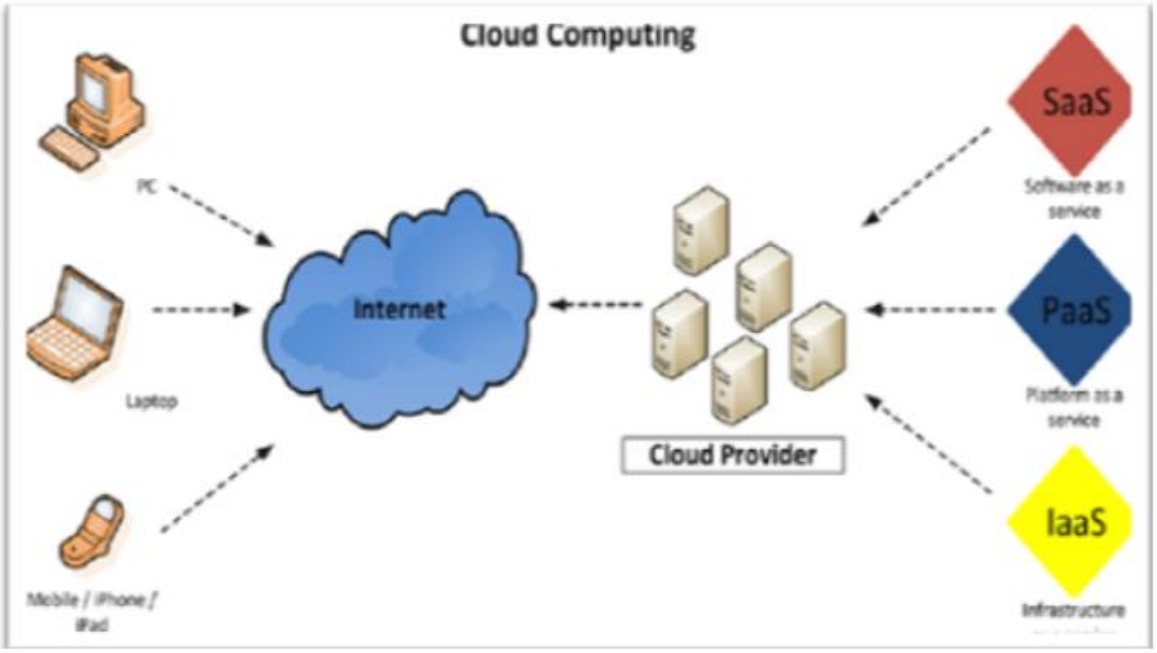
Amazon Web Services

What is cloud computing?

Cloud computing is the on-demand delivery of compute power, database storage,applications, and other IT resources through a cloud services platform via the internet with pay-as-you-go pricing.



What Are Different Types Of Clouds?

We have three types of cloud infrastructure

Private Cloud:

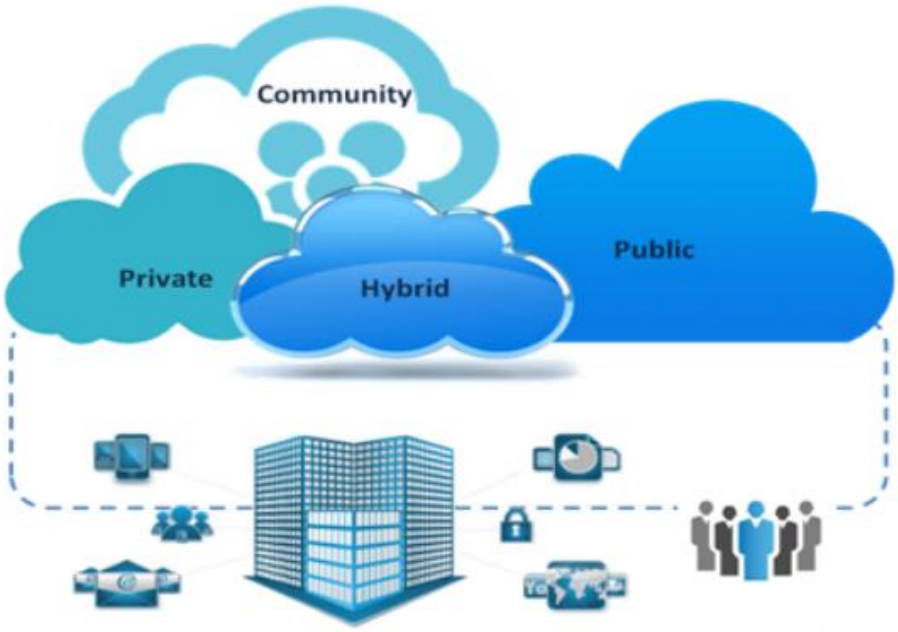
The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

Public Cloud:

The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

Hybrid Cloud:

The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).



Cloud Service Models:

Software As A Service (SaaS):

Software as a Service provides you with a completed product that is run and managed by the service provider. In most cases, people referring to Software as a Service are referring to end-user applications. With a SaaS offering you do not have to think about how the service is maintained or how the underlying infrastructure is managed; you only need to think about how you will use that particular piece software. A common example of a SaaS application is web-based email where you can send and receive email without having to manage feature additions to the email product or maintaining the servers and operating systems that the email program is running on.
Example: Google Apps, Salesforce, Workday, Concur, Citrix, Cisco WebEx...

Platform As A Service (PaaS):

Platforms as a service remove the need for organizations to manage the underlying infrastructure (usually hardware and operating systems) and allow you to focus on the deployment and management of your applications. This helps you be more efficient as you don't need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.
Example: Beanstalk

Infrastructure As A Service (IaaS):

Infrastructure as a Service, sometimes abbreviated as IaaS, contains the basic building blocks for cloud IT and typically provide access to networking features, computers (virtual or on dedicated hardware), and data storage space. Infrastructure as a Service provides you with
the highest level of flexibility and management control over your IT resources and is most similar to existing IT resources that many IT departments and developers are familiar with today. Example: AWS EC2, Microsoft Azure, Google Compute Engine, Joyent..

Advantages and Disadvantages of Cloud Computing

As stated earlier vagrant is a command line tool and windows command prompt is just sluggish to run vagrant commands. So, we will need a better command line utility, one of them is Git bash.

Advantages:

- Easy Implementation: Cloud hosting allows business to retain the same applications and business processes without having to deal with the backend technicalities. Readily manageable by the Internet, a cloud infrastructure can be accessed by enterprises easily and quickly.
- Accessibility: Access your data anywhere, anytime. An Internet cloud infrastructure maximizes enterprise productivity and efficiency by ensuring your application is always accessible. This allows for easy collaboration and sharing among users in multiple locations.
- No hardware required: Since everything will be hosted in the cloud, a physical storage center is no longer needed. However, a backup could be worth looking into in the event of a disaster that could leave your company's productivity stagnant.
- Cost per head: Overhead technology costs are kept at a minimum with cloud hosting services, enabling businesses to use the extra time and resources for improving the company infrastructure.
- Flexibility for growth: The cloud is easily scalable so companies can add or subtract resources based on their needs. As companies grow, their system will grow with them.

- Efficient recovery: Cloud computing delivers faster and more accurate retrievals of applications and data. With less downtime, it is the most efficient recovery plan.

Disadvantages:

- No longer in control: When moving services to the cloud, you are handing over your data and information. For companies who have an in-house IT staff, they will be unable to handle issues on their own. However, Stratosphere Networks has a 24/7 live help desk that can rectify any problems immediately.
- May not get all the features: Not all cloud services are the same. Some cloud providers tend to offer limited versions and enable the most popular features only, so you may not receive every feature or customization you want. Before signing up, make sure you know what your cloud service provider offers.
- Doesn't mean you should do away with servers: You may have fewer servers to handle which means less for your IT staff to handle, but that doesn't mean you can let go of all your servers and staff. While it may seem costly to have data centers and a cloud infrastructure, redundancy is key for backup and recovery.
- No Redundancy: A cloud server is not redundant nor is it backed up. As technology may fail here and there, avoid getting burned by purchasing a redundancy plan. Although it is an extra cost, in most cases it will be well worth it.

- Bandwidth issues: For ideal performance, clients have to plan accordingly and not pack large amounts of servers and storage devices into a small set of data centers.

What is AWS

Amazon Web Services (AWS) is a secure cloud services platform, offering compute power, database storage, content delivery and other functionality to help businesses scale and grow. Explore how millions of customers are currently leveraging AWS cloud products and solutions to build sophisticated applications with increased flexibility, scalability and reliability.

Regions and Availability Zones

Amazon cloud computing resources are hosted in multiple locations world-wide. These locations are composed of regions and Availability Zones. Each region is a separate geographic area. Each region has multiple, isolated locations known as Availability Zones.For most of the AWS services that you use, you will be prompted to select a region in which you want to deploy the service. Each region is completely isolated from the other and runs independently as well.

A list of regions and availability zones are given below for reference:

Region	Name	Endpoint
US East (N. Virginia) Region	us-east-1	https://rds.us-east-1.amazonaws.com
US East (Ohio) Region	us-east-2	https://rds.us-east-2.amazonaws.com
US West (N. California) Region	us-west-1	https://rds.us-west-1.amazonaws.com
US West (Oregon) Region	us-west-2	https://rds.us-west-2.amazonaws.com
Asia Pacific (Mumbai) Region	ap-south-1	https://rds.ap-south-1.amazonaws.com
Asia Pacific (Seoul) Region	ap-northeast-2	https://rds.ap-northeast-2.amazonaws.com

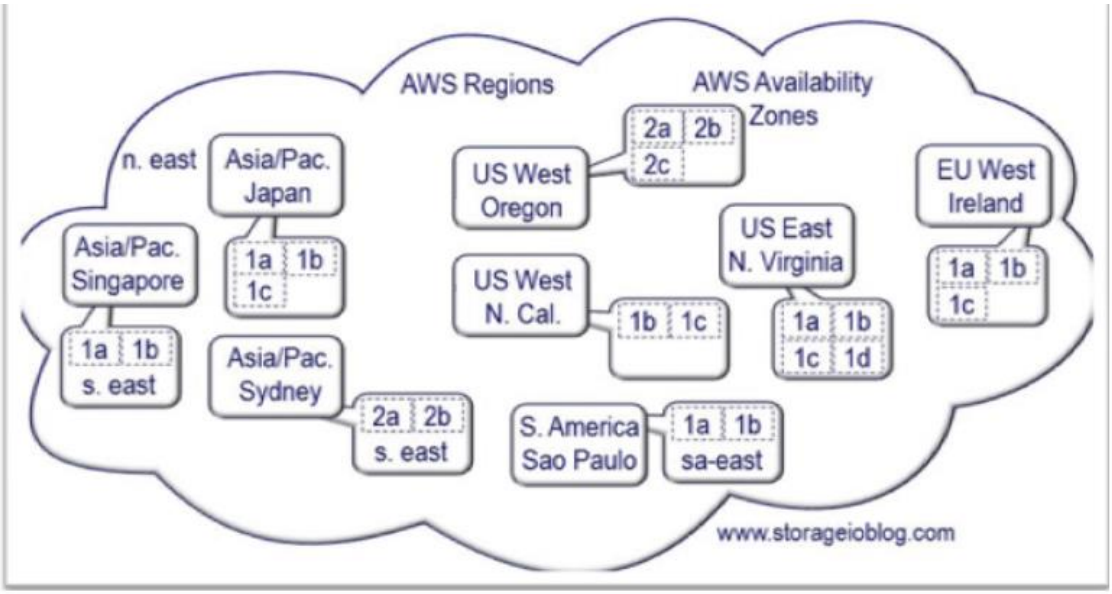
Asia Pacific (Singapore) Region	ap-southeast-1	https://rds.ap-southeast-1.amazonaws.com
Asia Pacific (Sydney) Region	ap-southeast-2	https://rds.ap-southeast-2.amazonaws.com
Asia Pacific (Tokyo) Region	ap-northeast-1	https://rds.ap-northeast-1.amazonaws.com
Canada (Central) Region	ca-central-1	https://rds.ca-central-1.amazonaws.com
China (Beijing) Region	cn-north-1	https://rds.cn-north-1.amazonaws.com.cn
EU (Frankfurt) Region	eu-central-1	https://rds.eu-central-1.amazonaws.com
EU (Ireland) Region	eu-west-1	https://rds.eu-west-1.amazonaws.com
EU (London) Region	eu-west-2	https://rds.eu-west-2.amazonaws.com
South America (São Paulo) Region	sa-east-1	https://rds.sa-east-1.amazonaws.com
AWS GovCloud (US)	us-gov-west-1	https://rds.us-gov-west-1.amazonaws.com

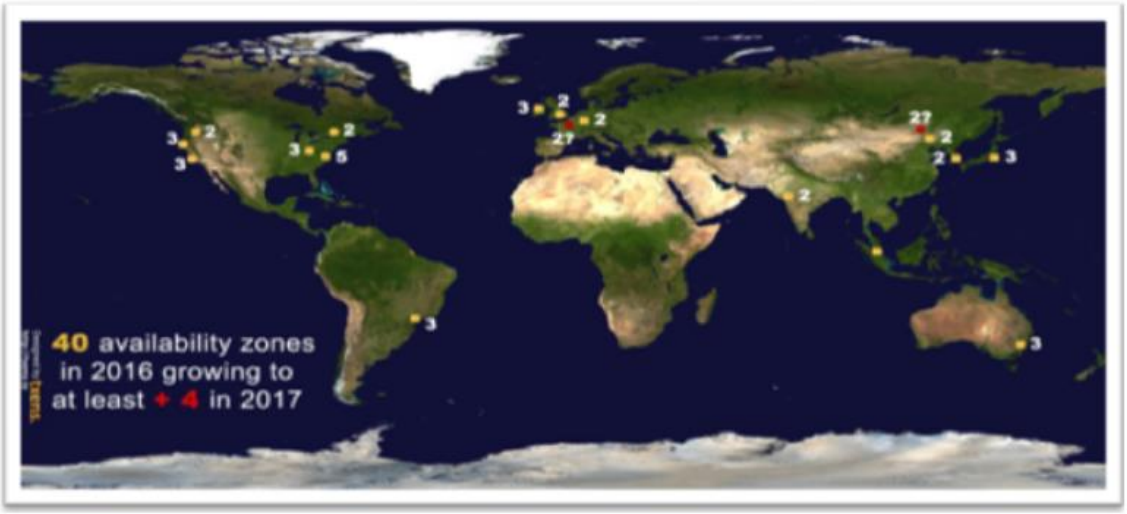
Each region is split up into one or more Availability Zones (AZs) and pronounced as AZees. An A Z is an isolated location inside a region. AZs are made up of one or more physical data centers that host AWS services on them. Just as with regions, even AZs have corresponding codes to identify them, generally they are regional names followed by a numerical value. For example, if you select and use us-east-1, which is the North Virginia region, then it would have AZs listed as us-east-1b, us-east-1c, us-east-1d, and so on.

AZs are very important from a design and deployment point of view. Being data centers, they are more than capable of failure and downtime, so it is always good practice to distribute your resources across multiple AZs and design your applications such that they can remain available even if one AZ goes completely offline. An

important point to note here is that AWS will always provide the services and products to you as a customer; however, it is your duty to design and distribute your applications so that they do not suffer any potential outages or failures.

The below structure represents the regions and their availability zones





Amazon RDS provides you the ability to place resources, such as instances, and data in multiple locations.

Note:

: AWS does not replicate resources across regions automatically. It is up to the end user to set up the replication process.

AWS Services

AWS provides us lot of services that we can use to build our infrastructure on AWS cloud. Our focus being into DevOps will be to leverage SysOps services that AWS provides. There are so many other Services consumed by Developers directly and thats not the focus of this book. Also AWS has some DevOps related services like codecommit, codedeploy, cloudformationetc, that would be out of the scope for this book to cover.

In this chapter we will look into below mentioned AWS services.

- IAM
- EC2
- VPC
- S3
- RDS
- Beanstalk
- Cloudwatch
- Route53

IAM

Security, Identity & Compliance:

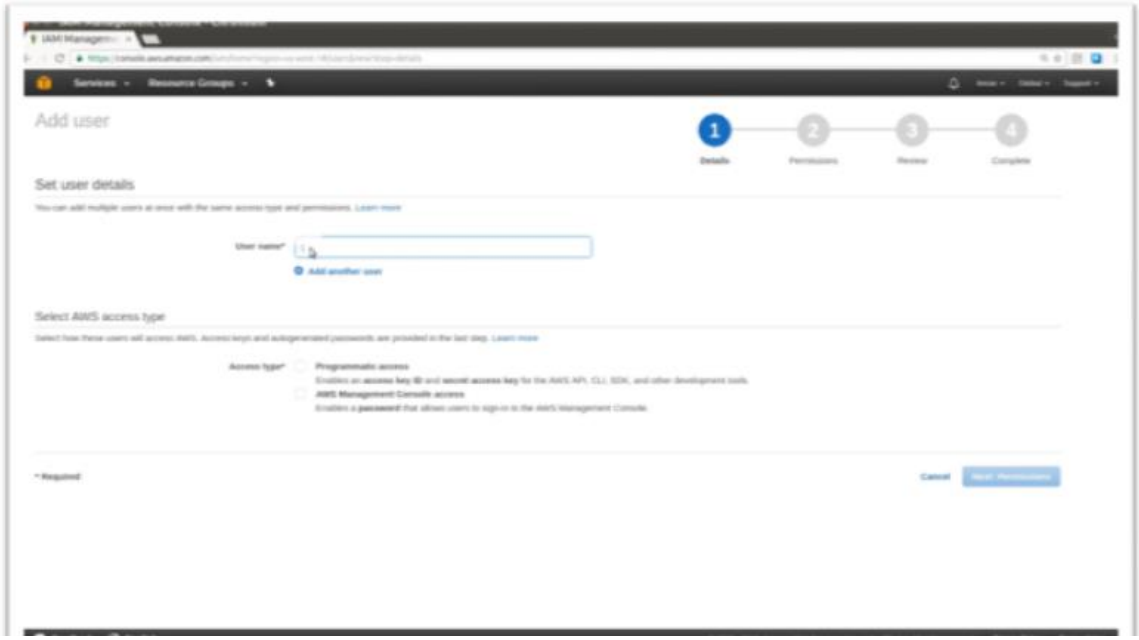
AWS Identity and Access Management (IAM) is a web service that enables Amazon Web Services (AWS) customers to manage users and user permissions in AWS. The service is targeted at organizations with multiple users or systems in the cloud that use AWS products such as Amazon EC2, Amazon SimpleDB, and the AWS Management Console. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users can access.

Creating IAM User:

1.1. Login to AWS account => Services => IAM

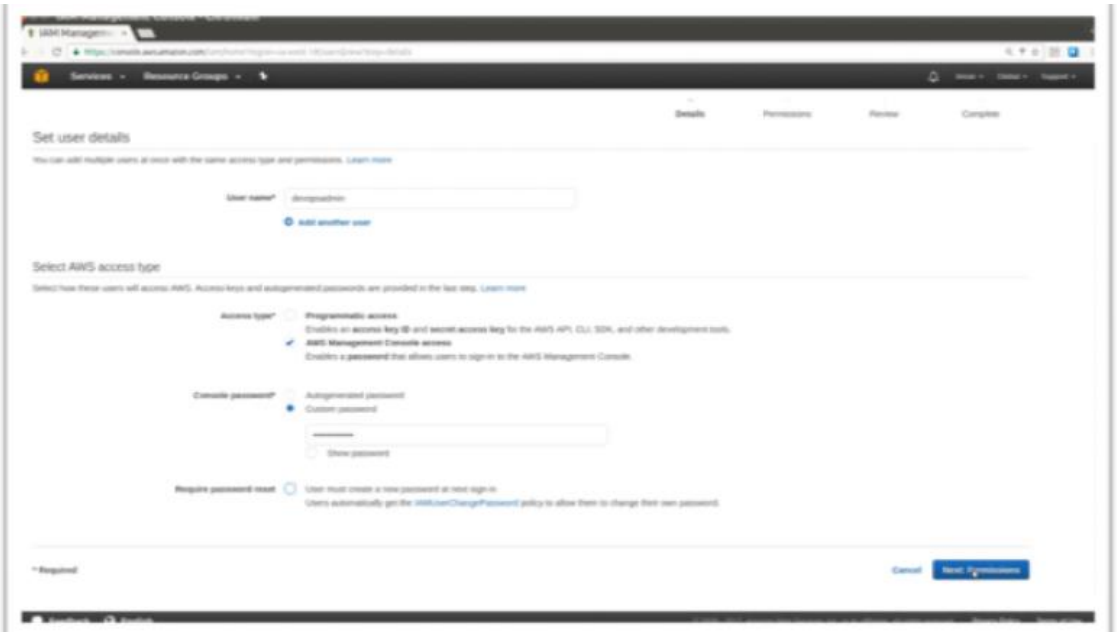
1.Select users => Add user => username

1.Select one of the options either Programmatic access (Key-based access for AWS-CLI) or AWS Management Console access.



1.If you select AWS Management Console access, you need to selectAutogenerated password or custom password.

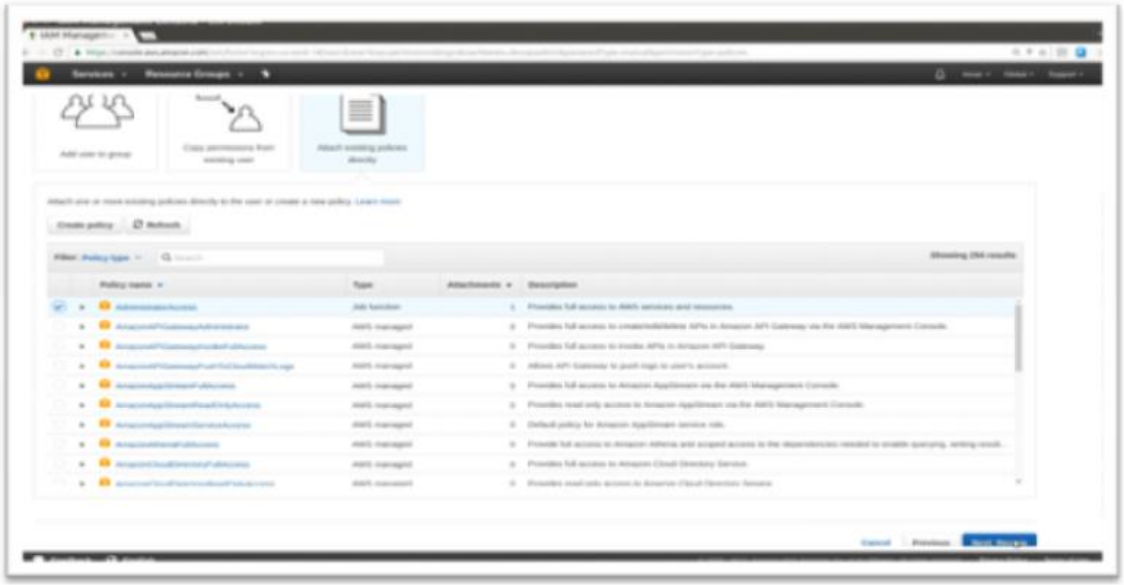
2.Click on Next Permissions

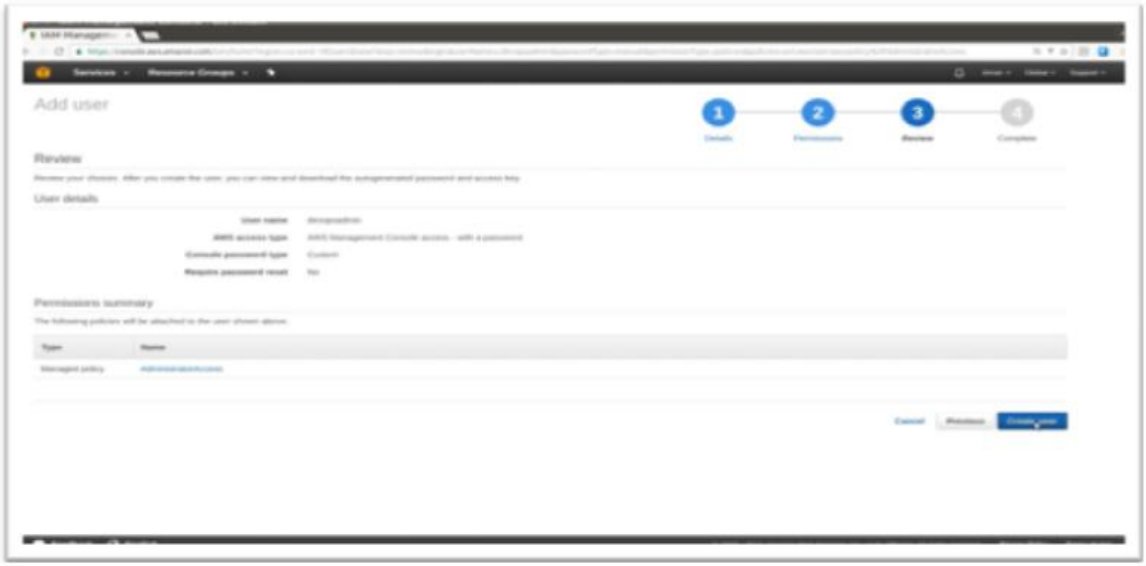


1.Attach Policy to the User.

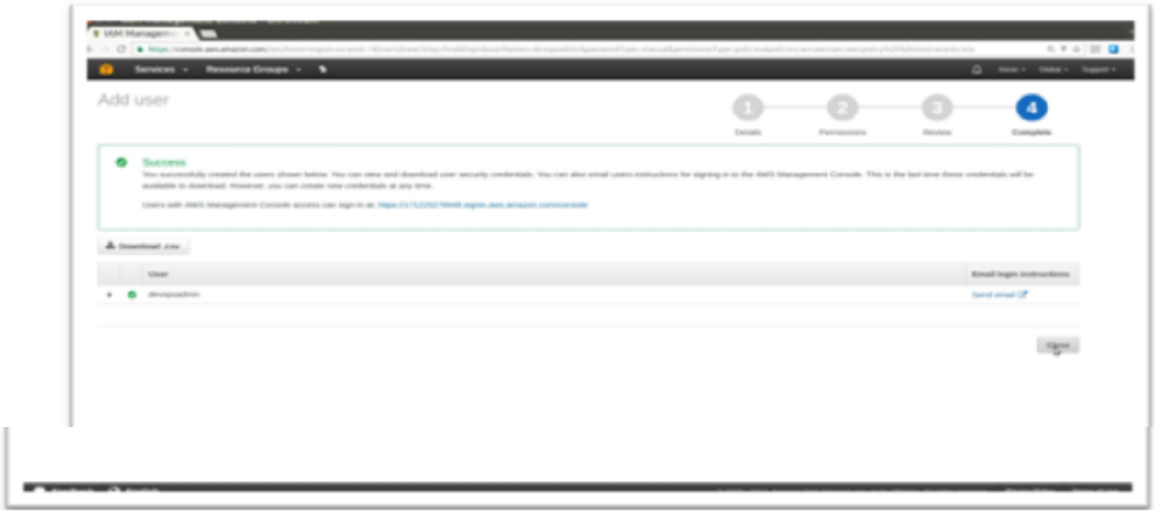
By default, IAM users don't have permission to create or modify Amazon EC2 resources, or perform tasks using the Amazon EC2 API. (This means that they also can't do so using the Amazon EC2 console or CLI).

When you attach a policy to a user or group of users, it allows or denies the users permission to perform the specified tasks on the specified resources.So while creating user we have to assign the policies which has to be performed by particular user.There are some predefined policies in IAM, we can attach the existing policies or create new policy.Here in this example we are attaching Administrator Access in which the user gets all administrator permissions.




Add user
1
2
3
4
Review
User details
Permissions summary

1. After creating user we get a Download.csv file, which contains the credentials of th user and url to login to the AWS console.



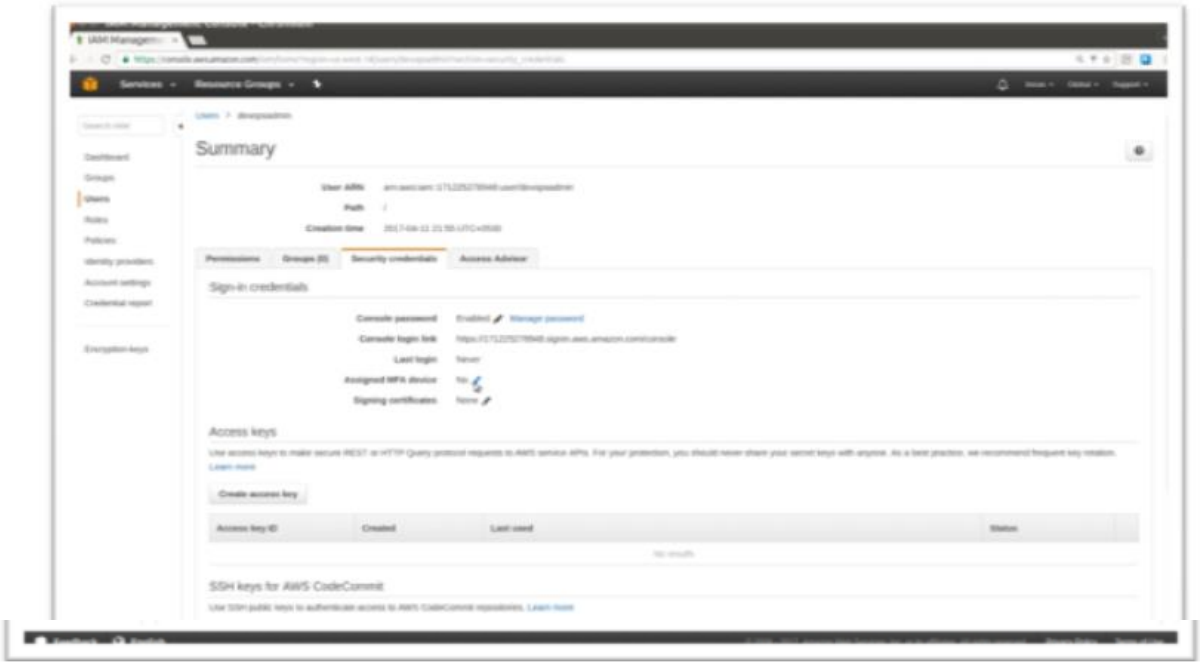
Setting up MFA:

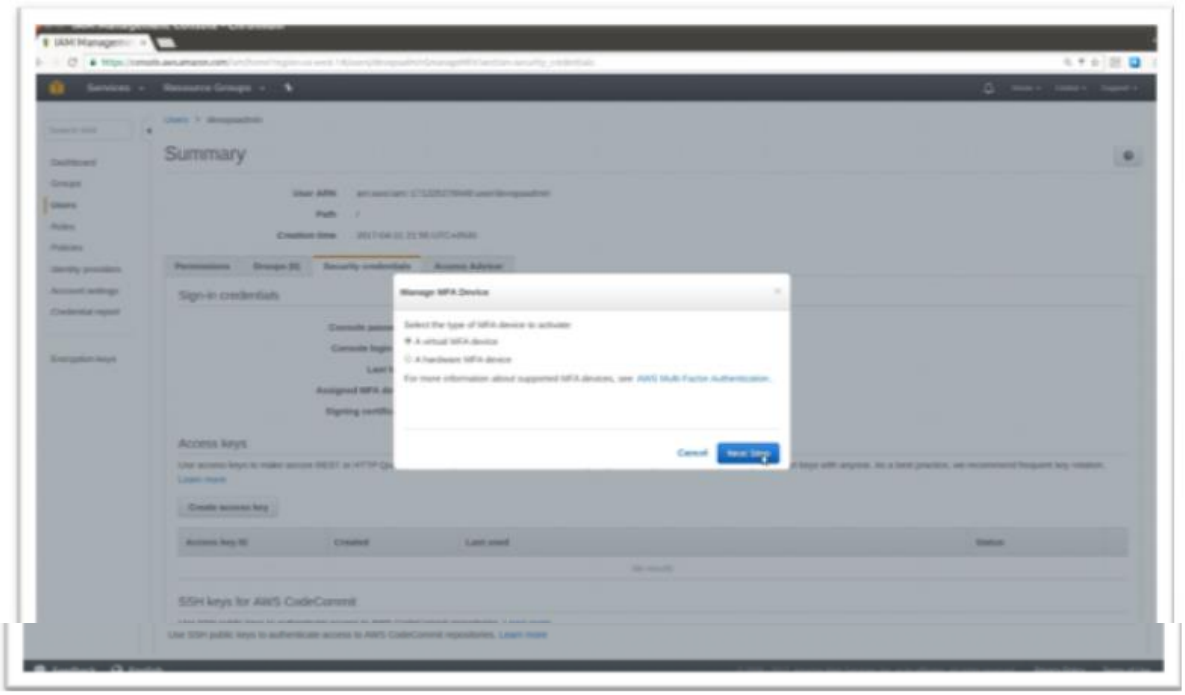
AWS Multi-Factor Authentication (MFA) is a simple best practice that adds an extra layer of protection on top of your user name and password. With MFA enabled, when a user signs in to an AWS website, they will be prompted for their user name and password (the first factor—what they know), as well as for an authentication code from their AWS MFA device (the second factor—what they have). Taken together, these multiple factors provide increased security for your AWS account settings and resources.

