Auto scaling group-Launch Configuration

Prepared by Sriganesh Pera

Launch Configurations:

A launch configuration is a template that an Auto Scaling group uses to launch EC2 instances.

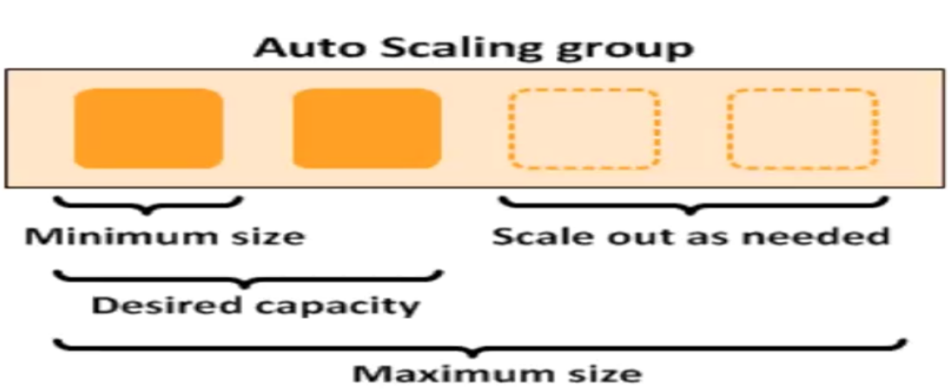
When you create an Auto Scaling group, you must specify a launch configuration. You can specify your launch configuration with multiple Auto Scaling groups. However, you can only specify one launch configuration for an Auto Scaling group at a time, and you can't modify a launch configuration after you've created it.

Therefore, if you want to change the launch configuration for your Auto Scaling group, you must create a launch configuration and then update your Auto Scaling group with the new launch configuration.

Auto Scaling:

Auto Scaling helps you maintain **application availability and allows you to scale your Amazon EC2** capacity up or down automatically per conditions you define.

Auto Scaling can also automatically increase the number of Amazon EC2 instances during demand spikes to maintain performance and decrease capacity during lulls to reduce costs



Example:

My application to be live – Minimum 1 server
My application to perform normal – Desired 3 servers
My application to perform during peak season – Maximum 20 servers
During peak load – Scale out by - 2 Servers every 10 minutes

AWS Auto Scaling components?

1. Launch Configuration (Template)

- AMI (Ex: Amazon Linux, Redhat Linux)
- Instance type (Ex: t2.micro, m4.large)
- Storage
- Security Group
- SSH-Key pair

2. Autoscaling Group

- Launch Configuration
- Network/Subnet's
- Scaling policies for increase
- Scaling policies for decrease
- Monitoring & Alarm

Scenario:

Auto-scaling test scenario

- Build webserver
- Create AMI (image)
- Create Launch configuration using webserver AMI
- Create Autoscale Group with minimum 1 instance, increase by 1 & maximum 4.
- Run the following script to Increase CPU load above 50%

```
vi dd.sh
dd if=/dev/zero of=/dev/null bs=50000 count=100000
dd if=/dev/zero of=/dev/null bs=50000 count=100000
dd if=/dev/zero of=/dev/null bs=50000 count=100000
dd if=/dev/zero of=/dev/null bs=50000 count=100000
dd if=/dev/zero of=/dev/null bs=50000 count=100000
dd if=/dev/zero of=/dev/null bs=50000 count=100000
dd if=/dev/zero of=/dev/null bs=50000 count=100000
:wq!
```

dd if=/dev/zero of=/dev/null bs=50000 count=100000

Lab :Launching an EC2 instance with auto sclaing group and notice hpw AWS automatically increase EC2 instance when the load is up and decrease EC2 instances when load is down

<u>Step 1: Login to AWS console and create an EC2 instance and install apache webserver</u>

As you know how to create an EC2 instance we straight away connect the instance and install Apache webserver.

Type in command prompt:

sudo yum update -y

sudo yum install -y httpd24 php70 mysql56-server php70-mysqlnd

sudo service httpd start

sudo chkconfig httpd on

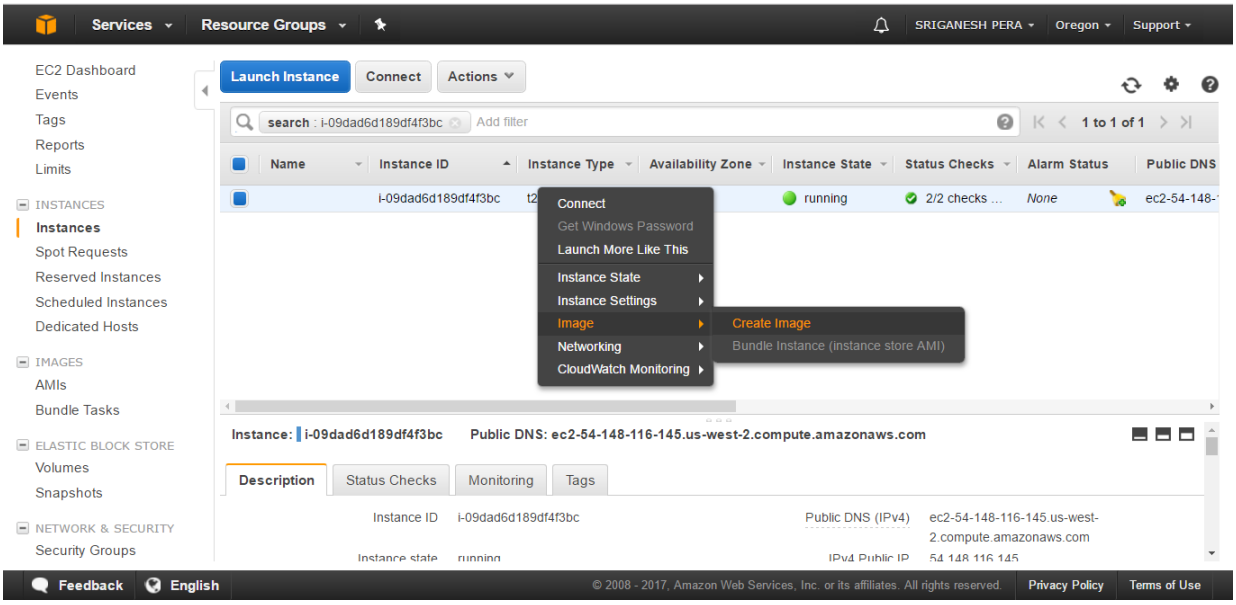
Now we have installed Apache webserver on our instace.

```
ec2-user@ip-172-31-19-112:~
  mysql-config.x86_64 0:5.5.54-1.16.amzn1
  mysql56.x86_64 0:5.6.35-1.23.amzn1
  mysql56-common.x86_64 0:5.6.35-1.23.amzn1
  mysql56-errmsg.x86_64 0:5.6.35-1.23.amzn1
  mysql56-libs.x86_64 0:5.6.35-1.23.amzn1
  perl-Compress-Raw-Bzip2.x86_64 0:2.061-3.11.amzn1
  perl-Compress-Raw-Zlib.x86_64 1:2.061-4.1.amzn1
  perl-DBD-MySQL56.x86_64 0:4.023-5.21.amzn1
  perl-DBI.x86_64 0:1.627-4.8.amzn1
  perl-Data-Dumper.x86_64 0:2.145-3.5.amzn1
  perl-IO-Compress.noarch 0:2.061-2.12.amzn1
  perl-Net-Daemon.noarch 0:0.48-5.5.amzn1
  perl-PlRPC.noarch 0:0.2020-14.7.amzn1
  php70-cli.x86_64 0:7.0.16-1.21.amzn1
  php70-common.x86_64 0:7.0.16-1.21.amzn1
  php70-json.x86_64 0:7.0.16-1.21.amzn1
  php70-process.x86_64 0:7.0.16-1.21.amzn1
  php70-xml.x86_64 0:7.0.16-1.21.amzn1

Complete!
[ec2-user@ip-172-31-19-112 ~]$ sudo service httpd start
Starting httpd:                                            [  OK  ]
[ec2-user@ip-172-31-19-112 ~]$ sudo chkconfig httpd on
[ec2-user@ip-172-31-19-112 ~]$
```

