

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY,  
GUWAHATI

# CS351 PROJECT

---

## INTELLIGENT HYPERVISOR

1. AILNENI SAI SHISHIR (1701002)
2. ANIVRAT GOEL (1701008)
3. SUSHANT AGGARWAL (1701069)

## **ABSTRACT:**

**Intelligent hypervisor design:** In the traditional cloud world, VMs are created when requested to the hypervisor. Design a solution where VMs are created by an intelligent system monitors the eco system, and requests the hypervisor to create VMs in order to comply with service level agreement of processing speed / turnaround etc.

- In the above problem statement we were asked to create an hypervisor, which is a very big task and cannot be done without specialized hardware. Hence, we have simulated the task of hypervisor on Amazon Web Services.
- In AWS we consider the server as our system and monitor the environment. When any parameters are not satisfied we launch instances to actually comply with the SLA.
- Each instance is considered as a separate virtual machine.
- This process can be achieved by AWS AutoScaling in which we can monitor the system and can launch instances according to it.

# **STEPS INVOLVED IN THE PROCESS**

---

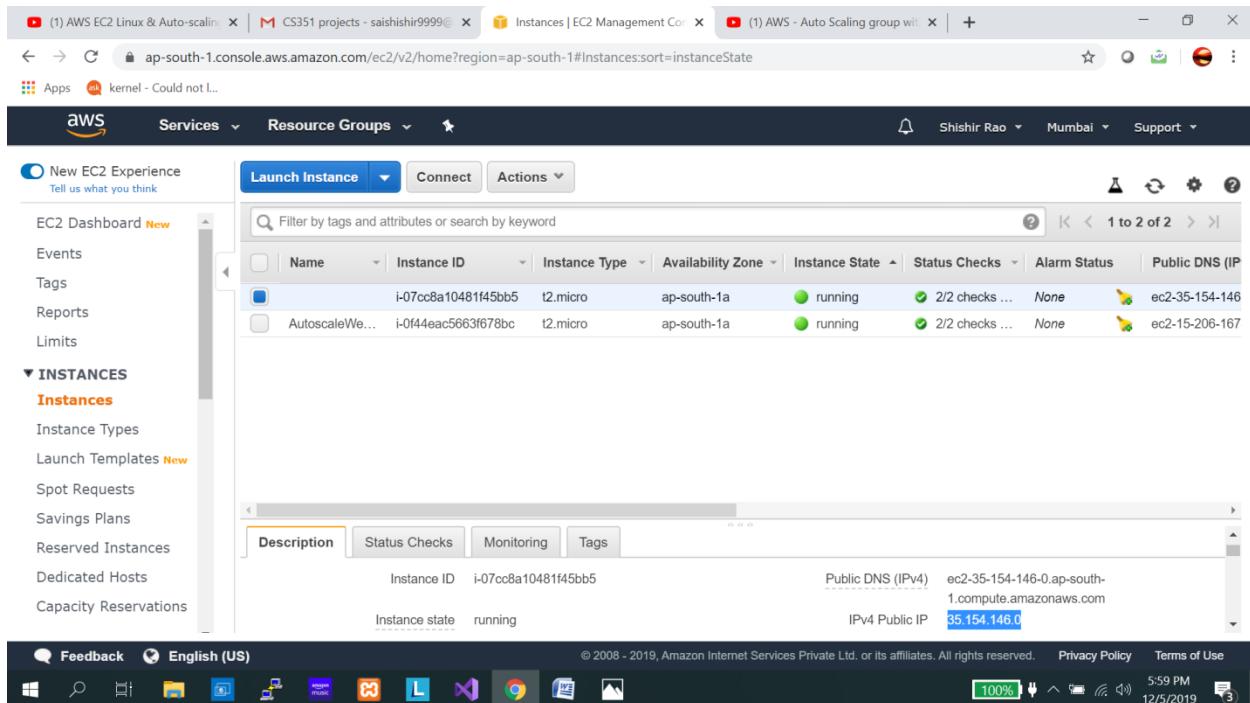
- 1.CREATE A WEB SERVER IN AWS.**
- 2.CREATE AN IMAGE OF THE WEB SERVER.**
- 3.CREATE AN AUTOSCALING GROUP.**
- 4.APPLY SCALE UP AND SCALE DOWN POLICIES.**
- 5.CREATE NETWORK/SUBNETS.**
- 6.CREATE ALARMS.**
- 7.OPEN YOUR INSTANCE FROM TERMINAL  
USING SSH KEY.**
- 8.EXECUTE A SCRIPT TO INCREASE THE LOAD ON  
CPU**
- 9.CHECK THE OUTPUT, WE CAN SEE NUMBER OF  
RUNNING INSTANCES INCREASED.**

## **DISCLAIMER:-**

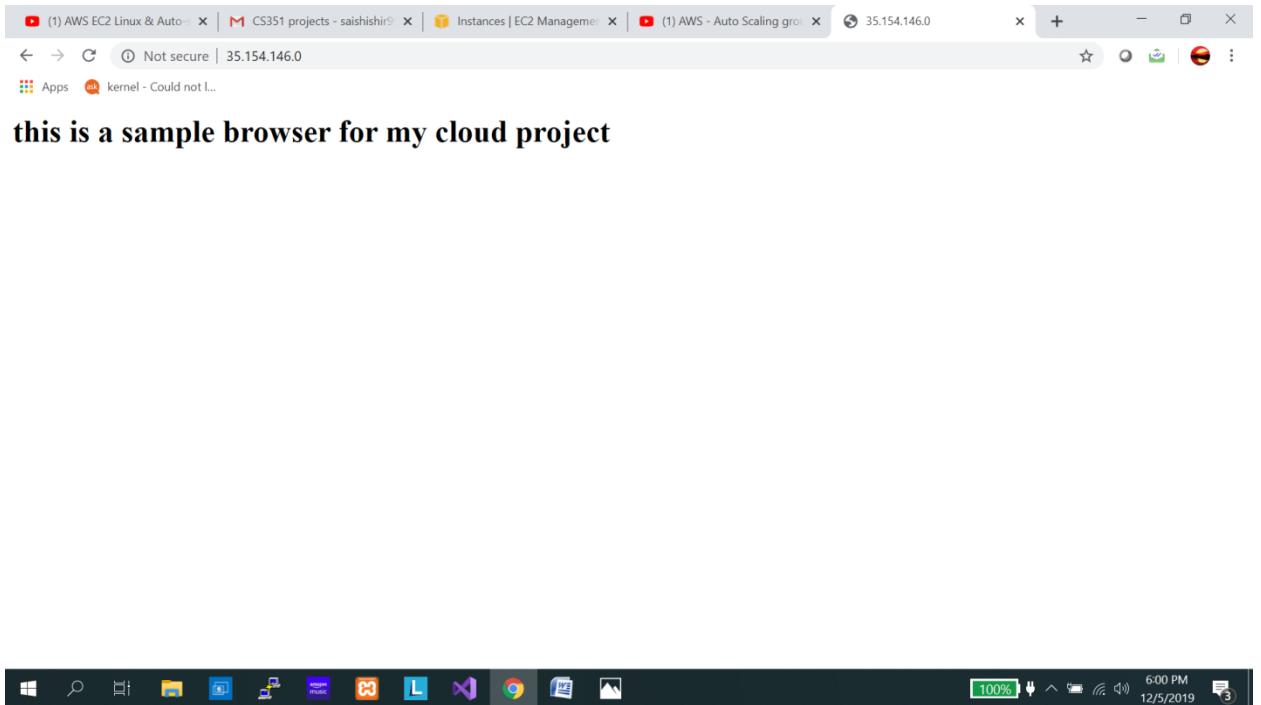
- 1. ALL THE CODES GIVEN BELOW ARE USING OUR KEYS  
AND OUR SYSTEMS. PLEASE DO CHANGE THEM TO  
EXECUTE THE CODES.**
- 2. PYTHON 3 IS REQUIRED TO EXECUTE THE CODES.**
- 3. FOR CONVENIENCE PLZ GO VIA GRAPHICAL APPROACH  
WHICH WILL BE A BIT EASY.**