You can enable MFA for your AWS account and for individual IAM users you have created under your account. MFA can be also be used to control access to AWS service APIs.

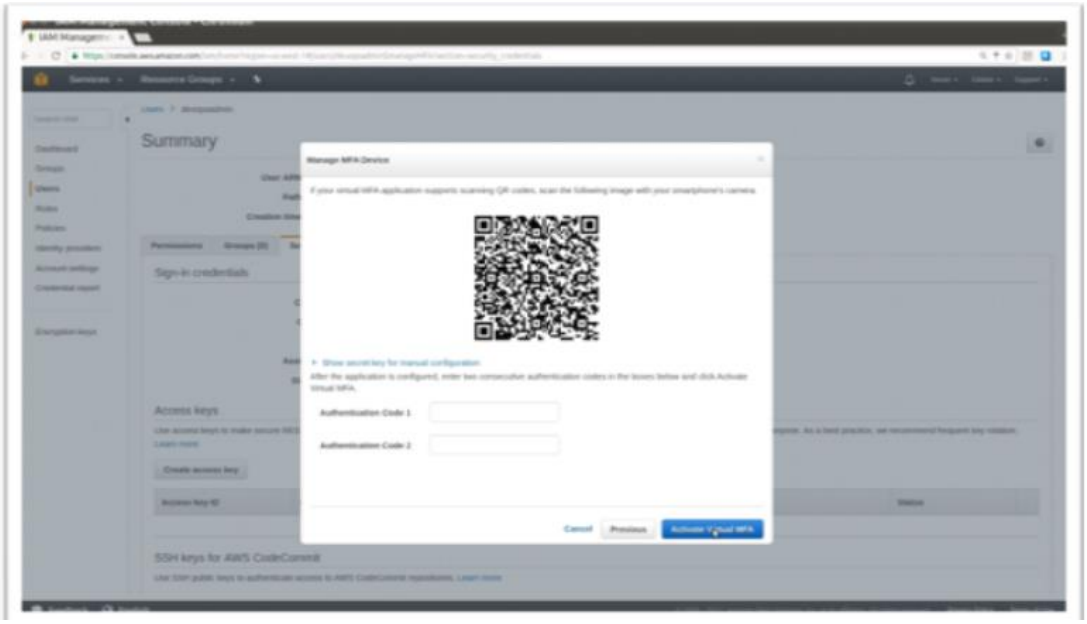
1.Click on Username

1.Click on Assigned MFA device

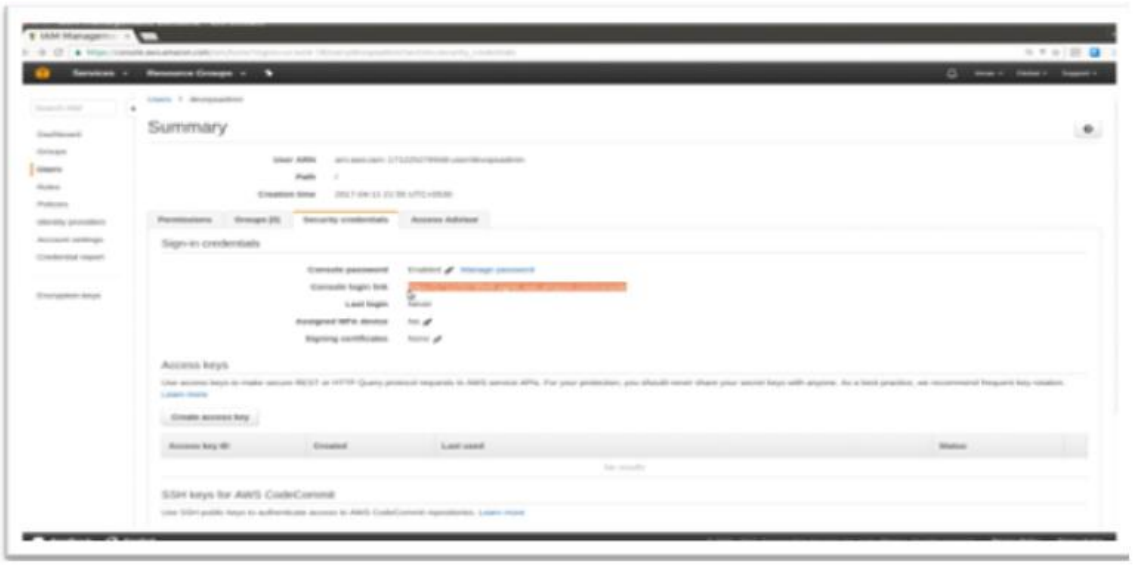




- Download Google Authenticator in your Smartphone
- Open Google Authenticator
- Click plus symbol
- Scan the barcode
- Enter Auth code1
- Enter second auth code.



1.Use the highlighted URL for your user to login to AWS account with IAM user.



Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate them from common failure scenarios.

Benefits Of EC2:

- Elastic Web-Scale Computing
- Completely Controlled
- Flexible Cloud Hosting Services
- Integrated
- Reliable
- Secure
- Inexpensive
- Easy to start

Amazon EC2 Instance Types

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instance types comprise varying combinations of CPU, memory, storage, and networking capacity and give you the flexibility to choose the appropriate mix of resources for your applications. Each instance type includes one or more instance sizes, allowing you to scale your resources to the requirements of your target workload.

Different Types Of Instances:

1.General Purpose: t2, m4, m3

T2 - T2 instance receives CPU Credits continuously at a set rate depending on the instance size. T2 instances accrue CPU Credits when they are idle, and use CPU credits when they are active. T2 instances are a good choice for workloads that don't use the full CPU often or consistently, but occasionally need to burst (e.g. web servers, developer environments and databases).

Model	vCPU	CPU Credits/hour	Mem(GiB)	Storage
t2.nano	1	3	0.5	EBS-Only
t2.micro	1	6	1	EBS-Only
t2.small	1	12	2	EBS-Only
t2.medium	2	24	4	EBS-Only
t2.large	2	36	8	EBS-Only
t2.xlarge	4	54	16	EBS-Only
t2.2xlarge	8	81	32	EBS-Only

M4 - M4 instances are the latest generation of General Purpose Instances. This family provides a balance of compute, memory, and network resources, and it is a good choice for many applications.

Model	vCPU	Mem(GiB)	SSD Storage(GB)	Dedicated EBS Bandwidth (Mbps)
m4.large	2	8	EBS-Only	450
m4.xlarge	4	16	EBS-Only	750
m4.2xlarge	8	32	EBS-Only	1,000
m4.4xlarge	16	64	EBS-Only	2,000
m4.10xlarge	40	160	EBS-Only	4,000
m4.16xlarge	64	256	EBS-Only	10,000

M3 - This family includes the M3 instance types and provides a balance of compute, memory, and network resources, and it is a good choice for many applications.

Model	vCPU	Mem(GiB)	SSD Storage(GB)
m3.medium	1	3.75	1 x 4
m3.large	2	7.5	1 x 32
m3.xlarge	4	15	2 x 40
m3.2xlarge	8	30	2 x 80

2.Compute Optimized: c4, c3

C4 - C4 instances are the latest generation of Compute-optimized instances, featuring the highest performing processors and the lowest price/compute performance in EC2.

Model	vCPU	Mem(GiB)	Storage(GB)	Dedicated EBS Bandwidth (Mbps)
c4.large	2	3.75	EBS-Only	500
c4.xlarge	4	7.5	EBS-Only	750
c4.2xlarge	8	15	EBS-Only	1,000
c4.4xlarge	16	30	EBS-Only	2,000
c4.8xlarge	36	60	EBS-Only	4,000

Model	vCPU	Mem(GiB)	Storage(GB)	Dedicated EBS Bandwidth (Mbps)
c4.large	2	3.75	EBS-Only	500
c4.xlarge	4	7.5	EBS-Only	750
c4.2xlarge	8	15	EBS-Only	1,000
c4.4xlarge	16	30	EBS-Only	2,000
c4.8xlarge	36	60	EBS-Only	4,000

C3 - C3 instances will provide you with the highest performance processors and the lowest price/compute performance compared to all other Amazon EC2 instances. C3 instances also feature Enhanced Networking and SSD-based instance storage.

Model	vCPU	Mem(GiB)	SSD Storage(GB)
c3.large	2	3.75	2 x 16
c3.xlarge	4	7.5	2 x 40
c3.2xlarge	8	15	2 x 80
c3.4xlarge	16	30	2 x 160
c3.8xlarge	32	60	2 x 320

3.Memory Optimized: x1, r4, r3

X1 - X1 Instances are optimized for large-scale, enterprise-class, in-memory applications and have the lowest price per GiB of RAM among Amazon EC2 instance types.

Model	vCPU	Mem(GiB)	SSD Storage(GB)	Dedicated EBS Bandwidth (Mbps)
x1.32xlarge	128	1,952	2 x 1,920	10,000
x1.16xlarge	64	976	2 x 1,920	5,000

R4 - R4 instances are optimized for memory-intensive applications and offer better price per GiB of RAM than R3.

Model	vCPU	Mem(GiB)	Networking Performance	SSD Storage(GB)
r4.large	2	15.25	Up to 10 Gigabit	EBS-Only
r4.xlarge	4	30.5	Up to 10 Gigabit	EBS-Only
r4.2xlarge	8	61	Up to 10 Gigabit	EBS-Only
r4.4xlarge	16	122	Up to 10 Gigabit	EBS-Only
r4.8xlarge	32	244	10 Gigabit	EBS-Only
r4.16xlarge	64	488	20 Gigabit	EBS-Only

R3 - R3 instances are optimized for memory-intensive applications and offer lower price per GiB of RAM.

Model	vCPU	Mem(GiB)	SSD Storage(GB)
r4.large	2	15.25	1 x 32
r4.xlarge	4	30.5	1 x 80
r4.2xlarge	8	61	1 x 160
r4.4xlarge	16	122	1 x 320
r4.8xlarge	32	244	2 x 320

4.Accelerated Computing Instances: p2, g2, f1

Model	GPUs	vCPU	Mem(GiB)	GPU Memory (GiB)
p2.xlarge	1	4	61	12
p2.8xlarge	8	32	488	96
p2.16xlarge	6	64	732	192

G2 - G2instances are optimized for graphics-intensive applications.

Model	GPUs	vCPU	Mem(GiB)	SSD Storage (GB)
g2.2xlarge	1	8	15	1 x 60
g2.2xlarge	4	32	60	2 x 120

F1 - F1 instances offer customizable hardware acceleration with field programmable gate arrays (FPGAs).

Model	FPGAs	vCPU	Mem(GiB)	SSD Storage (GB)
f1.2xlarge	1	8	122	470
f1.16xlarge	8	64	976	4 x 940

5.Storage Optimized: i3, d2

I3 – High I/O instances This family includes the High Storage Instances that provide Non-Volatile Memory Express (NVMe) SSD backed instance storage optimized for low latency, very high random I/O performance, high sequential read throughput and provide high IOPS ata low cost.

Model	vCPU	Mem(GiB)	Networking Performance	Storage (TB)
i3.large	2	15.25	Up to 10 Gigabit	1 x 0.475 NVMe SSD
i3.xlarge	4	30.5	Up to 10 Gigabit	1 x 0.95 NVMe SSD
i3.2xlarge	8	61	Up to 10 Gigabit	1 x 1.9 NVMe SSD
i3.4xlarge	16	122	Up to 10 Gigabit	2 x 1.9 NVMe SSD
i3.8xlarge	32	244	10 Gigabit	4 x 1.9 NVMe SSD
i3.16xlarge	64	488	20 Gigabit	8 x 1.9 NVMe SSD

D2 - D2 instances feature up to 48 TB of HDD-based local storage, deliver high disk throughput, and offer the lowest price per disk throughput performance on Amazon EC2.

Model	vCPU	Mem(GiB)	Storage (TB)
d2.xlarge	4	30.5	3 x 2000 HDD
d2.2xlarge	8	61	6 x 2000 HDD
d2.4xlarge	16	122	12 x 2000 HDD
d2.8xlarge	36	244	24 x 2000 HDD

Amazon EC2 Pricing

Amazon EC2 is free to try. There are four ways to pay for Amazon EC2 instances: On-Demand, Reserved Instances, and Spot Instances. You can also pay for Dedicated Hosts which provide you with EC2 instance capacity on physical servers dedicated for your use.

On-Demand - On-Demand instances are recommended for:

- Users that prefer the low cost and flexibility of Amazon EC2 without any up-front payment or long-term commitment.
- Applications with short-term, spiky, or unpredictable workloads that cannot be interrupted.
- Applications being developed or tested on Amazon EC2 for the first time.

Spot Instances - Amazon EC2 Spot instances allow you to bid on spare Amazon EC2 computing capacity for up to 90% off the On-Demand price.

Spot instances are recommended for:

- Applications that have flexible start and end times
- Applications that are only feasible at very low compute prices
- Users with urgent computing needs for large amounts of additional capacity

Reserved Instances - Reserved Instances provide you with a significant discount (up to 75%) compared to On-Demand instance pricing. In addition, when Reserved Instances are assigned to a specific Availability Zone, they provide a capacity reservation, giving you additional confidence in your ability to launch instances when you need them.

Reserved Instances are recommended for:

- Applications with steady state usage.
- Applications that may require reserved capacity.
- Customers that can commit to using EC2 over a 1 or 3 year term to reduce their total computing costs.

Dedicated Hosts - A Dedicated Host is a physical EC2 server dedicated for your use. Dedicated Hosts can help you reduce costs by allowing you to use your existing server-bound software licenses, including Windows Server, SQL Server, and SUSE Linux Enterprise Server.

- Can be purchased On-Demand (hourly).
- Can be purchased as a Reservation for up to 70% off the On-Demand price.

Security Groups:

Security Group act as a firewall for associated Amazon EC2 instances, controlling both inbound and outbound traffic at the instance level. When you launch an instance, you associate one or more security groups with the instance. You add rules to each security group that allow traffic to or from its associated instances. You can modify the rules for a security group at any time; the new rules are automatically applied to all instances that are associated with the security group. When we decide whether to allow traffic to reach an instance, we evaluate all the rules from all the security groups that are associated with the instance.

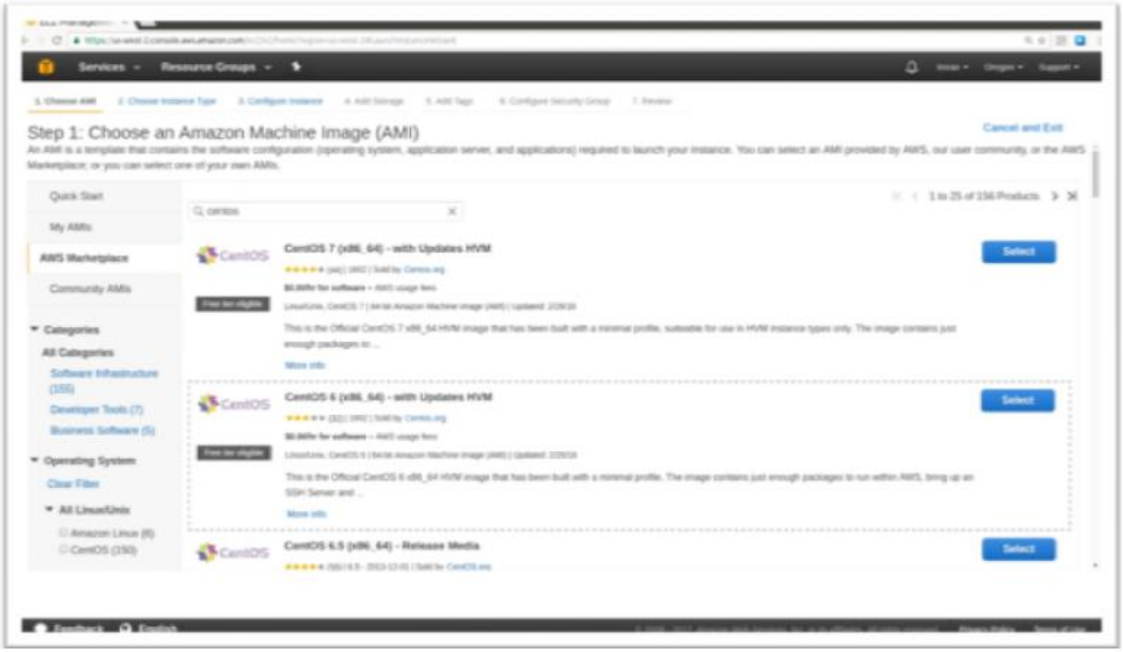
Key Pairs:

Amazon EC2 uses public–key cryptography to encrypt and decrypt login information. Public–key cryptography uses a public key to encrypt a piece of data, such as a password, then the recipient uses the private key to decrypt the data. The public and private keys are known as a key pair. To log in to your instance, you must create a key pair, specify the name of the key pair when you launch the instance, and provide the private key when you connect to the instance. Linux instances have no password, and you use a key pair to log in using SSH. With Windows instances, you use a key pair to obtain the administrator password and then log in using RDP.

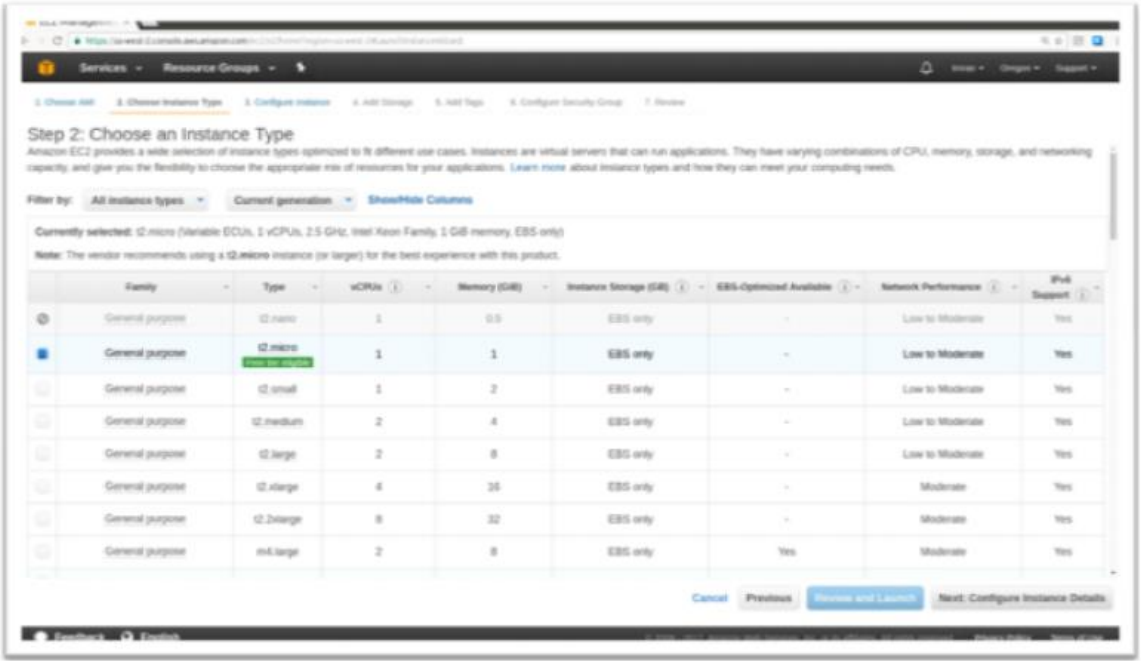
Creating Ec2 Instance.

AWS Marketplace contains different types of Amazon Machine Images (AMIs) like Centos, Ubuntu, Amazon Linux, Windows,...etc

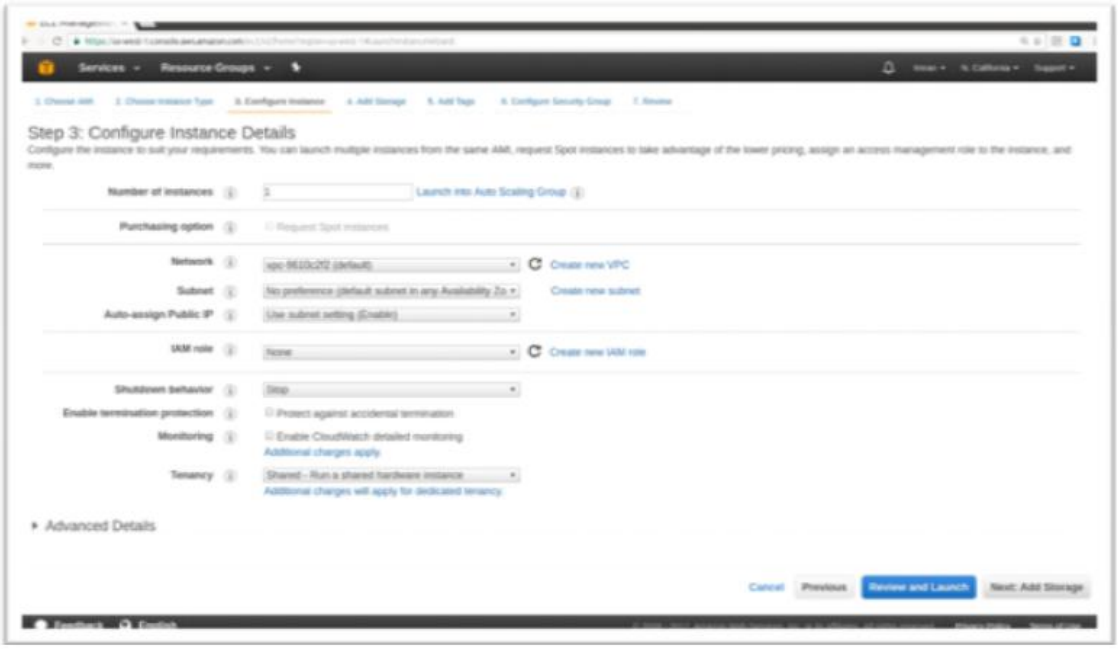
- Login to AWS Management Console and set up your root account.
- Launch an Amazon EC2 instance.
- In the Amazon EC2 Dashboard select "Launch Instance" to create and configure your virtual machine.
- Configure the instance
- In the AWS Marketplace select your required AMI (Ex: Centos 6 AMI)

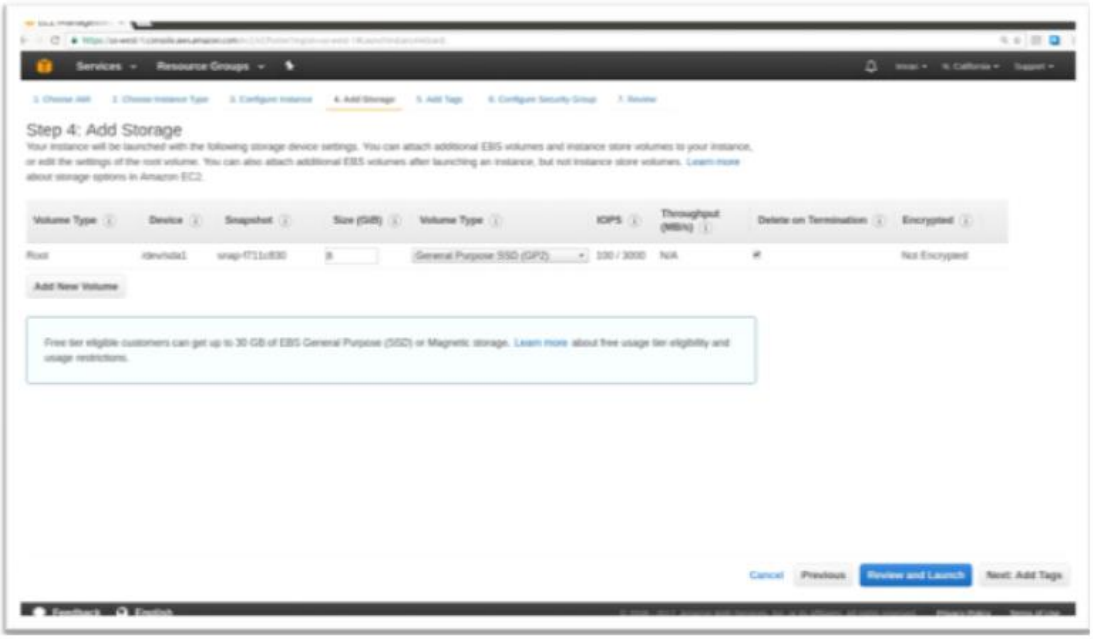


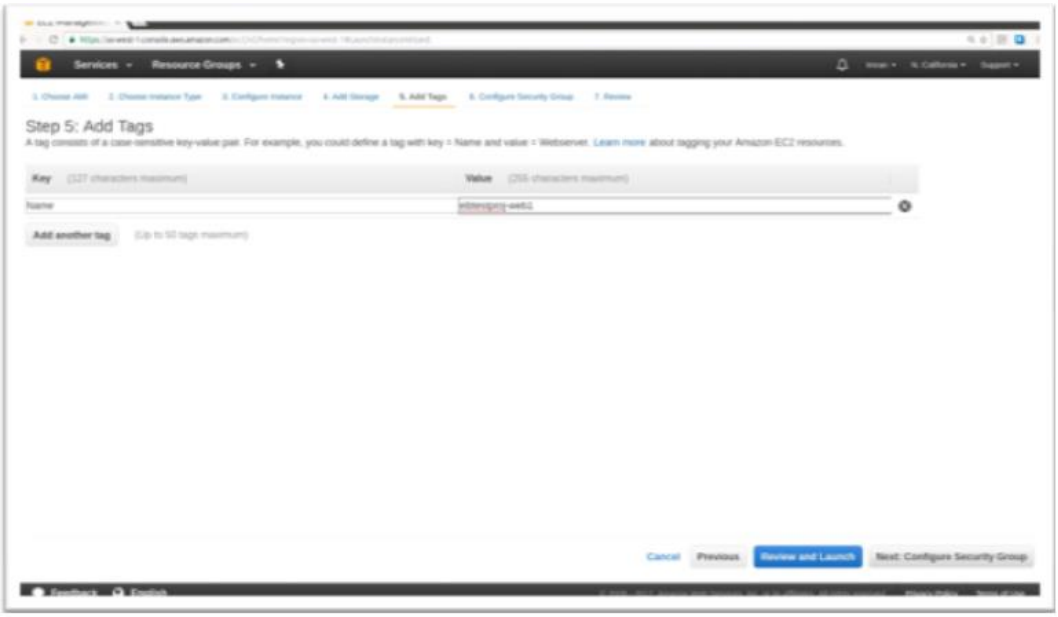
1.Choose an instance type: In the wizard choose an instance type, we recommend t2.micro (free-tier eligible).



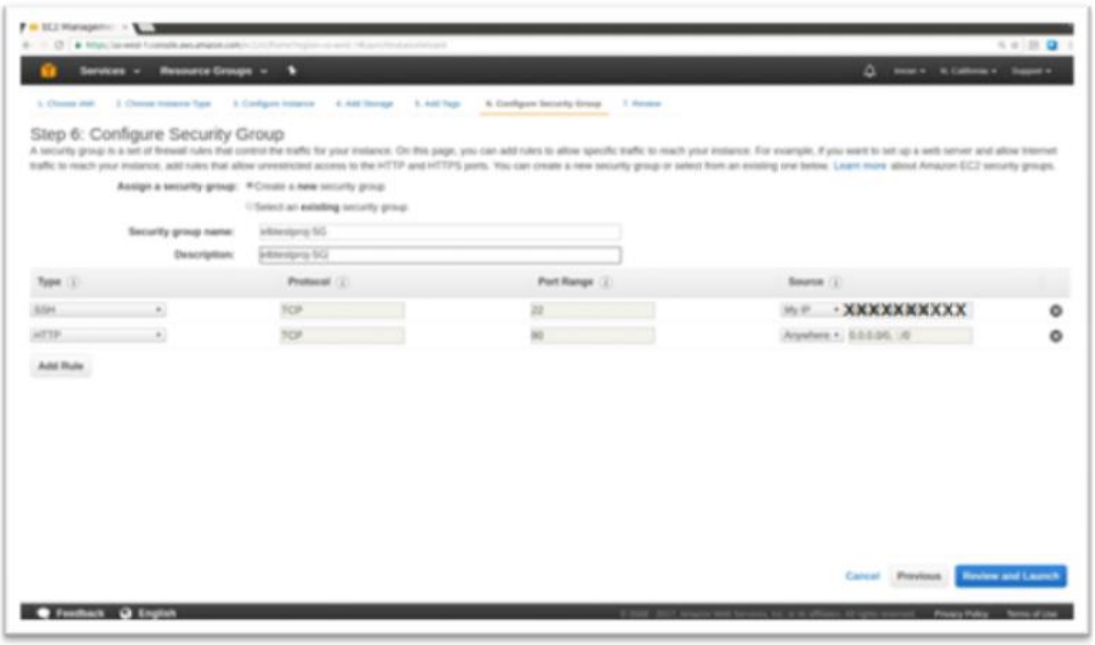
1.Configure instance details with all default settings.



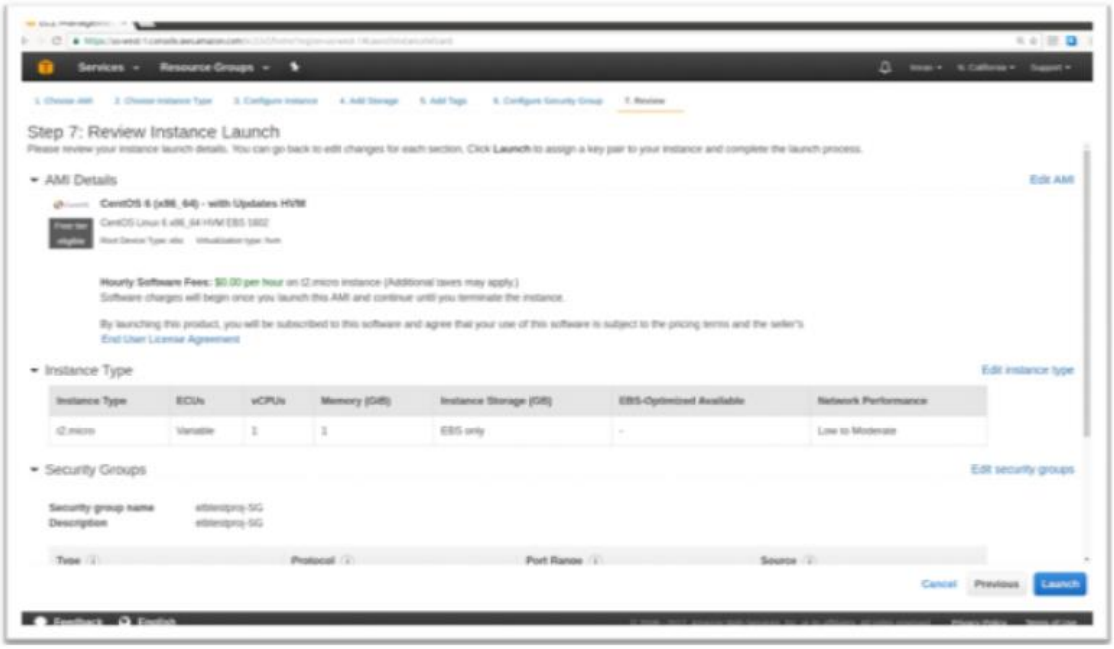




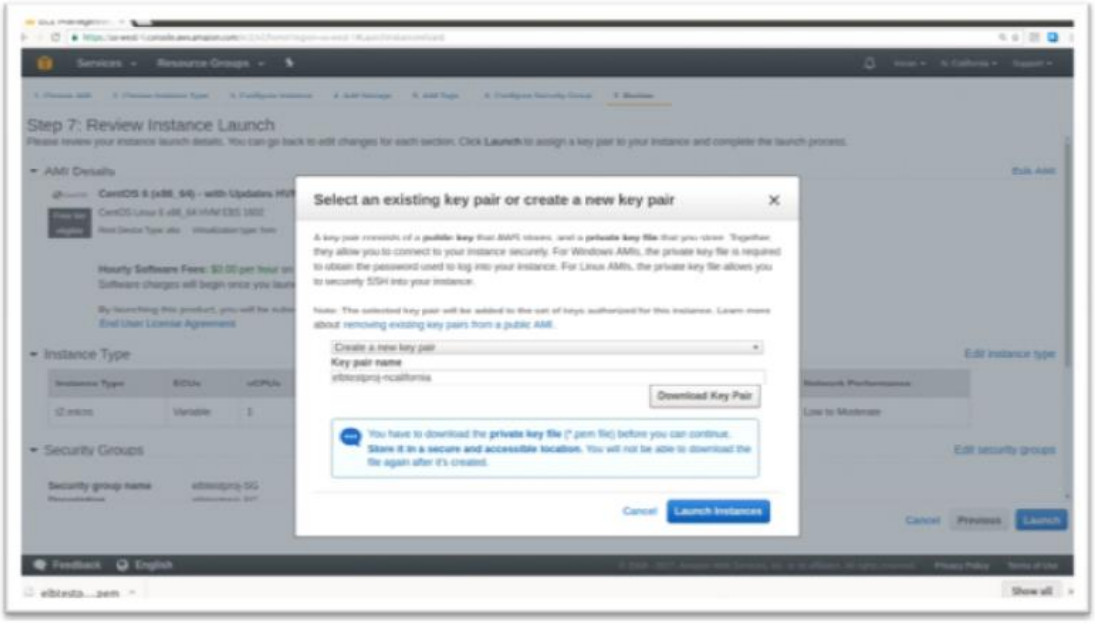
1.Security Group: We have to create the security group in order to control the traffic to your instance.