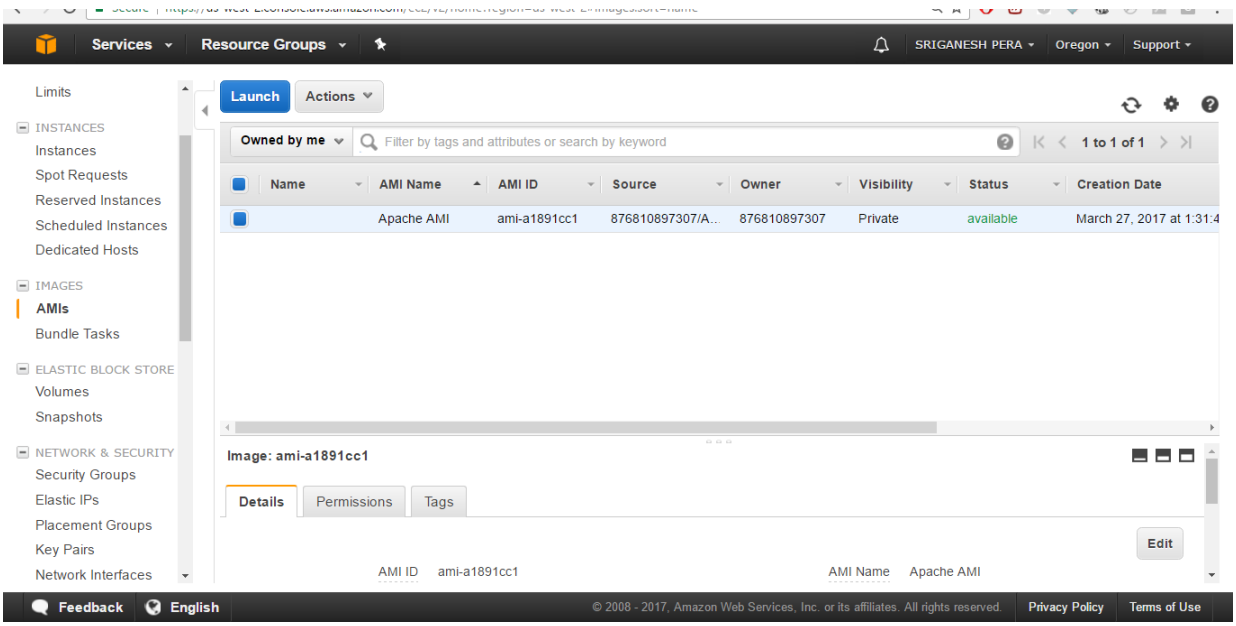
Now we have to ***<u>create an AMI</u>*** for this EC2 instance

Go to EC2 dash board and right click on the instance to create an image



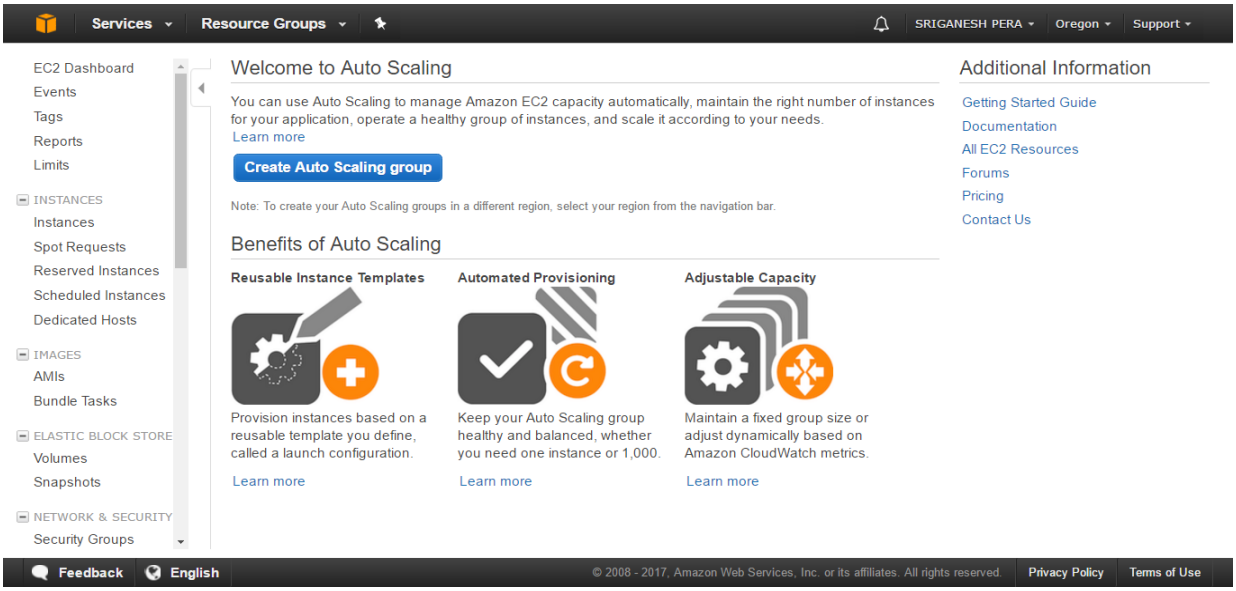
Now we have created an AMI with pre-loaded Apache webserver on it

Which can be found in EC2 dashboard under AMI section

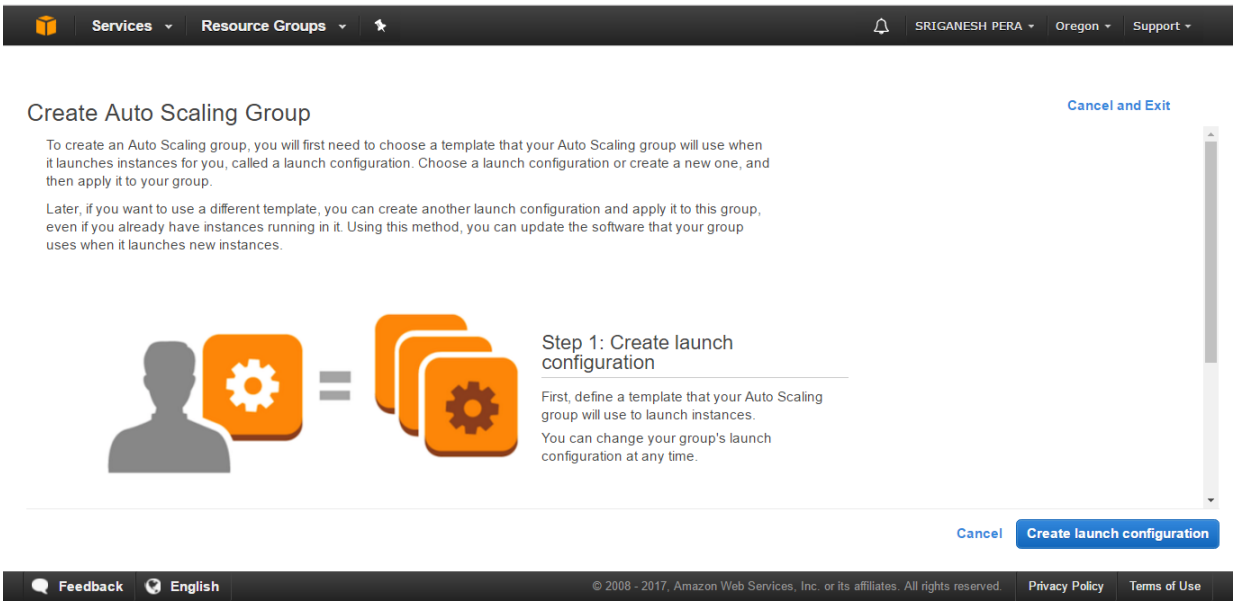


So as we know this AMI contains Apache webserver pre installed in it.After AMI is created terminate the instance.