## STEP-1:-

- **GO TO EC2 IN AWS**
- **CLICK LAUNCH INSTANCE**
- **SELECT A LINUX INSTANCE**
- **SET TO T2.MICRO**
- **IN INSTANCE DETAILS CLICK ON ADVANCE AND PASTE THE FOLLOWING SCRIPT.**
- **#!/bin/bash**
- **Yum install httpd -y**
- **Echo '<h1>this is a sample website for cloud project<h1>' >/var/www/html/index.html**
- **Chconfig httpd on**
- **Service httpd start**
- **AFTER ENTERING THE FOLLOWING SCRIPT DON'T CHANGE ANYTHING AND JUST LAUNCH THE INSTANCE.**



- IN THE ABOVE PICTURE LOOK AT FIRST INSTANCE WE CAN SEE THE AUTOSCALING WEB SERVER.



- IN THIS PICTURE WE HAVE THE SAMPLE WEBSITE WHICH WE HAVE HOSTED.
- THE IP ADDRESS IN FIRST PICTURE AND THE SECOND ONE IS THE SAME.

## STEP 2:-

- CREATE AN IMAGE OF THE ABOVE INSTANCE.
- WE CREATE AN IMAGE BECAUSE WE USE THIS IMAGE TO ACTUALLY CREATE OUR AUTOSCALING GROUP RATHER THAN A NEW SERVER.

The screenshot shows the AWS EC2 Management Console. On the left sidebar, under the 'AMIs' section, there is a table listing an AMI named 'Project-Image'. The table includes columns for Name, AMI Name, AMI ID, Source, Owner, Visibility, Status, and Creation Date. The AMI ID is 'ami-0c8716b43f1ecd868'. The status is 'available' and it was created on November 25, 2019. Below the table, a modal window displays the details for the selected AMI, including its AMI ID ('ami-0c8716b43f1ecd868'), AMI Name ('Project-Image'), and other metadata.

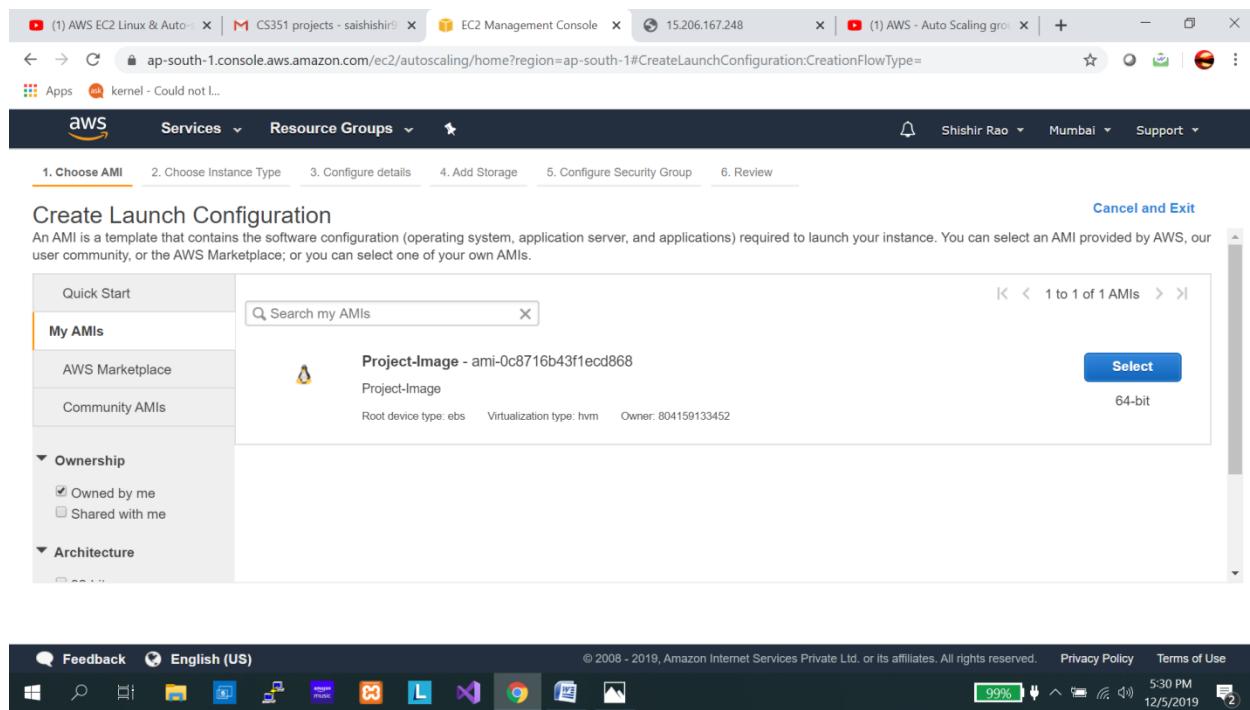
## - IMAGE OF MY WEB SERVER.