1.Launch Instance: Review your instance configuration and choose "Launch



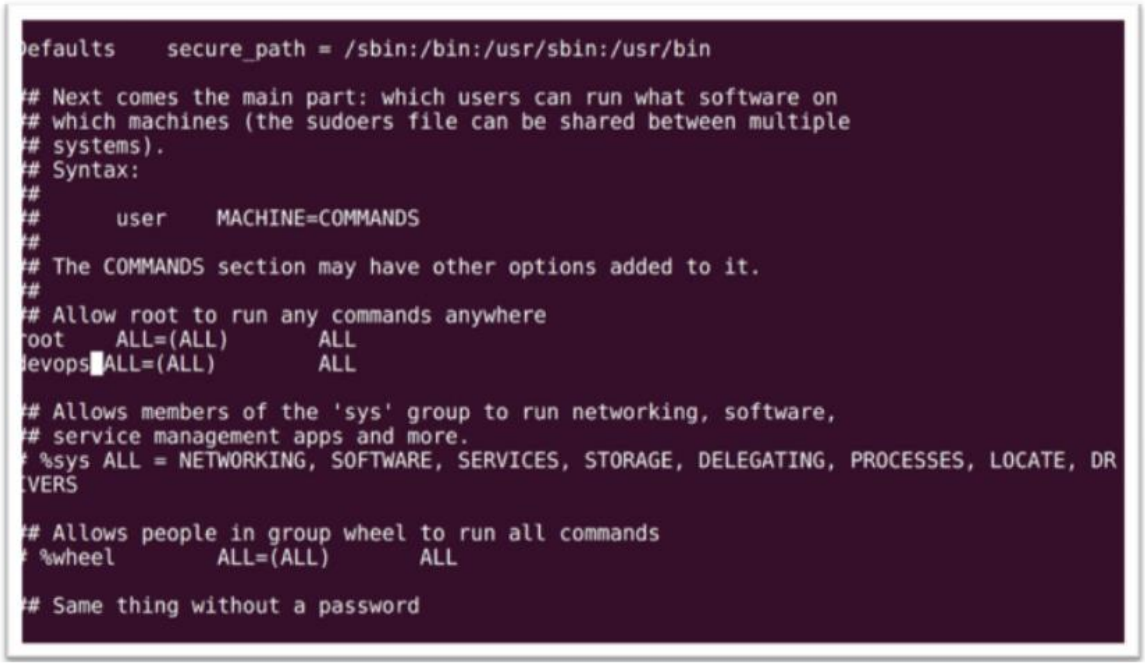
Create a key pair: Select "Create a new key pair" and assign a name. The key pair file (.pem) will download automatically - save this in a safe place as we will later use this file to log in to the instance. Finally, choose "Launch Instances" to complete the set up.



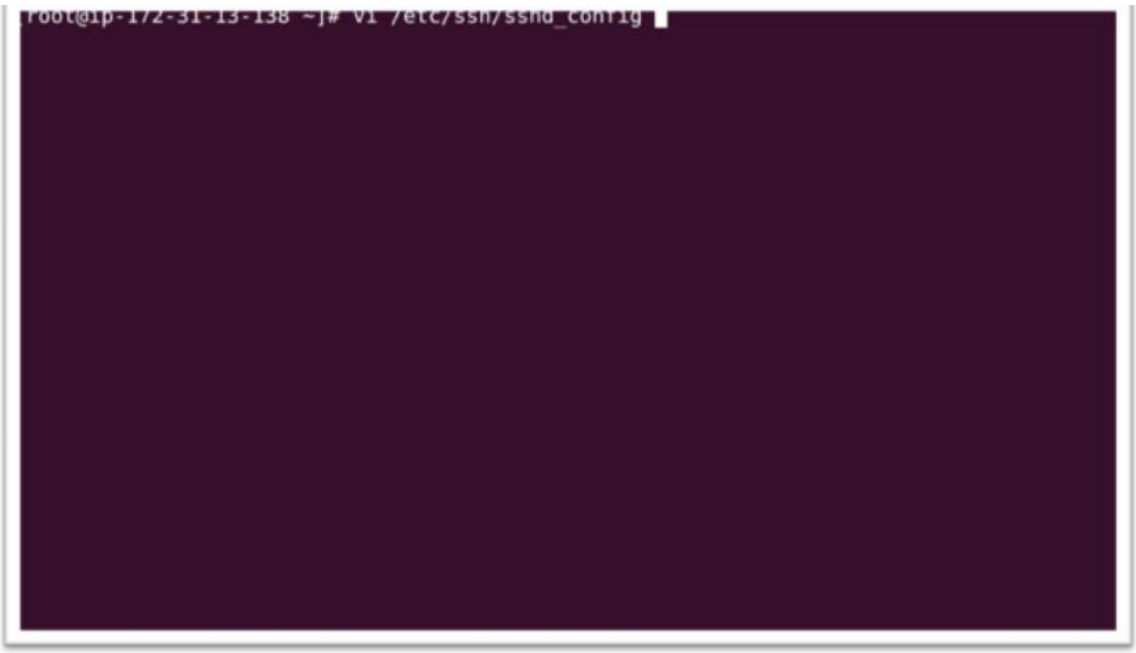
1.Login to ec2 instance, switch to root user & create one user and open sudoers file as shown below:

```
mran@DevOps:~/keys$ ssh -i elbtestproj-ncaligornia.pem centos@54.215.249.185
he authenticity of host '54.215.249.185 (54.215.249.185)' can't be established.
SA key fingerprint is SHA256:t79U8qI3X7oonppHac7puSDusdY256jcSBhxb1ibFbk.
re you sure you want to continue connecting (yes/no)? yes
arning: Permanently added '54.215.249.185' (RSA) to the list of known hosts.
centos@ip-172-31-13-138 ~]$ sudo -i
root@ip-172-31-13-138 ~]# useradd devops
root@ip-172-31-13-138 ~]# passwd devops
hanging password for user devops.
ew password:
AD PASSWORD: it is too simplistic/systematic
etype new password:
orry, passwords do not match.
ew password:
AD PASSWORD: it is based on a dictionary word
etype new password:
asswd: all authentication tokens updated successfully.
root@ip-172-31-13-138 ~]# visudo
```

1.Find entry for root user, below that add similar entry for your user.



1.Open SSHD_CONFIG file for Enabling password authentication



```
# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#RhostsRSAAuthentication no
# similar for protocol version 2
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# RhostsRSAAuthentication and HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication no
#PermitEmptyPasswords no
PasswordAuthentication yes

# Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes
ChallengeResponseAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no
```

1.Restart SSHD service. And now you can login to your user.

```
root@ip-172-31-13-138 ~]# vi /etc/ssh/sshd_config
root@ip-172-31-13-138 ~]# service sshd restart
Stopping sshd:                                             [  OK  ]
Starting sshd:                                             [  OK  ]
root@ip-172-31-13-138 ~]#
```

Amazon EBS

Amazon Elastic Block Store (Amazon EBS) provides persistent block storage volumes for use with Amazon EC2 instances in the AWS Cloud. Each Amazon EBS volume is automatically replicated within its Availability Zone to protect you from component failure, offering high availability and durability. Amazon EBS volumes offer the consistent and low-latency performance needed to run your workloads. With Amazon EBS, you can scale your usage up or down within minutes, all while paying a low price for only what you provision.

Some of the key features and benefits that EBS volumes have to offer:

- High performance volumes
- Availability
- Encryption capabilities
- Snapshot capabilities
- Access Management
- Elastic Volumes

Note: EBS volumes cannot be copied from one AWS region to another. To make a volume available outside of the Availability Zone, you can create a snapshot and restore that snapshot to a new volume anywhere in that region. You can copy snapshots to other regions and then restore them to new volumes there, making it easier to leverage multiple AWS regions for geographical expansion, data centre migration, and disaster recovery.

Amazon EBS Volume Types:

There are three different types of EBS volumes, each with their own sets of performance characteristics and associated costs:

General Purpose Volumes (SSD):

This volume provides base performance of 3 IOPS/GiB, with the ability to burst to 3,000 IOPS for extended periods of time. Gp2 volumes are ideal for a broad range of use cases such as boot volumes, small and medium-size databases, and development and test environments. Gp2 volumes support up to 10,000 IOPS and 160 MB/s of throughput.

Provisioned IOPS Volumes (SSD):

With Provisioned IOPS SSD (io1) volumes, you can provision a specific level of I/O performance. Io1 volumes support up to 20,000 IOPS and 320 MB/s of throughput. This allows you to predictably scale to tens of thousands of IOPS per EC2 instance.

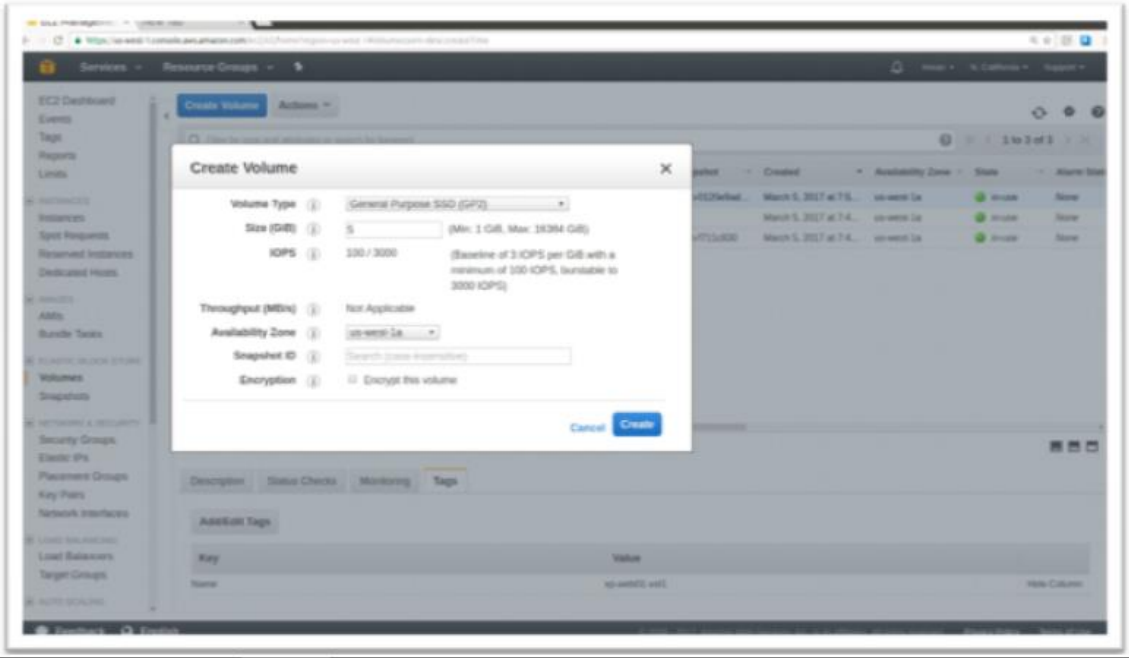
Magnetic Volumes:

Magnetic volumes are backed by magnetic drives and are suited for workloads where data is accessed infrequently, and scenarios where low-cost storage for small volume sizes is important. These volumes deliver approximately 100 IOPS on average, with burst capability of up to hundreds of IOPS, and they can range in size from 1 GiB to 1 TiB.

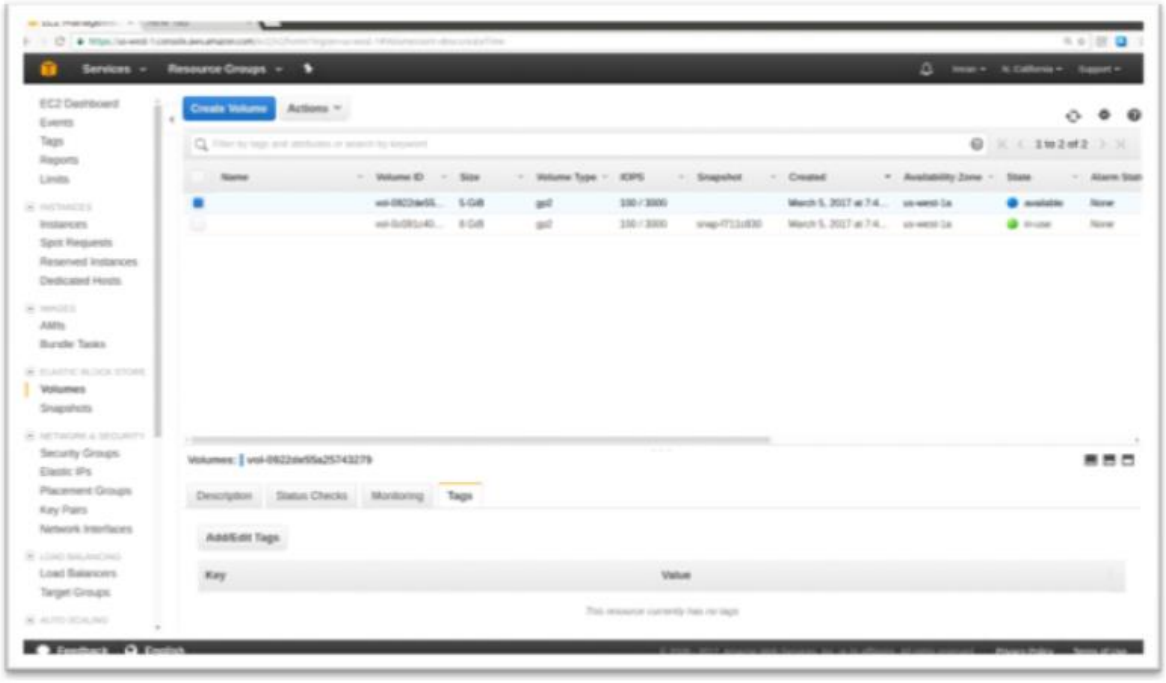
Creating, Attaching, Formatting & Mounting EBS Volume To An EC2 Instance:

Before we start knowing how to create Volume, let us create an EC2 instance of CentOS 6. To view and access your account's EBS Volumes using AWS Management Console, simply select the Volumes option from the EC2 dashboard's navigation pane. Click Volumes in the left pane, we can see Volume Management Dashboard. From that select the Create Volume option.

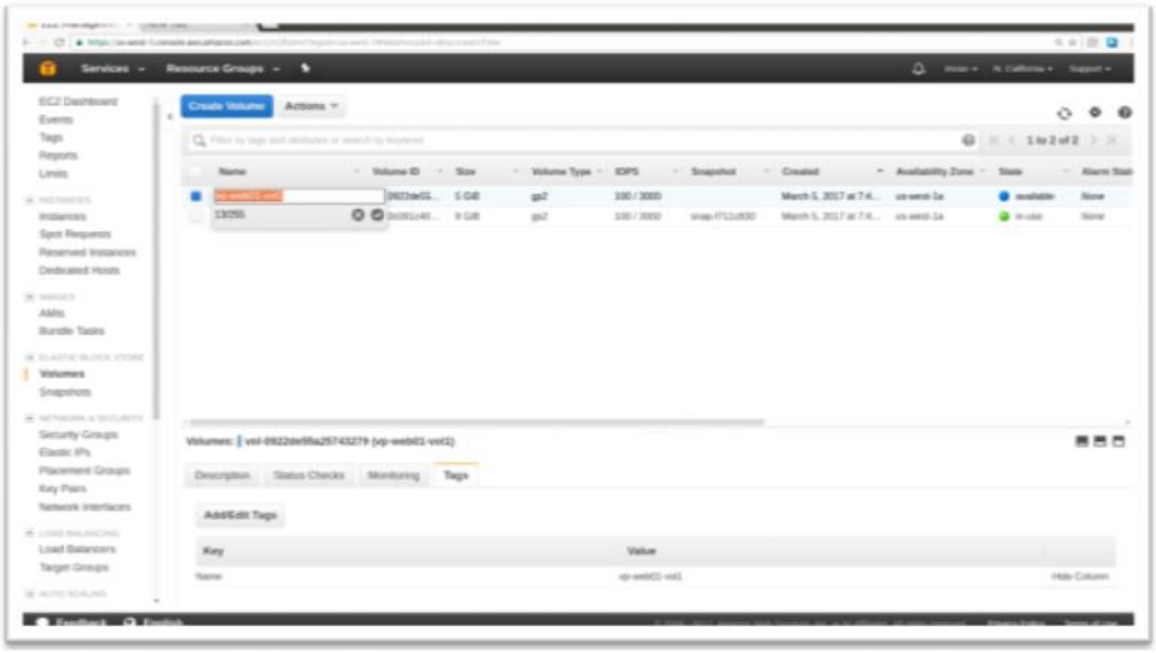
> This will pop up the Create Volume Dashboard as shown below:



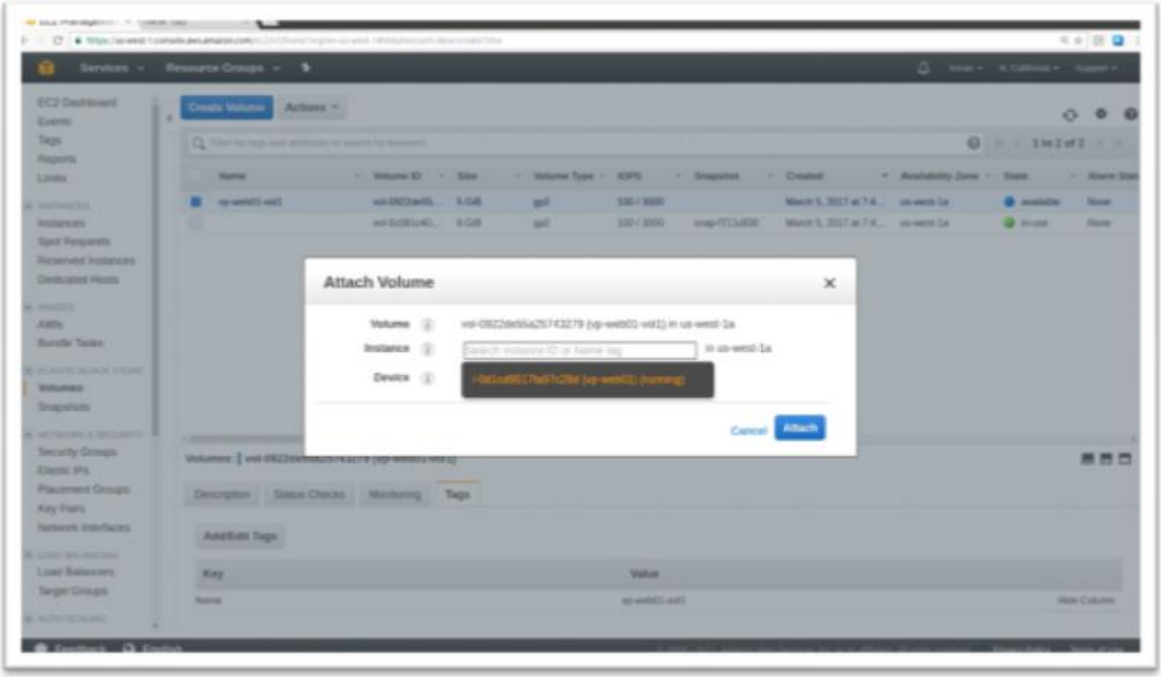
- Fill in the details as required in the Create Volume dialog box. Here I created a sample 5-GB general purpose volume
- After filling the configuration settings, select Create to complete the volume creation process. The new volume will take a few minutes to be available for use as shown in the below figure. Once the volume is created, we can now attach this volume to your running instance.



The created volume is available for use. We will tag the volume by a name which is used for future identification.



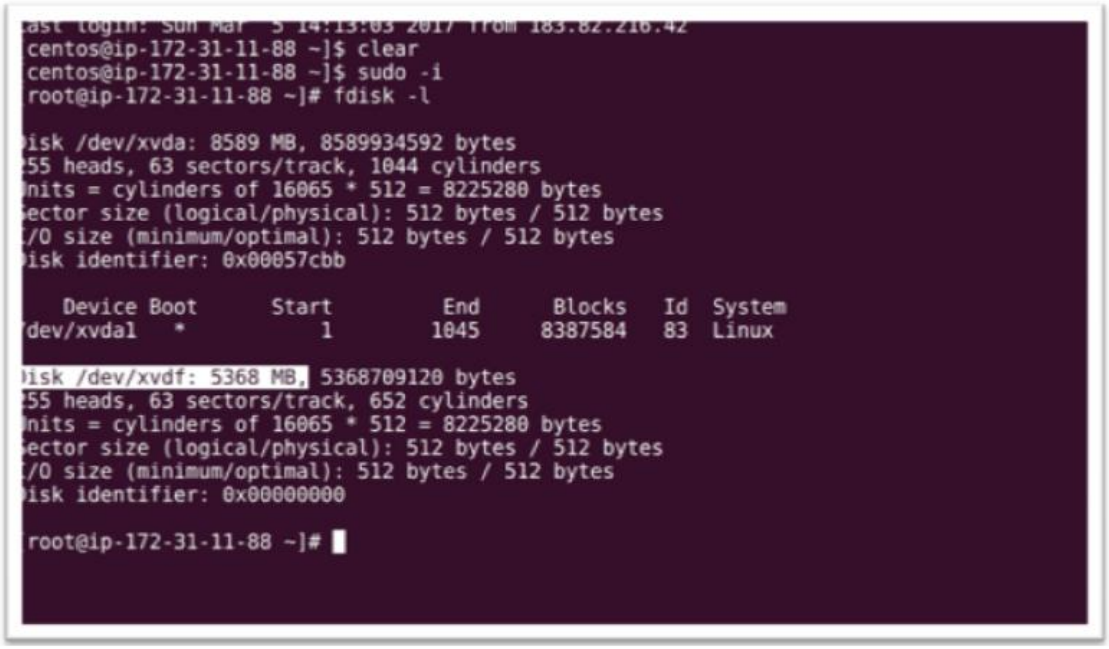
- Finally attach the volume to ec2 instance. We can attach multiple volumes to a single instance at a time, with each volume having a unique device name. Some of these device names are reserved, for example, /dev/sda1 is reserved for the root device volume.
- 2. To attach a volume, select the volume which is available for use from the Volume Management dashboard. Then select the Actions tab and click on the Attach Volume option. This will pop up the Attach Volume dialog box, as shown below:



Type your instance ID in the Instance field and provide a suitable name in the Device field as shown. Here, I provided the recommended device name of /dev/sdf to this volume. Click on Attach once you given the details. The volume attachment process takes a few minutes to complete. You are now ready to make the volume accessible from your instance.

Mounting Volume To The Instance:

After the volume is attached to an instance, you can format it and use it like other block device. Here I'm using the same EC2 instance (CentOS6) that we created earlier. To get started, login to the running instance using SSH. As it is a CentOS machine by default it will login to the centos user. So, run the following command to login to the root user and run the command to list the partitions of your instance. You should see a default /dev/xvda partition along with its partition table and an unformatted disk partition with the name /dev/xvdf as shown in the following screenshot. The /dev/xvdf command is the newly added EBS volume that need to be formatted.



Now create new partition with the available disk partition as shown below.

 # Fdisk /Dev/Xvdf

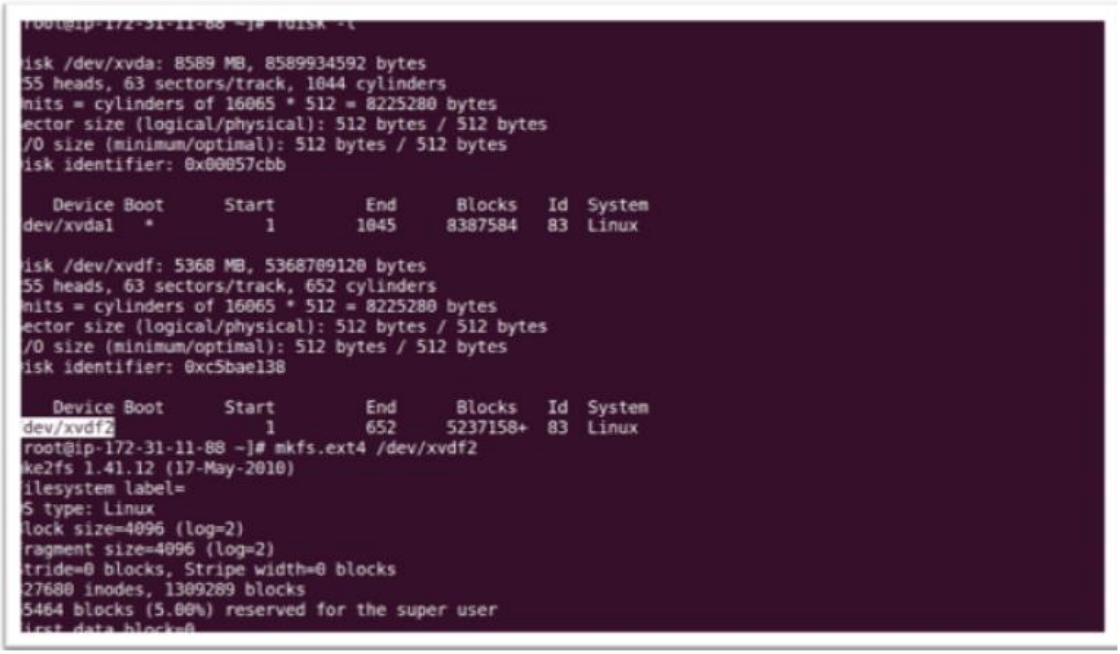
1.Use m to list out various options that can be used in fdisk.

2.Use p to list out the partition information first and

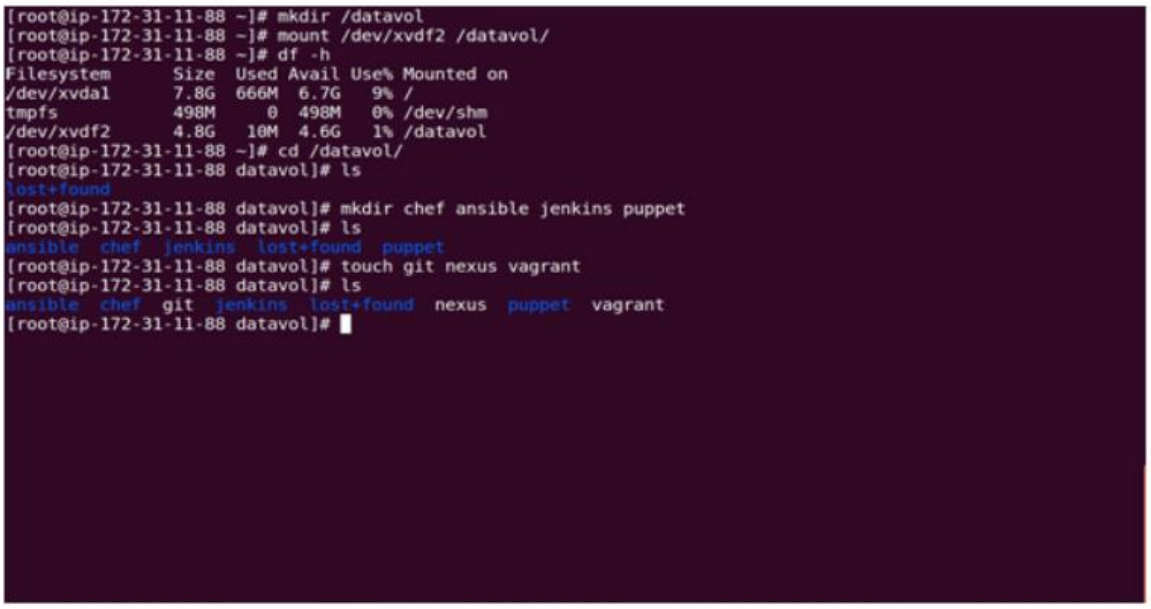
3.Use n to create a new partition.

Follow the steps as shown in the below screenshot.

After creating the partition, we have to format it with a filesystem of your choice. Here I have chosen ext4 filesystem.



Once formatting is done, we can mount that partition to a directory. Create a directory and mount it to a volume as shown below:



Backup & Restore

We will create a situation where we need to restore the lost data. We will delete few files from the mount point after taking the backup and then restore the deleted data.

Backup The EBS Volume By Taking Its Snapshot:

Amazon EBS provides the ability to save point-in-time snapshots of your volumes to Amazon S3. Amazon EBS Snapshots are stored incrementally: only the blocks that have changed after your last snapshot are saved, and you are billed only for the changed blocks. If you have a device with 100 GB of data but only 5 GB has changed after your last snapshot, a subsequent snapshot consumes only 5 additional GB and you are billed only for the additional 5 GB of snapshot storage, even though both the earlier and later snapshots appear complete.

When you delete a snapshot, you remove only the data not needed by any other snapshot. All active snapshots contain all the information needed to restore the volume to the instant at which that snapshot was taken. The time to restore changed data to the working volume is the same for all snapshots.

Snapshots can be used to instantiate multiple new volumes, expand the size of a volume, or move volumes across Availability Zones. When a new volume is created, you may choose to create it based on an existing Amazon EBS snapshot. In that scenario, the new volume begins as an exact replica of the snapshot.

The following are key features of Amazon EBS Snapshots:

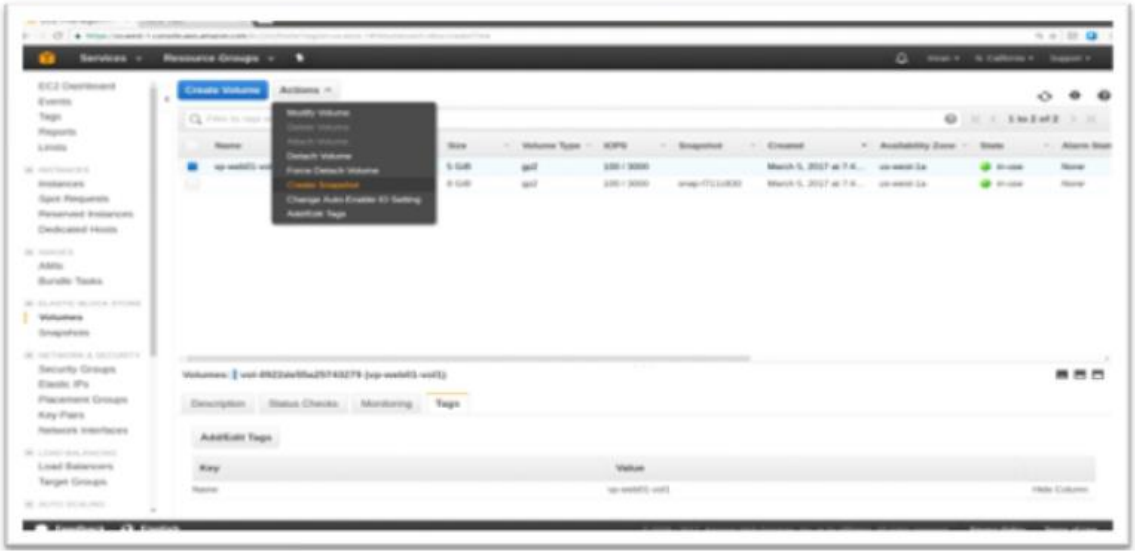
1.Immediate access to Amazon EBS volume data

2.Resizing Amazon EBS volumes

3.Sharing Amazon EBS Snapshots

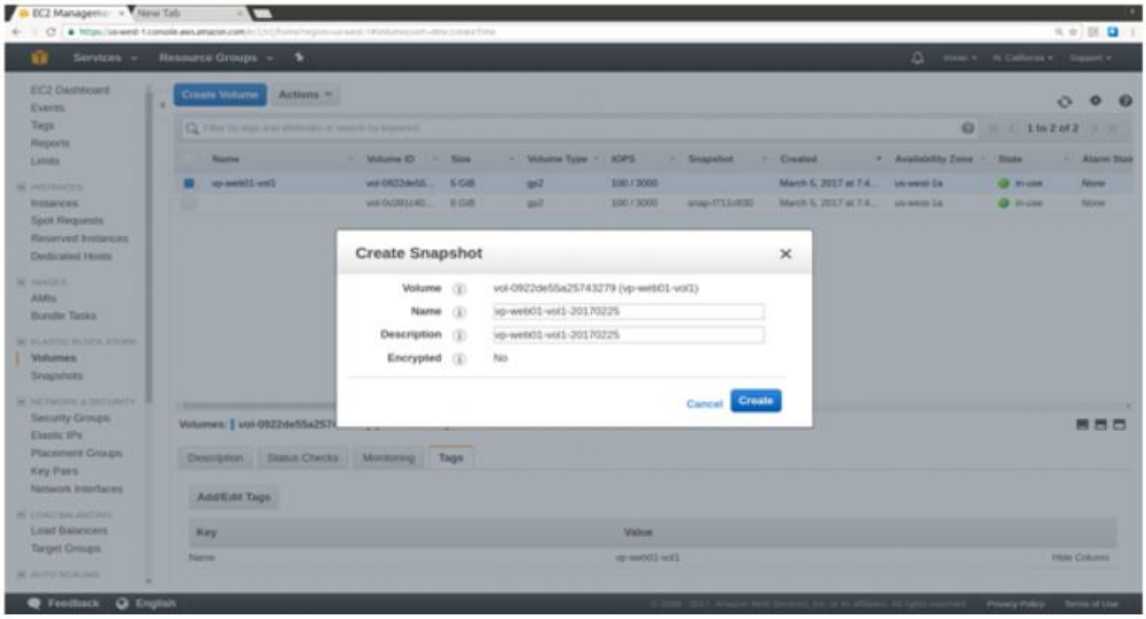
4.Copying Amazon EBS Snapshots across AWS regions

To create a snapshot from the AWS Management Console, go to the Volume Management Dashboard, Select the Volume, Click on Actions and Click on Create Snapshot.

