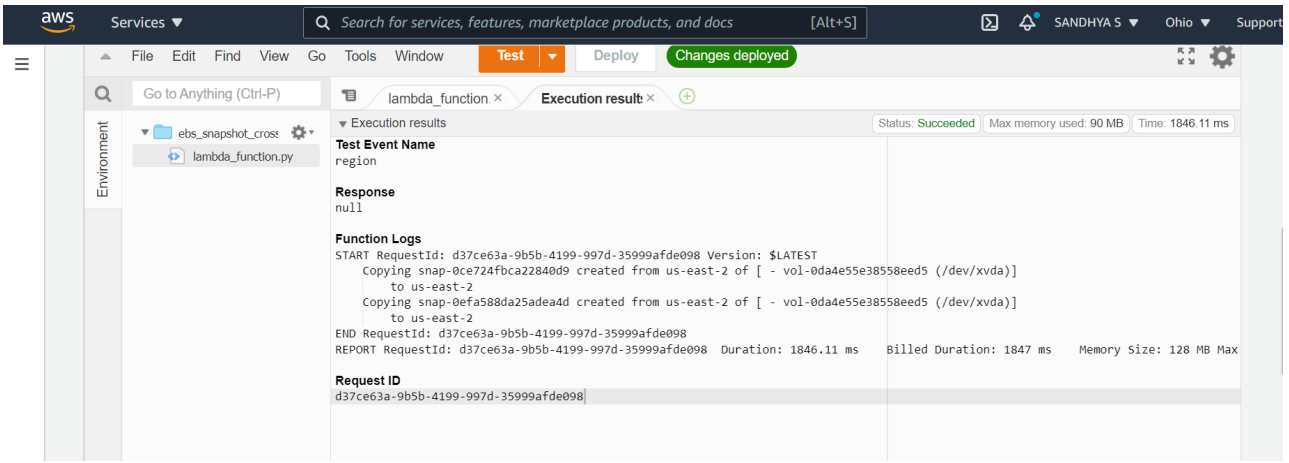


We try changing the value of deleted on to see the update in snapshot test

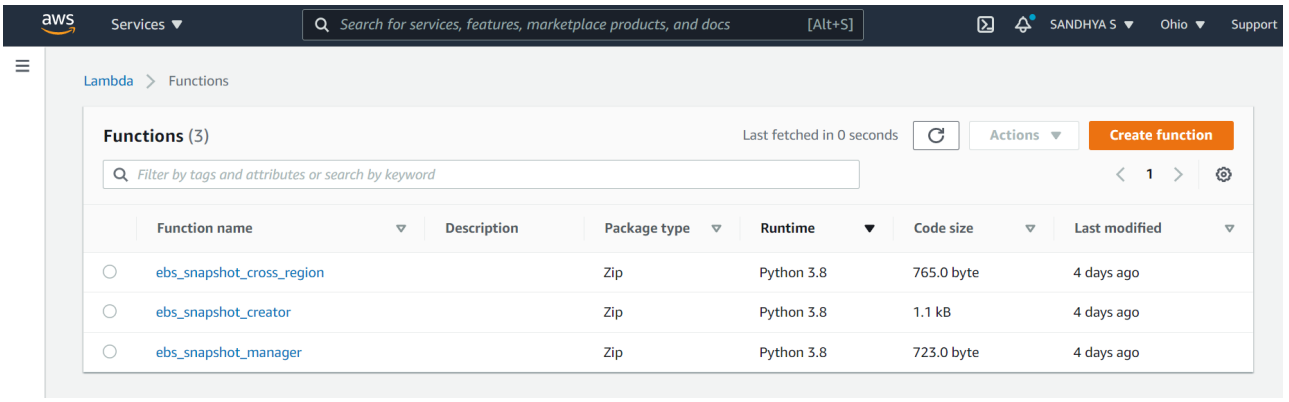