The screenshot shows the AWS EC2 Management Console. On the left sidebar, under the 'Instances' section, there is a table listing two instances. One instance has an Instance ID of 'i-07cc8a10481f45bb5' and another with 'i-0f44eac5663f678bc'. Both instances are in the 'running' state. A context menu is open over the first instance, with the 'Image' option highlighted. The 'Create Image' option is also visible in the menu. The public DNS for the second instance is 'ec2-15-206-167-248.ap-south-1.compute.amazonaws.com'.

## - THIS IS HOW WE GRAPHICALLY CREATE AN IMAGE.

## STEP 3:-

- **CREATE A LAUNCH CONFIGURATION.**
- **BEFORE CREATING AUTOSCALING GROUP WE NEED TO CREATE A LAUNCH CONFIGURATION.**
- **GO TO LAUNCH CONFIGURATION IN AWS AND SELECT CREATE LAUNCH CONFIGURATION AND SET IT TO T2.MICRO.**



- **SELECT THE AMI YOU HAVE CREATED RATHER THAN A LINUX SERVER.**
- **SELECT T2.MICRO IN TYPE AND ADD A NAME TO THE LAUNCH CONFIGURATION.**
- **DON'T CHANGE ANYTHING EXCEPT THE SECURITY SETTINGS.**

**Create Launch Configuration**

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group:  Create a **new** security group  Select an **existing** security group

**Security group name:** AutoScaling-Security-Group-3

**Description:** AutoScaling-Security-Group-3 (2019-12-05 17:30:33.379+05:30)

Type	Protocol	Port Range	Source
HTTP	TCP	80	Anywhere 0.0.0.0/0
SSH	TCP	22	Anywhere 0.0.0.0/0

**Add Rule**

**Warning:** This launch configuration does not have a security group assigned.

**Cancel** **Previous** **Review**

## - ADD THE ABOVE SECURITY SETTINGS AND CREATE THE LAUNCH CONFIGURATION.

**Create launch configuration** **Create Auto Scaling group** **Copy to launch template** **Actions**

**Launch Configuration: Project-launch**

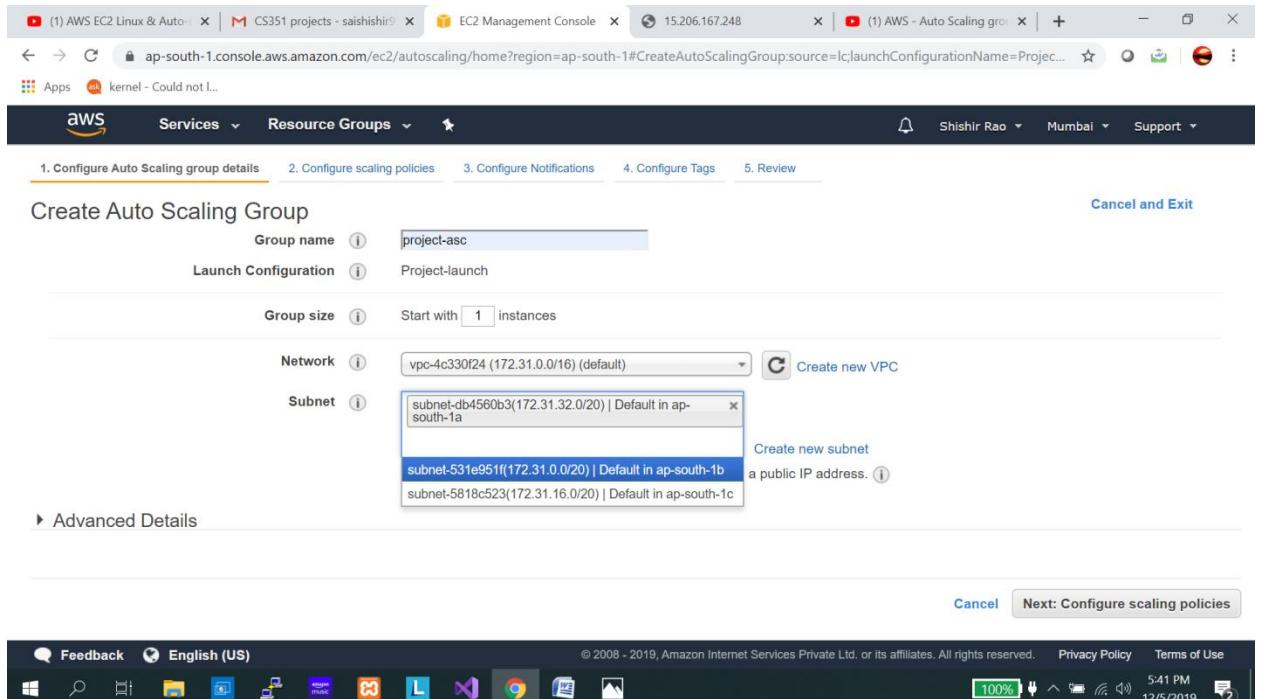
**Details**

AMI ID	ami-0c8716b43f1ecd868	Instance Type	t2.micro
IAM Instance Profile	vettel	Kernel ID	0dfc1453b3881b62b
EBS Optimized	false	Monitoring	false
Spot Price		Security Groups	sg-0dfc1453b3881b62b
RAM Disk ID		Creation Time	Mon Nov 25 12:49:24 GMT+530 2019
User data	-	Block Devices	/dev/xvda
		IP Address Type	Only assign a public IP address to instances launched in the default VPC and subnet. (default)

## - THIS IS MY LAUNCH CONFIGURATION.

## STEP 4:-

- **CREATE THE AUTOSCALING GROUP USING THE ABOVE LAUNCH CONFIGURATION.**
- **WE ONLY SET THE SCALING UP AND SCALING DOWN POLICIES IN THE AUTOSCALING GROUP AND SETUP NOTIFICATIONS.**



- **WHILE LAUNCHING WE SELECT THE SUBNETS IN WHICH OUR SERVER WILL WORK.( DONOT SELECT AP-SOUTH-1C, BECAUSE IT WILL NOT SUPPORT AUTOSCALING. ONLY SELECT 1A AND 1B)**
- **SET A GROUP NAME AND START WITH ANY NUMBER OF INSTANCES AS U WANT. I AM STARTING WITH ONE INSTANCE.**

**Increase Group Size**

**Name:** Increase Group Size

**Execute policy when:** awsec2-project-asc-CPU-Utilization Edit Remove  
breaches the alarm threshold: CPUUtilization >= 30 for 60 seconds  
for the metric dimensions AutoScalingGroupName = project-asc

**Take the action:** Add ▾ 0 capacity units ▾ when 30 <= CPUUtilization < +infinity  
Add step ⓘ

**Instances need:** 3 seconds to warm up after each step

[Create a simple scaling policy ⓘ](#)