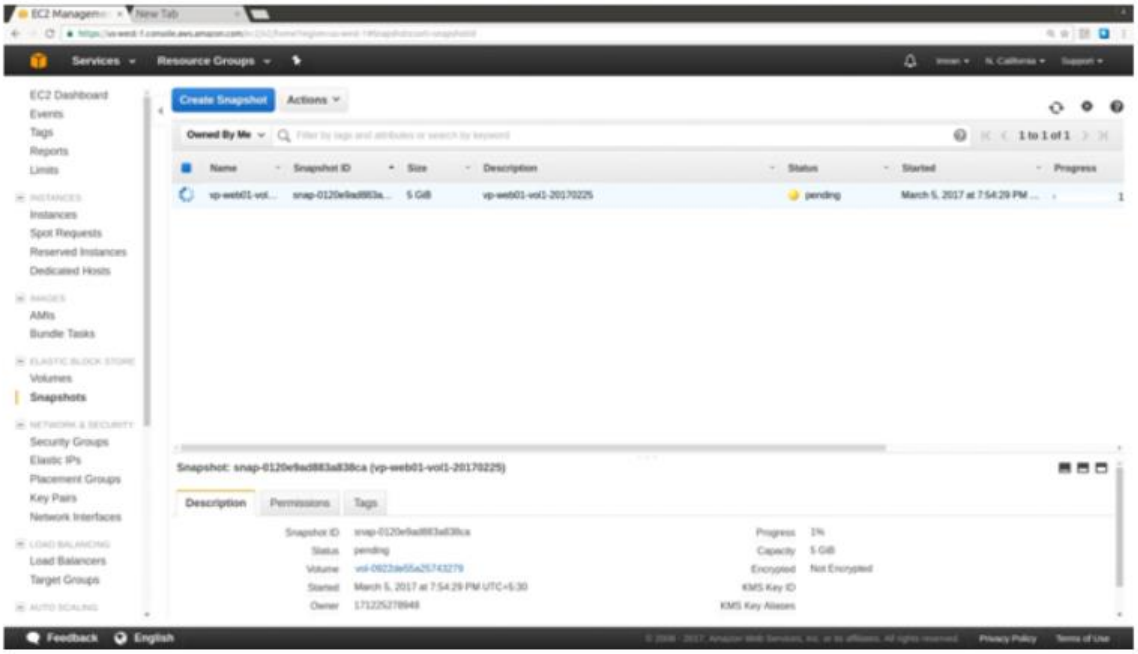


Note: It's a good practice to stop your instance before taking a snapshot if you are taking a snapshot of its root volume.

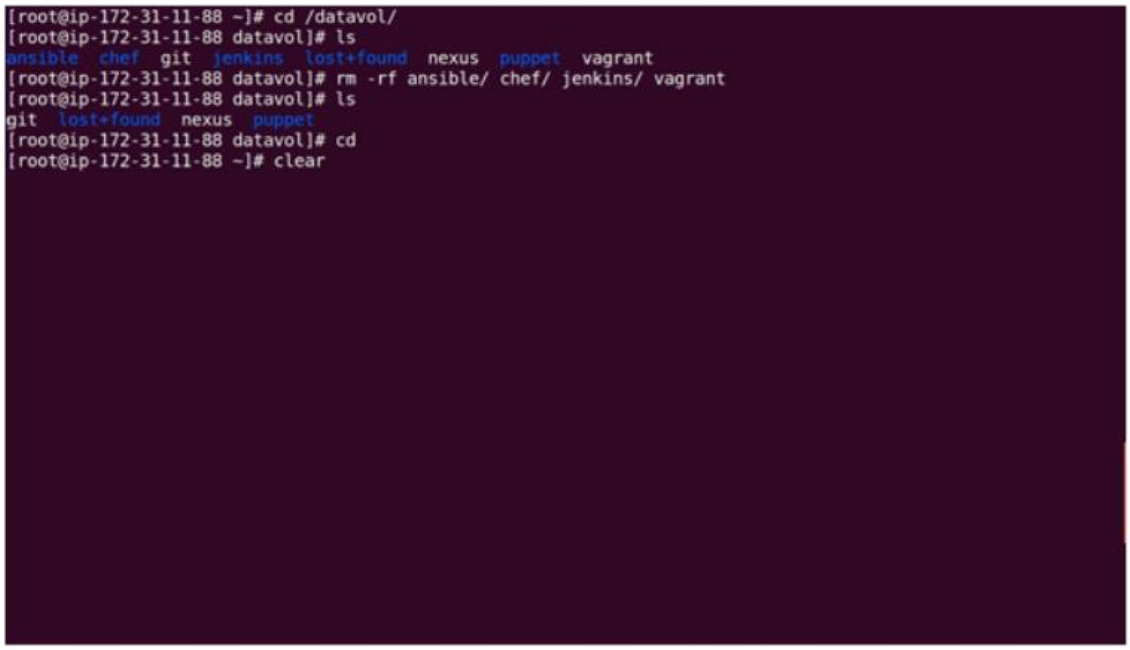
You will see Create Snapshot dialog box as shown in the following screenshot. Provide Suitable Name and Description for your new Snapshot. Here there is no Encryption for your snapshot because the Volume is not encrypted. Snapshots of encrypted volumes will be encrypted automatically.



After filling the required details Click on Create to complete the snapshot process. The process will take few minutes to complete.

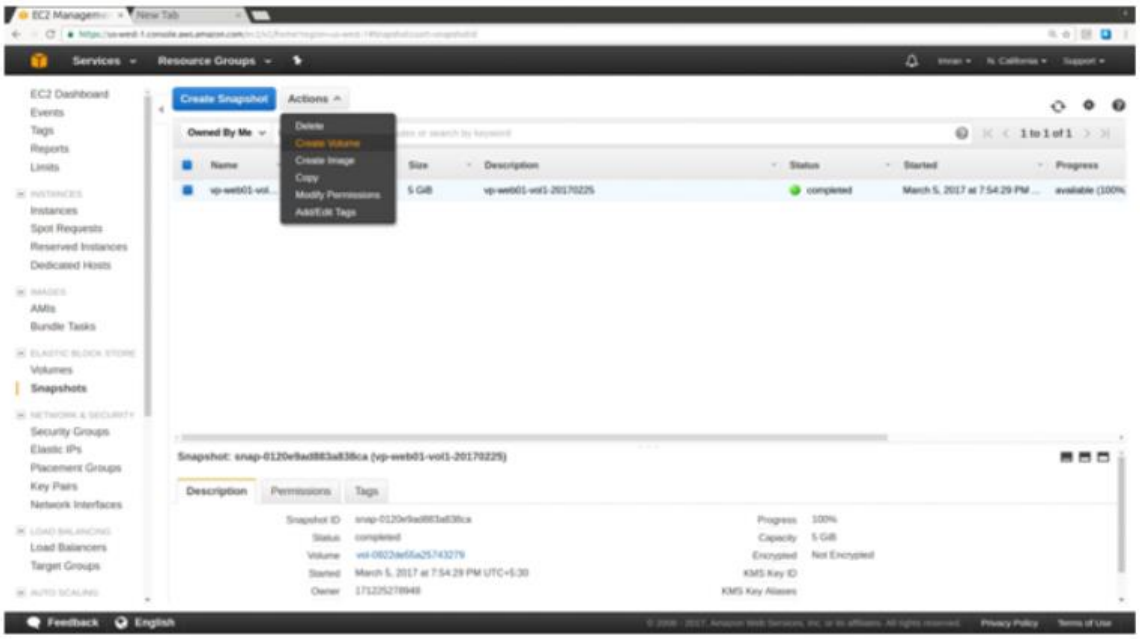


After creating the snapshot, we will delete few files from the mount point.

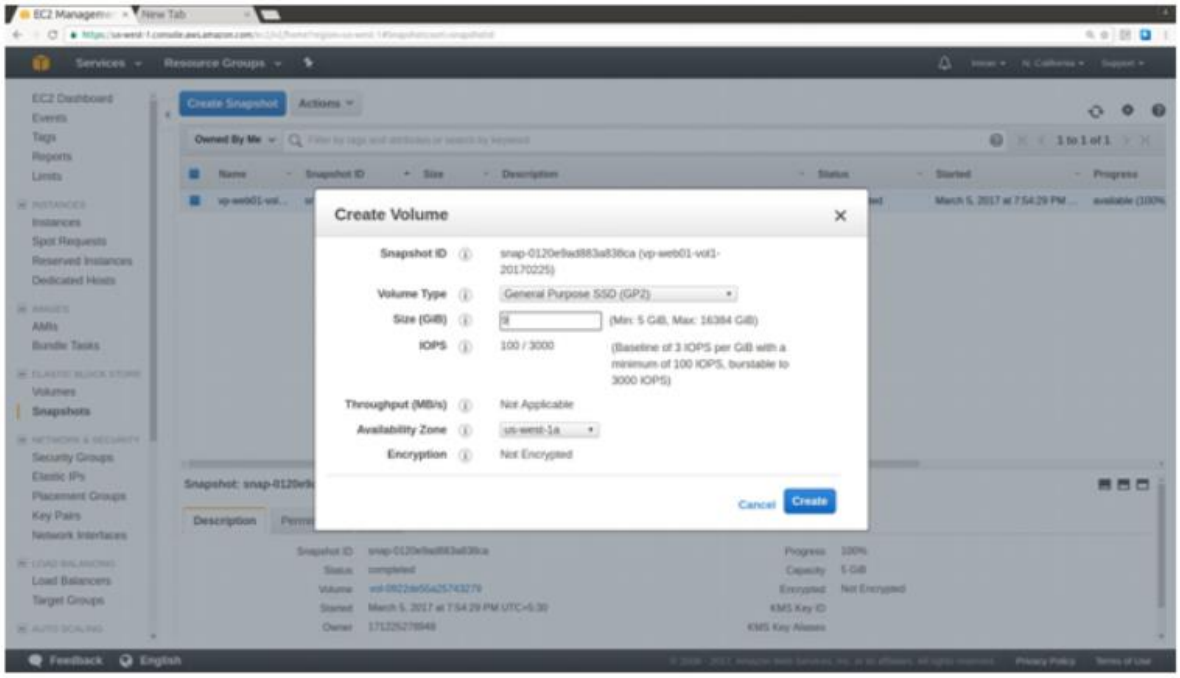


Create Volume From Snapshot & Resize:

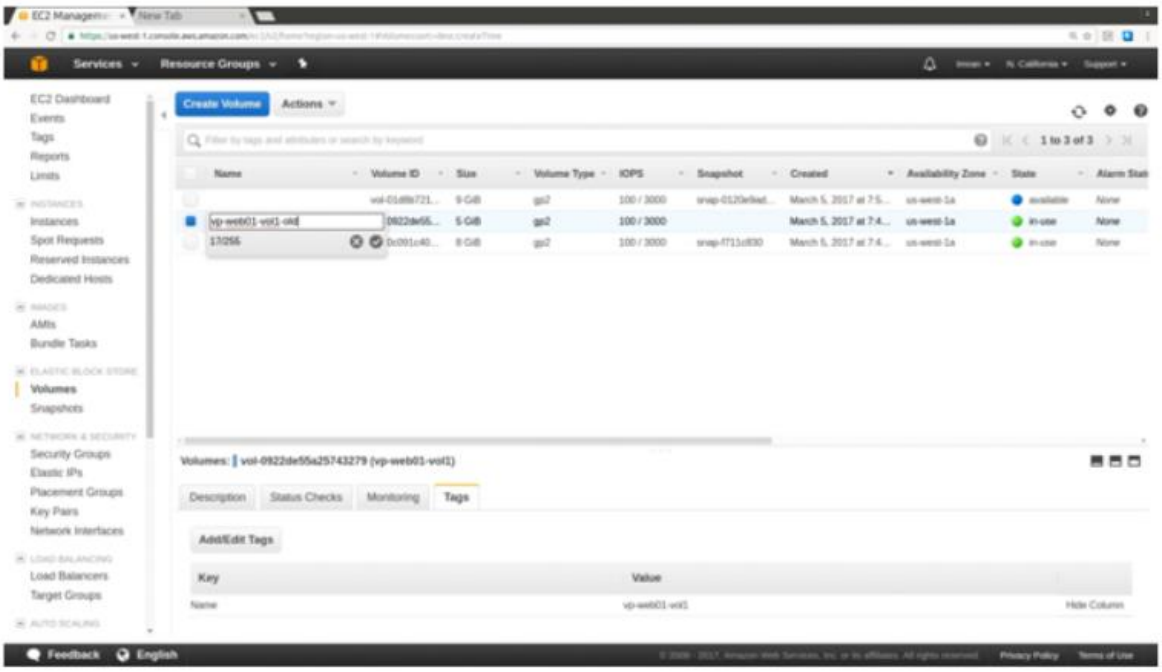
Now if we want to recover the lost data, we have to create a new volume from snapshot & replace old volume with the new one. We will also increase the volume size. Click Snapshots in left pane of EC2 Dashboard, Select the snapshot from which new volume has to be created, Click on Actions and select Create volume, you will get a create Volume pop up dialog box as shown below:



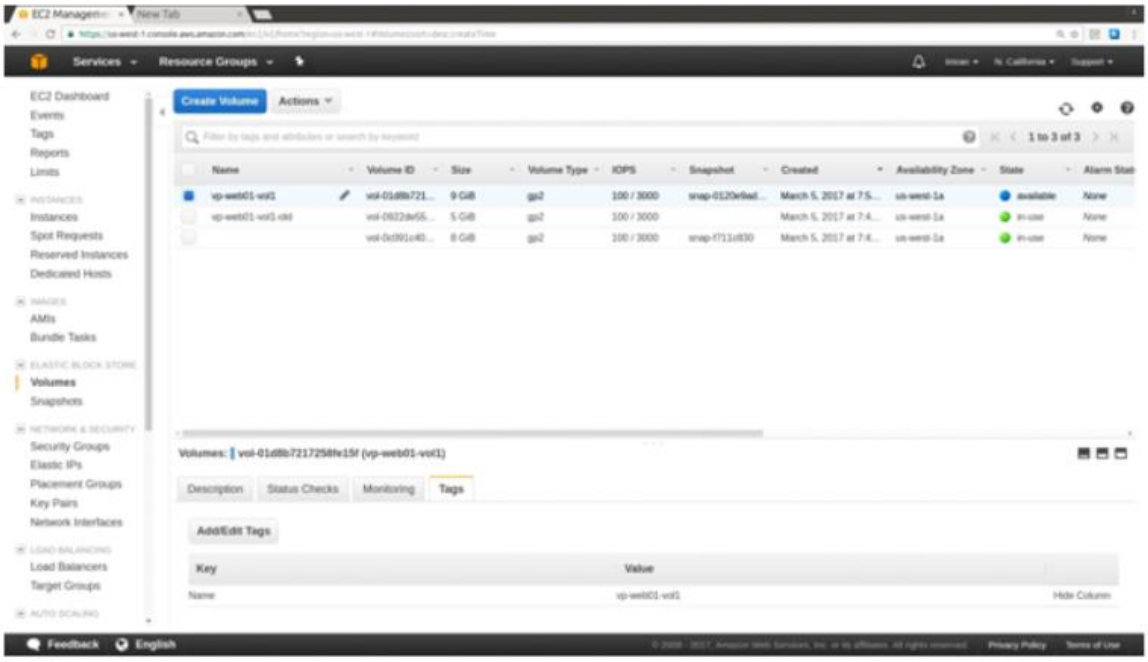
Fill the details to create new volume. Here the new volume size must be greater than 5GB because we created this snapshot from 5GB volume which is attached to the instance. And the Availability Zone should be same where the instance is created. Once this is done Click on Create to complete the process.



Tag the old volume with -old extension for identification and tag new volume with other name as shown below:



Once the volume is created it is available for use.

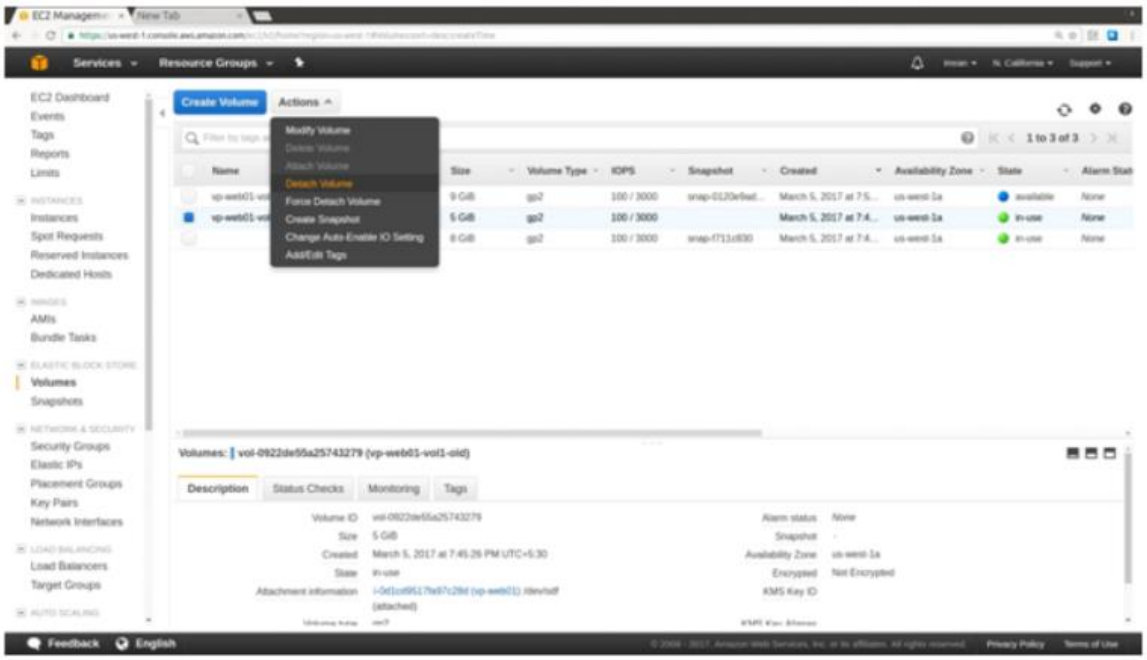


Unmount& Detach Old Volume:

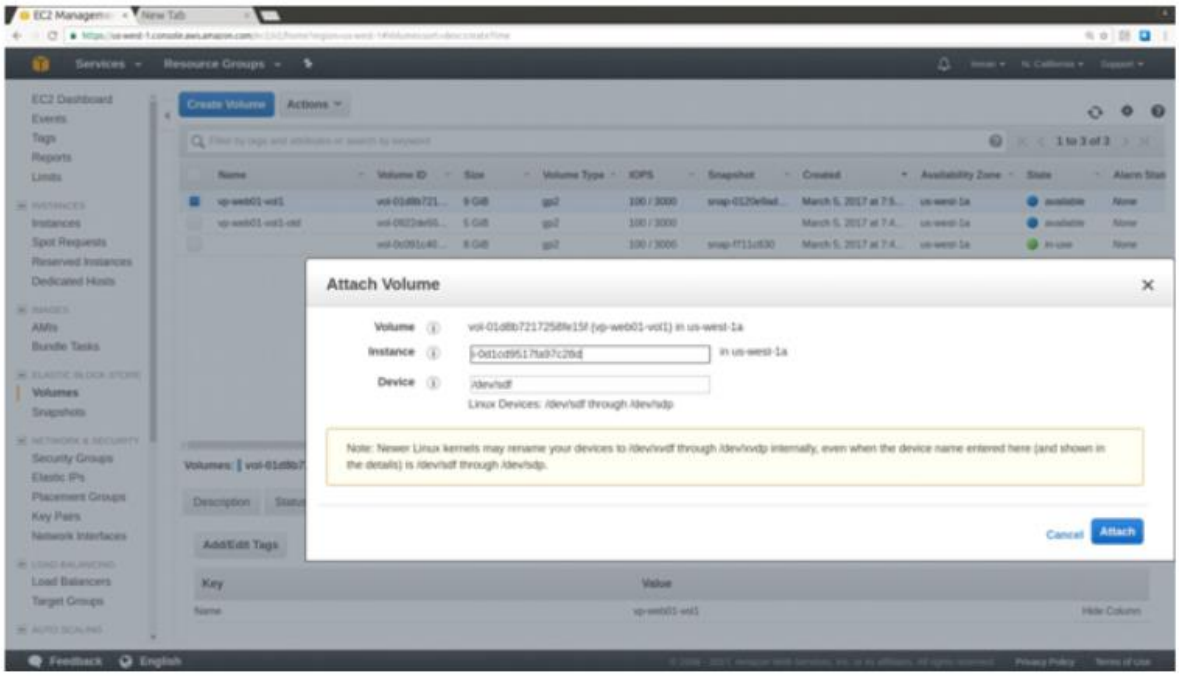
To unmount and detach old volume from the instance first check the mount points in the created partition and then unmount the mounted directory as shown below:

```
[root@ip-172-31-11-88 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1      7.8G  666M  6.7G   9% /
tmpfs           498M     0  498M   0% /dev/shm
/dev/xvdf2      4.8G   10M  4.6G   1% /datavol
[root@ip-172-31-11-88 ~]# umount /datavol/
[root@ip-172-31-11-88 ~]#
```

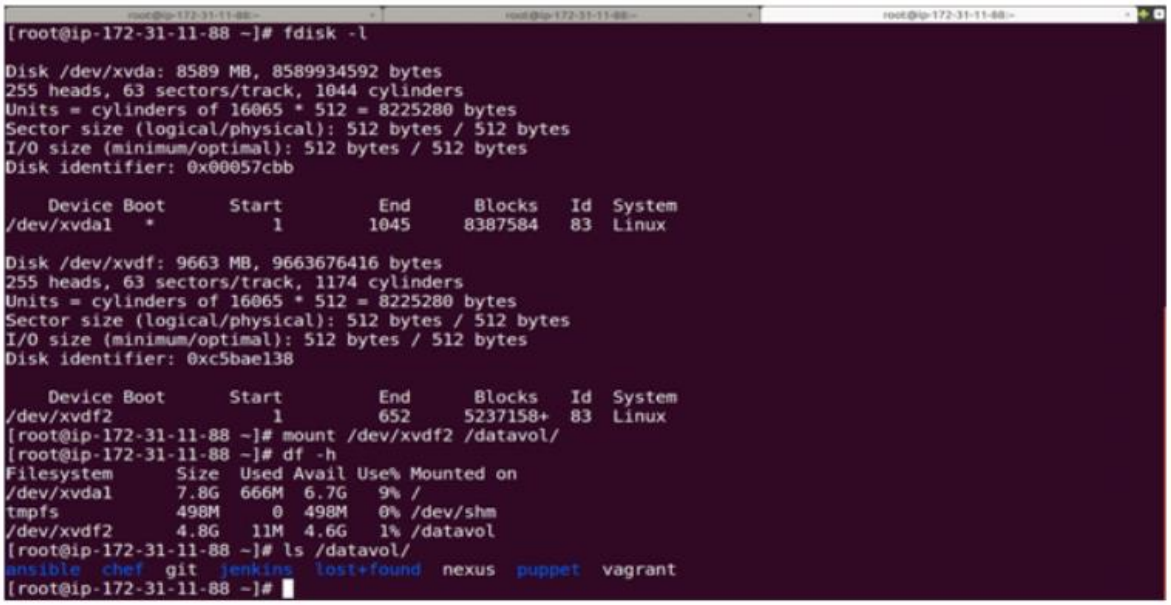
Once we are done with unmounting then detach the old volume from the instance. Select the volume to be detached, click on actions and select detach volume as shown below:



Finally attach the new volume to the instance by selecting the new volume, Click on Actions and select Attach Volume. We will get Attach Volume pop up dialog box as shown below: Provide the instance ID device partition.



After attaching the volume mount, it to the instance as we done before

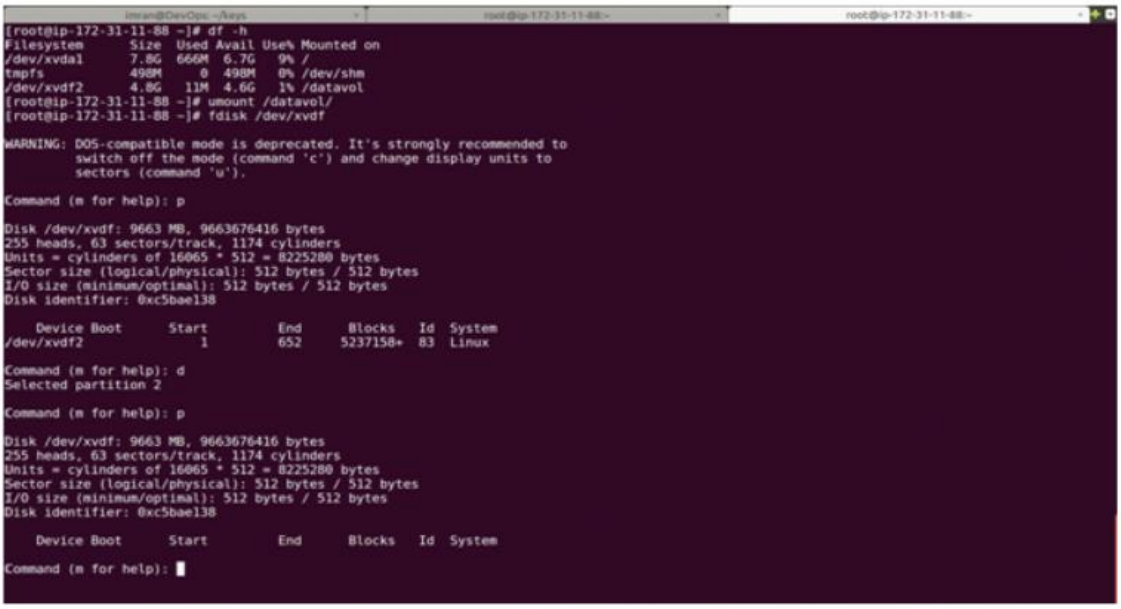


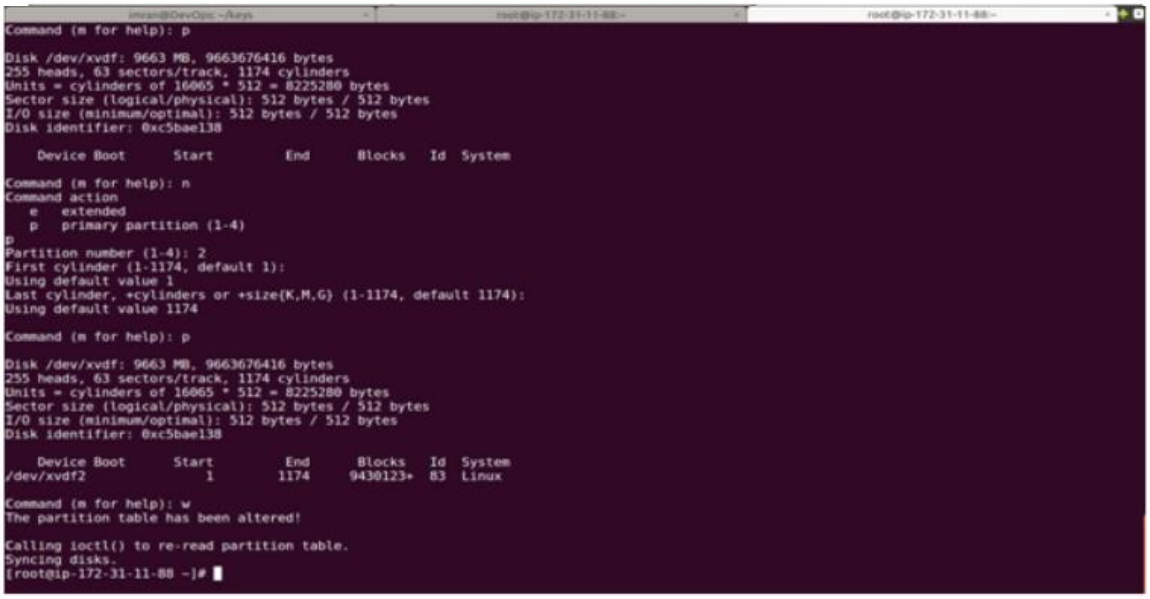
In the above screenshot, we can see that the data is restored.

But the mount point size is still around 5 GB even though we changed the EBS volume size to 9 GB. This is because the rest of the volume is still not resized and we need to resize it.

Resizing The Increased Volume:

We have already mounted the new volume but to resize it we need to unmount, delete partition, recreate it with new size, resize it and mount it. Follow the same steps to unmount, delete and create new partition as shown below:

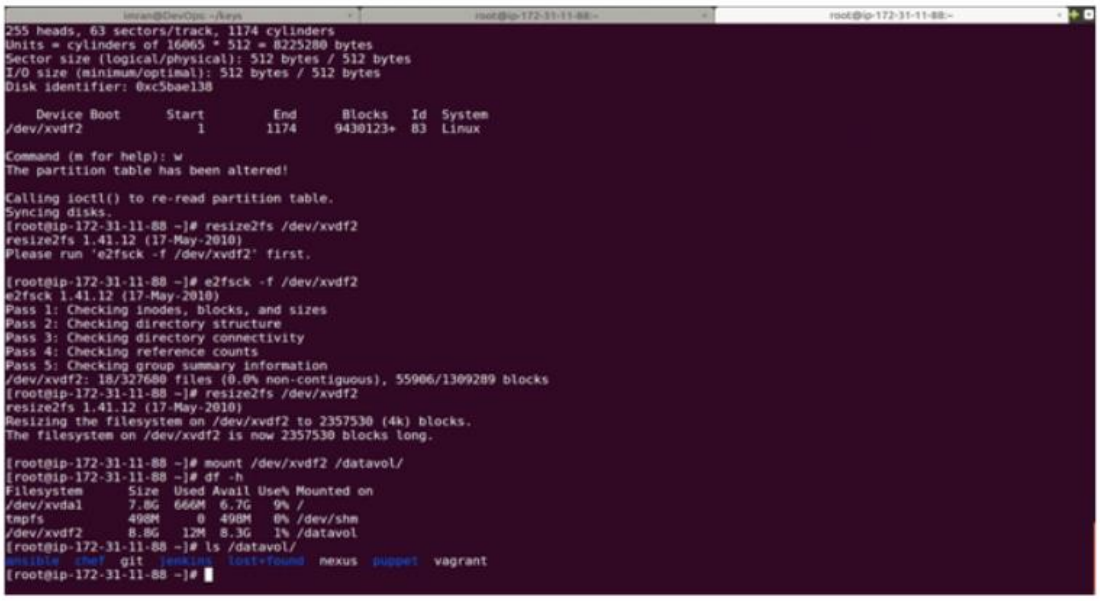




Resize The Volume:

In order to resize the volume, we need to run the command # resize2fs /dev/xvdf2 (If we run this command it gives an error to run another command because new volume is greater than old volume) Follow the steps as shown in the below screenshot to resize the volume.

Note: If the new volume is also of same size as old volume then no need of resizing it, just we have to mount it after attaching to the instance.



Now the Volume size is increased and data is also restored.

AWS VPC

Virtual private cloud (VPC) is a virtual network dedicated to your AWS account. It is logically isolated from other virtual networks in the AWS cloud. You can launch your AWS resources, such as Amazon EC2 instances, into your VPC. You can configure your VPC; you can select its IP address range, create subnets, and configure route tables, network gateways, and security setting.

Subnet is a range of IP addresses in your VPC. You can launch AWS resources into a subnet that you select. Use a public subnet for resources that must be connected to the Internet, and a private subnet for resources that won't be connected to the Internet.

Private Subnet: A private subnet sets that route to a NAT instance. Private subnet instances only need a private IP and internet traffic is routed through the NAT in the public subnet. You could also have no route to 0.0.0.0/0 to make it a truly private subnet with no internet access in or out.

Public Subnet: A public subnet routes 0.0.0.0/0 through an Internet Gateway (igw). Instances in a public subnet require public IPs to talk to the internet.