Creation of the lambda function ebs-cross-region-backup

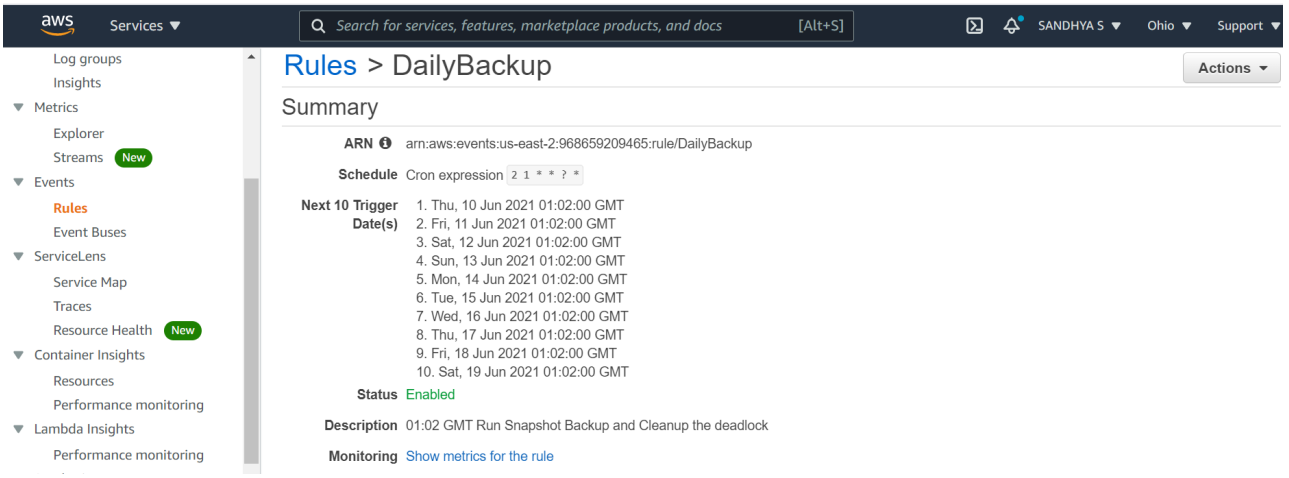


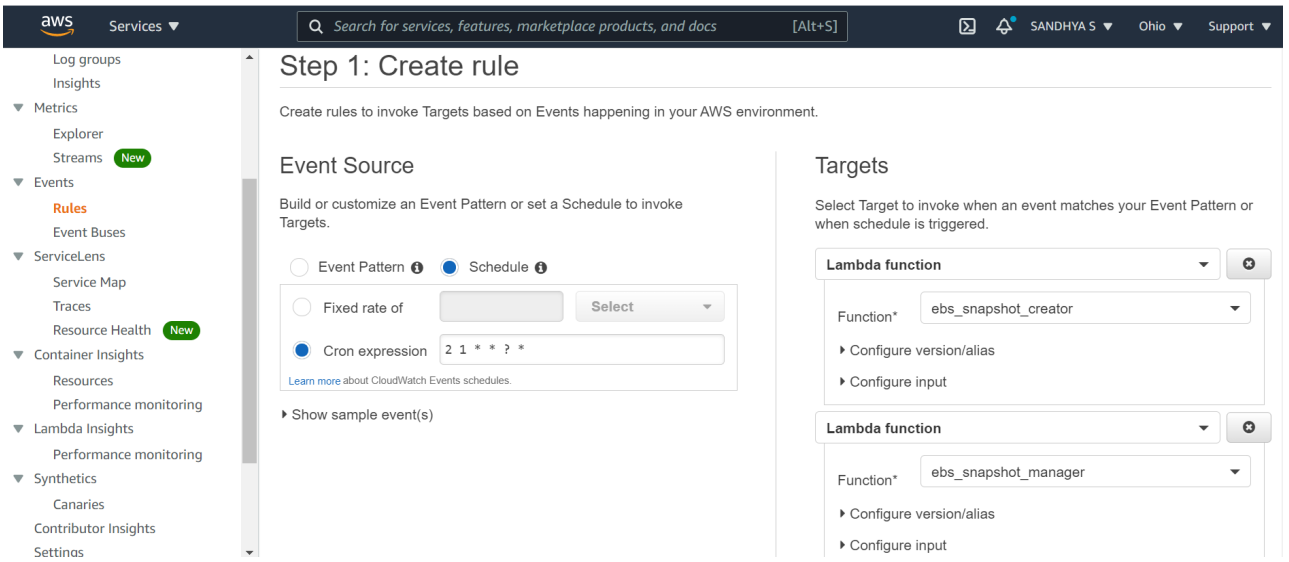