***Step 2*: Create Launch configuration which can be found in EC2 dashboard and click on Create Auto Scaling group**

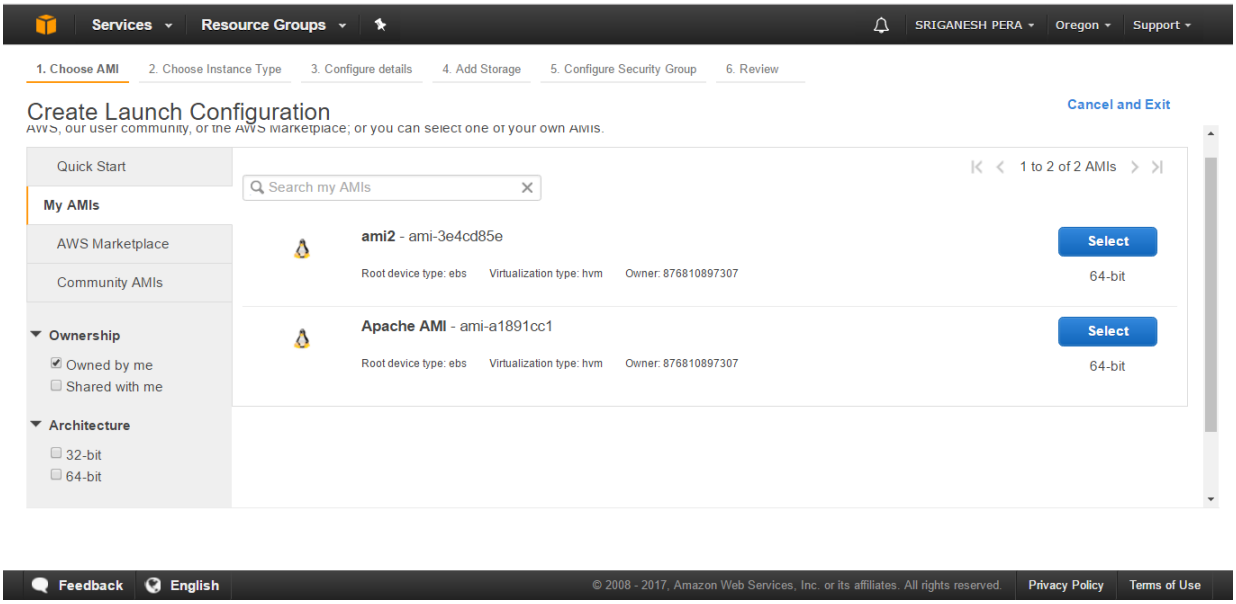


Step 3: Click on launch configuration

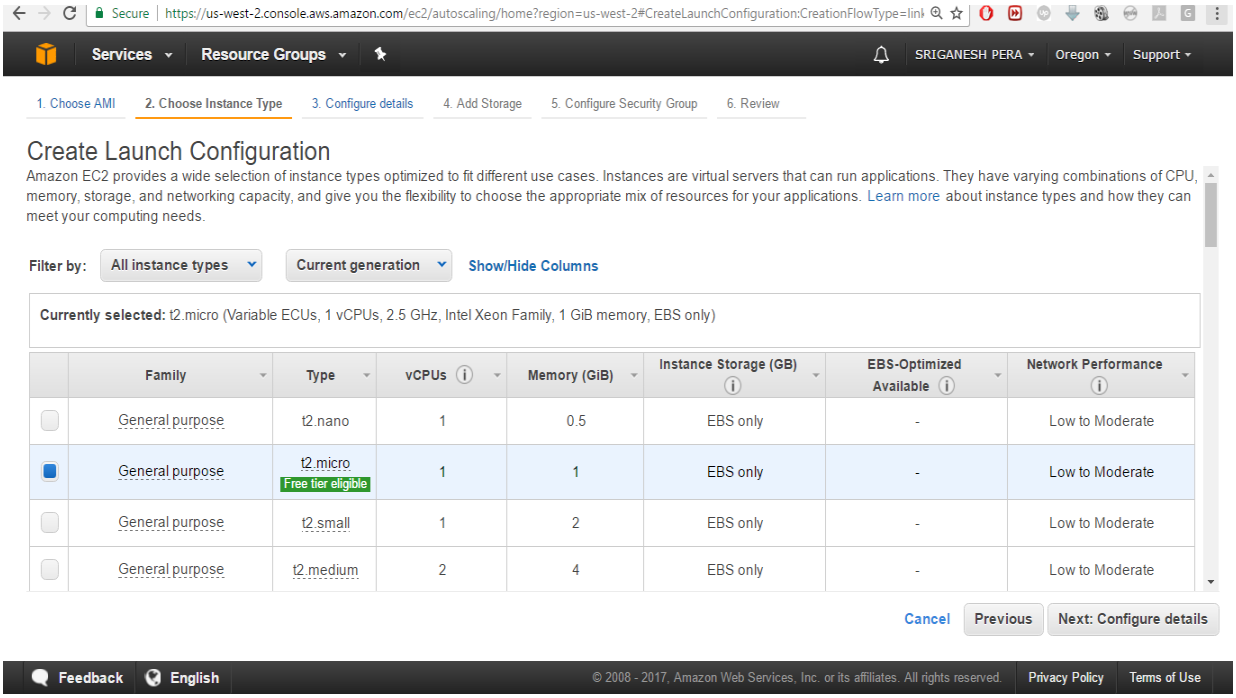


Now this launch configuration is a template which launches an EC2 instance and also allows Auto scaling group associate with it.

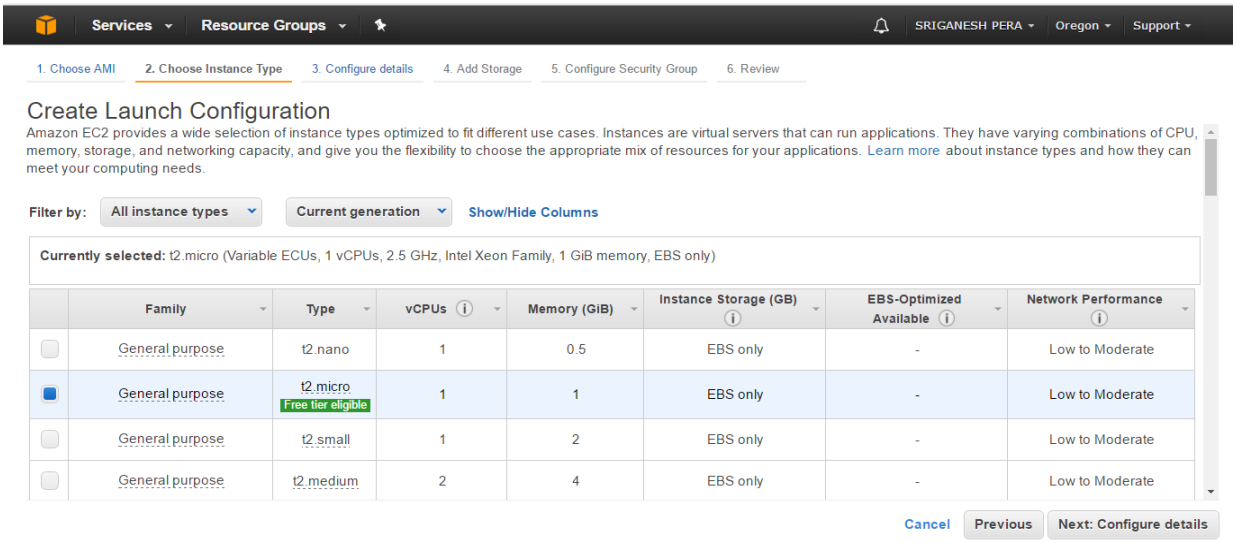
Step 4: Click on my AMI and choose your AMI you just created.Here I choose Apache AMI



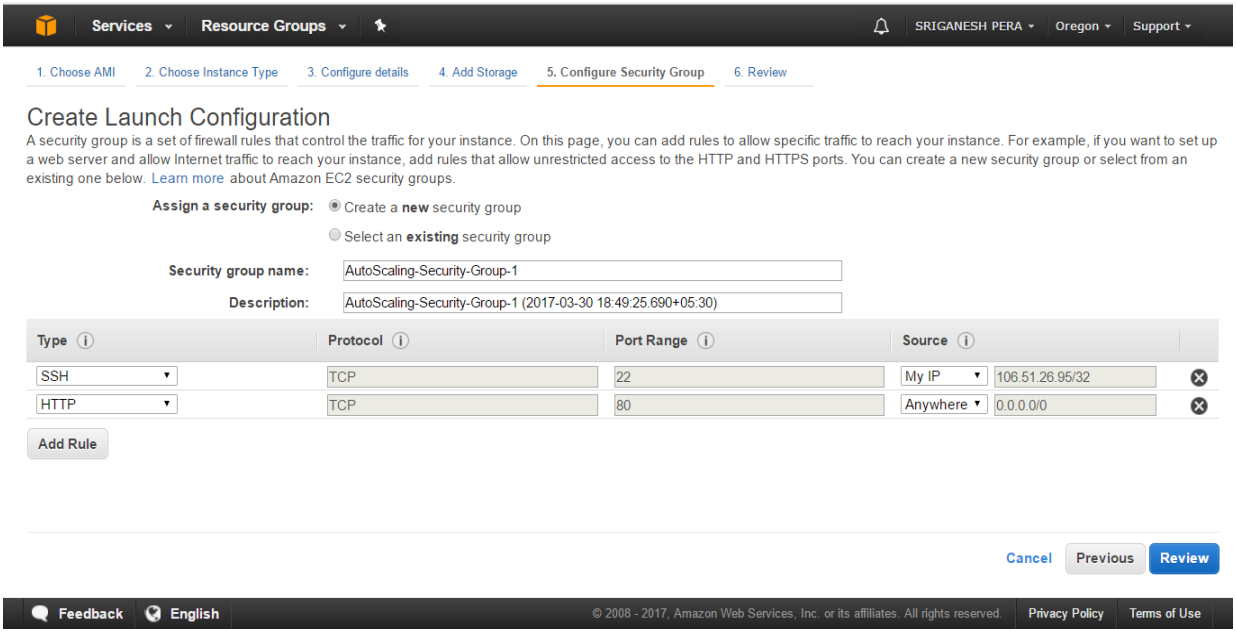
Step 5:Select t2.micro instace which is a free tier