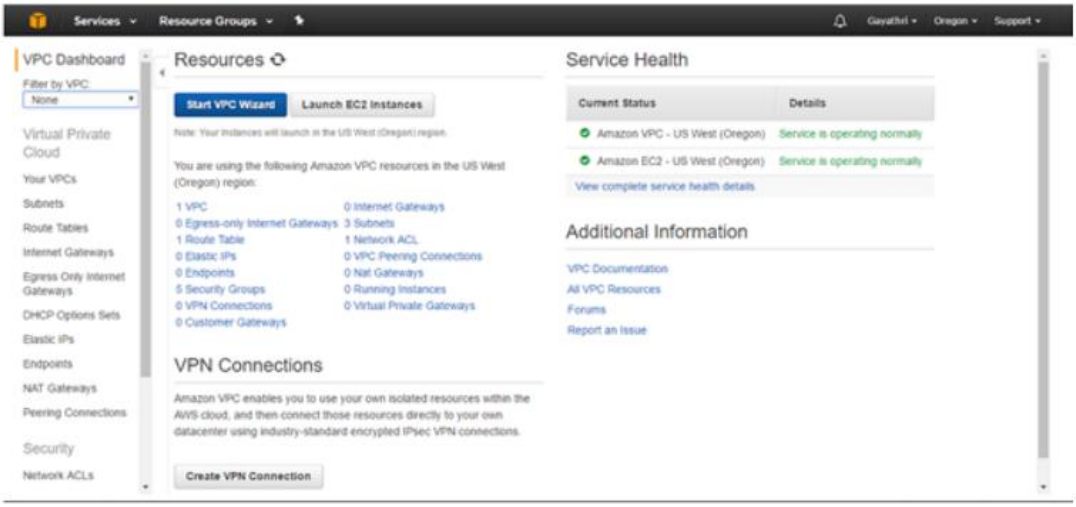
Network Address Translation (NAT) gateway is used to enable instances in a private subnet to connect to the Internet or other AWS services, but prevent the Internet from initiating a connection with those instances.

An Internet Gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between instances in your VPC and the Internet. It therefore imposes no availability risks or bandwidth constraints on your network traffic. An Internet gateway serves two purposes: to provide a target in your VPC route tables for Internet-routable traffic, and to perform network address translation (NAT) for instances that have been assigned public IPv4 addresses. An Internet gateway supports IPv4 and IPv6 traffic.

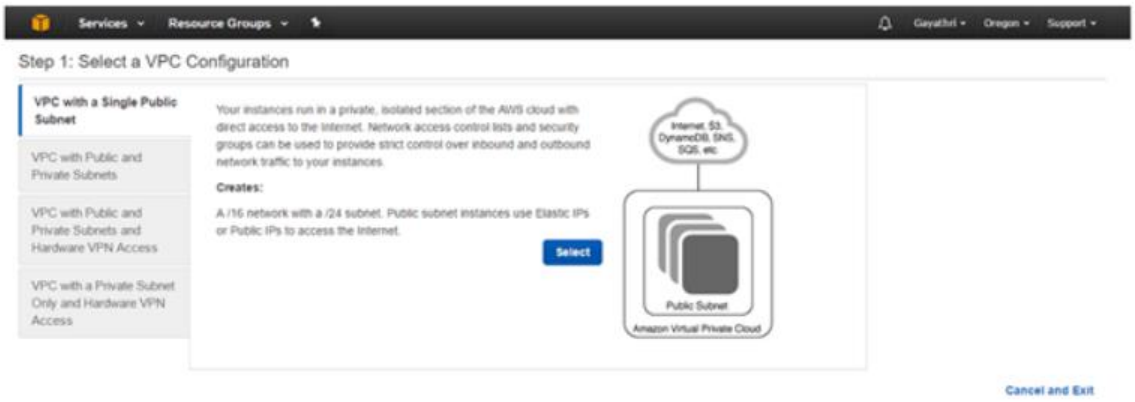
A Route Table contains a set of rules, called routes that are used to determine where network traffic is directed.
Each subnet in your VPC must be associated with a route table; the table controls the routing for the subnet. A subnet can only be associated with one route table at a time, but you can associate multiple subnets with the same route table.

Creating VPC:

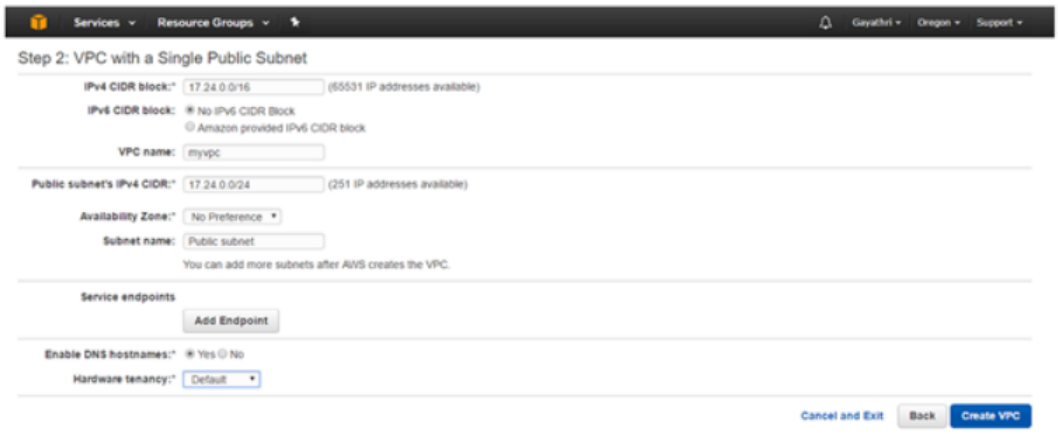
Go to VPC Dashboard from AWS main Dashboard as shown below, and click on Start VPC Wizard



You will a VPC configuration page, select VPC with a single Public subnet and click on select

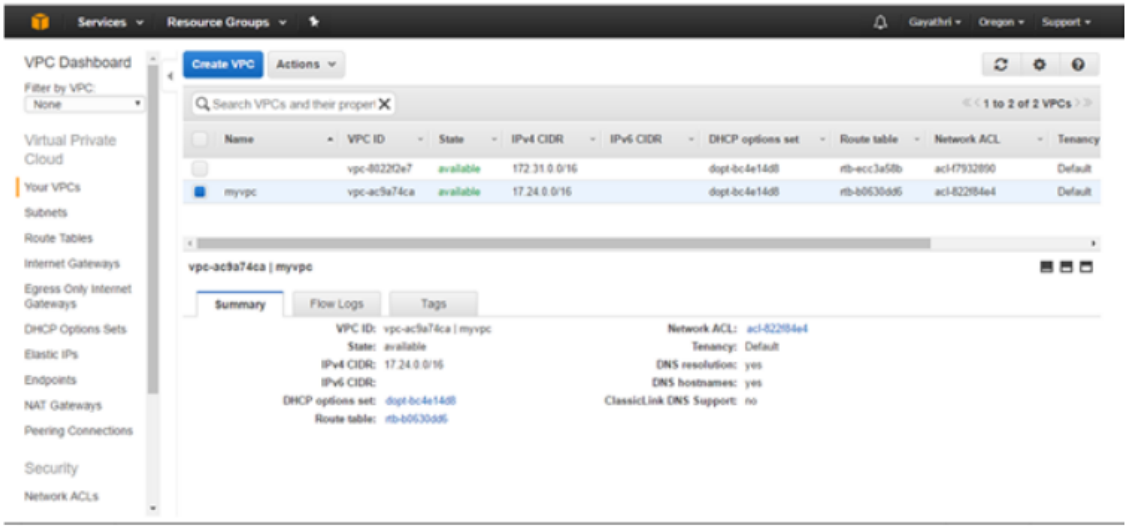


Specify the IPV4 CIDR block range for subnet, provide a name for vpc and click on create vpc.



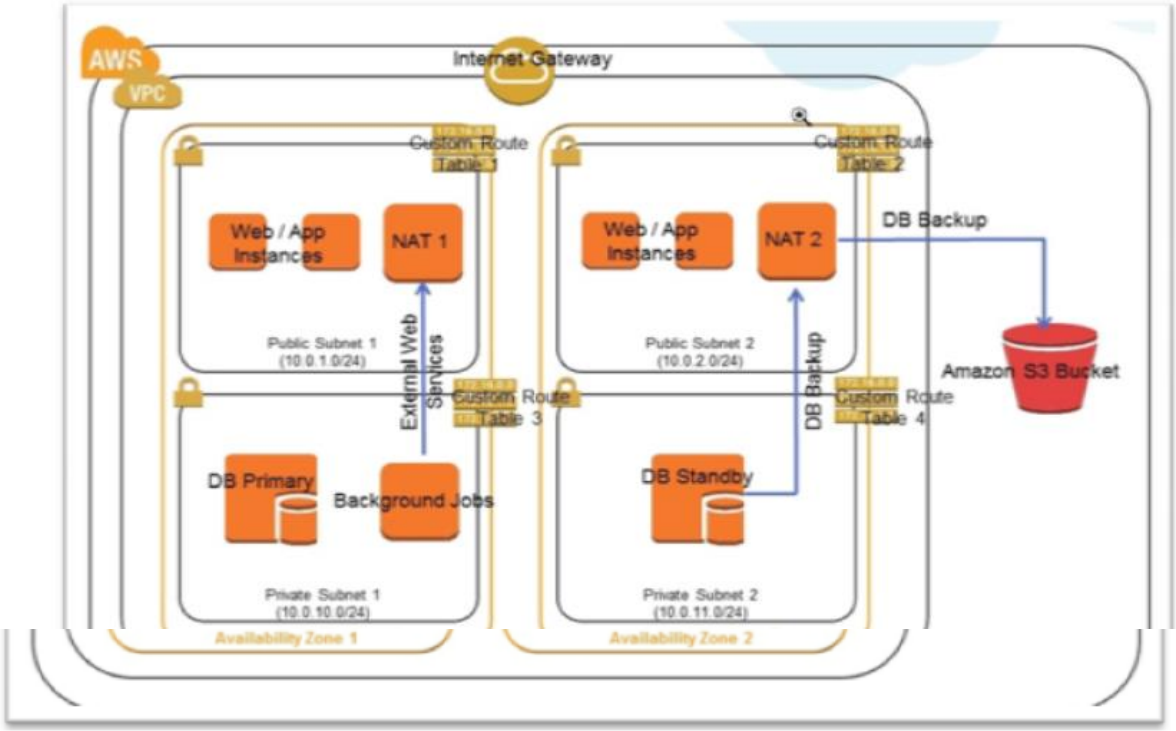
Your vpc is successfully created and is available to attach it to instances.

It takes few minutes come to available state. Once it is available select the vpc and click on actions to attach it to the instance.



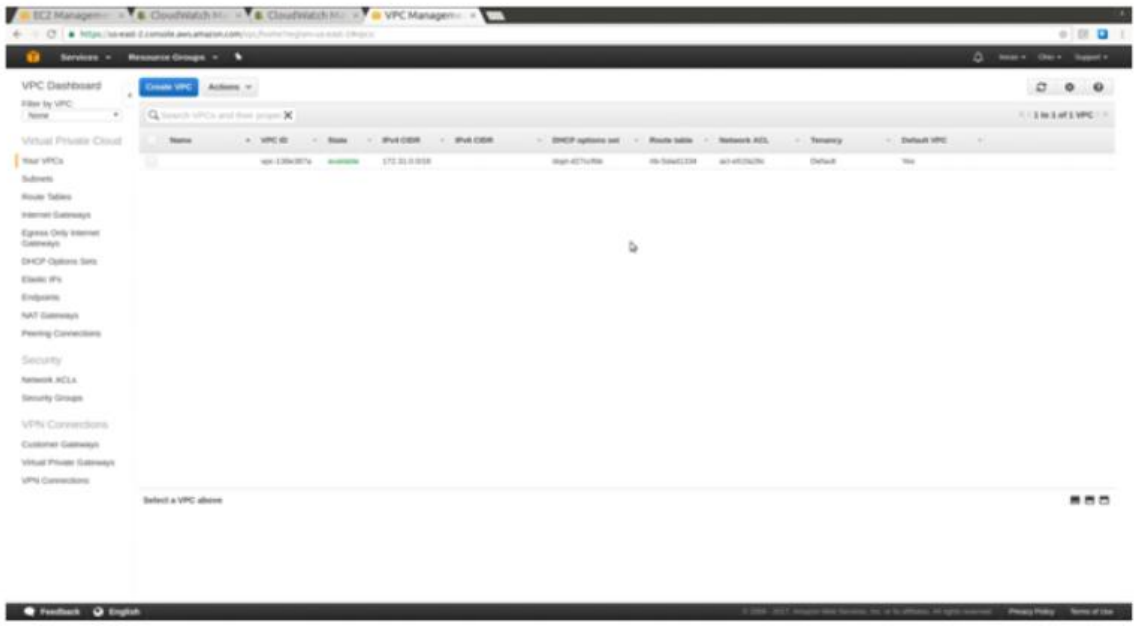
In the summary tab of vpc you can see the complete details of vpc

Creating Highly Available VPC



Highly available VPC spans over multiple zones. Even if one zone goes down our services spanned over the other zone will be still serving the user traffic. If you see from above diagram we have web, DB and backend services in two zones. While creating Ec2 instance we can decide now on which subnet our instance to create. So, for example we will create web01 in one subnet (located in zone 1a) and web02(located in zone 1b) in other subnet. So, if zone 1a goes down we still have web02 serving user traffic from zone 1b. We are going to create HA VPC manually and not with the wizard.

Creating VPC: Go to VPC from AWS main Dashboard



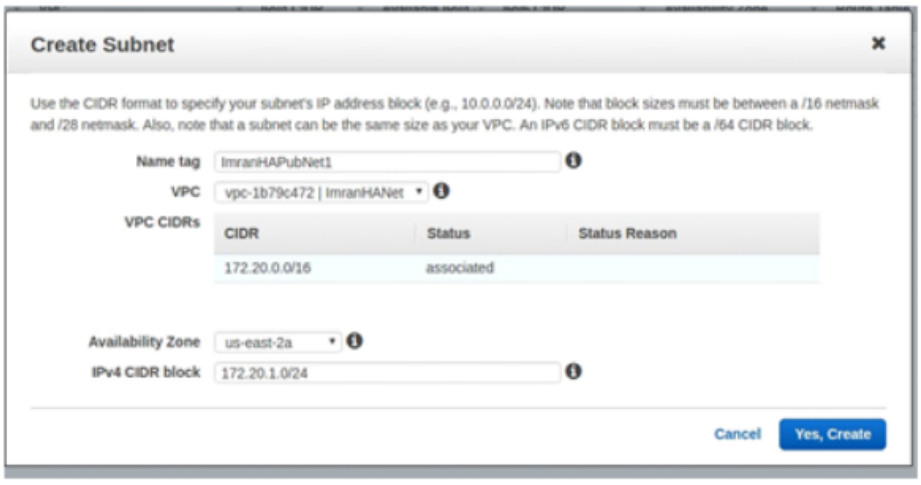
1.Click on your VPCs on left side of navigation pane and Click on Create VPC.

2.Create CIDR block /16 and private IP range of your choice as shown below

1.Go to Subnets and Click on create Subnets.

2.Create first public subnet from same VPC range with /24 CIDR block.

3.Select the Availability Zone as us-east-2a.

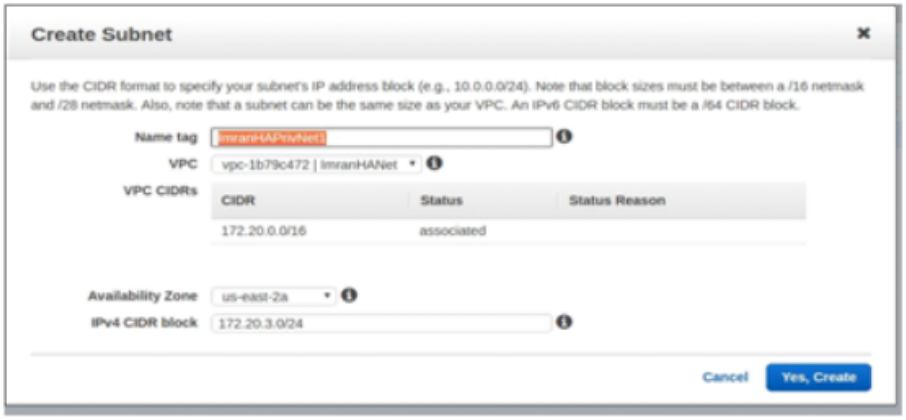


1.Create second public subnet from same VPC range with /24 CIDR block.

2.Select the Availability Zone as us-east-2b.

1.Create First PRIVATE subnet from same VPC range with /24 CIDR block.

2.Select the Availability Zone as us-east-2a.

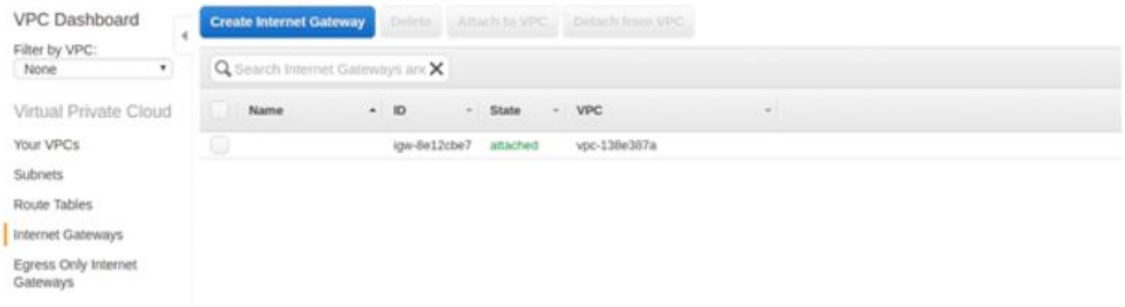


1.Create Second PRIVATE subnet from same VPC range with /24 CIDR block.

2.Select the Availability Zone as us-east-2b.

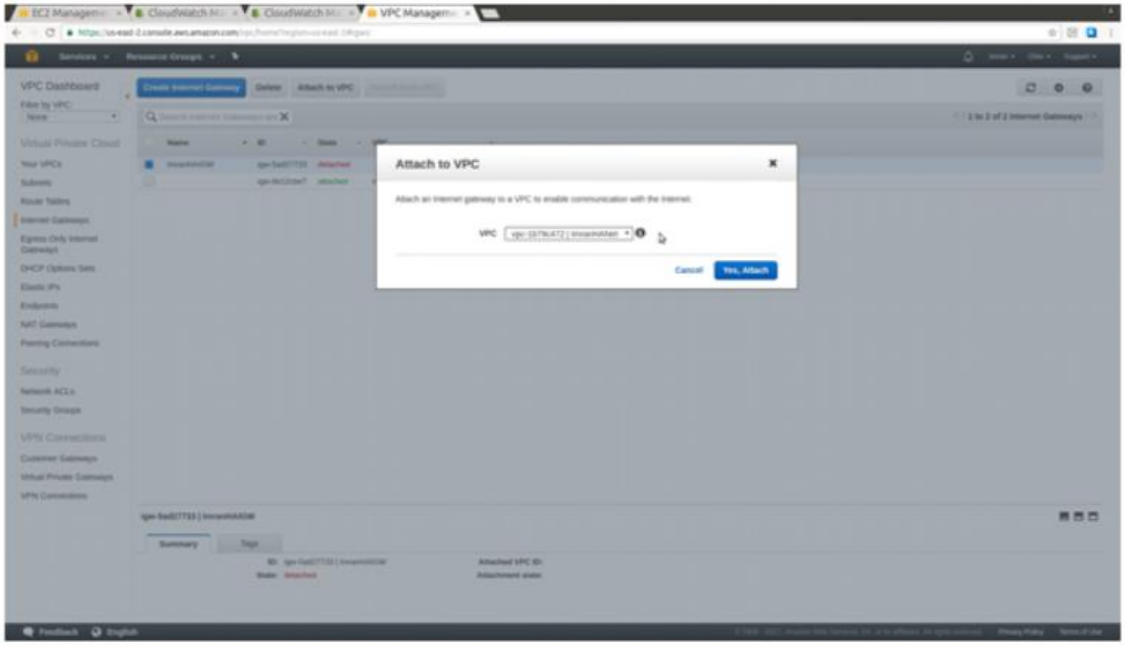
Verify all your subnet settings.

Go to Internet Gateway to Create Internet Gateway to map to Public subnet.



Provide the name for Internet Gateway. Click on Create

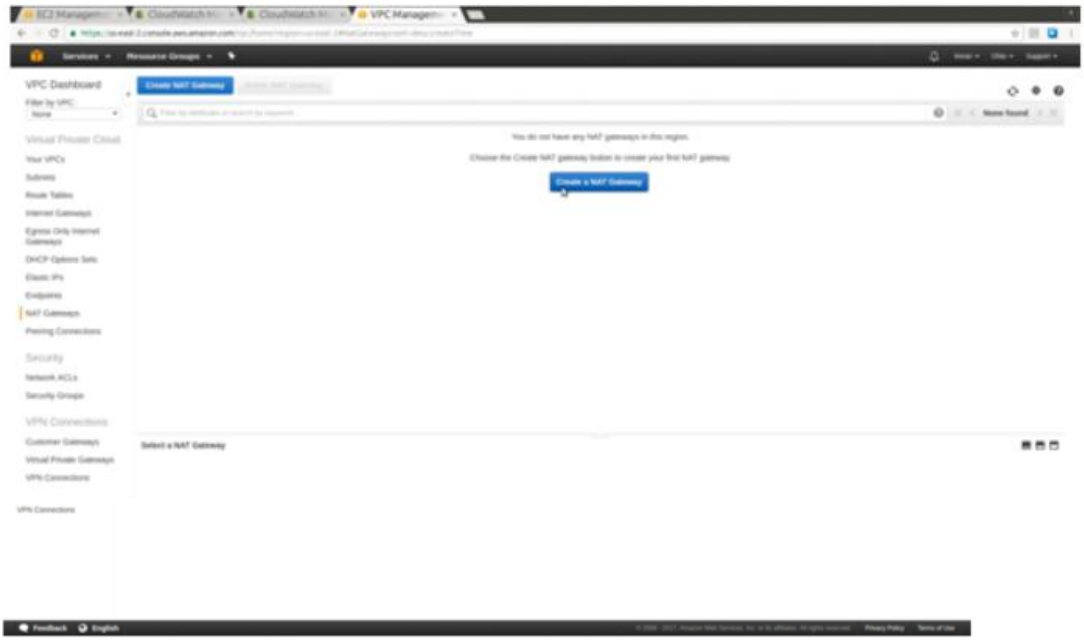
Attach IGW to your VPC: Select Internet Gateway you created, choose the VPC to be attached and click on click on Attach to VPC.



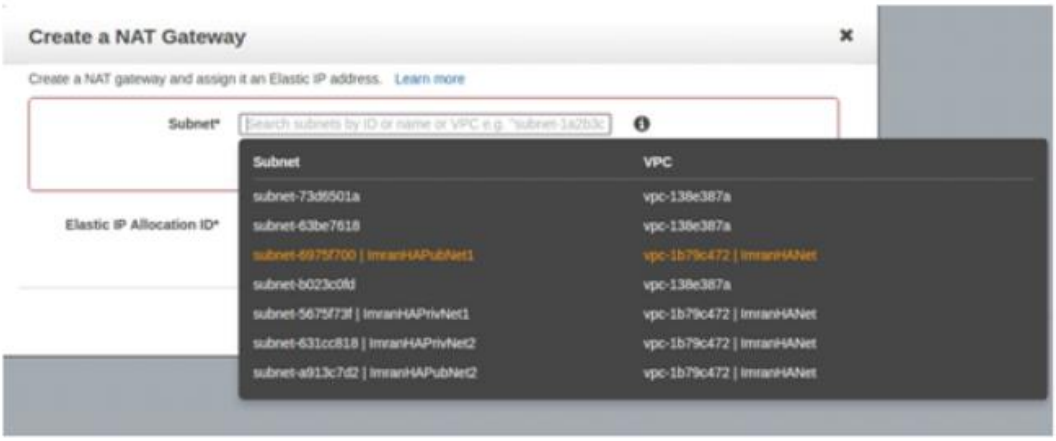
1.Create NAT Gateway to Map to your Private subnets.

2.You also need an Elastic IP (EIP) to assign to your NAT Gateway.

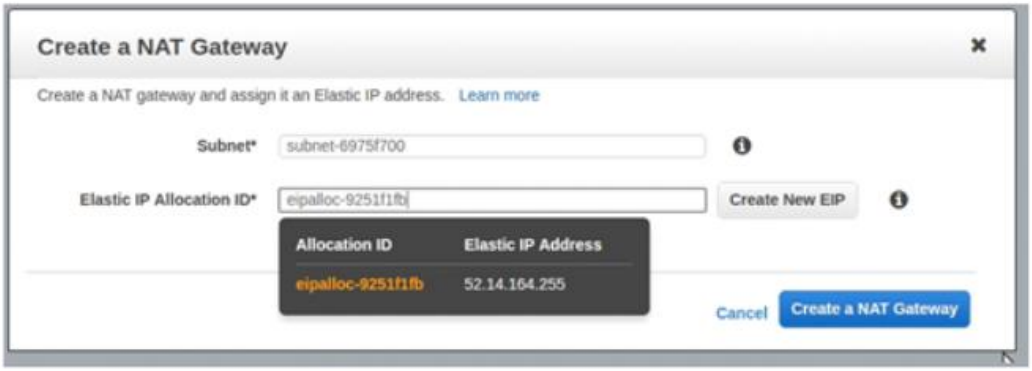
3.Go to NAT Gateway navigation pane and click on create NAT Gateway

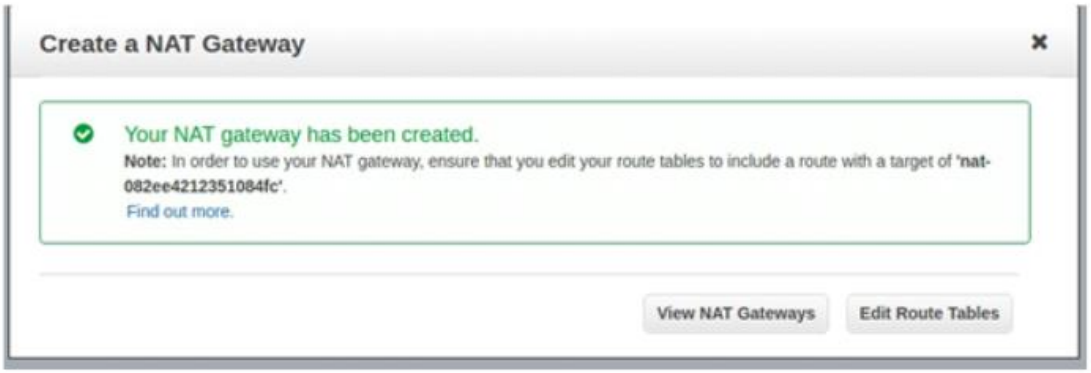


Select any of the PUBLIC subnet that we created earlier, Create New EIP and assign to Gateway.



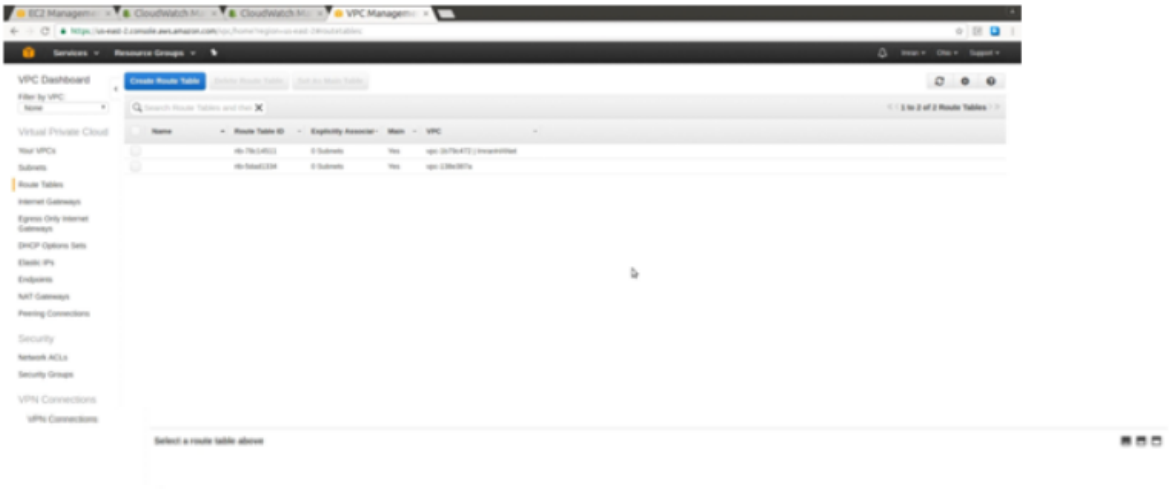
Once done click on create NAT Gateway





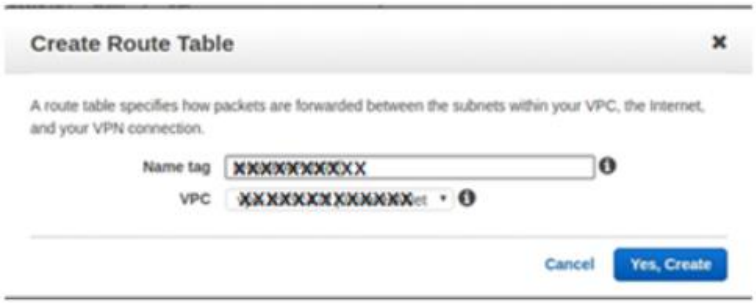
NAT Gateway is successfully created as shown below.

Go to Route Table to create route tables for subnets.

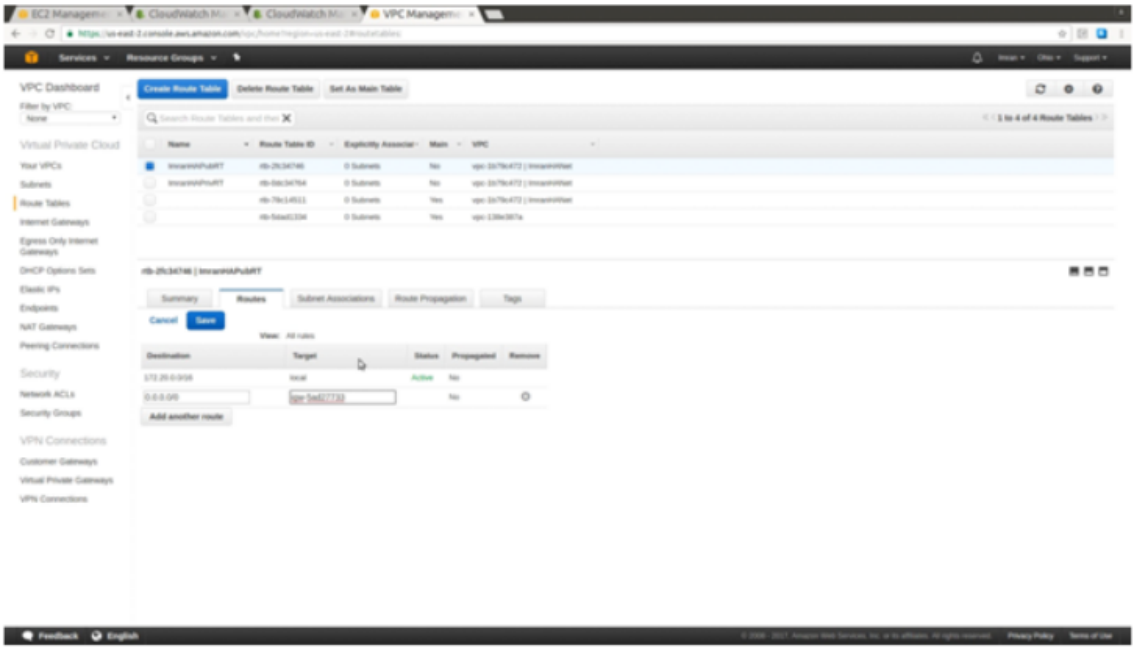


1.Click on create Route Table and Give a name, Click on create.

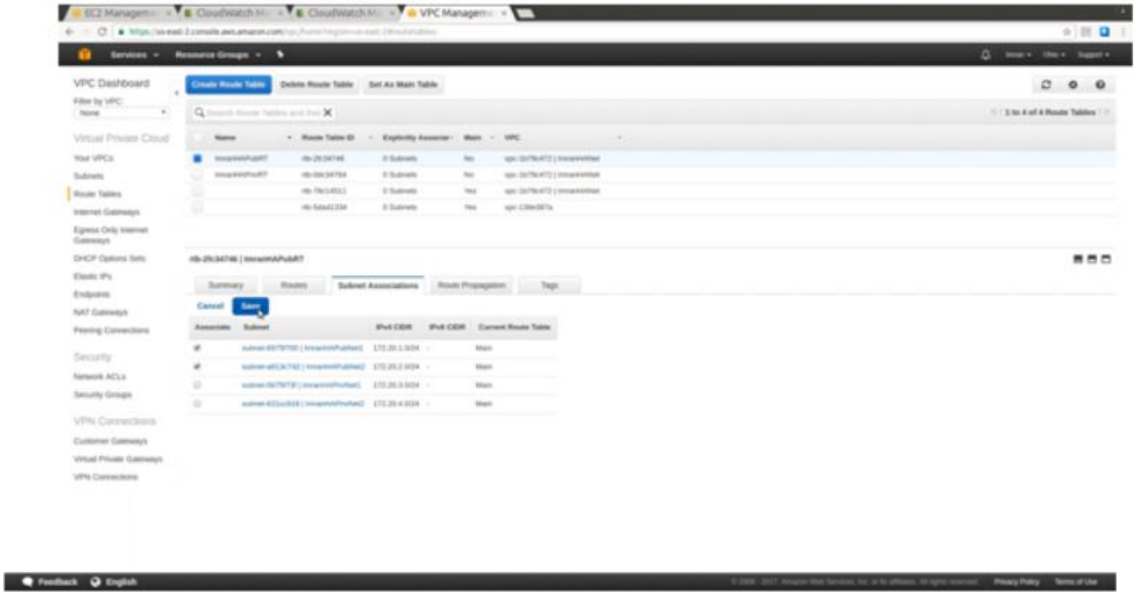
2.We need two route tables, one for Public subnet & one for Private Subnet.