Ebs-cross-region-backup code and execution

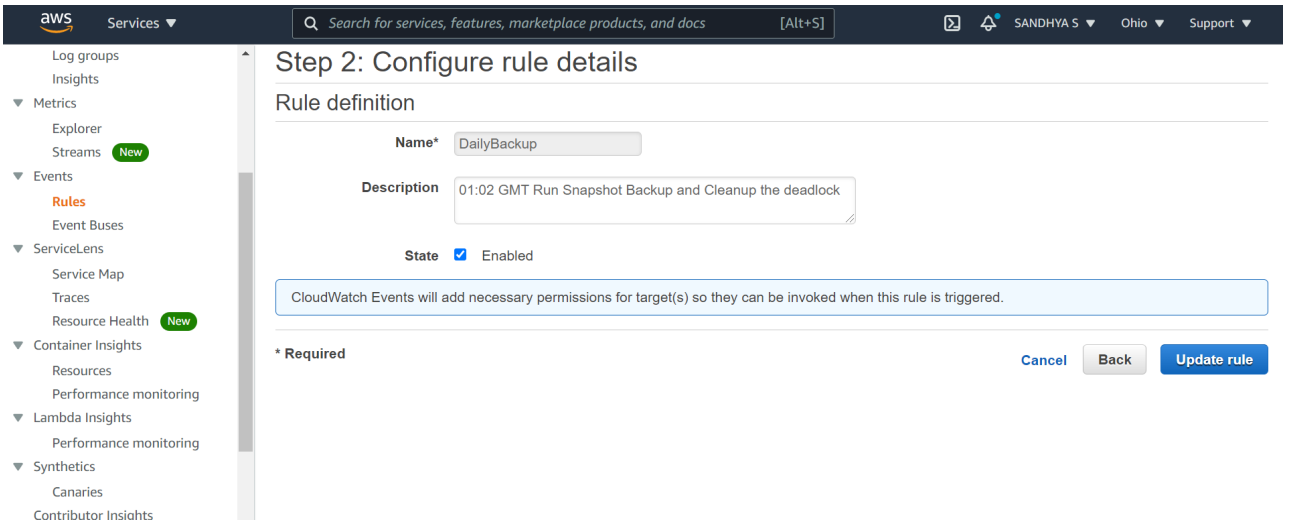


Creation of the three lambda functions