Step 6:Name your launch configuration and click next:Addstorage and also click on Next:Security configuration



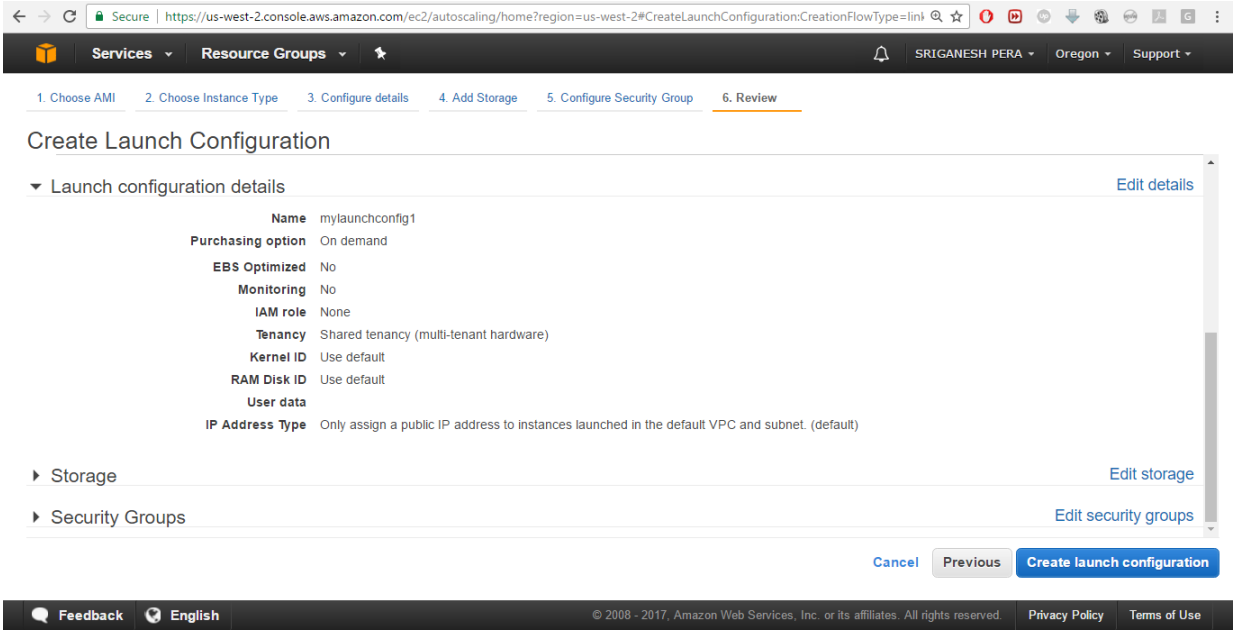
Step 7:Configure security settings and select create new security group and keep settings as shown in this screenshot and click on review



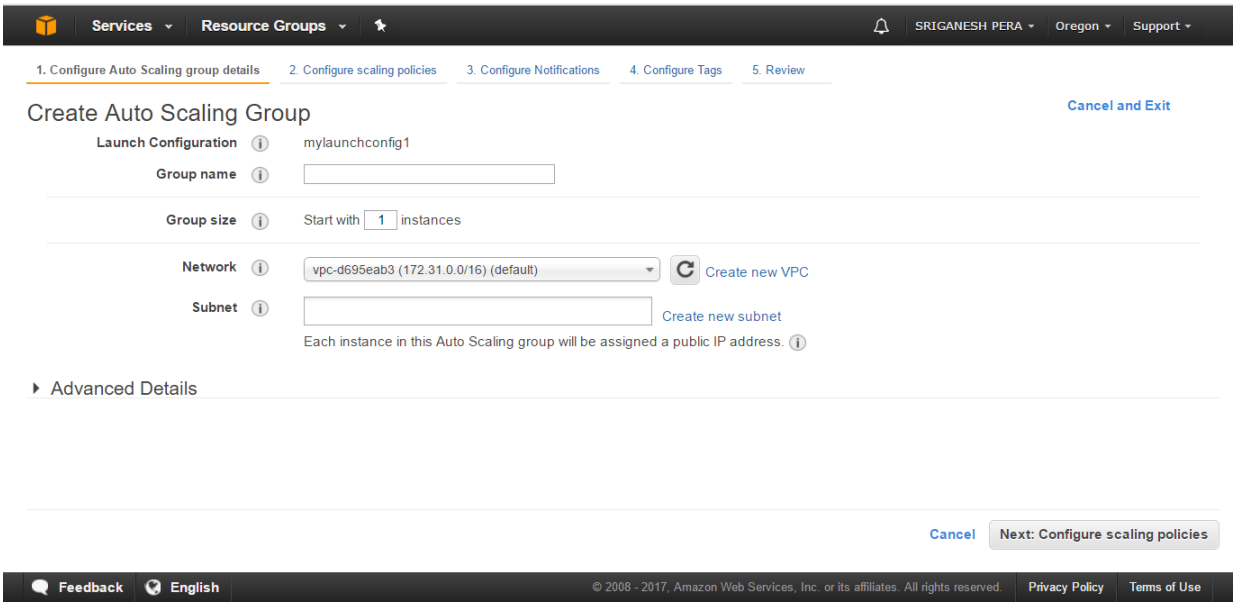
HTTP port 80 makes sure the client/server interaction is availiable

SSH my IP suggests that only I can securely login to my command prompt

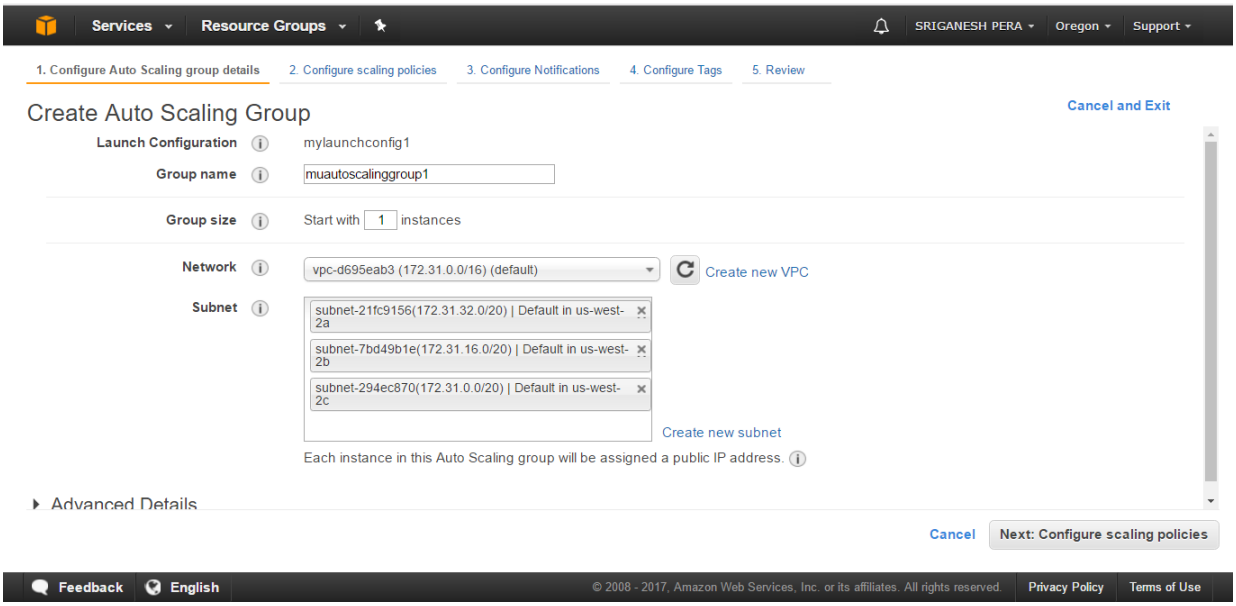
Step 8:Review whatever the settings you made and click on Create Launch conifguration and it asks for a key.Choose a key and click on Create Launch configuration



Now as soon as you click on Create launch configuration it takes you to Auto scaling group for setting up policies for you instance as you can see in the screenshot.