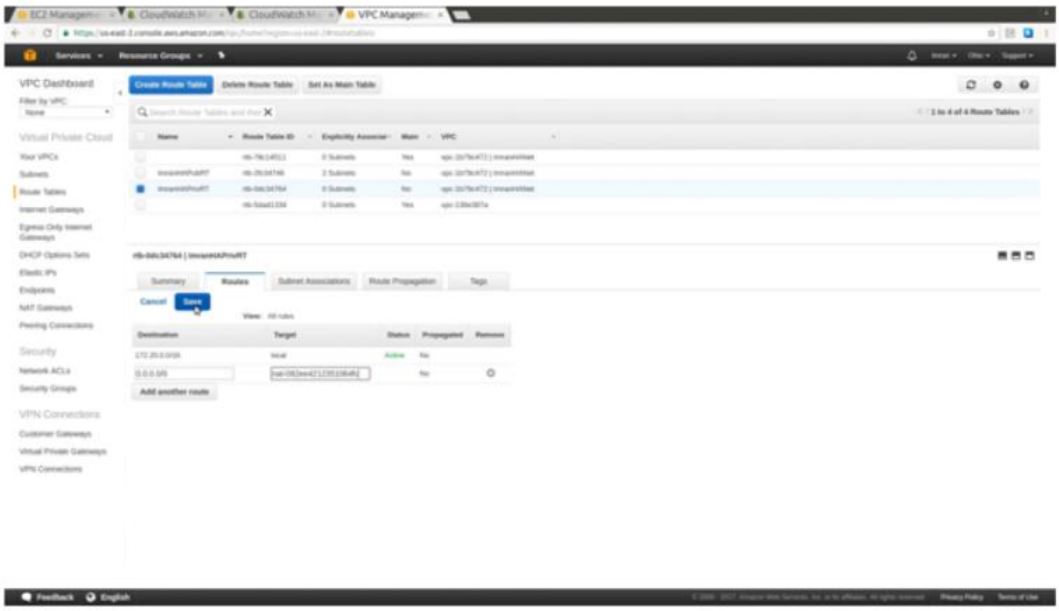
Add Route Table rule on HAPubRT to route to IGW that we created and save.



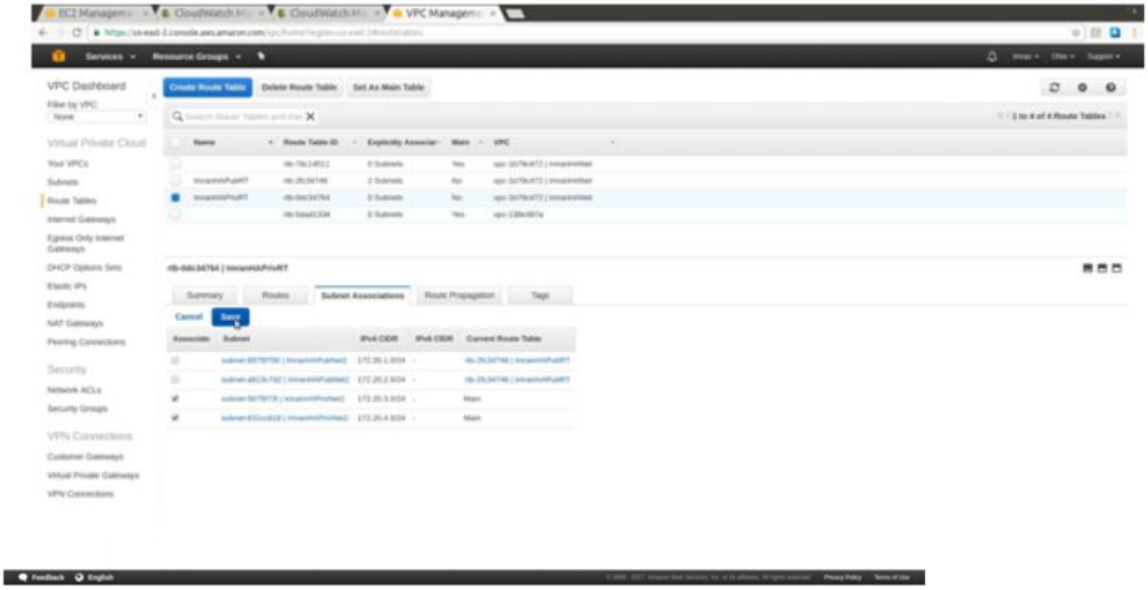
Associate HAPubRT to Public subnets and save.



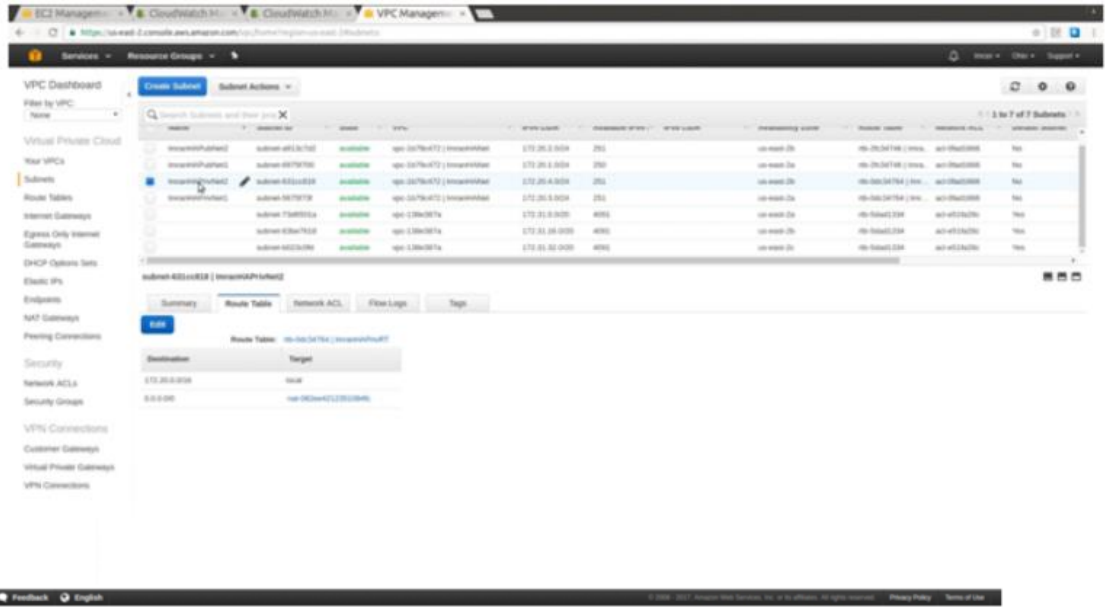
Add Route Table rule on HAPrivRT to route to NAT GW and save.



Associate HAPrivRT to PRIVATE subnets and save.



Go to your Subnets and verify every subnet.



You can attach this VPC to EC2 instances with public subnet and private subnet. If you want to connect to the private subnet instance first you need to connect to the public subnet instance.

Elastic Load Balancer

Elastic Load Balancing distributes incoming application traffic across multiple EC2 instances, in multiple Availability Zones. This increases the fault tolerance of your applications. A load balancer accepts incoming traffic from clients and routes requests to its registered EC2 instances in one or more Availability Zones. The load balancer also monitors the health of its registered instances and ensures that it routes traffic only to healthy instances. When the load balancer detects an unhealthy instance, it stops routing traffic to that instance, and then resumes routing traffic to that instance when it detects that the instance is healthy again.

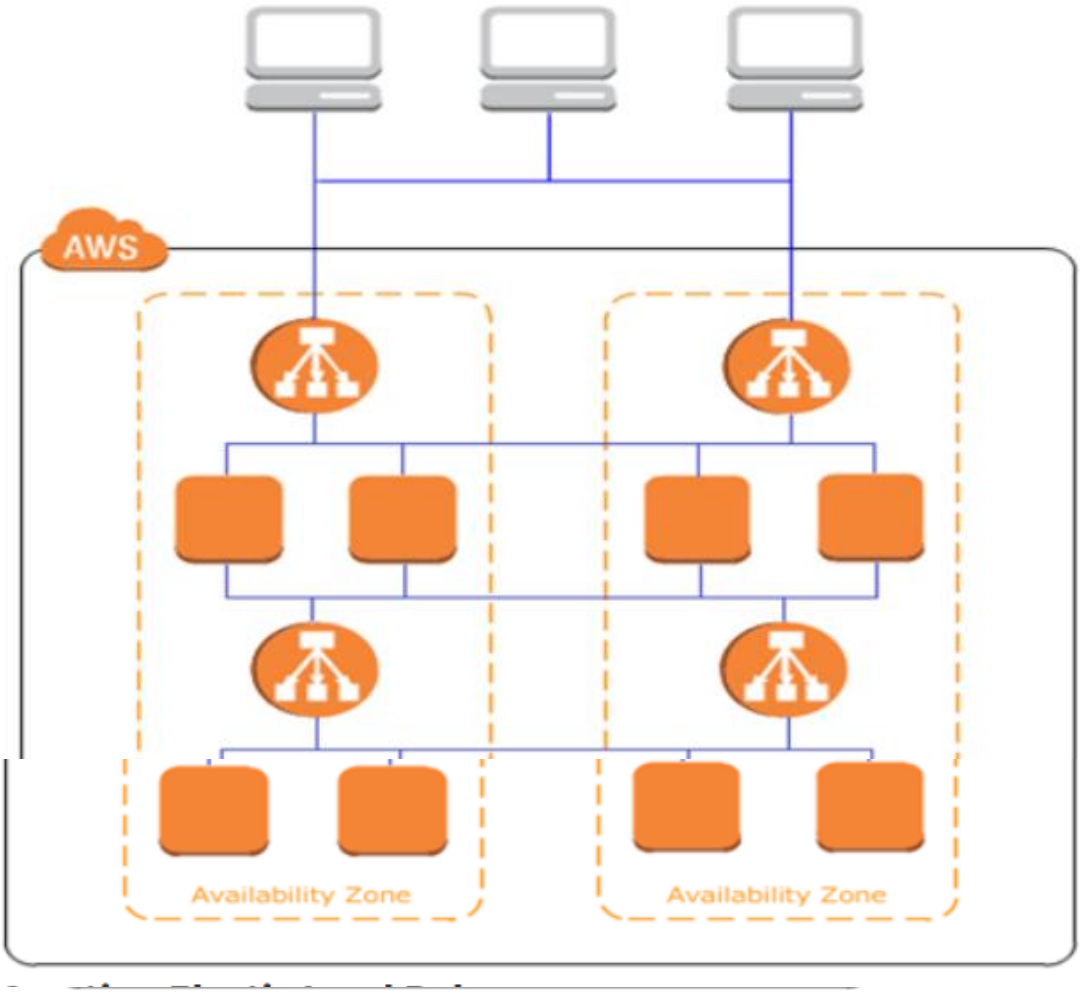
Elastic Load Balancing supports two types of load balancers: Application Load Balancers and Classic Load Balancers. There is a key difference between the way you configure these load balancers. With a Classic Load Balancer, you register instances

with the load balancer. With an Application Load Balancer, you register the instances as targets in a target group, and route traffic to a target group.

When you create a load balancer, you must choose whether to make it an internal load balancer or an Internet-facing load balancer. Note that when you create a Classic Load Balancer in EC2-Classic, it must be an Internet-facing load balancer. The nodes of an Internet-facing load balancer have public IP addresses. The DNS name of an Internet facing load balancer is publicly resolvable to the public IP addresses of the nodes. Therefore, Internet facing load balancers can route requests from clients over the Internet.

The nodes of an internal load balancer have only private IP addresses. The DNS name of an internal load balancer is publicly resolvable to the private IP addresses of the nodes. Therefore, internal load balancers can only route requests from clients with access to the VPC for the load balancer. Note: Both Internet-facing and internal load balancers route requests to your instances using private IP addresses. Therefore, your instances do not need public IP addresses to receive requests from an internal or an Internet-facing load balancer.

If your application has multiple tiers, for example web servers that must be connected to the Internet and database servers that are only connected to the web servers, you can design an architecture that uses both internal and Internet-facing load balancers. Create an Internet-facing load balancer and register the web servers with it. Create an internal load balancer and register the database servers with it. The web servers receive requests from the Internet-facing load balancer and send requests for the database servers to the internal load balancer. The database servers receive requests from the internal load balancer.



Creating Elastic Load Balancer:

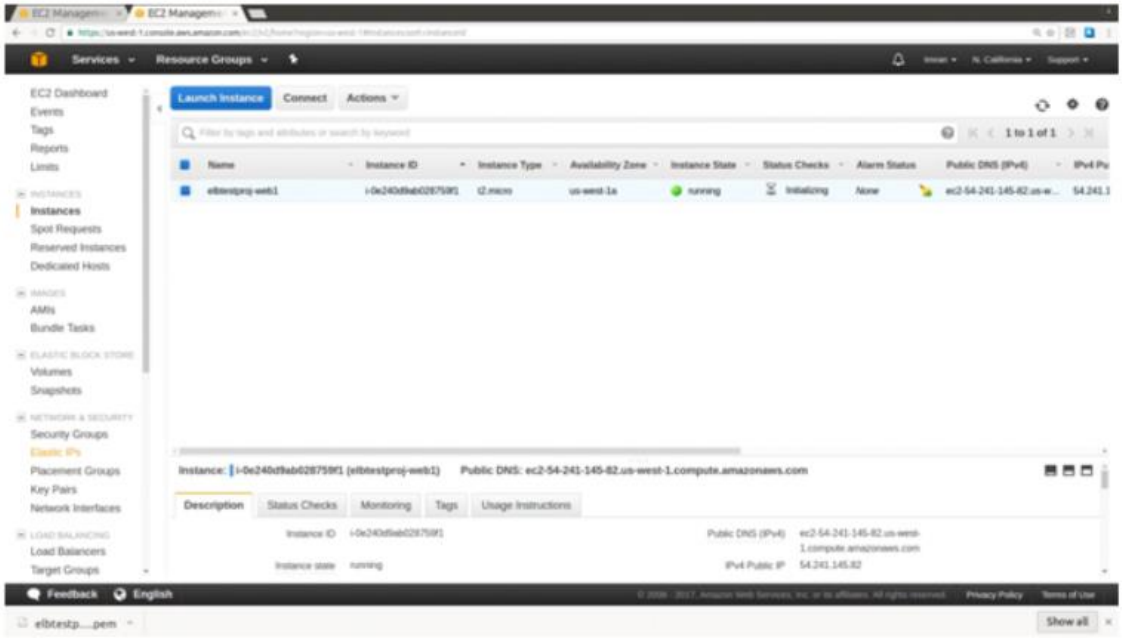
Prerequisites:

1.Choose any two Availability Zones you will use for your EC2 instances. Verify that your virtual private cloud (VPC) has at least one public subnet in each of these Availability Zones.

2.Launch at least one EC2 instance in each Availability Zone.

3.Ensure that the security group for your EC2 instances allows HTTP access on port 80. To test the web server, copy the DNS name of the instance and verify whether browser displays the default page of the web server or not.

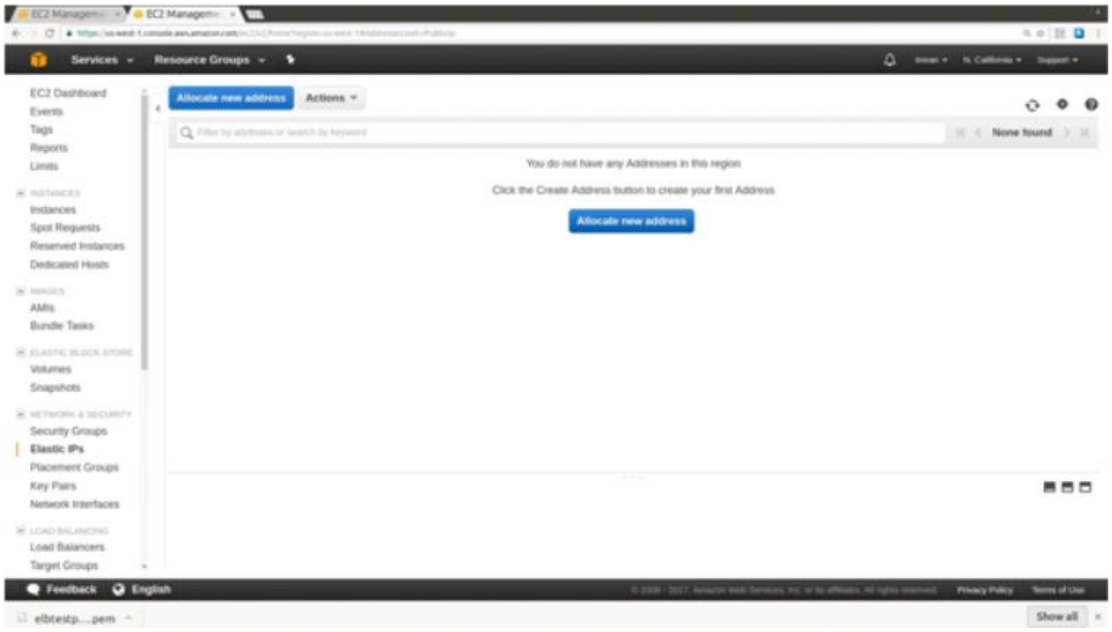
The below screenshot shows the instance created named elbtestproj-web1, it has a public IP address but this IP is dynamic and changes after every reboot of the instance. We need to assign an Elastic IP to this instance which is static and does not change.



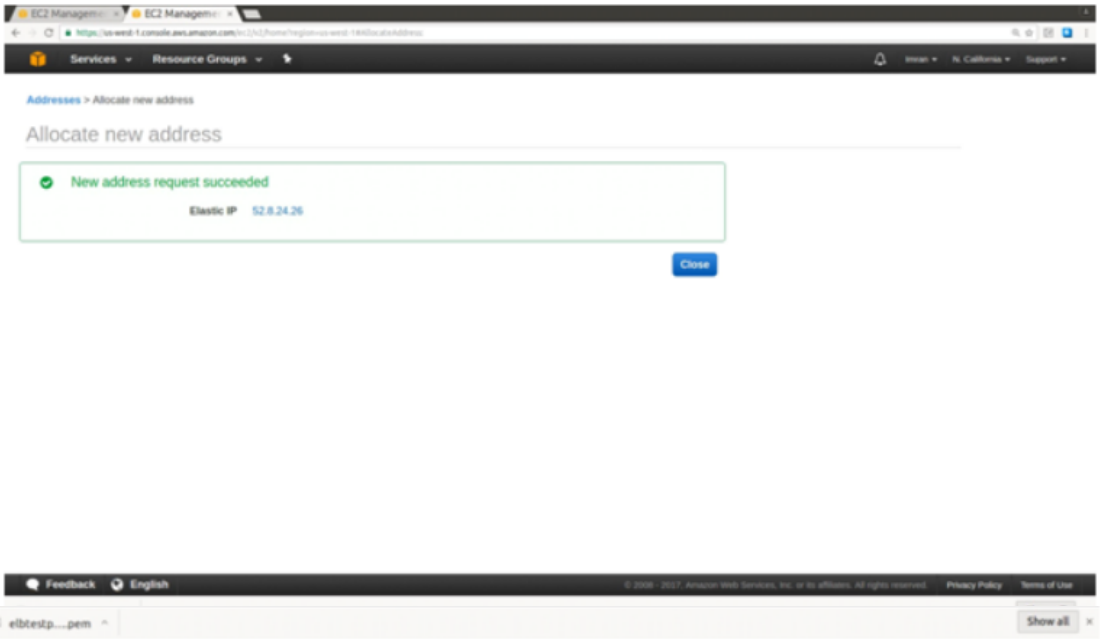
Elastic IP: An Elastic IP address is a static IPv4 address designed for dynamic cloud computing. An Elastic IP address is associated with your AWS account. With an Elastic IP address, you can mask the failure of an instance or software by rapidly remapping the address to another instance in your account.

Assigning Elastic IP:

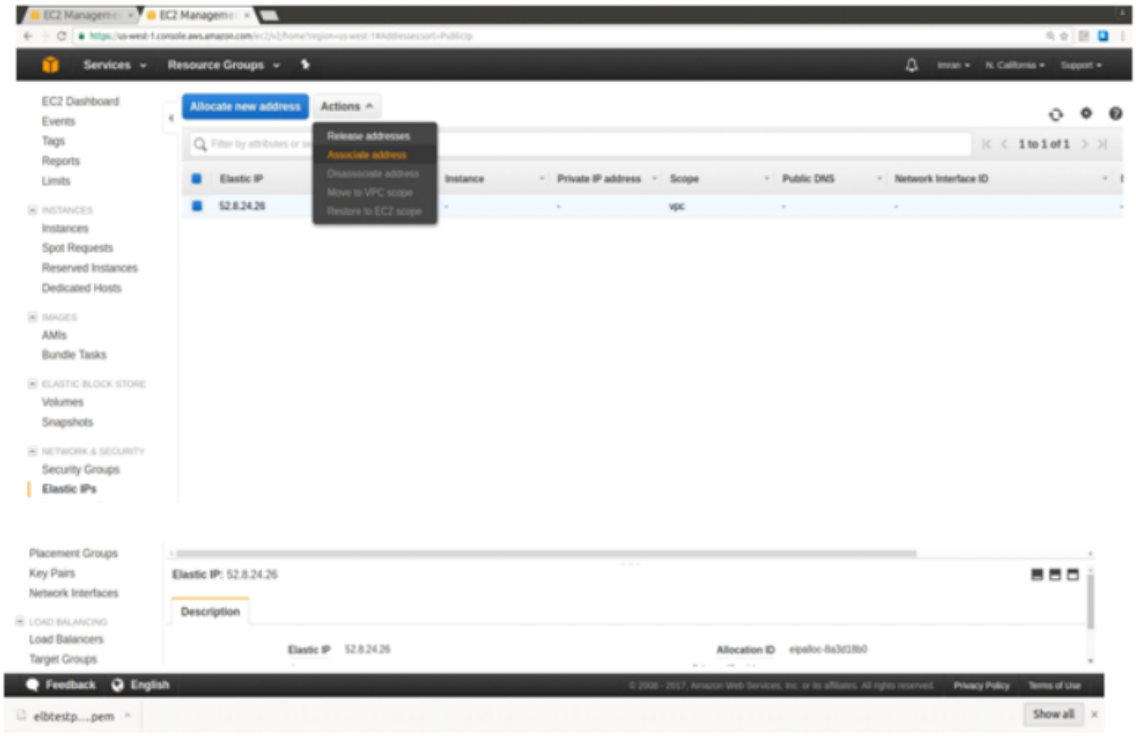
Navigate to the Elastic IPs tab on the EC2 Dashboard and click on allocate new address



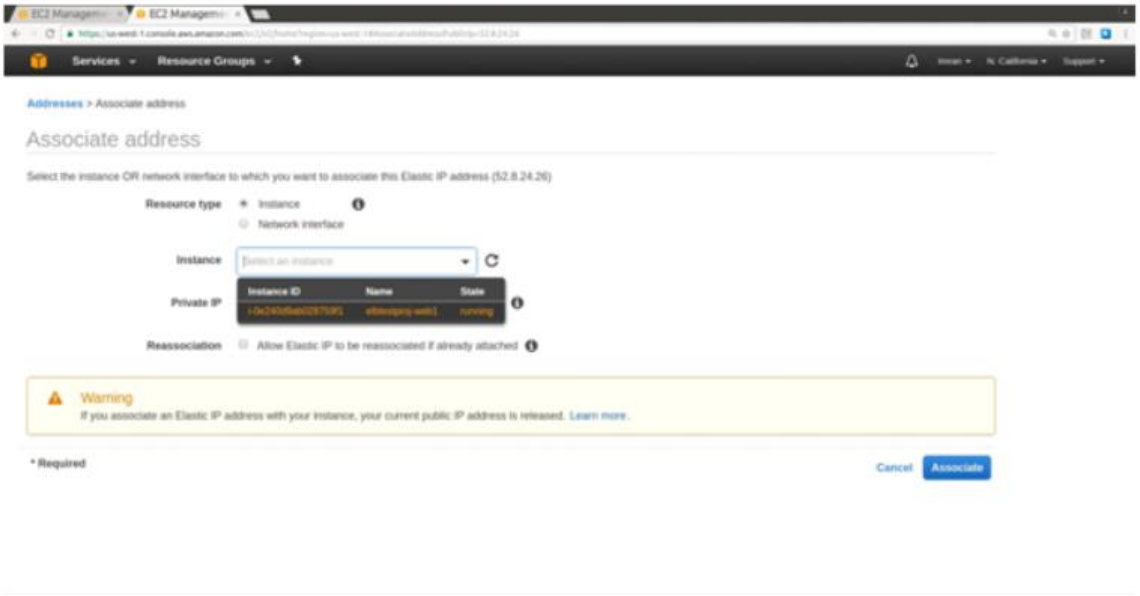
You will get a message 'New address request succeeded'



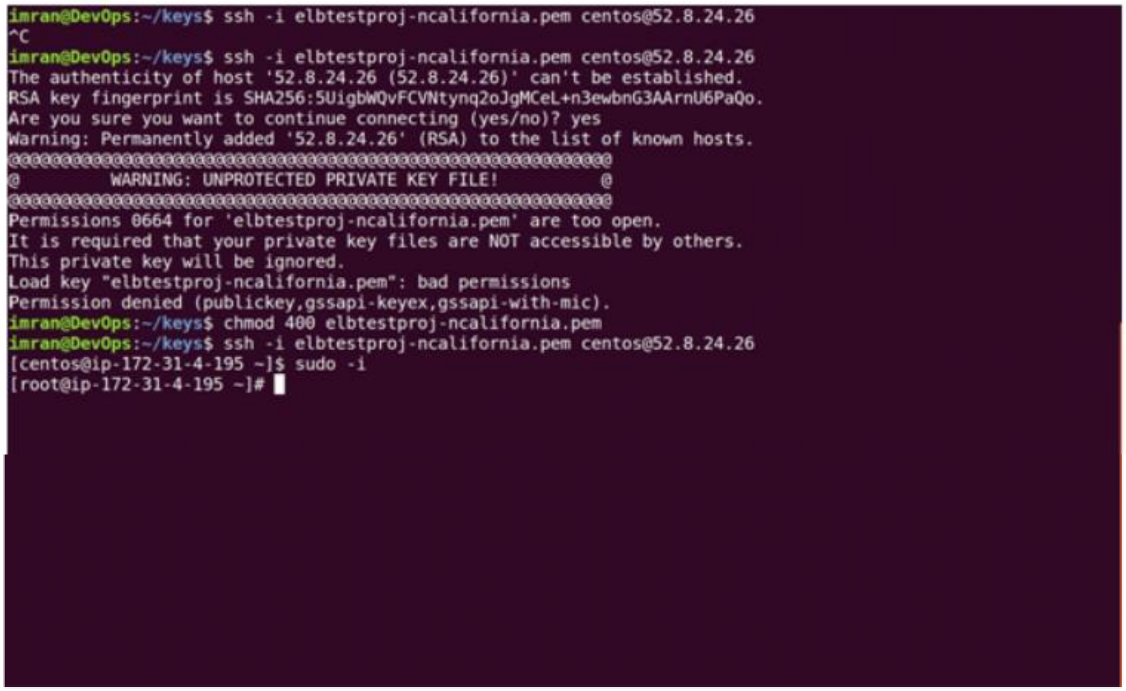
Click on actions and select Associate address



Provide the instance id and click on associate. Then your instance will be allocated with this elastic IP.



Login to the instance by using public IP



Install Apache Service By Using Below Command

yum install httpd

Start Apache Service

servicehttpd start

Enable Apache Service

chkconfighttpd on

Stop & Disable Firewall

serviceiptables stop

chkconfigiptables off

Create A Test Webpage For Apache Using Html.

Test The Webpage

Enter the ec2 inst public IP in browser.

http://52.8.24.26/

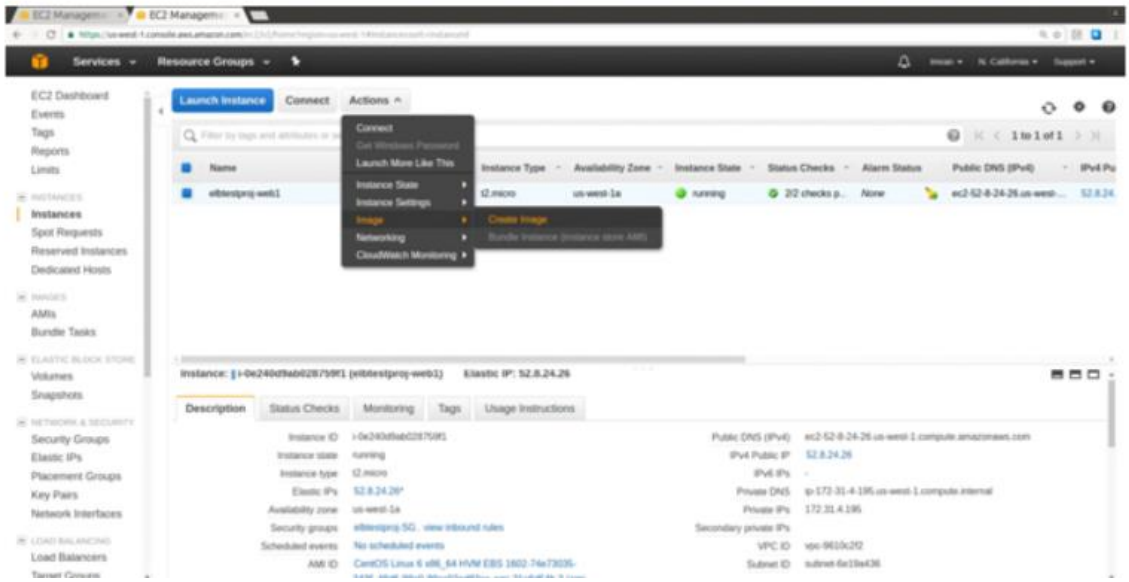
AWS AMI

An Amazon Machine Image (AMI) is a special type of virtual appliance that is used to create a virtual machine within the Amazon Elastic Compute Cloud ("EC2"). It serves as the basic unit of deployment for services delivered using EC2.

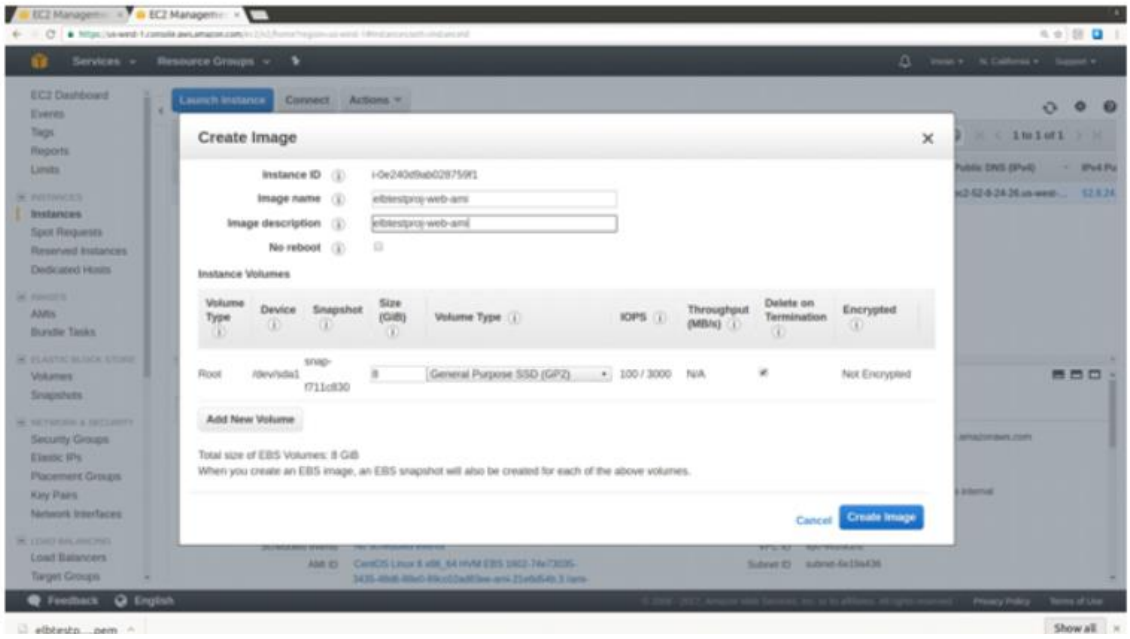
AMI Creation

Create AMI of the instance which we will use to spin web02 instance. Web02 instance is exactly similar to web01, so instead to creating new instance from scratch and setting up apache, we can create an AMI (image) of web01 instance and can spin as many as web instances we want.