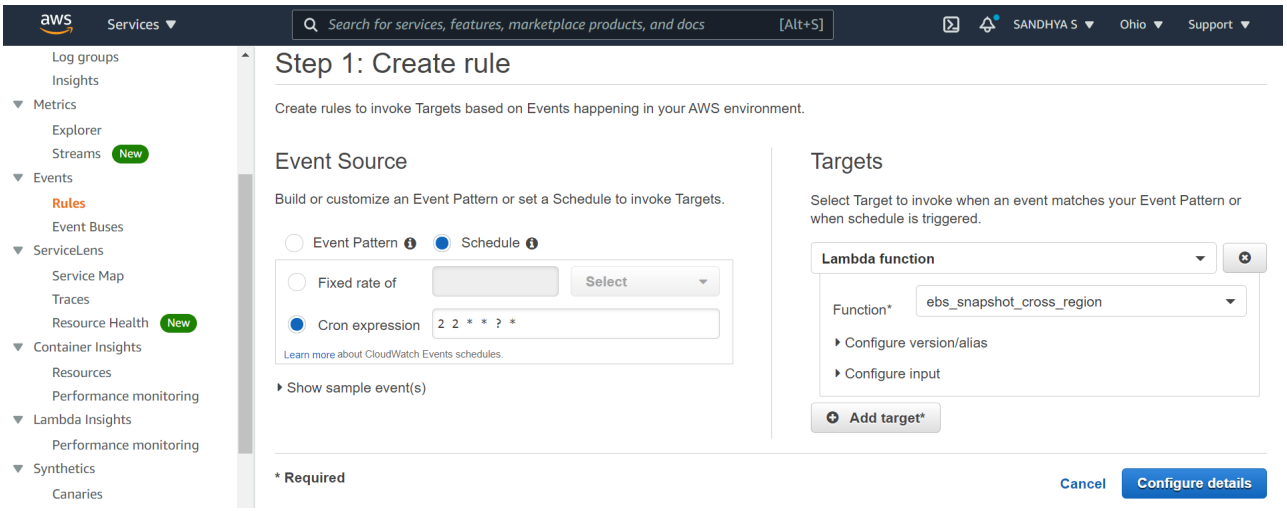




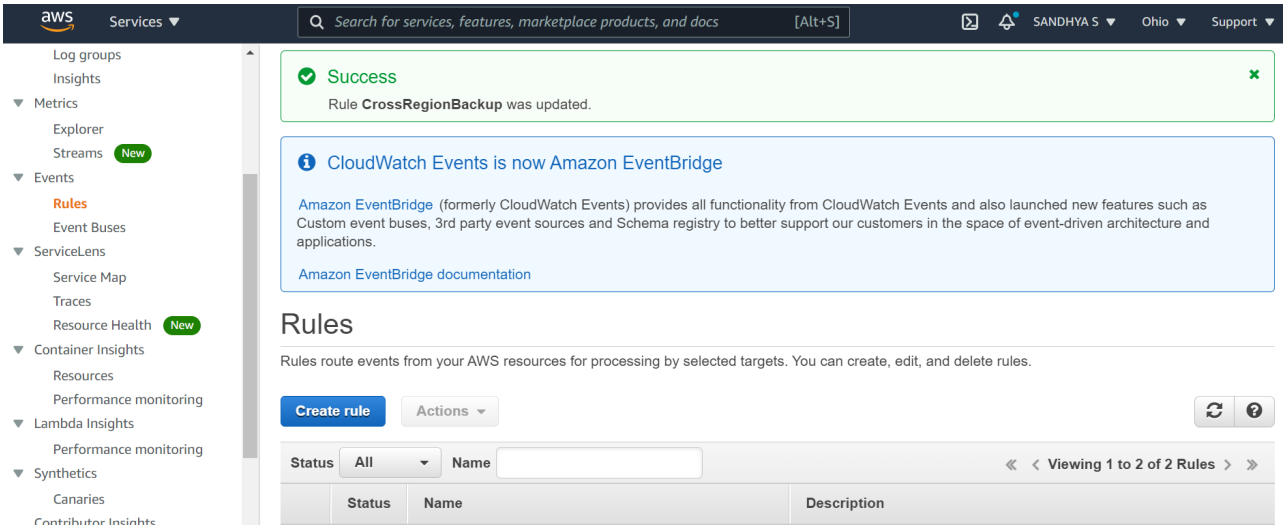
Creation of trigger for ebs-snapshot-creator and ebs-snapshot-manager