[Cancel](#) [Previous](#) [Review](#) [Next: Configure Notifications](#)

- **WE CREATE A SCALING UP POLICY HERE, THAT IS WE DECIDE AFTER WHAT PARAMETER WILL WE LAUNCH INSTANCES OR VIRTUAL MACHINES SO AS TO COMPLY WITH THE SYSTEM.**

**Decrease Group Size**

**Name:** Decrease Group Size

**Execute policy when:** awsec2-Project-Auto-High-CPU-Utilization C Add new alarm  
breaches the alarm threshold: CPUUtilization <= 20 for 60 seconds  
for the metric dimensions AutoScalingGroupName = Project-Auto

**Take the action:** Remove ▾ 0 capacity units ▾ when 20 >= CPUUtilization > -infinity  
Add step ⓘ

[Create a simple scaling policy ⓘ](#)

[Scale the Auto Scaling group using a target tracking scaling policy ⓘ](#)

[Cancel](#) [Previous](#) [Review](#) [Next: Configure Notifications](#)

- HERE WE HAVE CREATED THE SCALING DOWN POLICY SO AS TO WHEN WE SHOULD TERMINATE THE INSTANCES WHICH ARE NO MORE REQUIRED BECAUSE THE AVERAGE CPU UTILIZATION IS UNDER MARK AND OK.
- ONLY FOR DEMO PURPOSE WE HAVE SET THE AVG UTILIZATION AS 30 FOR SCALE UP AND 20 FOR SCALE DOWN.
- WE ALSO PUT THE TIME AS 1 MINUTE BECAUSE THIS IS A DEMO AND WE DON'T WANT TO WASTE MUCH TIME.

The screenshot shows the AWS EC2 Management Console interface for creating an Auto Scaling group. The user is on step 3 of the wizard, "Configure Notifications". The page title is "Create Auto Scaling Group". It displays a list of notification topics, with one selected: "MyProjectNotifications (shishir27b@gmail.com)". Below this, there's a section titled "Whenever instances:" with checkboxes for "launch", "terminate", "fail to launch", and "fail to terminate". A blue "Add notification" button is located below this section. At the bottom of the page, there are navigation buttons: "Cancel", "Previous", "Review", and "Next: Configure Tags". The browser's address bar shows the URL: "ap-south-1.console.aws.amazon.com/ec2/autoscaling/home?region=ap-south-1#CreateAutoScalingGroup:source=lc|launchConfigurationName=Projec...". The top of the browser window shows other tabs and the AWS logo.

- WE THE SET THE NOTIFICATIONS WITH EMAIL, SO THAT EVERYTIME A SERVER IS LAUNCHED OR TERMINATED WE GET A NOTIFICATION FROM AWS.
- LAUNCH THE AUTOSCALING GROUP.

Try the new design for Amazon EC2 Auto Scaling  
This older console is being replaced with the new EC2 Auto Scaling console. No new features or improvements will be made in this older console. Go to the new console.

Name	Launch Configuration	Instances	Desired	Min	Max	Availability Zones	Default Cooldown	Health
Project-Auto	Project-launch	1	1	1	4	ap-south-1b, ap-south-1a	300	56

Auto Scaling Group: Project-Auto

Details Activity History Scaling Policies Instances Monitoring Notifications Tags Scheduled Actions Lifecycle Hooks

Launch Configuration: Project-launch Availability Zone(s): ap-south-1b, ap-south-1a Subnet(s): subnet-db4560b3

## - AUTOSCALING GROUP HAS BEEN LAUNCHED

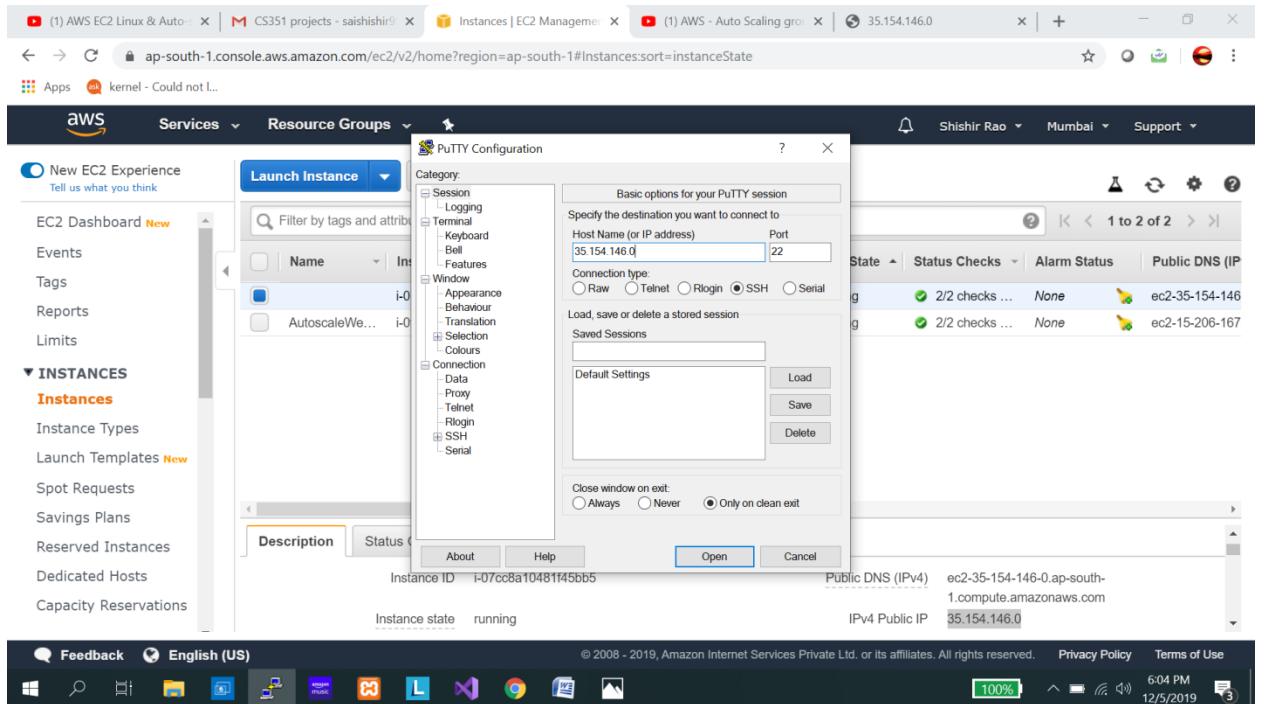
Filter: Filter Auto Scaling groups... 1 to 1 of 1 Auto Scaling Groups