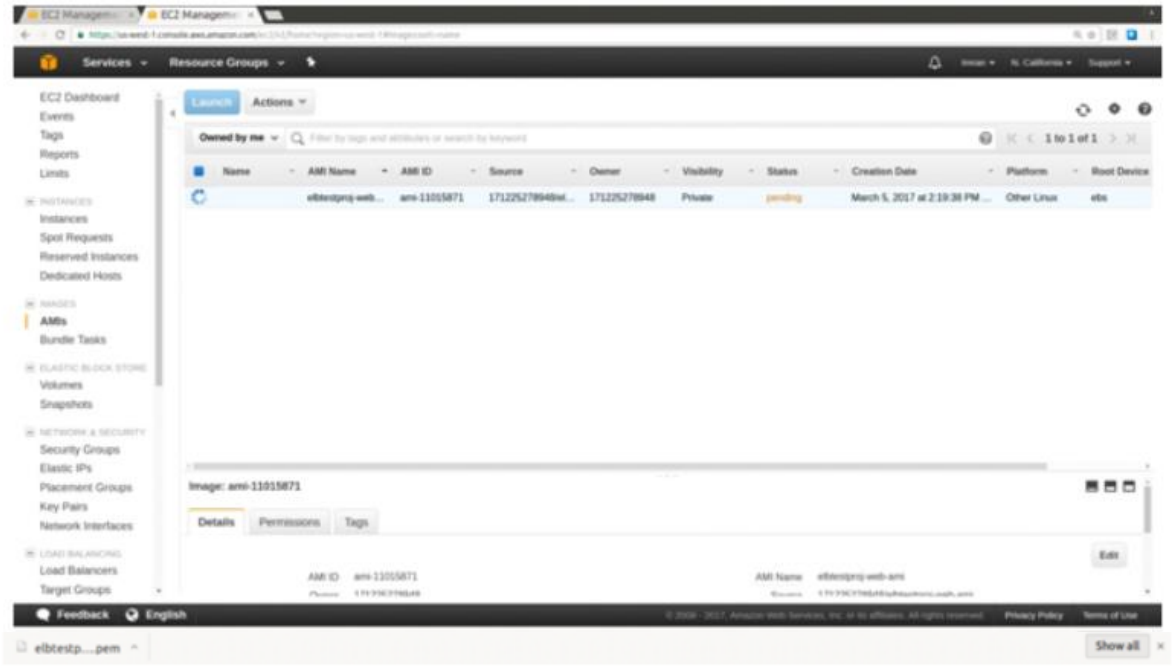
Select the instance in which we have to create an image. Click on actions, select image and click on create image as shown in the below screenshot:



You will get a create image dialog box as shown below. Give proper name and description for image and click on create image.



It takes few minutes to create an image which you can see on AMIs navigation pane



Create Web02 Instance From Elbtestproj-Web-Ami.

Click on Launch instance --> My AMI --> Select your AMI --> Follow the wizard and create the instance similar to web01.

Tag Name: elbtestproj-web02

Security: Select existing security group -->elbtestproj-SG

Select an existing key pair -->

Assign Elastic IP

Test the webpage

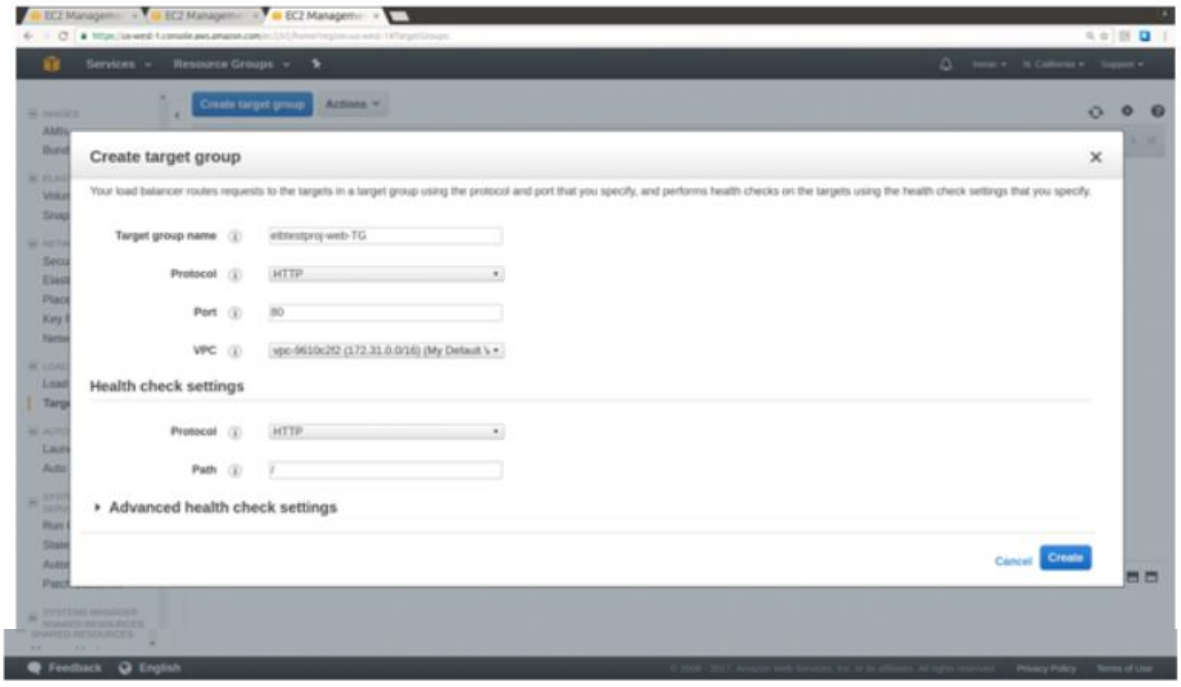
Enter the ec2 instance public IP in browser.

http://54.215.191.250//

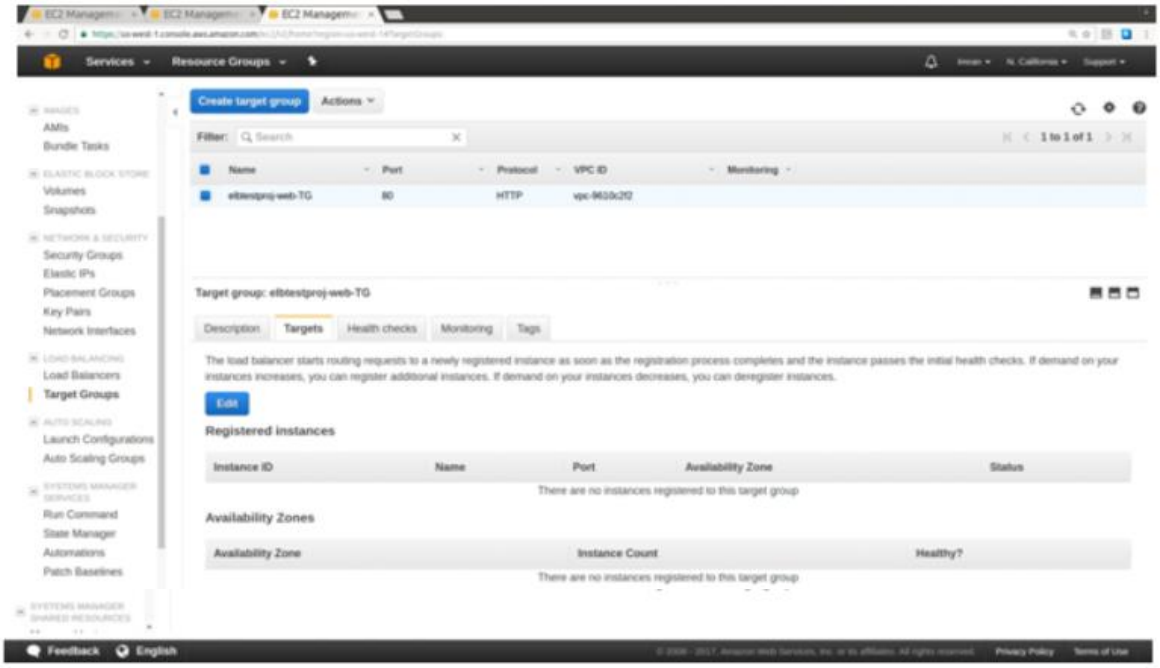
Load Balancer Setup

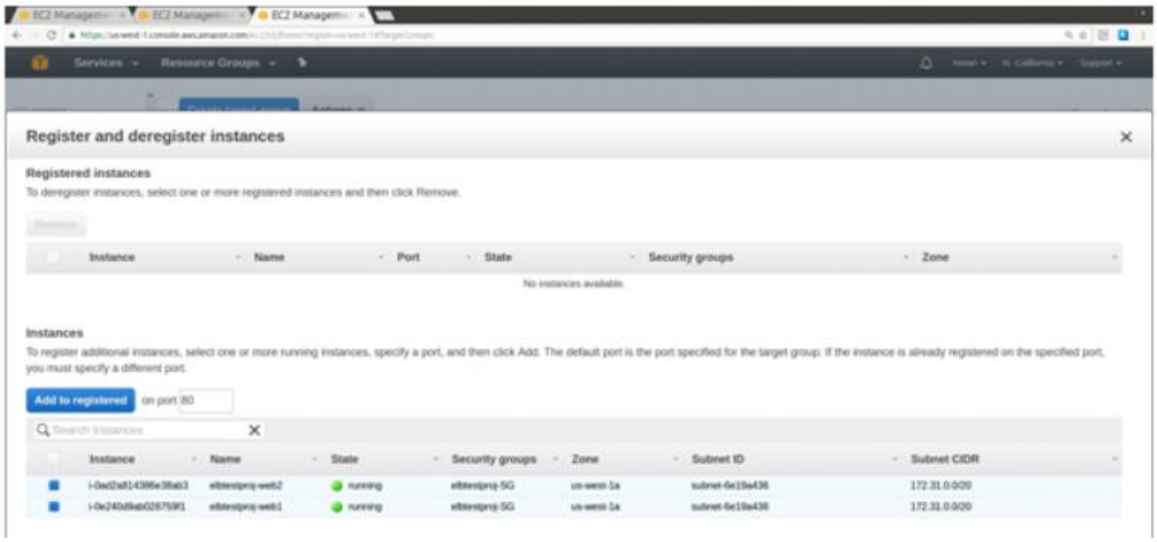
1.Click on Target Group on the left side of the navigation pane, click on create target group.

2.Provide name for target group and click on create as shown in below screenshot:

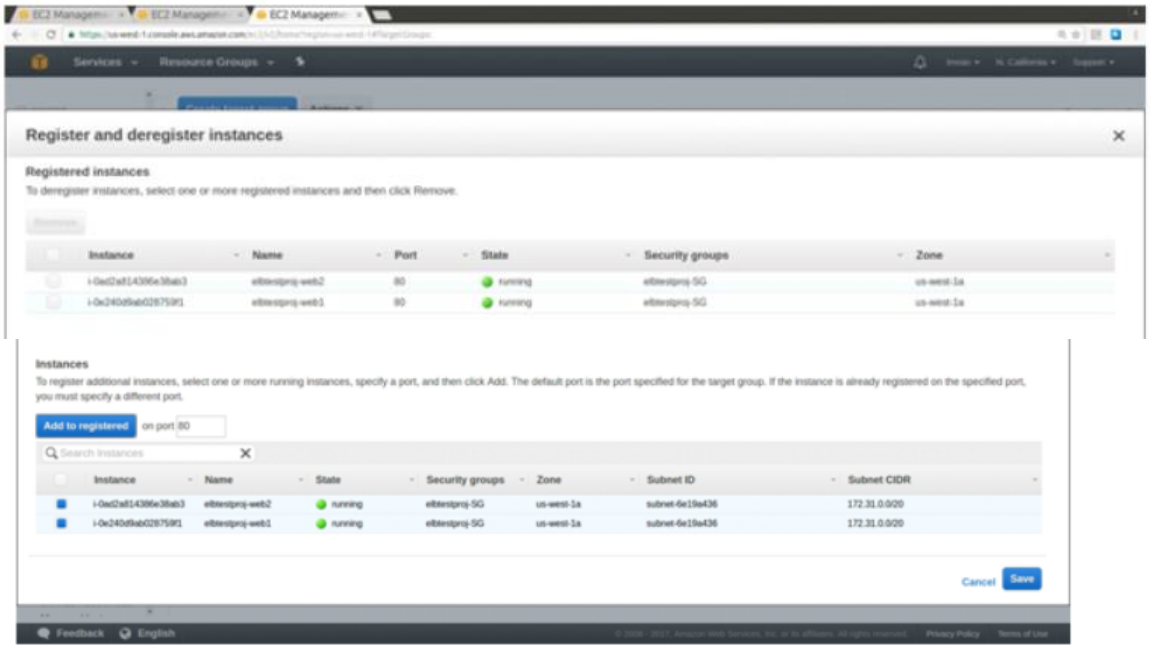


1.Once the Target group is created click on the targets tab, Edit and Select web01 & web02 instances. Add to registered and click on Save.


Services
Resource Groups
Create target group
Actions
Name
Port
Protocol
VPC ID
Monitoring
elbtestproj-web-TG
80
HTTP
Target group: elbtestproj-web-TG
Description
Targets
Health checks
Monitoring
Tags
Edit
Registered instances
Instance ID
Name
Port
Availability Zone
Status
There are no instances registered to this target group
Availability Zones
Availability Zone
Instance Count
Healthy?
There are no instances registered to this target group
Feedback
English

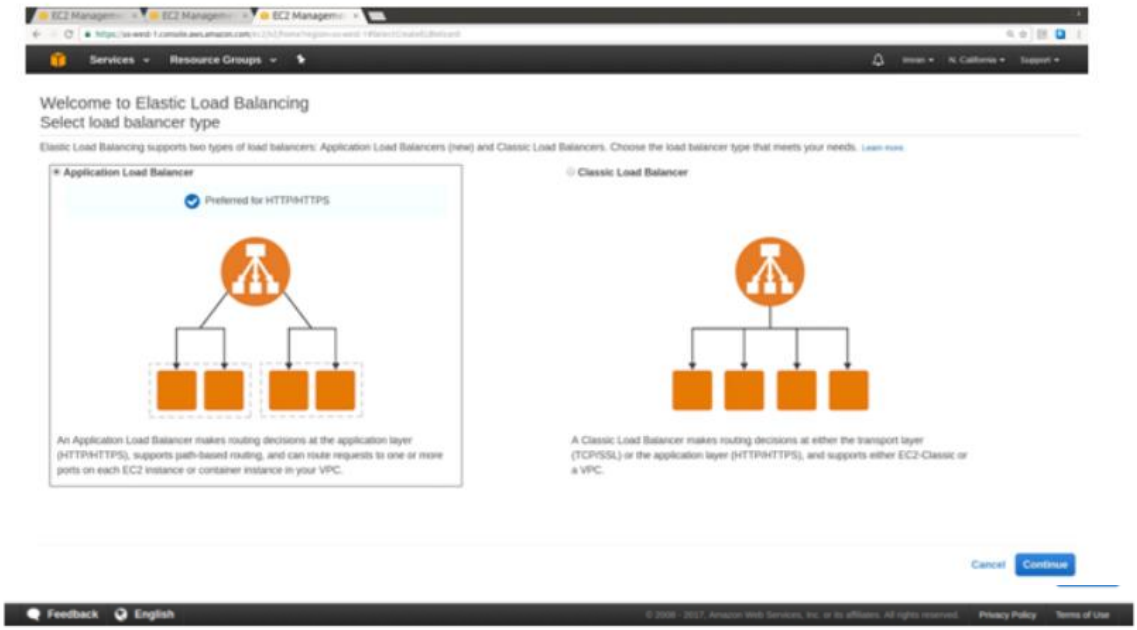

Register and deregister instances
Registered instances
To deregister instances, select one or more registered instances and then click Remove.
Instance
Name
Port
State
Security groups
Zone
No instances available.
Instances
To register additional instances, select one or more running instances, specify a port, and then click Add. The default port is the port specified for the target group. If the instance is already registered on the specified port, you must specify a different port.
Add to registered
on port 80
Search instances
Instance
Name
State
Security groups
Zone
Subnet ID
Subnet CIDR
running
elbtestproj-SG
172.31.0.0/20

Then the instances will be added to the Target Group.

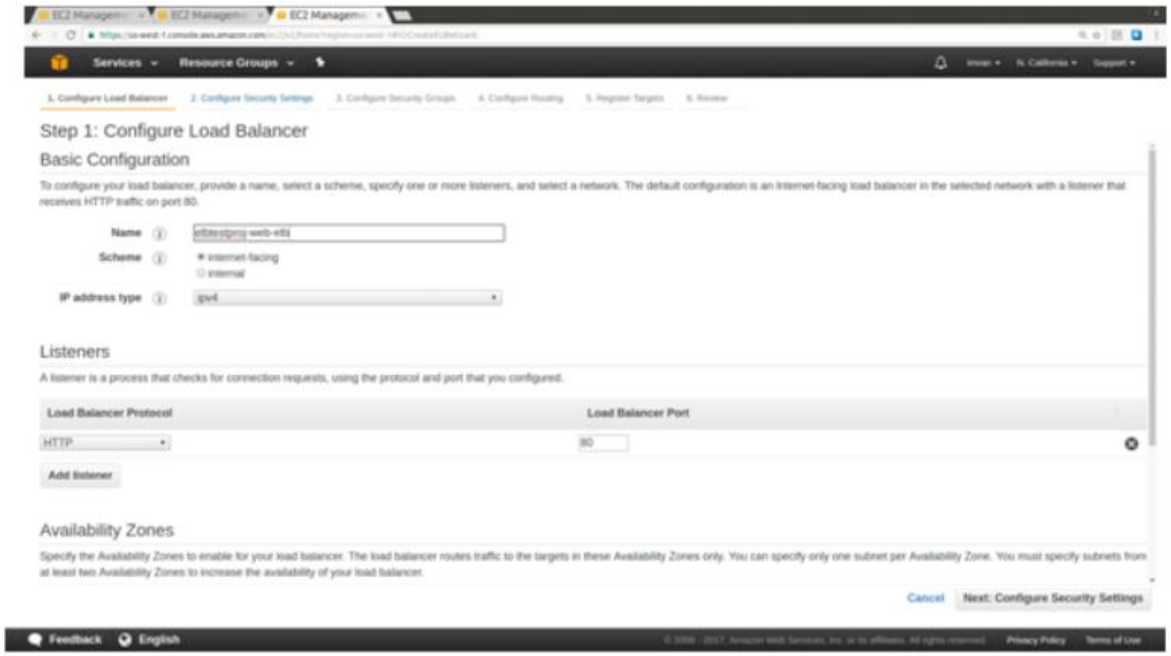


Creating Load Balancer

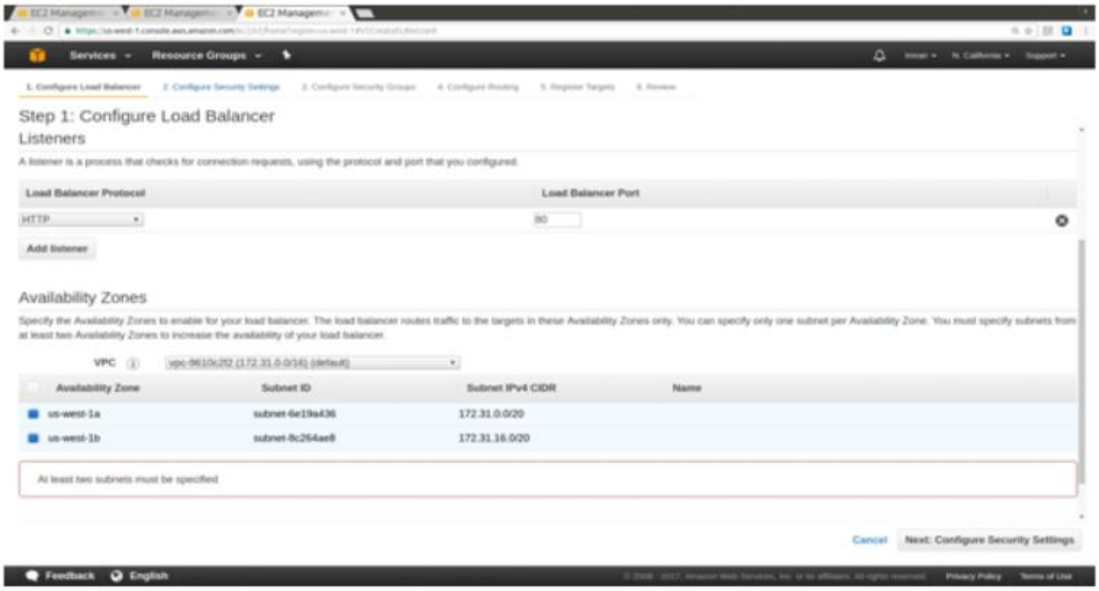
Click on Load Balancers in Left Pane and select Create Load Balancer, you will get two types of load balancers, Application load balancer and Classic load balancer. Select Application Load Balancer and click on Continue.



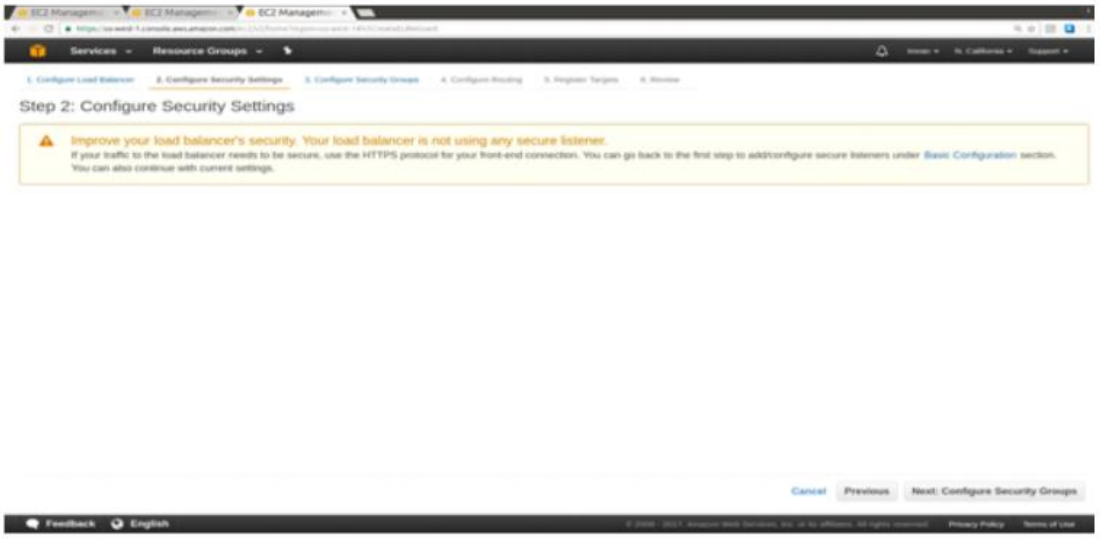
Configure the details of the load balancer. Provide the name for load balancer, the name of your Application Load Balancer must be unique within your set of Application Load Balancers. For one region, it can have a maximum of 32 characters and contain only alphanumeric characters and hyphens, and must not begin or end with a hyphen. Click on next configure security group.



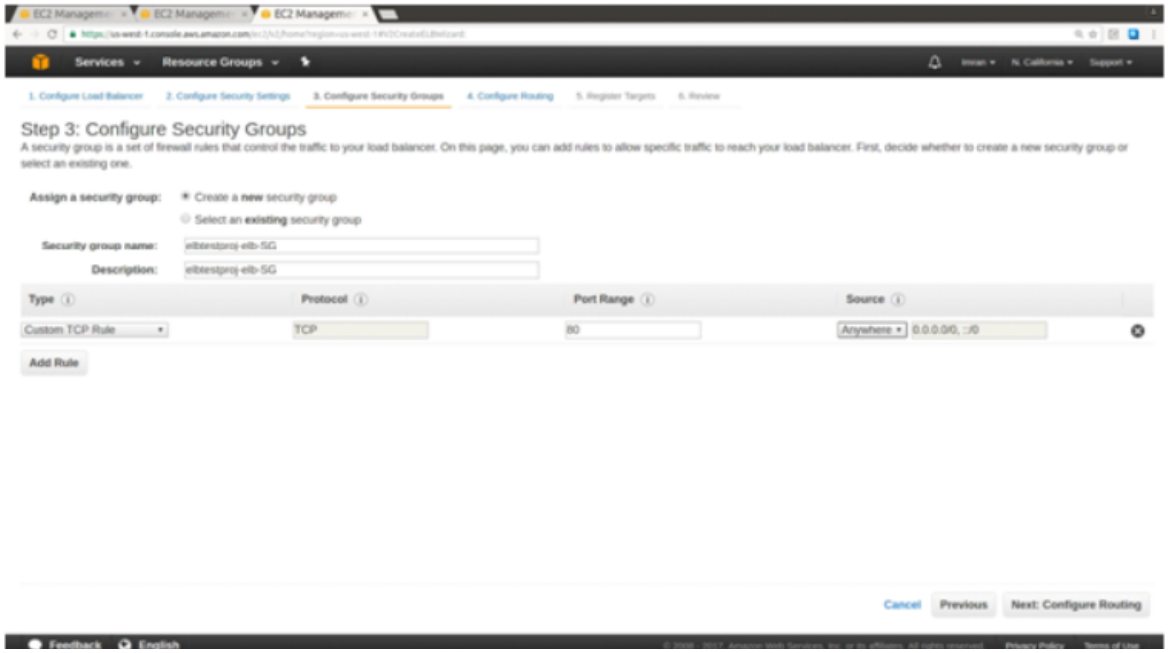
For Availability Zones, select the VPC that you used for your EC2 instances. For each of the two Availability Zones that contain your EC2 instances, select the Availability Zone and then select the public subnet for that Availability Zone. Add atleast two Availability Zones to increase the availability of the load balancer. Click on next



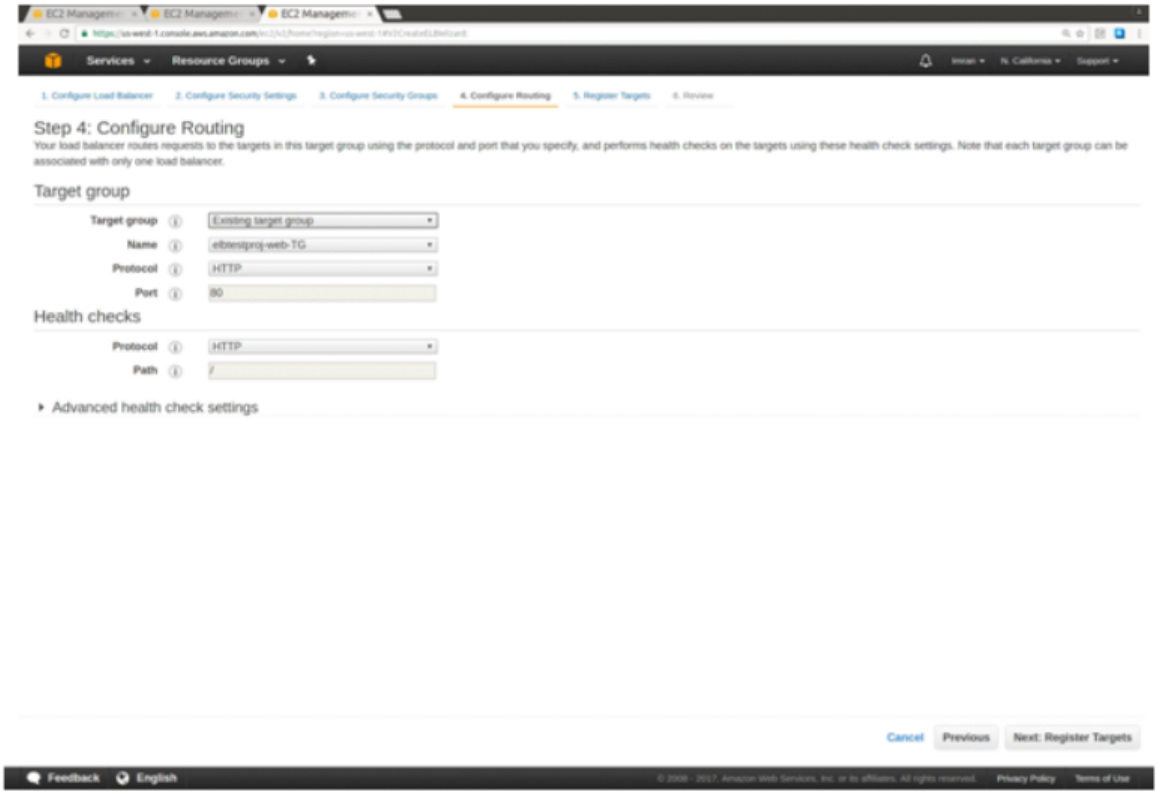
Configure the details of security group to improve the load balancer's security.



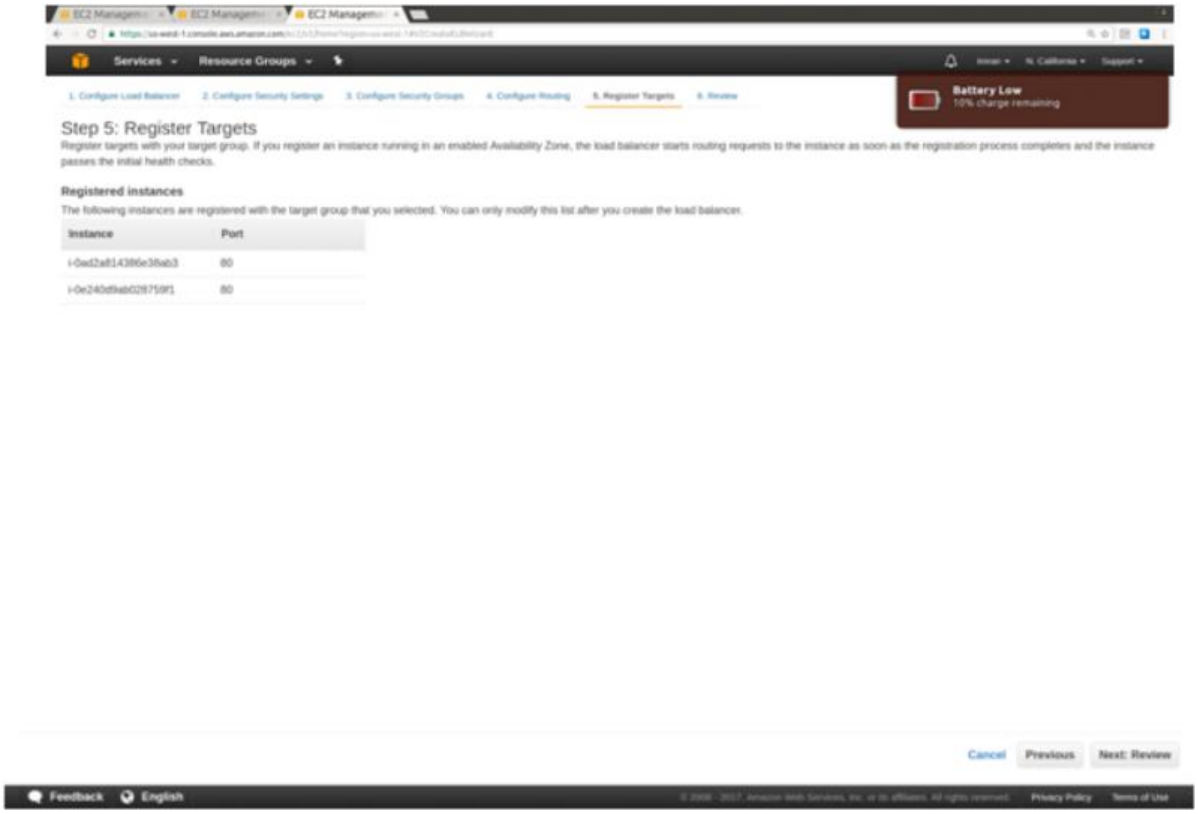
Create a new security group for load balancer which accepts HTTP traffic on port 80.



Configure the target group. The default rule for your listener routes requests to the registered targets in this target group. The load balancer checks the health of targets in this target group using the health check settings defined for the target group.



Register targets with the Target Groups



Validate the Load Balancer:

Enter Load Balancers DNS name in the browser to test the connection

http://elbtestproj-web-elb-289295805.us-west-1.elb.amazonaws.com/

If everything is good you will see webpages from web01 and web02 instances.

AWS Auto Scaling

Auto Scaling is a web service designed to launch or terminate Amazon EC2 instances automatically based on user-defined policies, schedules, and health checks. You can use Auto Scaling to help ensure that you are running your desired number of Amazon EC2 instances. 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. Auto Scaling is well suited both to applications that have stable demand patterns or that experience hourly, daily, or weekly variability in usage.