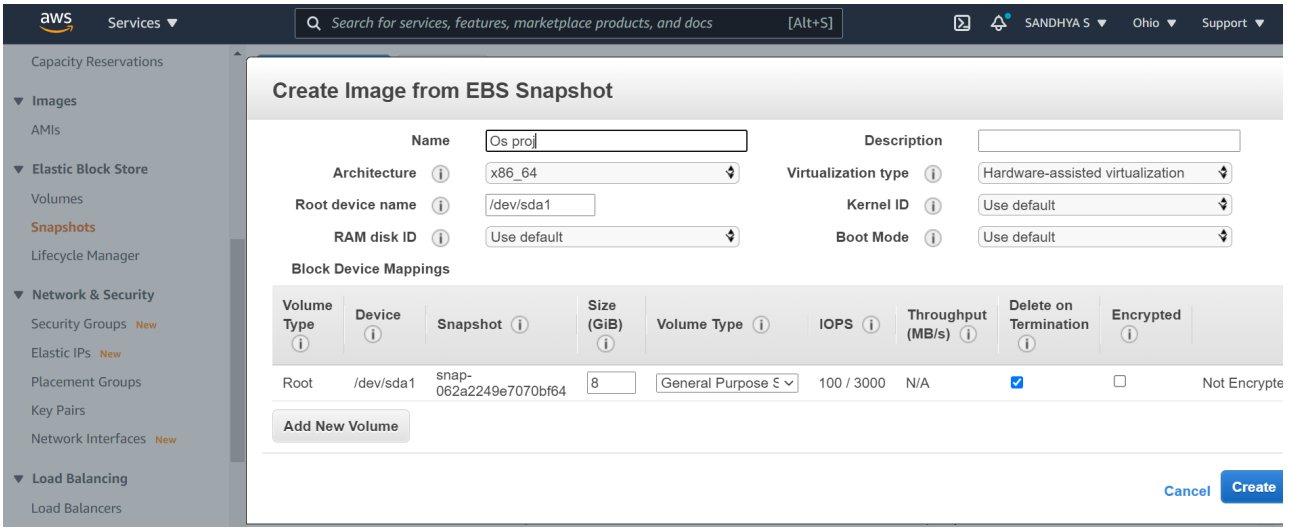
Configuring the trigger as a daily backup