Instance ID	Lifecycle	Launch Configuration / Template	Availability Zone	Health Status	Protected from
i-0f44eac5663f678bc	InService	Project-launch	ap-south-1a	Healthy	

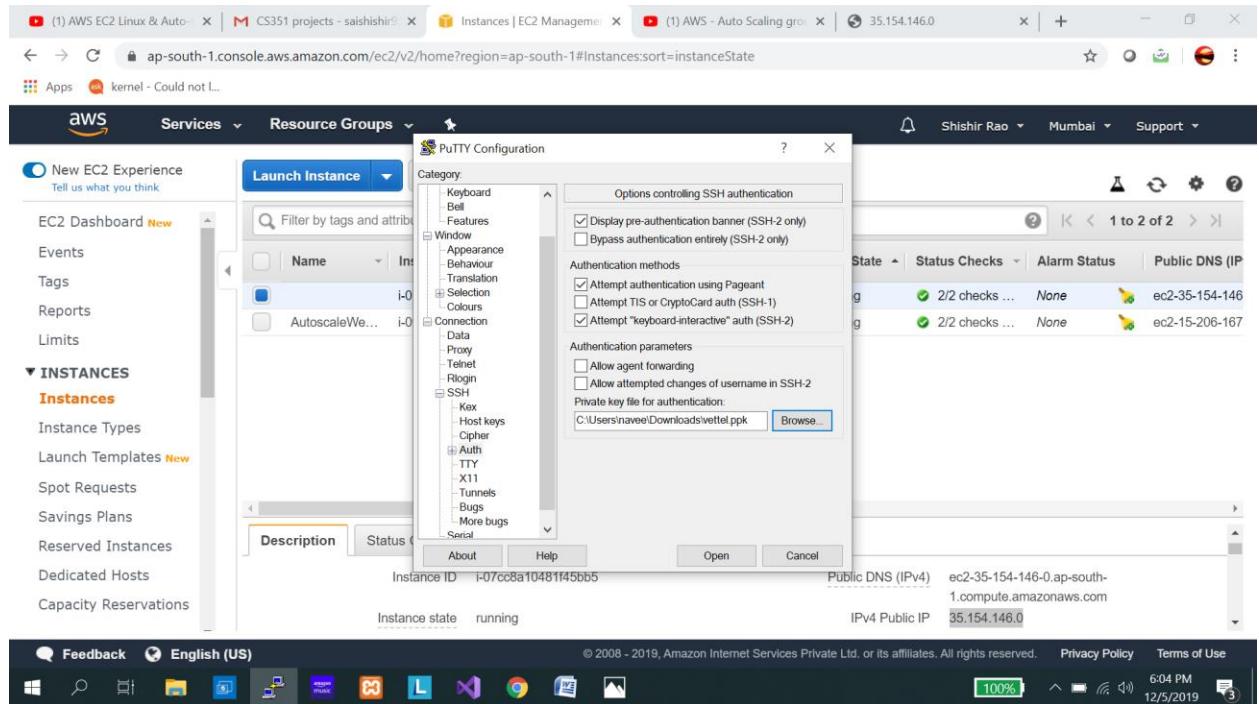
- **ONE INSTANCE HAS ALREADY BEEN LAUNCHED BECAUSE WE HAVE MENTIONED IT ALREADY THAT WE WILL START WITH ONE INSTANCE.**

## **STEP 5:-**

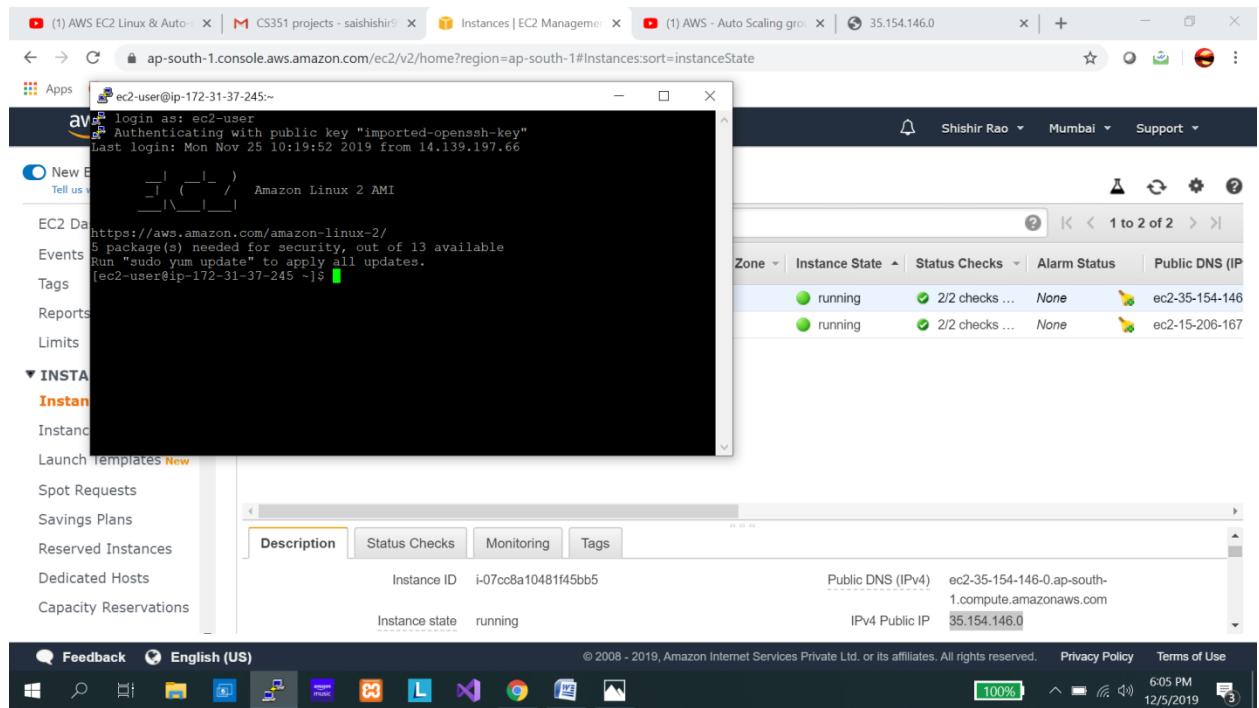
- **USE YOUR SSH KEY TO LOGIN INTO THE SERVER USING PPK AND PEM FILES.**



- **USE PUTTY AND SET THE IP ADDRESS TO THE ONE IN YOUR WEB SERVER INSTANCE.**



- WE ALSO PUT THE PPK FILE WHICH WE HAVE CREATED FROM PEM FILE.
- WE THEN CLICK OPEN.