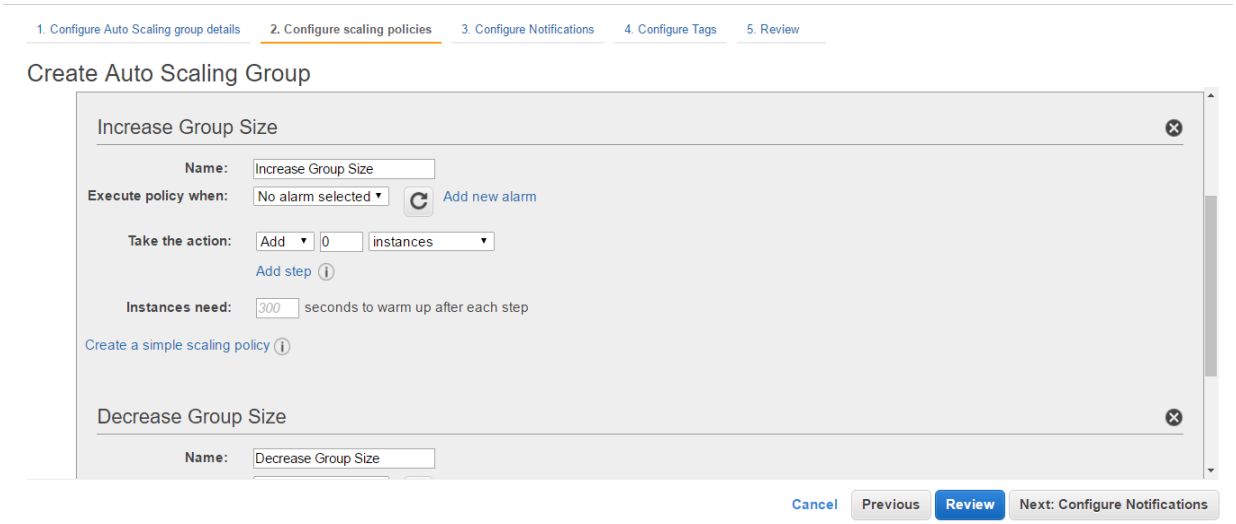


Step 9: Create a name for you Auto scaling group and choose default VPC and choose all Avaliability zones for your subnet and then click Next



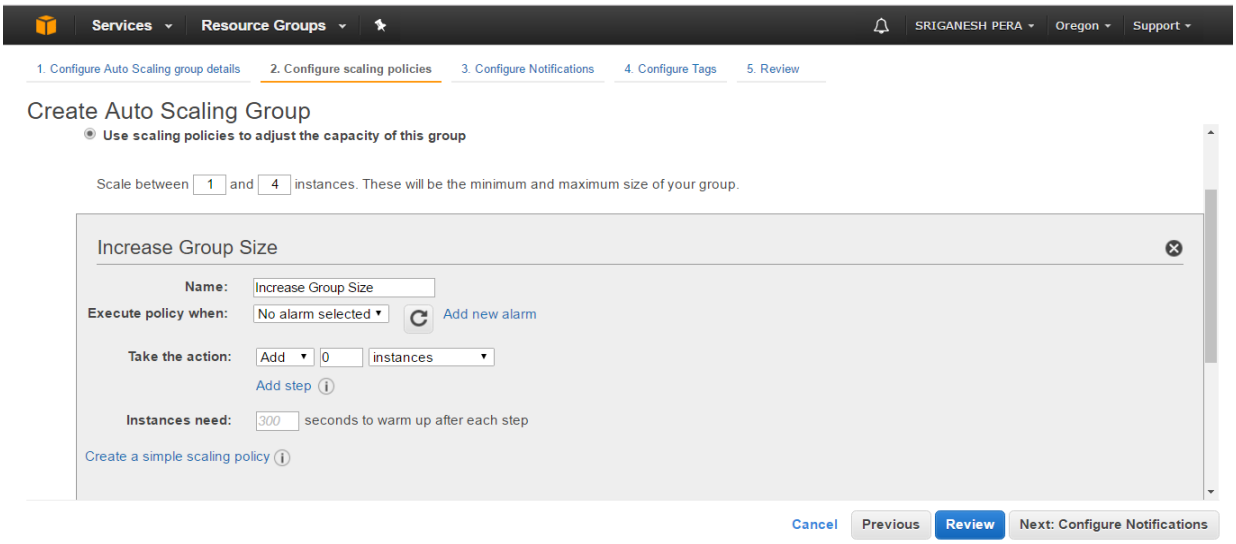
Step 10: Now choose your scaling policy and click on Use scaling policies to adjust the capacity of this group.

This is a very important where in you choose number of instances for scaling up and scaling down which keeps your instance highly avalible when there is a increase/decrease of workload.

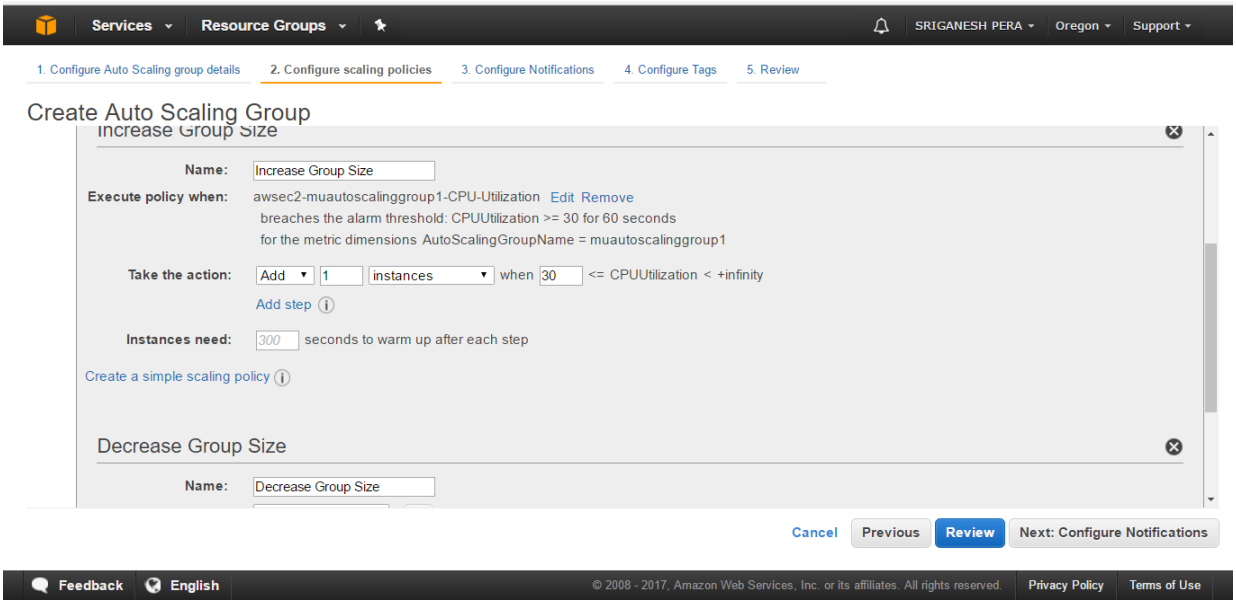


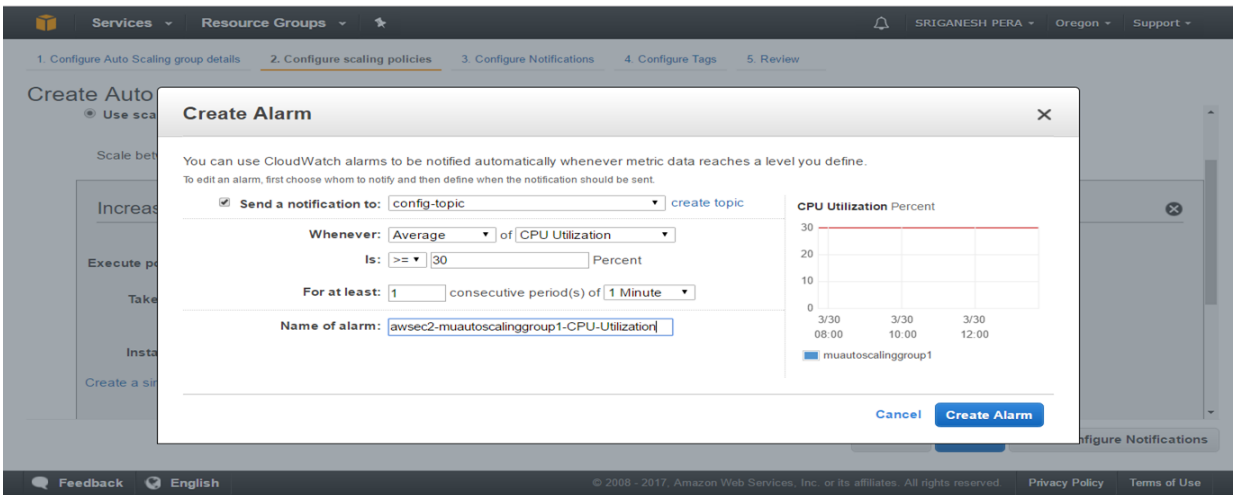
As you can see I have two polices that I need to choose 'Increase group size' when there is a load spike and 'decrease group size' when the load is less.