Creation of trigger or rule for ebs-cross-region-backup



After creation of both the rules/trigger



Creating image for EBS snapshot

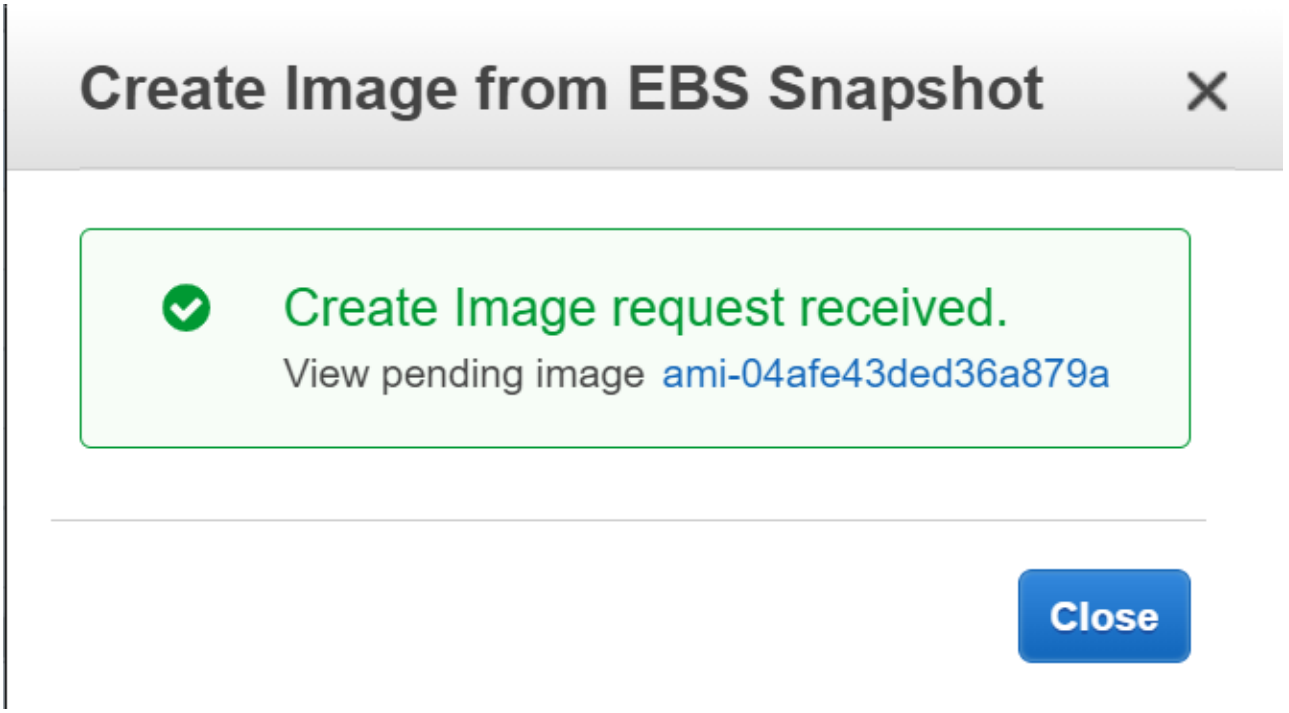
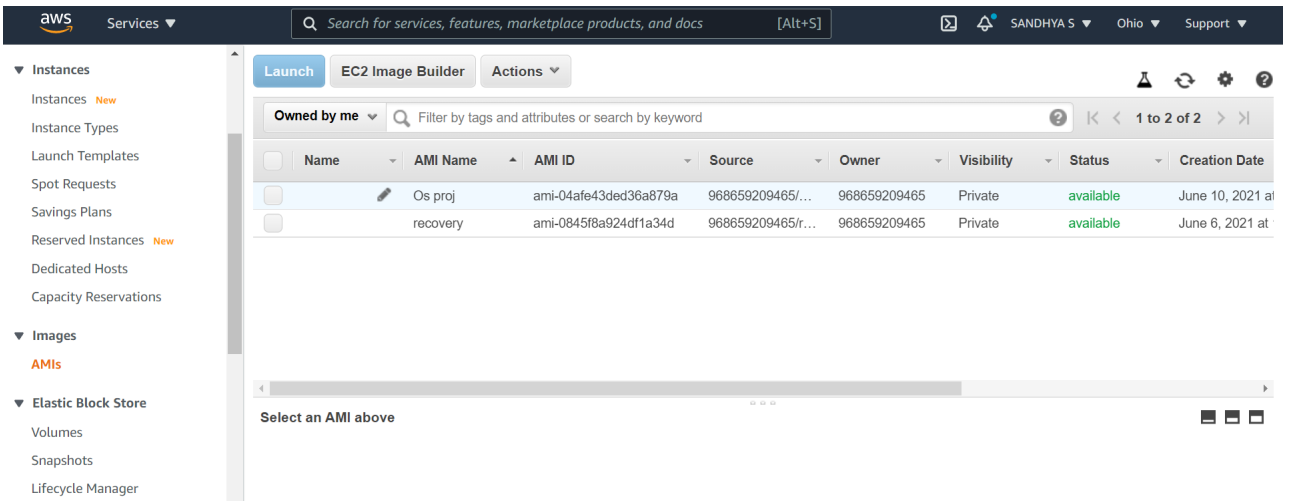


Image being requested and received



The recovered backup being generated

Eb-snapshot-creator code :

```
import boto3

import

collections

import datetime
session =

boto3.session.Session()

region = session.region_name

ec = boto3.client('ec2’)
```

```
def lambda_handler(event, context):
reservations =
ec.describe_instances( Filters=[
{'Name': 'tag:Backup', 'Values': ['Yes']},
]
).get(
'Reservations',
[]
 )
instances =
sum( [
[i for i in
r['Instances']] for r in
reservations
 ], [])
 print("\tFound %d instances that need backing up" %
 len(instances)) to_tag = collections.defaultdict(list)
 instance_cross_region
= {} for instance in
instances:
 try:
 retention_days = [
 int(t.get('Value')) for t in
 instance['Tags'] if t['Key'] ==
 'Retention'][0]
except
IndexError:
retention_days =
```