- LOGIN AS “ec2-user”.

## STEP 6:-

- INCREASE THE LOAD ON WEB SERVER.
- WE USE THE FOLLOWING SCRIPT TO DO IT SO.
- WE CREATE A FILE WITH NAME INCREASE.SH
- dd if=/var/zero of=/dev/null bs=50000 count=10000
- PUT THE ABOVE LINE INTO THE FILE AND THEN RUN IT UNTILL THE LOAD INCREASES.
- WE USE THE “TOP” COMMAND TO FIND OUT THE CPU UTILIZATION.

```

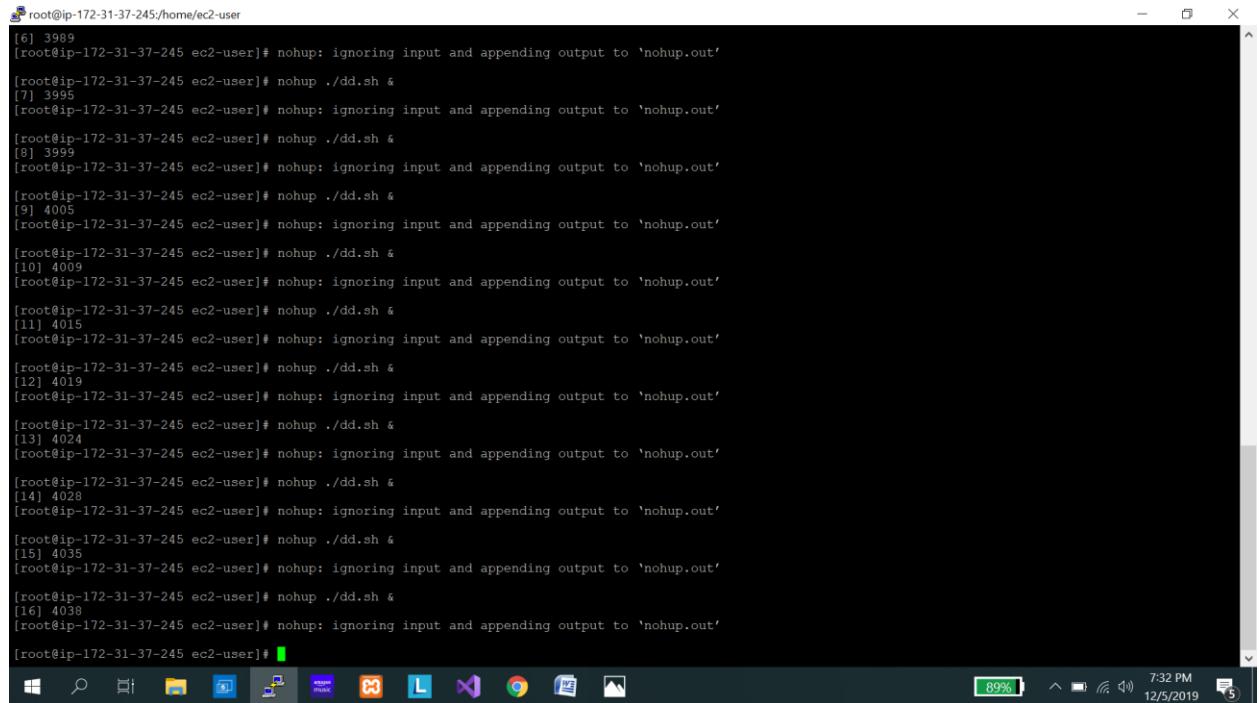
top - 13:54:19 up 2:21, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 93 total, 1 running, 56 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
Kib Mem: 1007276 total, 742028 free, 85424 used, 179824 buff/cache
Kib Swap: 0 total, 0 free, 0 used. 777752 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
3250 root 20 0 216568 6260 5364 S 0.3 0.6 0:00.39 rsyslogd
3640 apache 20 0 296464 6160 3776 S 0.3 0.6 0:00.60 httpd
3932 root 20 0 171056 4304 3780 R 0.3 0.4 0:00.01 top
1 root 20 0 43596 5208 3844 S 0.0 0.5 0:01.69 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 mm_percpu_wq
7 root 20 0 0 0 0 S 0.0 0.0 0:00.02 ksoftirqd/0
8 root 20 0 0 0 0 I 0.0 0.0 0:00.13 rcu_sched
9 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_bh
10 root rt 0 0 0 0 S 0.0 0.0 0:00.00 migration/0
11 root rt 0 0 0 0 S 0.0 0.0 0:00.01 watchdog/0
12 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/0
14 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kdevtmpfs
15 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 netns
16 root 20 0 0 0 0 I 0.0 0.0 0:00.03 kworker/u30:1
21 root 20 0 0 0 0 S 0.0 0.0 0:00.00 xenbus
22 root 20 0 0 0 0 S 0.0 0.0 0:00.01 xenwatch
172 root 20 0 0 0 0 S 0.0 0.0 0:00.00 khungtaskd
173 root 20 0 0 0 0 S 0.0 0.0 0:00.00 oom_reaper
174 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 writeback
176 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kcompactd0
177 root 25 5 0 0 0 S 0.0 0.0 0:00.00 ksm
178 root 39 19 0 0 0 S 0.0 0.0 0:00.00 khugepaged
179 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 crypto
180 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 integrityd
182 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kblockd
535 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 md
538 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 edac-poller
543 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 watchdogd
684 root 20 0 0 0 0 S 0.0 0.0 0:00.01 kaudiod
690 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kswapd0
822 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kthrotld
872 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kstpp
900 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 ipv6_addrconf
1737 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 ata_sff
1754 root 20 0 0 0 0 S 0.0 0.0 0:00.00 scsi_eh_0

```

- WE CAN SEE THAT THE CPU UTILIZATION IS ZERO WHEN THERE IS LOAD ON THE SERVER.
- NOW CHANGE THE SCRIPT TO EXECUTABLE FORM AND RUN IT.