Step 11: Choose Scale between 1 and 4 instances for this lab.This means when the load is normal 1 instance works and when the load increases by 30% or 50%(CPU utilization rate) it launches upto 4 instances



Step 12: Increase group size by 1 in the take action field and click on create an alarm and set CPU utilization to 30% for every 1 minute and click on create alarm

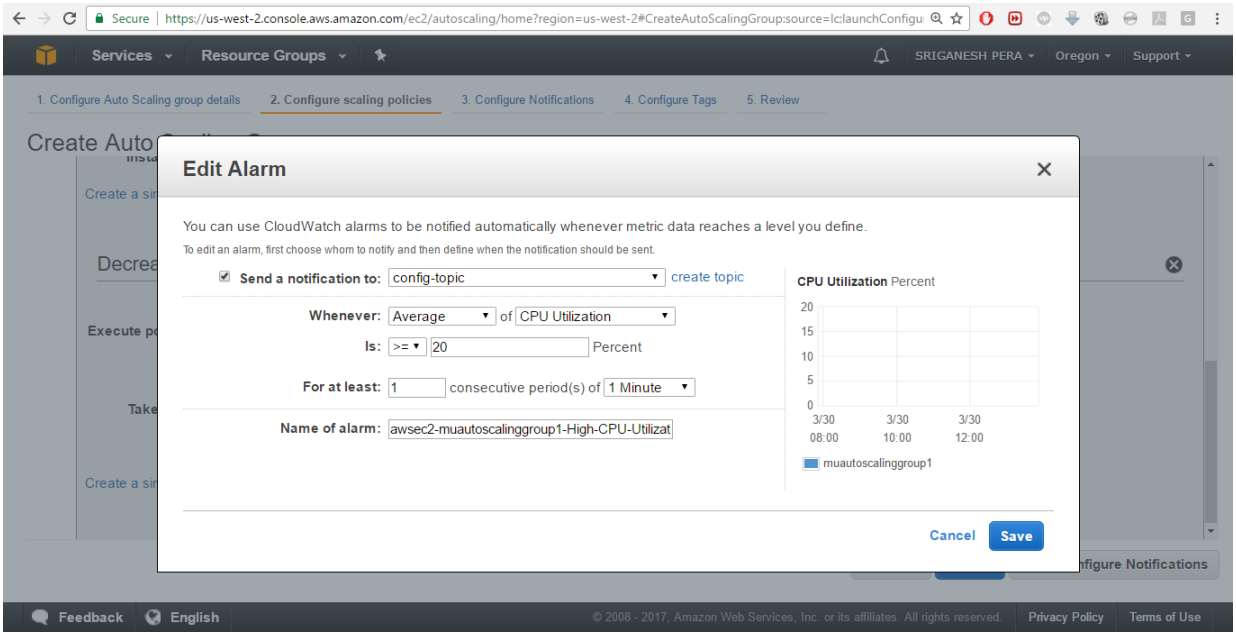




As you can see I set the CPU utilizatoin to 30% and 1 instance for every 1 minute which means if CPU utilization reaches 30% for every minute 1 instance gets launches.

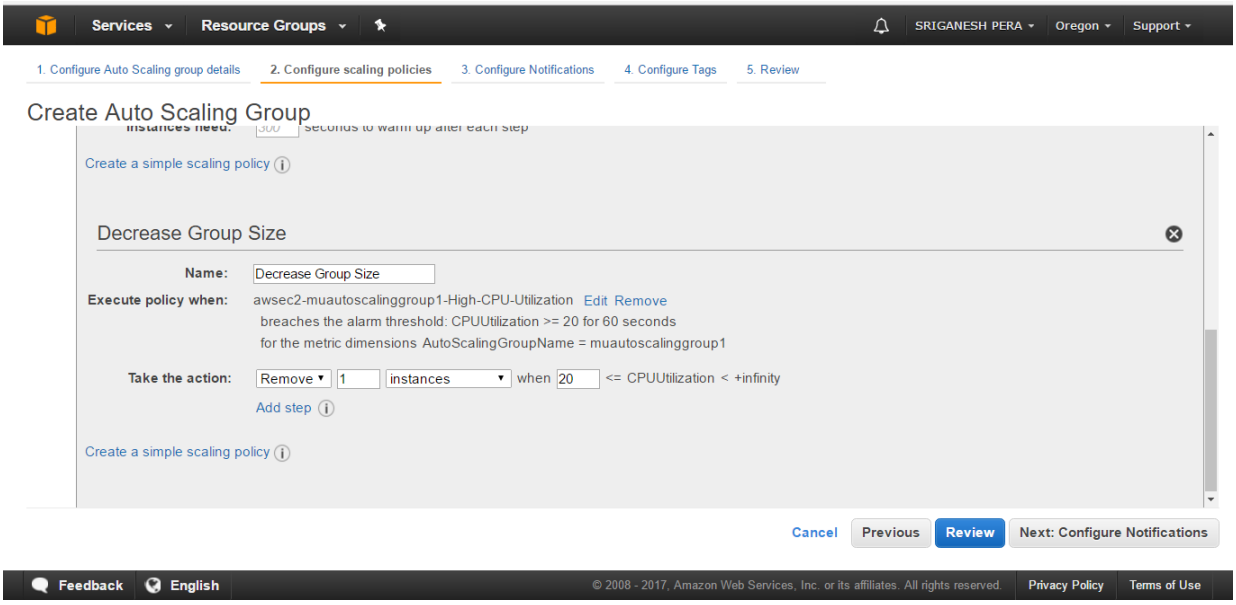
This is setting is for demo purpose but for more practical scenario we need to choose time as 5 minutes or 15 minutes which is the best practice for choosing Auto scaling group

<u>Step 13:Decrease group size and now click on create an alarm and set CPU utilization to 20% and select instance 1 and set timiing as 1 minute</u>