7

```
for dev in
instance['BlockDeviceMappings']: if
dev.get('Ebs', None) is None:
Continue
```

```
vol_id =
dev['Ebs']['VolumeId']
dev_name =
dev['DeviceName’]
print("\tFound EBS volume %s (%s) on instance %s" % (vol_id,
dev_name, in stance['InstanceId']))

  # figure out instance name & if cross region backup
 wanted instance_name = ""
cross_region = ""
for tag in instance['Tags']:
if tag['Key'] == 'Name':
instance_name = tag['Value']
if tag['Key'] == 'BackupCrossRegion':
cross_region = tag['Value']
description = '%s - %s (%s)' % (instance_name, vol_id,
dev_name) snap = ec.create_snapshot(
VolumeId=vol_id,
Description=descripti
on
)
if (snap):
print("\tSnapshot %s created in %s of [%s]" % (snap['SnapshotId'],
region, d escription))
to_tag[retention_days].append(snap['SnapshotId’])
print(
"\tRetaining snapshot %s of volume %s from instance %s (%s) for %d
days"
% (
snap['SnapshotId
```

'], vol_id,

```
instance['InstanceId
'], instance_name,
retention_days,
))
 today_fmt = datetime.date.today().strftime('%Y-%m-%d')
delete_date = datetime.date.today() +
datetime.timedelta(days=retention_ days)
delete_fmt =
delete_date.strftime('%Y-%m-%d')
ec.create_tags(
 Resources=[snap['SnapshotId'
], ], Tags=[
{'Key': 'CreatedOn', 'Value': today_fmt},
{'Key': 'DeleteOn', 'Value': delete_fmt},
{'Key': 'Type', 'Value': 'Automated'},
]
)
if cross_region:
ec.create_tags(
Resources=[snap['SnapshotId'
], ], Tags=[
{'Key': 'BackupCrossRegion', 'Value': cross_region}
])
```

Eb-snapshot-manager code :

```
import boto3
import re
import
datetime
```

```
import time
ec = boto3.client('ec2')

iam =

boto3.client('iam')

def lambda_handler(event, context):

account_ids =

list() try:

iam.get_user()

except Exception as e:
today = datetime.datetime.now()

print("\tDelete snapshots < %s " %

today) filters = [

{'Name': 'tag:Type', 'Values': ['Automated']},]

snapshot_response = ec.describe_snapshots(OwnerIds=account_ids,
Filters
=filters)

for snap in

snapshot_response['Snapshots']:

skipping_this_one = False

snapshot_delete_on = ""

for tag in snap['Tags']:

if tag['Key'] == 'KeepForever':

skipping_this_one =

True if tag['Key'] ==

'DeleteOn':

snapshot_delete_on =

tag['Value'] if skipping_this_one

== True:
```

```
print("\tSkipping snapshot %s marked KeepForever" %
(snap['SnapshotId'])) else:
if snapshot_delete_on == "":
```

```
print("\There is no DeleteOn Tag for this backup: %s" %
(snap['SnapshotId'])
)

continue

snapshot_expires = datetime.datetime.strptime(snapshot_delete_on,
"%Y-
%m-%d")

if snapshot_expires < today:

print("\tDeleting %s with expiry date %s" % (snap['SnapshotId'],
snapshot_d elete_on))

 ec.delete_snapshot(SnapshotId=snap['Snapsh

otId']) else:

print("\tKeeping %s until %s" % (snap['SnapshotId'],
snapshot_delete_on))
```