- **CHMOD 755 INCREASE.SH**
- **THE COMMAND TO EXECUTE IS “nohup ./increase.sh &”**
- **EXECUTE THE COMMAND AS MANY TIMES AS YOU CAN SO AS TO INCREASE THE LOAD FROM 0 TO SOMETHING MORE THAN 30.**



The screenshot shows a Windows Command Prompt window titled 'root@ip-172-31-37-245/home/ec2-user'. The window contains a series of command-line entries, each starting with '[root@ip-172-31-37-245 ec2-user]# nohup' followed by a file name like 'dd.sh' or 'increase.sh' and '&'. The process IDs (PIDs) for these nohup processes range from 3989 to 4038. The taskbar at the bottom of the screen shows various icons for common Windows applications such as File Explorer, Task View, and the Start button.

```

root@ip-172-31-37-245/home/ec2-user
[6] 3989
[root@ip-172-31-37-245 ec2-user]# nohup: ignoring input and appending output to 'nohup.out'
[root@ip-172-31-37-245 ec2-user]# nohup ./dd.sh &
[7] 3995
[root@ip-172-31-37-245 ec2-user]# nohup: ignoring input and appending output to 'nohup.out'
[root@ip-172-31-37-245 ec2-user]# nohup ./dd.sh &
[8] 3999
[root@ip-172-31-37-245 ec2-user]# nohup: ignoring input and appending output to 'nohup.out'
[root@ip-172-31-37-245 ec2-user]# nohup ./dd.sh &
[9] 4005
[root@ip-172-31-37-245 ec2-user]# nohup: ignoring input and appending output to 'nohup.out'
[root@ip-172-31-37-245 ec2-user]# nohup ./dd.sh &
[10] 4009
[root@ip-172-31-37-245 ec2-user]# nohup: ignoring input and appending output to 'nohup.out'
[root@ip-172-31-37-245 ec2-user]# nohup ./dd.sh &
[11] 4015
[root@ip-172-31-37-245 ec2-user]# nohup: ignoring input and appending output to 'nohup.out'
[root@ip-172-31-37-245 ec2-user]# nohup ./dd.sh &
[12] 4019
[root@ip-172-31-37-245 ec2-user]# nohup: ignoring input and appending output to 'nohup.out'
[root@ip-172-31-37-245 ec2-user]# nohup ./dd.sh &
[13] 4024
[root@ip-172-31-37-245 ec2-user]# nohup: ignoring input and appending output to 'nohup.out'
[root@ip-172-31-37-245 ec2-user]# nohup ./dd.sh &
[14] 4028
[root@ip-172-31-37-245 ec2-user]# nohup: ignoring input and appending output to 'nohup.out'
[root@ip-172-31-37-245 ec2-user]# nohup ./dd.sh &
[15] 4035
[root@ip-172-31-37-245 ec2-user]# nohup: ignoring input and appending output to 'nohup.out'
[root@ip-172-31-37-245 ec2-user]# nohup ./dd.sh &
[16] 4038
[root@ip-172-31-37-245 ec2-user]# nohup: ignoring input and appending output to 'nohup.out'
[root@ip-172-31-37-245 ec2-user]#