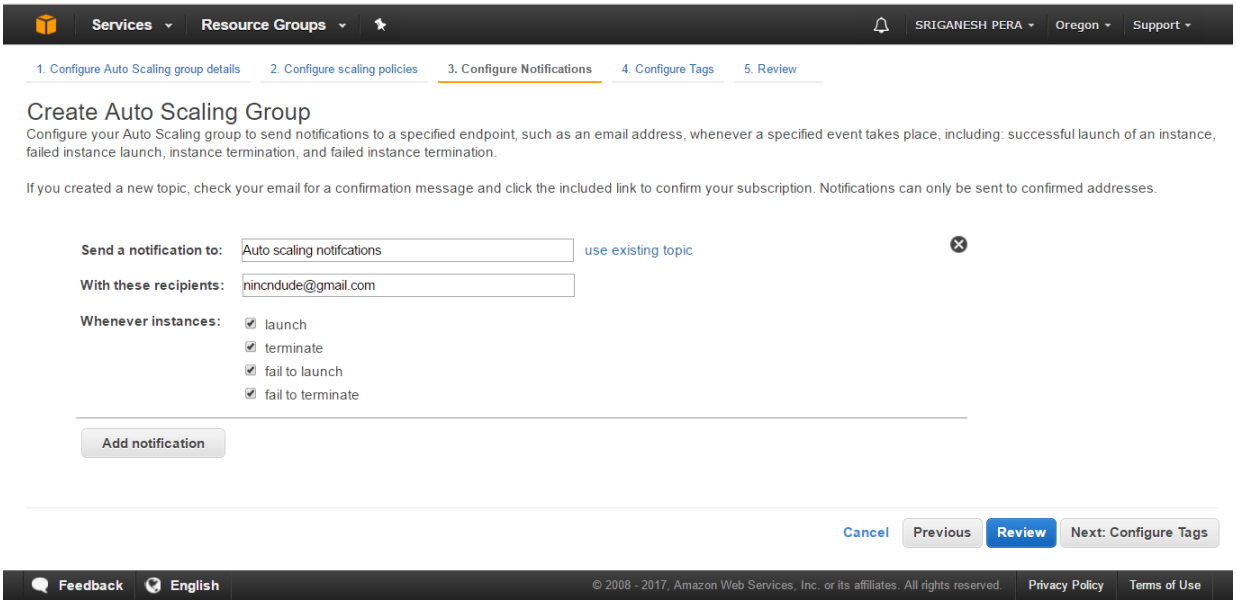


Now this says that whenever CPU utilization decrease to 20% terminate 1 instance

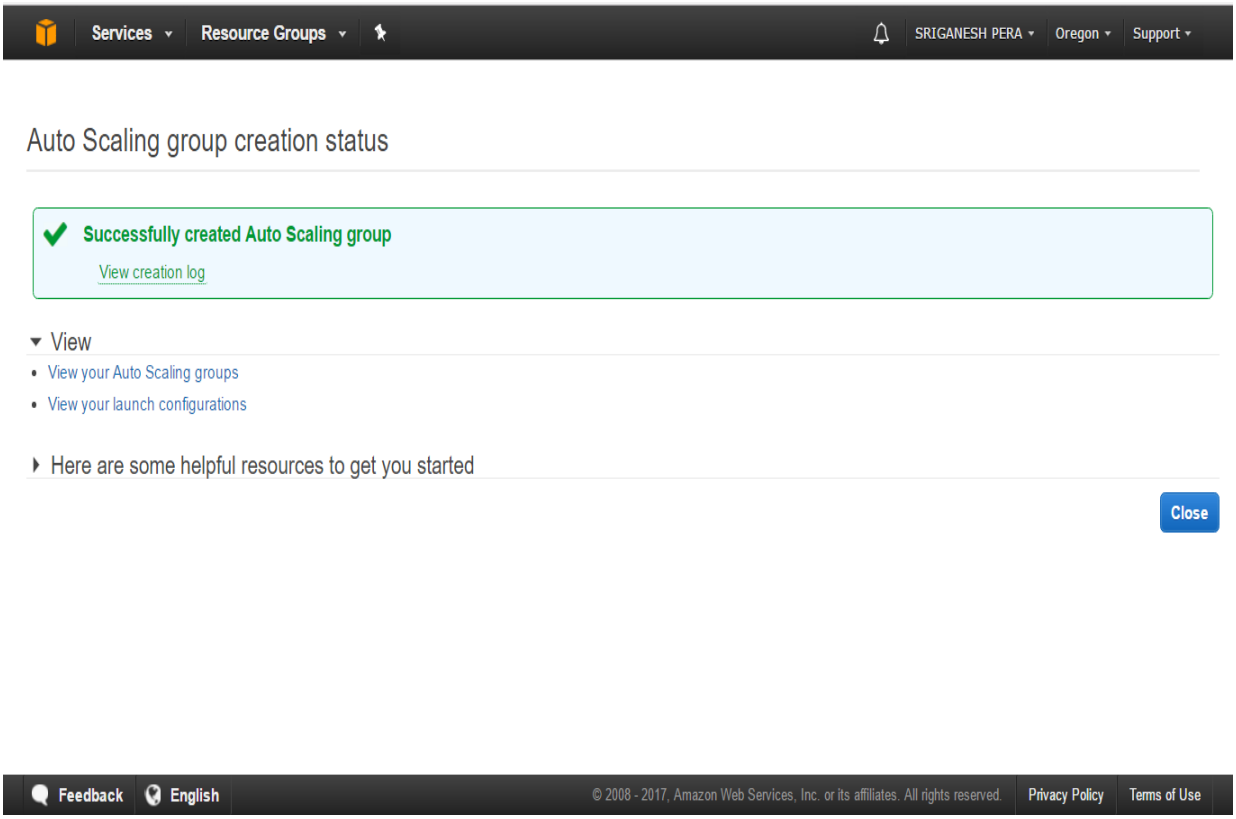
Now select 1 in the take action field choose **'Remove'** and then click on Configure notifications



<u>Step 14: Click on add notifcation and create new topic and set notifications as per your requirement and make AWS send an E-mail(your aws account email) to you in case of events of triggered as per our requirement</u>

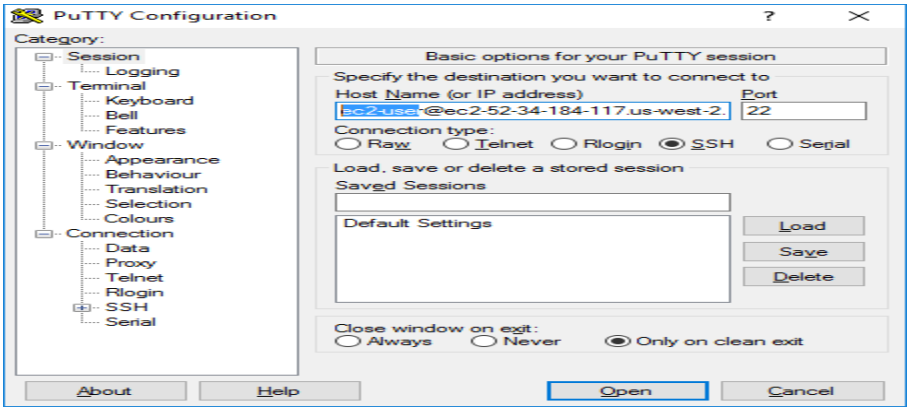


Now click on Tags and create one(optional) if you want.If you don't want then click review and check everything before you click on create Autoscaling group

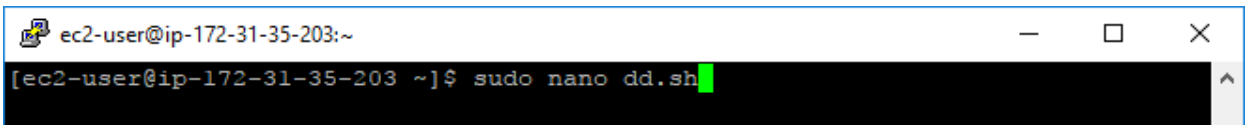


Now go back to EC2 dashboard to see an instance running which is automatically created for us by Auto scaling group.Can you tell why an instance is already created for us by AWS?

Step 15: Now SSH into your instance(Launch Configuration one) and install a script using dd command(which sends garbage data to dev/null) and execute it with nohup command!

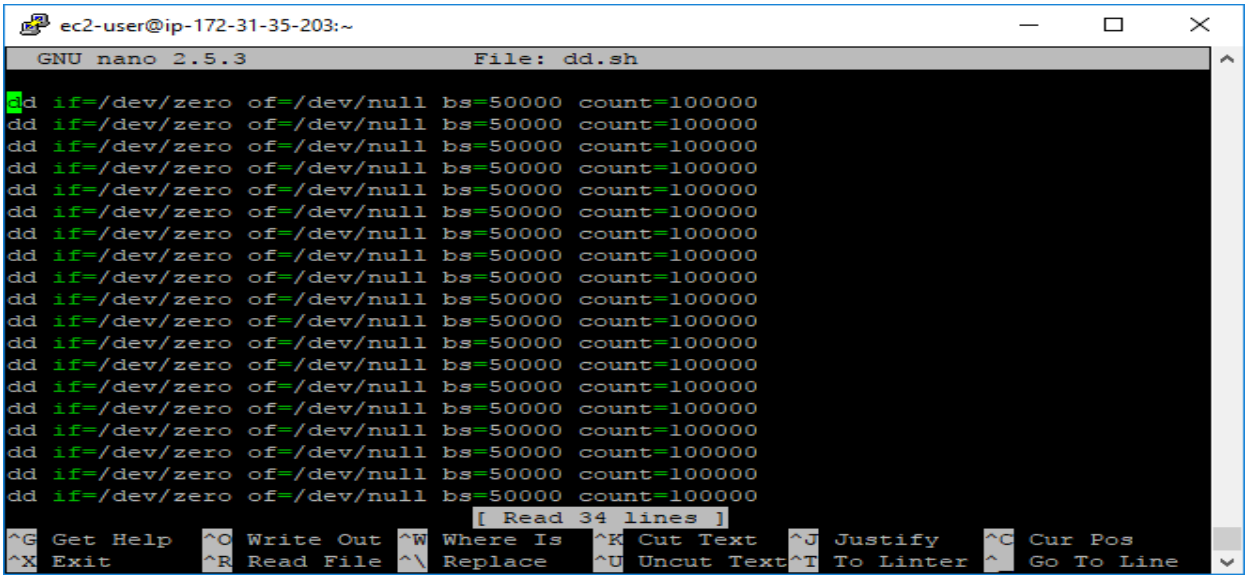


Now create a file dd.sh



Now paste this script multiple times as shown:

dd if=/dev/zero of=/dev/null bs=50000 count=100000



Now lets see our CPU utilization

As you can see my cpu utilization is 0%

```
ec2-user@ip-172-31-20-109:~
top - 14:52:32 up 14 min,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:  79 total,   1 running,  78 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
Mem:   1019116k total,   170968k used,   848148k free,     8724k buffers
Swap:        0k total,        0k used,        0k free,    92036k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2631 apache    20   0  401m  14m 6916 S  0.3  1.4   0:00.07 httpd
    1 root      20   0 19632 2424 2108 S  0.0  0.2   0:00.94 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kworker/0:0
    5 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kworker/0:0H
    6 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kworker/u30:0
    7 root      20   0     0    0    0 S  0.0  0.0   0:00.02 rcu_sched
    8 root      20   0     0    0    0 S  0.0  0.0   0:00.00 rcu_bh
    9 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
   10 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kdevtmpfs
   11 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 netns
   12 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 perf
   13 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kworker/u30:1
   14 root      20   0     0    0    0 S  0.0  0.0   0:00.00 xenwatch
   19 root      20   0     0    0    0 S  0.0  0.0   0:00.00 xenbus
   20 root      20   0     0    0    0 S  0.0  0.0   0:00.05 kworker/0:1
```

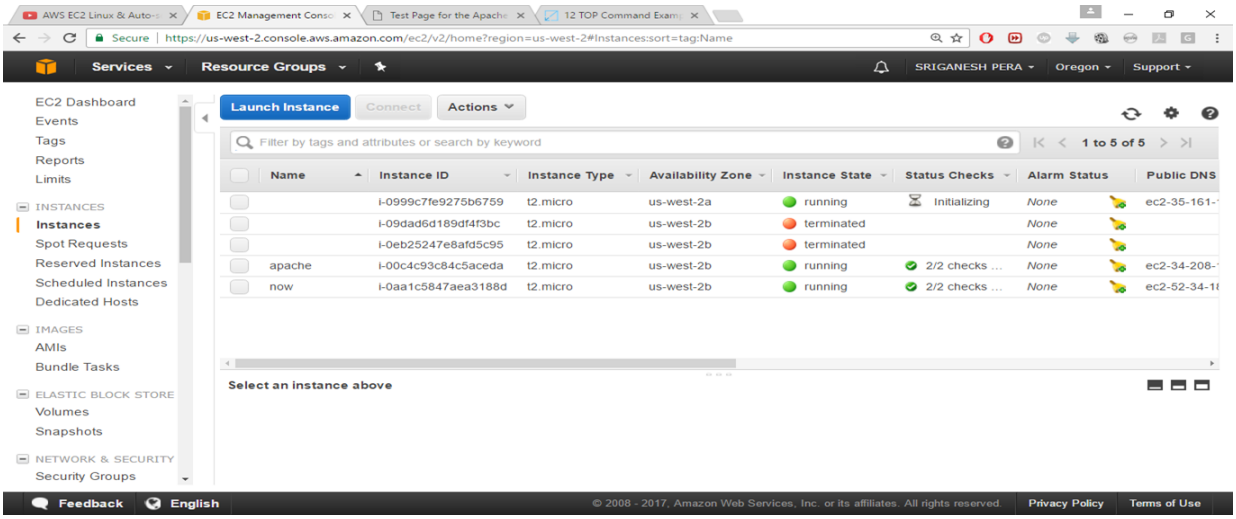
Make sure you give read write permission to your file

```
ec2-user@ip-172-31-20-109:~
[ec2-user@ip-172-31-20-109 ~]$ chmod 755 dd.sh
[ec2-user@ip-172-31-20-109 ~]$
```

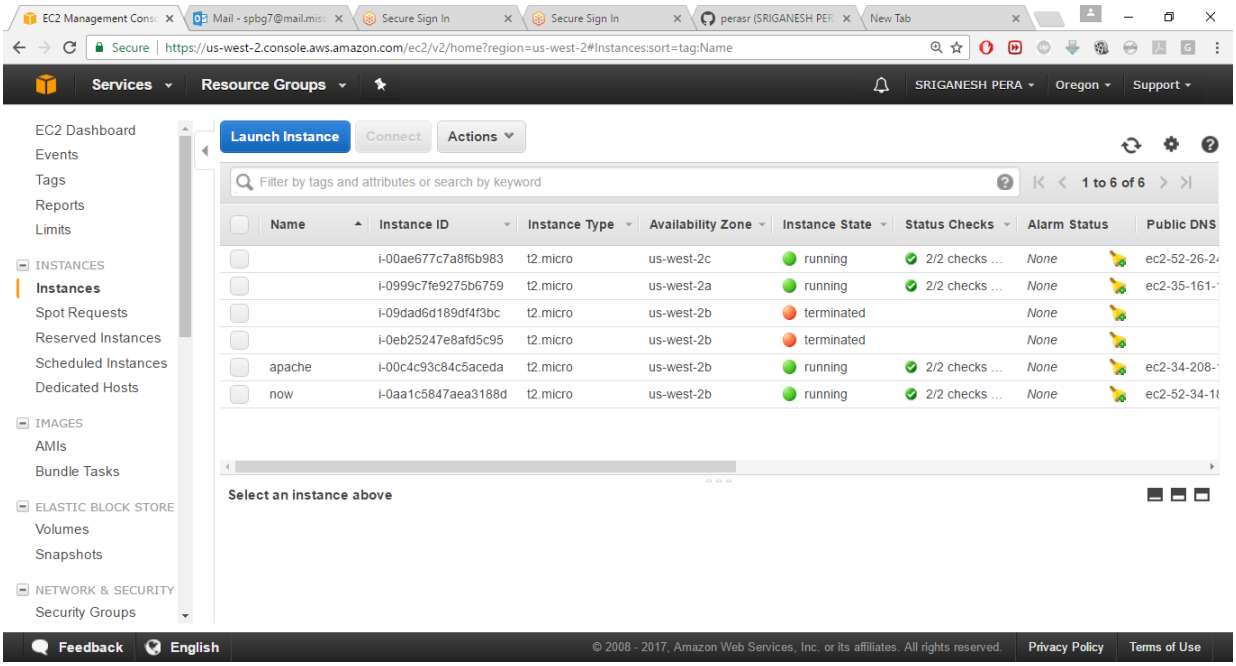
Now run the script multiple times which spikes the CPU utilization using the following command

$ nohup ./dd.sh &

After 5 minutes(cooling period) check your EC2 dashboard which will show the EC2 instance added automatically

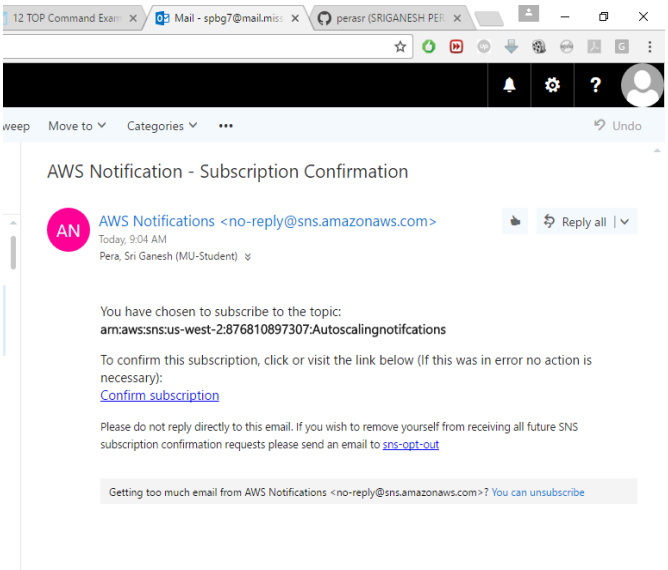


The first instance is the automatically created instance after the CPU utilization spike



Now I have got one more instance running.

I got a Notification as well from AWS autoscaling group with regards to the CPU util spike.



Step 15:Cleanup

Terminate all instances,auto scaling,launch scaling groups!