Eb-snapshot-cross-regions code :

```
import

boto3

import re

import datetime
session = boto3.session.Session()

source_region =
session.region_name ec =
boto3.client('ec2')

iam = boto3.client('iam')

def lambda_handler(event,

 context): account_ids = list()

 try:

iam.get_user()

   except Exception as e:
   today_fmt =
```

```
datetime.date.today().strftime('%Y-%m-%d') filters =

[
```

```
        {'Name': 'tag:CreatedOn', 'Values': [today_fmt]},
        {'Name': 'tag:Type', 'Values': ['Automated']},
        {'Name': 'tag:BackupCrossRegion', 'Values': ['*']}
    ]
    snapshot_response =
ec.describe_snapshots(OwnerIds=account_ids, Filte rs=filters)
  for snap in snapshot_response['Snapshots']:
 print("\tCopying %s created from %s of [%s]" % (snap['SnapshotId'],
 source
_region,
snap['Description']))
target_regions = [
t.get('Value') for t in snap['Tags']
if                t['Key']                ==
 'BackupCrossRegion'][0]        for
 target  in  target_regions.split(','):
 print("\t\tto %s" % target)
addl_ec = boto3.client('ec2',
region_name=target) addl_snap =
addl_ec.copy_snapshot(
SourceRegion=source_region,
SourceSnapshotId=snap['SnapshotId'],
Description=snap['Description'],
DestinationRegion=target
)
addl_ec.create_tags(
Resources=[addl_snap['SnapshotI
d']], Tags=snap['Tags']
)
```

```
addl_ec.delete_tags(
Resources=[addl_snap['SnapshotI
d']],
```

```
Tags=[{"Key": "BackupCrossRegion"}]
 )
addl_ec.create_tags(
Resources=[addl_snap['SnapshotI
d']],
Tags=[{'Key': 'BackupFromRegion', 'Value': source_region}]
)
```

Conclusion:

The advancement in the Cloud computing model has led to an increase in demand for its services. An efficient and effective performance of this model is essential to have better business returns. The performance of this model is influenced by various factors like Scalability, Reliability, Security, Disaster Management etc.Disaster is one of the factors which deteriorate the Cloud performance. Though it is an unpredictable event it can be prevented if the system is designed with efficient Scheduling, Resource allocation, Load Balancing etc. Disaster management is a challenging task, it provides a platform for a Cloud Service provider to exhibit the best service. It is essential for the Cloud Service providers to apply appropriate metrics to monitor the Cloud performance. Due to the growing demand for the cloud services a short period of downtime would result in significant financial loss. Disaster is one of the several factors which influence the downtime to occur in this computing environment. In the proposed Throttle Algorithm to balance the load in the computing model the Deadlock Probability is Less and Job Rejection Ratio is Less. The algorithm maintains the status of the resources periodically. It rejects the jobs exceeding the Job queue size.

Scopes of Improvement :

In the future we would like to remove the queue limit and decrease the execution time.

References:

1. Mahitha.O , Suma. V , “Deadlock Avoidance through Efficient Load Balancing to Control Disaster in Cloud Environment.”, Proceedings of the IEEE - 31661 International Conference on Computer Science and Information Technology, July 4 - 6, 2013.

2. Mr.Akshay A. Gharat, Mr. Devendra E. Mhamunkar , “Disaster Recovery in Cloud Computing”, In: 2010 10th Annual International Symposium on Applications and the Internet IJARCET, pp. 304–307 2015

3. Chintureena Thingom,”A Study on Tools for Cloud Disaster Management”, International Journal of Interdisciplinary and Multidisciplinary Studies (IJIMS), 2014, Vol 1, No.4, 82- 86.

4. Mahitha.O , Suma. V, “Effective Disaster Management to Enhance the Cloud Stability”,International Conference on Electrical Engineering and Computer Science (IEEE),2nd june 2013, pp.50-54,2013,Bangalore,India.

5. Sinung Suakanto, Suhono H. Supangkat, Suhardi, Roberd Saragih, Tunggul Arief Nugroho, I Gusti Bagus Baskara Nugraha, “Environmental and Disaster Sensing Using Cloud Computing Infrastructure”, In: International Conference on Cloud Computing and Social Networking (ICCCSN), pp. 1–6 (2012)