```

- **RUNNING THE COMMAND SO MANY TIMES.**

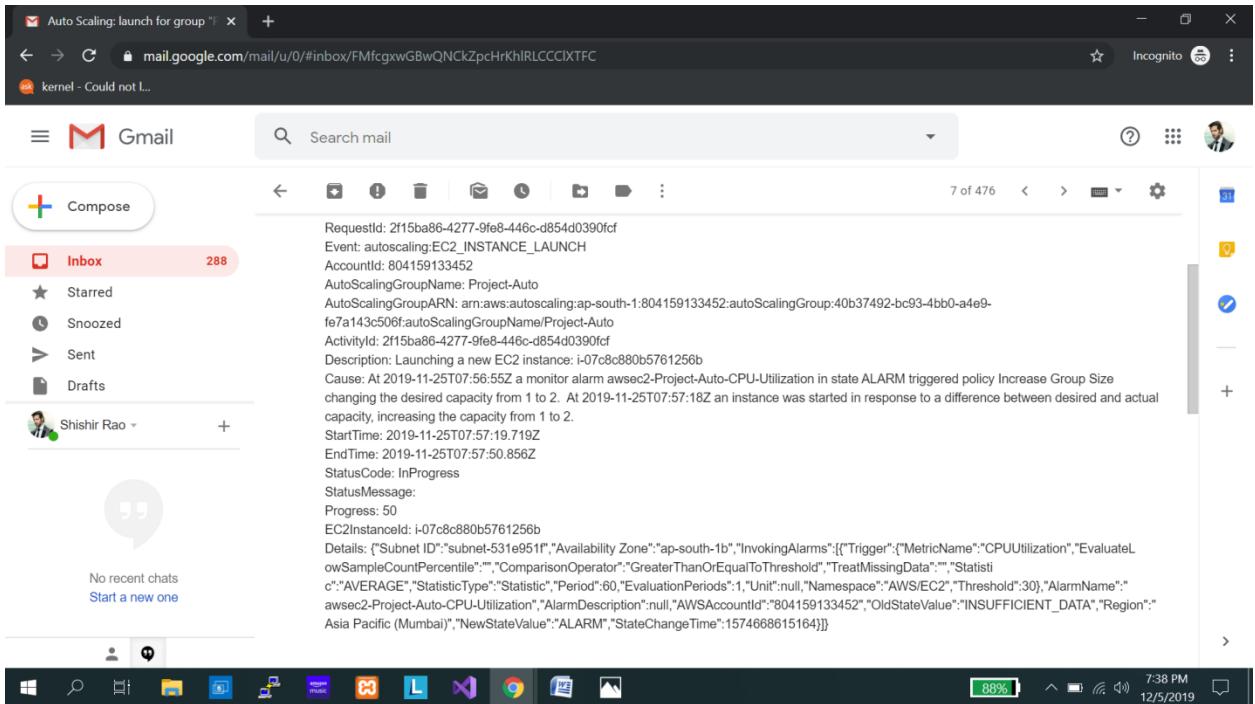
```

root@ip-172-31-37-245:/home/ec2-user
top - 14:02:33 up 2:30, 1 user, load average: 4.87, 1.14, 0.38
Tasks: 125 total, 17 running, 72 sleeping, 0 stopped, 0 zombie
%Cpu(s): 18.0 us, 82.0 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 1007276 total, 732836 free, 94384 used, 180056 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 768740 avail Mem

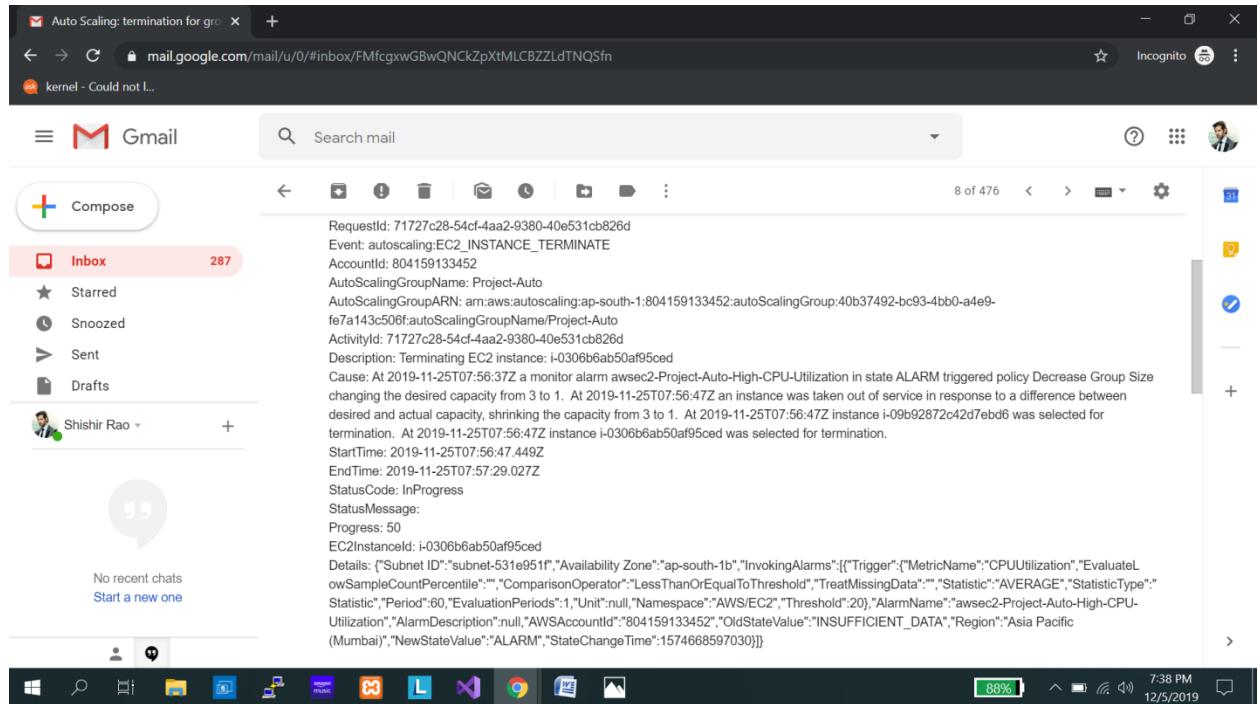
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
4059 root 20 0 114680 792 728 R 6.7 0.1 0:00:39 dd
4061 root 20 0 114680 764 700 R 6.7 0.1 0:00:31 dd
4057 root 20 0 114680 800 736 R 6.3 0.1 0:00:40 dd
4062 root 20 0 114680 760 696 R 6.3 0.1 0:00:26 dd
4063 root 20 0 114680 796 732 R 6.3 0.1 0:00:23 dd
4058 root 20 0 114680 784 716 R 6.0 0.1 0:00:39 dd
4064 root 20 0 114680 764 700 R 6.0 0.1 0:00:20 dd
4065 root 20 0 114680 784 724 R 5.7 0.1 0:00:17 dd
4066 root 20 0 114680 820 756 R 5.0 0.1 0:00:15 dd
4069 root 20 0 114680 836 772 R 4.0 0.1 0:00:12 dd
4067 root 20 0 114680 792 728 R 3.7 0.1 0:00:11 dd
4068 root 20 0 114680 808 744 R 3.7 0.1 0:00:11 dd
4070 root 20 0 114680 828 764 R 3.7 0.1 0:00:11 dd
4072 root 20 0 114680 724 656 R 1.0 0.1 0:00:03 dd
4073 root 20 0 114680 824 764 R 0.7 0.1 0:00:02 dd
4060 root 20 0 171020 4268 3704 R 0.3 0.4 0:00:01 top
1 root 20 0 43596 5208 3844 S 0.0 0.5 0:01:70 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00:00 kthreadd
4 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 kworker/0:0H
6 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 mm_percpu_wq
7 root 20 0 0 0 0 S 0.0 0.0 0:00:03 ksoftirqd/0
8 root 20 0 0 0 0 I 0.0 0.0 0:00:14 rcu_sched
9 root 20 0 0 0 0 I 0.0 0.0 0:00:00 rcu_bh
10 root rt 0 0 0 0 S 0.0 0.0 0:00:00 migration/0
11 root rt 0 0 0 0 S 0.0 0.0 0:00:01 watchdog/0
12 root 20 0 0 0 0 S 0.0 0.0 0:00:00 cpuhp/0
14 root 20 0 0 0 0 S 0.0 0.0 0:00:00 kdevtmpfs
15 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 netns
16 root 20 0 0 0 0 I 0.0 0.0 0:00:04 kworker/u30:1
21 root 20 0 0 0 0 S 0.0 0.0 0:00:00 xenbus
22 root 20 0 0 0 0 S 0.0 0.0 0:00:01 xenwatch
172 root 20 0 0 0 0 S 0.0 0.0 0:00:00 khungtaskd
173 root 20 0 0 0 0 S 0.0 0.0 0:00:00 oom_reaper
174 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 writeback
176 root 20 0 0 0 0 S 0.0 0.0 0:00:00 kcompactd0
177 root 25 5 0 0 0 S 0.0 0.0 0:00:00 ksm
178 root 39 19 0 0 0 S 0.0 0.0 0:00:00 khugepaged

```

- WE CAN SEE THAT THE CPU UTILIZATION HAS GONE UP FROM 0 TO 18.
- EXECUTE UNTILL IT CROSSES 30 AND STAYS THERE FOR ATLEAST A MINUTE.
- ONCE IT CROSSES 30 AND STAYS THERE FOR 1 MINUTE YOU WILL RECEIVE NOTIFICATION TO YOUR MAIL ADDRESS.



- **MAIL I RECEIVED REGARDING THE INSTANCE LAUNCH.**
- **ONCE THE CPU COMES BELOW 20 AND STAYS FOR ONE MINUTE YOU WILL RECEIVE ANOTHER NOTIFICATION REGARDING INSTANCE TERMINATION.**



- **NOTIFICATION WE RECEIVE REGARDING INSTANCE TERMINATION.**

## CONCLUSION:-

- **WE HAVE FINALLY SIMULATED THE ENTIRE HYPERVISOR BY CONSIDERING THE ONLINE SERVER AS OUR SYSTEM AND INSTANCES AS VIRTUAL MACHINES.**
- **WE ALSO MADE IT INTELLIGENT BECAUSE WE NEVER LAUNCHED AN INSTANCE OR TERMINATED ANY WE JUST INCREASED THE CPU UTILIZATION AND IT AUTOMATICALLY LAUNCHED BY ITSELF.**