Listing local boxes.

Benefits Of Auto Scaling:

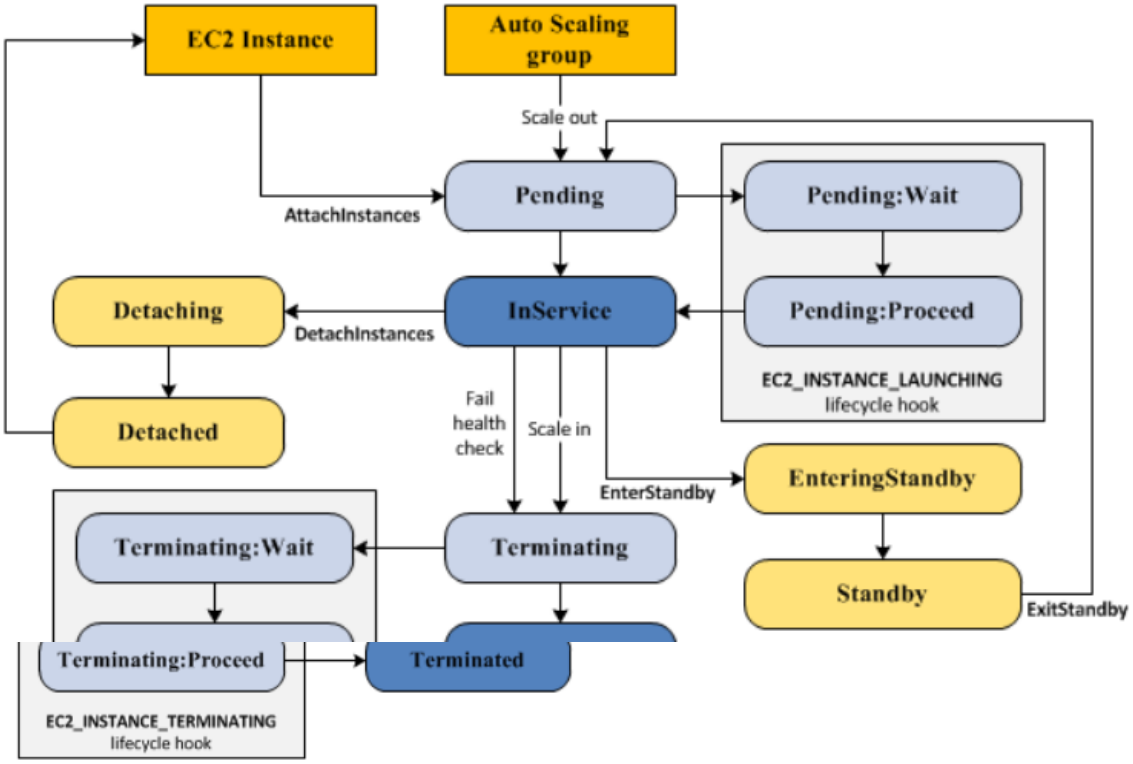
Adding Auto Scaling to your application architecture is one way to maximize the benefits of the AWS cloud. When you use Auto Scaling, your applications gain the following benefits:

1.Better fault tolerance. Auto Scaling can detect when an instance is unhealthy, terminate it, and launch an instance to replace it. You can also configure Auto Scaling to use multiple Availability Zones. If one Availability Zone becomes unavailable, Auto Scaling can launch instances in another one to compensate.
2.Better availability. Auto Scaling can help you ensure that your application always has the right amount of capacity to handle the current traffic demands.
3.Better cost management. Auto Scaling can dynamically increase and decrease capacity as needed. Because you pay for the EC2 instances you use, you save money by launching instances when they are actually needed and terminating them when they aren't needed.

Auto Scaling Lifecycle:

The EC2 instances in an Auto Scaling group have a path, or lifecycle, that differs from that of other EC2 instances. The lifecycle starts when the Auto Scaling group launches an instance and puts it into service. The lifecycle ends when you terminate the instance, or the Auto Scaling group takes the instance out of service and terminates it.

Note: You are billed for instances as soon as they are launched, including the time that they are not yet in service. The following illustration shows the transitions between instance states in the Auto Scaling lifecycle.



Scale Out

The following scale out events direct the Auto Scaling group to launch EC2 instances and attach them to the group:

1.You manually increase the size of the group.

2.You create a scaling policy to automatically increase the size of the group based on a specified increase in demand.

3.You set up scaling by schedule to increase the size of the group at a specific time.

When a scale out event occurs, the Auto Scaling group launches the required number of EC2 instances, using its assigned launch configuration. These instances start in the Pending state. If you add a lifecycle hook to your Auto Scaling group, you can perform a custom action here. When each instance is fully configured and passes the Amazon EC2 health checks, it is attached to the Auto Scaling group and it enters the In-service

state. The instance is counted against the desired capacity of the Auto Scaling group.

Instances In Service

Instances remain in the In-service state until one of the following occurs:

1.A scale in event occurs, and Auto Scaling chooses to terminate this instance in order to reduce the size of the Auto Scaling group.
2.You put the instance into a Standby state.
3.You detach the instance from the Auto Scaling group.
4.he instance fails a required number of health checks, so it is removed from the Auto Scaling group, terminated, and replaced.

Scale In

It is important that you create a scale in event for each scale out event that you create. This helps ensure that the resources assigned to your application match the demand for those resources as closely as possible.
The following scale in events direct the Auto Scaling group to detach EC2 instances from the group and terminate them:

1.You manually decrease the size of the group. 2.You create a scaling policy to automatically decrease the size of the group based on a specified decrease in demand. 3.You set up scaling by schedule to decrease the size of the group at a specific time.

When a scale in event occurs, the Auto Scaling group detaches one or more instances. The Auto Scaling group uses its termination policy to determine which instances to terminate. Instances that are in the process of detaching from the Auto Scaling group and shutting down enter the Terminating state, and can't be put back into service. If you add a lifecycle hook to your Auto Scaling group, you can perform a custom action here. Finally, the instances are completely terminated and enter the Terminated state.

Attach An Instance

You can attach a running EC2 instance that meets certain criteria to your Auto Scaling group. After the instance is attached, it is managed as part of the Auto Scaling group.

Detach An Instance

You can detach an instance from your Auto Scaling group. After the instance is detached, you can manage it separately from the Auto Scaling group or attach it to a different Auto Scaling group.

Auto Scaling Limits

The following table lists the default limits related to your Auto Scaling resources.

Resource	Default Limit
Launch configurations per region	100
Auto Scaling groups per region	20
Scaling policies per Auto Scaling group	50
Scheduled actions per Auto Scaling group	125
Lifecycle hooks per Auto Scaling group	50
SNS topics per Auto Scaling group	10
Classic Load Balancers per Auto Scaling group	50*
Target groups per Auto Scaling group	50*
Step adjustments per scaling policy	20

* Note that you can attach or detach at most 10 at a time.

Setup for Auto Scaling: Here is process for setting up the basic infrastructure for Auto Scaling.

For practice of auto scaling we need EC2 instances with following details.

1.Create an EC2 instance of CentOS6 with instance type t2.micro,Security Group – SSH =>MyIP, HTTP => Anywhere.

2.Download the Private Keypair for Access.

3.Install HTTPD service and start it. Make sure that it should serve some html website.

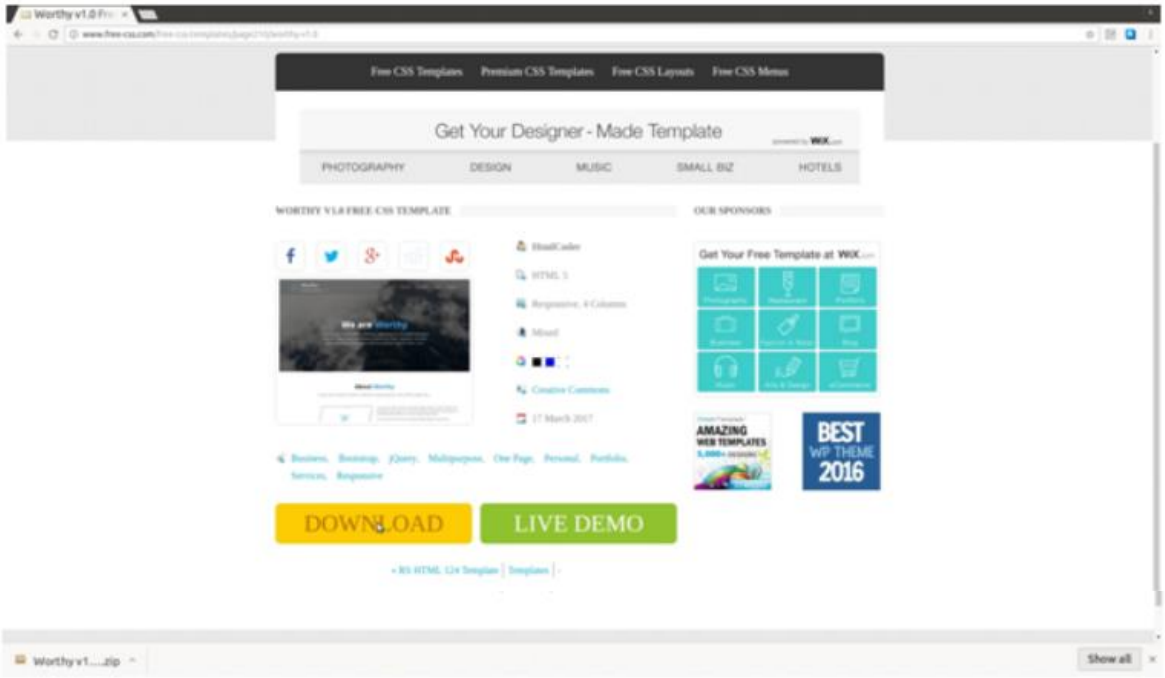
4.You can get a sample website from below link

http://www.free-css.com/free-css-templates

For example let us take sample website template named Worthy. Follow below steps to setup worthy website on our ec2 instance.

1. Setup Website On EC2:

- Download worthy website to your local computer.

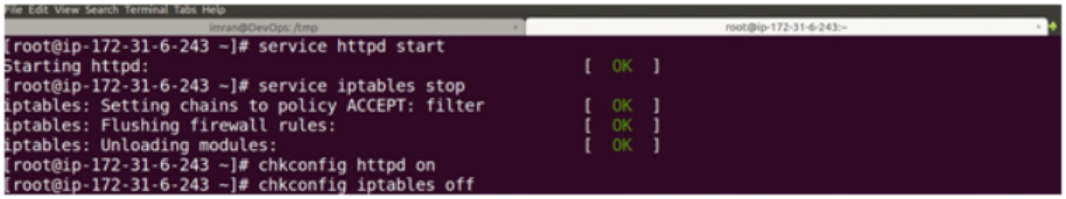


- Copy the website file from local computer to the ec2 instance using scp command.

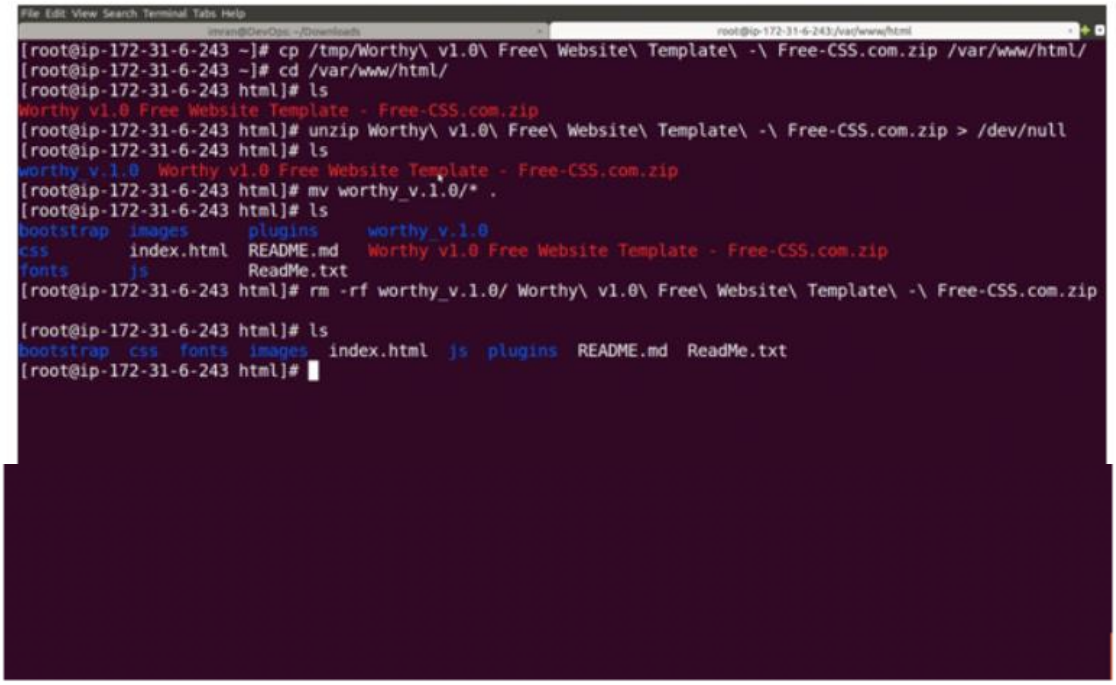
```
imran@DevOps:~/keys$ scp -i worthy-ncalifornia.pem /home/imran/Downloads/Worthy\ v1.0\ Free\ Website\ Templa
te\ -\ Free-CSS.com.zip centos@54.193.0.30:/tmp
Worthy v1.0 Free Website Template - Free-CSS.com.zip                    100% 1381KB 276.3KB/s   00:05
imran@DevOps:~/keys$ ssh -i worthy-ncalifornia.pem centos@54.193.0.30
[centos@ip-172-31-6-243 ~]$ sudo -i
```

- Install HTTPD and setup website..

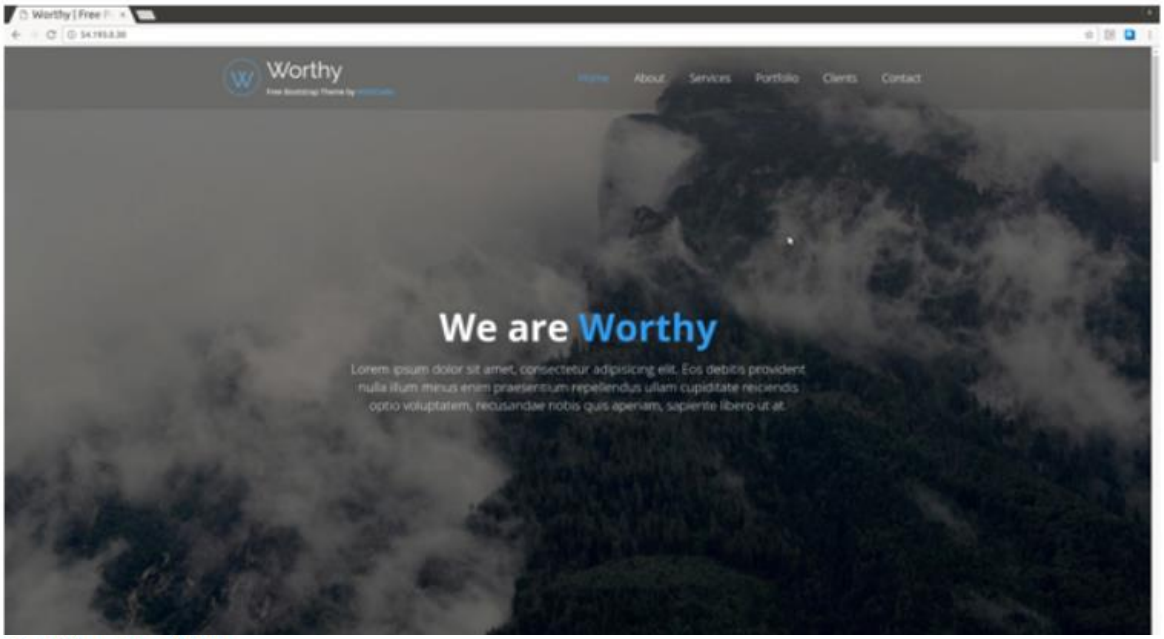
Start httpdservice, enable httpd service even after rebooting the system and stop the firewalls.



Copy website files to the Document root directory and unzip it in order to access the website.



Test the website by accessing ec2 instance public IP from browser.

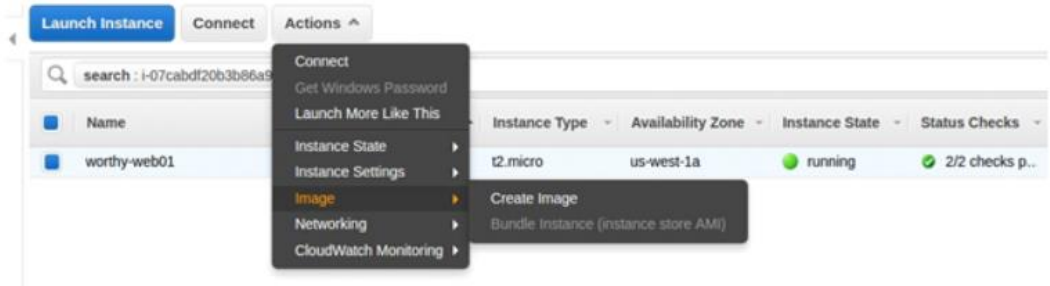


Setting Up AMI:

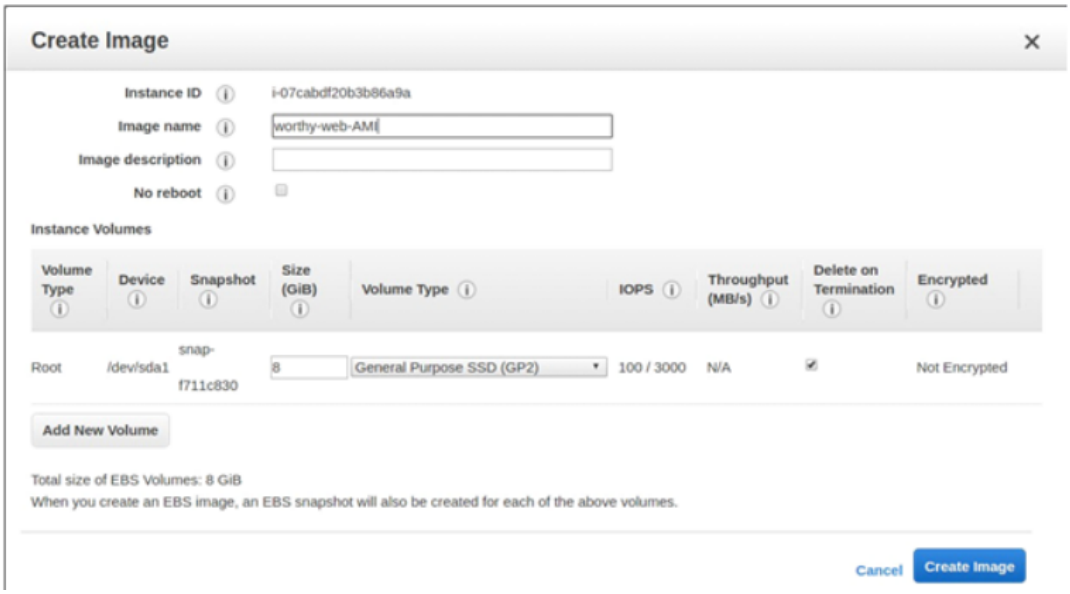
For more details about AMI refer ELB-exercise. Now once we have our instance ready we will create an AMI for the purpose of autoscaling.

Creating AMI:

Select an instance, click on Actions and select ImageCreate Image

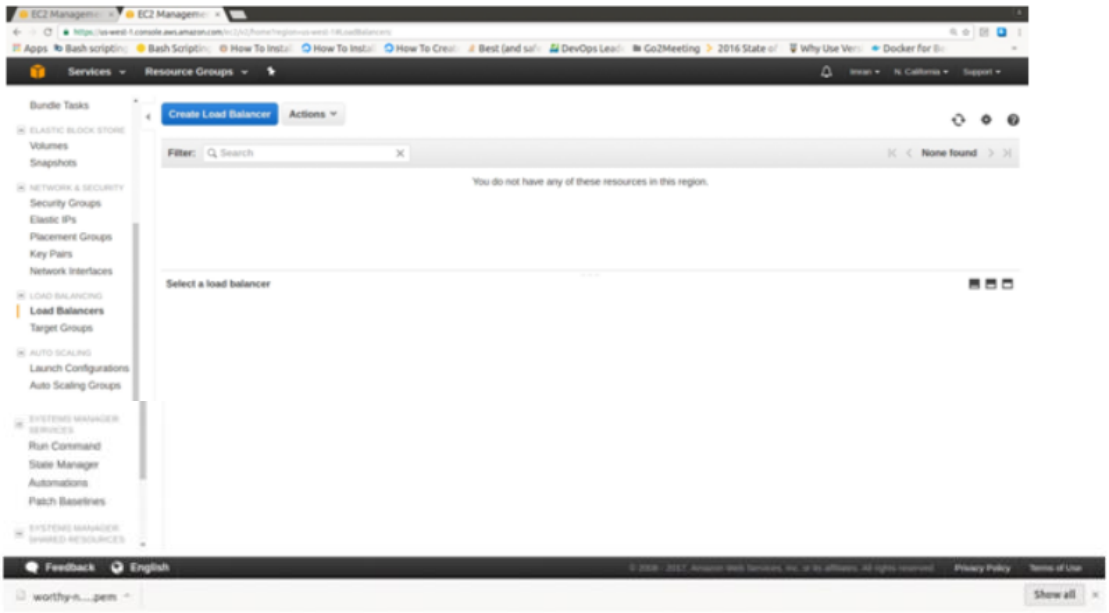


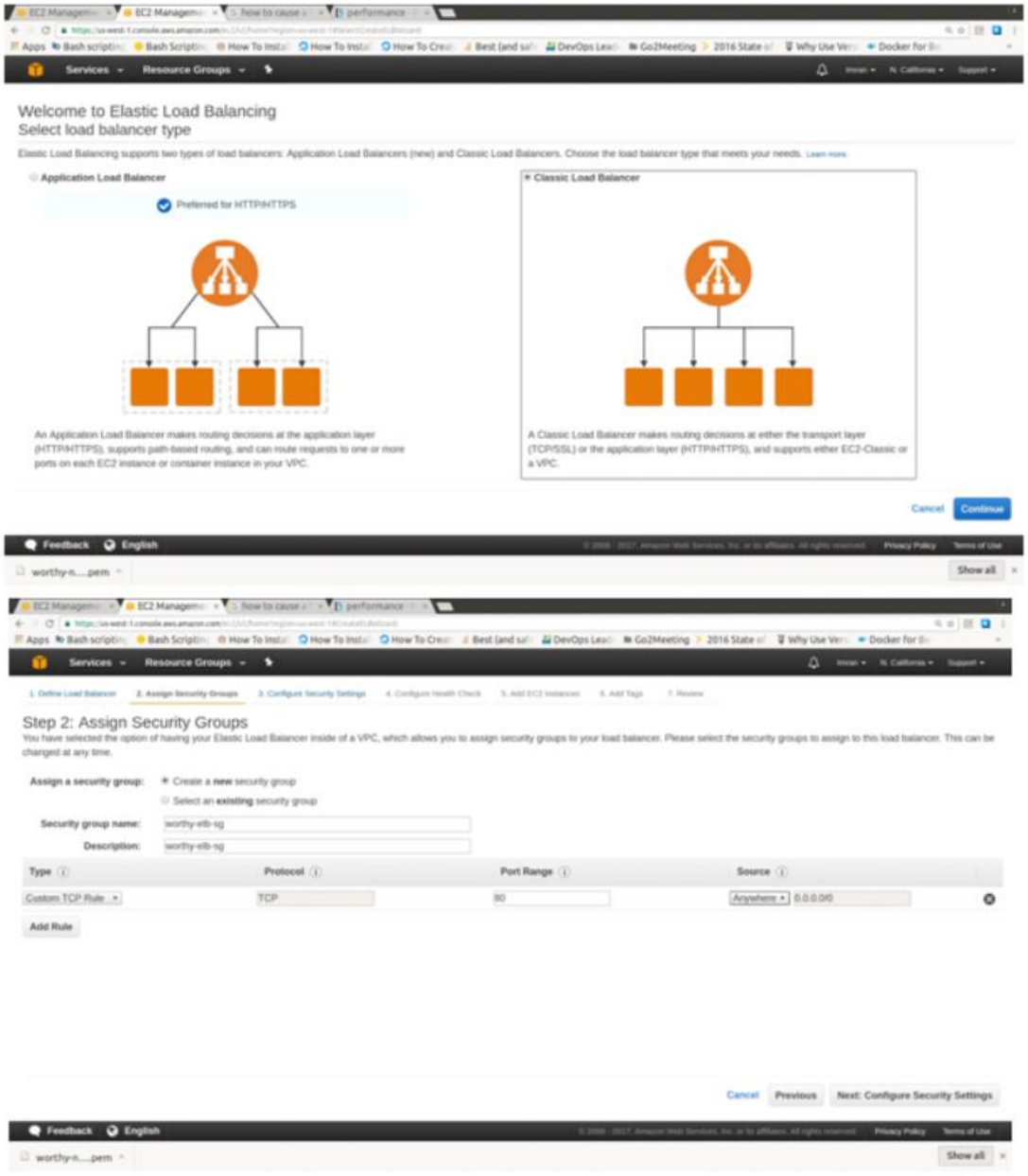
You will see create image dialog box as shown below:

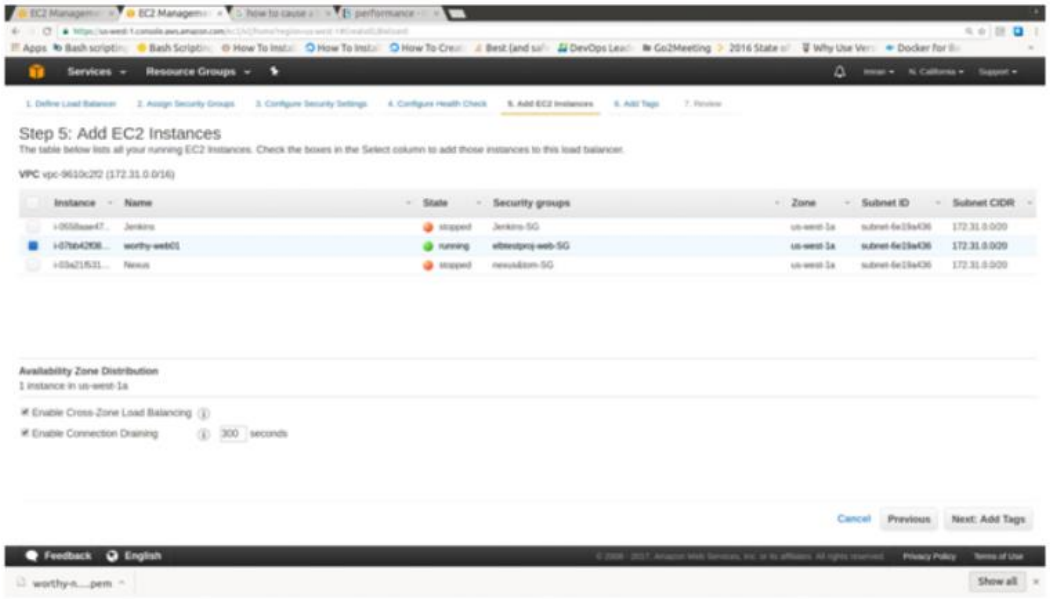


Creating AWS ELB:

Once we have our AMI ready we can start creating ELB, please refer to below screenshots. Select Load Balancer on the left pane of the EC2 Dashboard. To create Load Balancer, follow the steps mentioned in the ELB-Exercise.



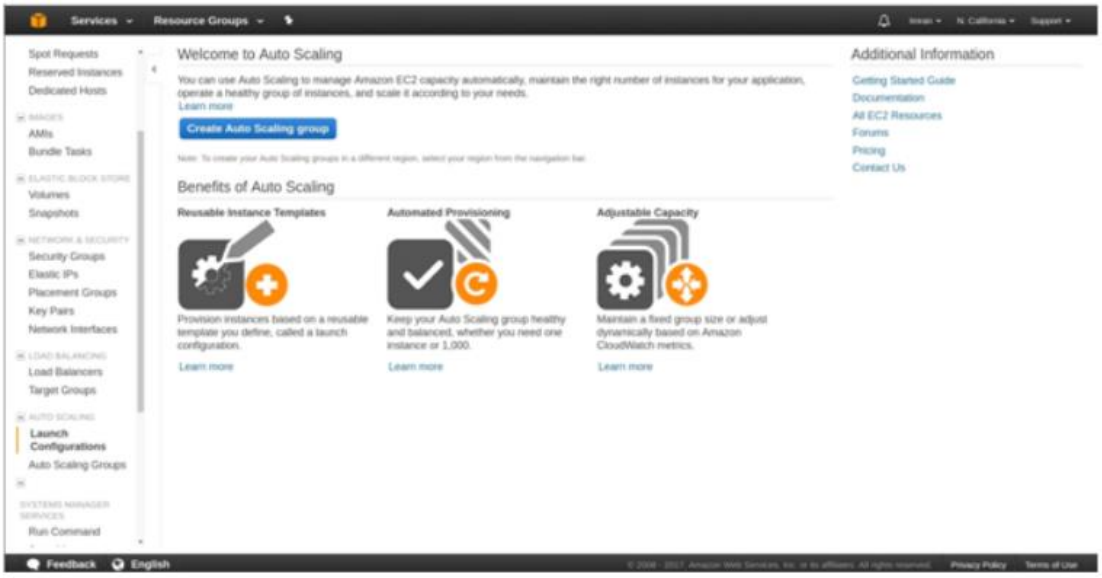

Services
Resource Groups
Welcome to Elastic Load Balancing
Select load balancer type
Application Load Balancer
Preferred for HTTP/HTTPS
An Application Load Balancer makes routing decisions at the application layer (HTTP/HTTPS), supports path-based routing, and can route requests to one or more ports on each EC2 instance or container instance in your VPC.
Classic Load Balancer
A Classic Load Balancer makes routing decisions at either the transport layer (TCP/SSL) or the application layer (HTTP/HTTPS), and supports either EC2-Classic or a VPC.
Cancel
Continue
Feedback
English
Show all
Step 2: Assign Security Groups
You have selected the option of having your Elastic Load Balancer inside of a VPC, which allows you to assign security groups to your load balancer. Please select the security groups to assign to this load balancer. This can be changed at any time.
Assign a security group:
Create a new security group
Select an existing security group
Security group name:
Description:
Type
Protocol
Port Range
Source
Custom TCP Rule
TCP
80
Anywhere
0.0.0.0/0
Add Rule
Cancel
Previous
Next: Configure Security Settings



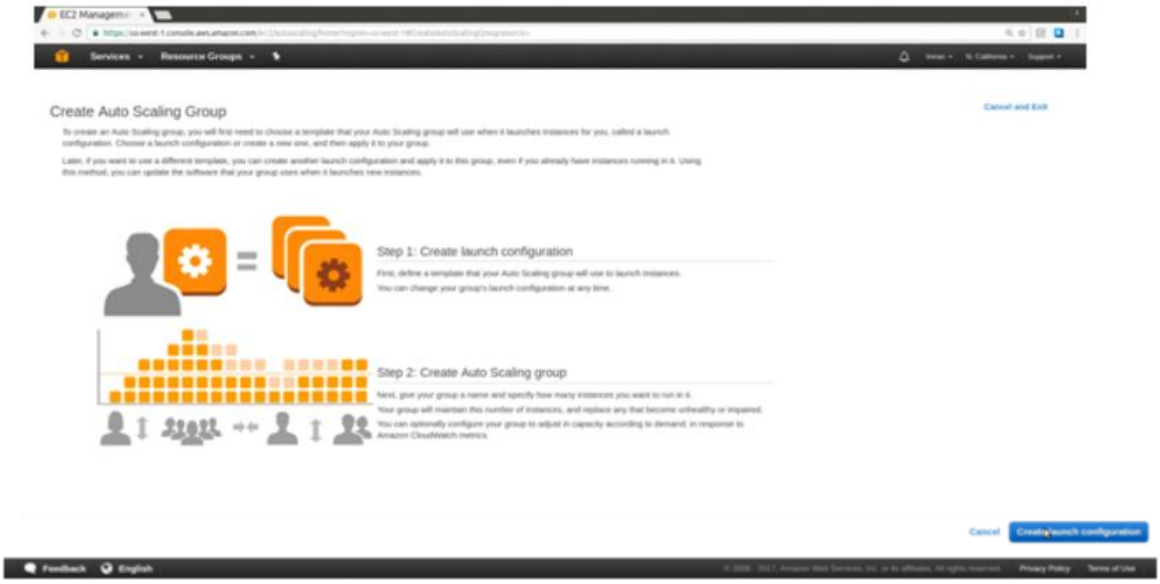
Once we have our ELB ready, wait for the instance in ELB to become healthy and use ELB DNS name to verify if the Worthy website is accessible.

Create Launch Configuration:

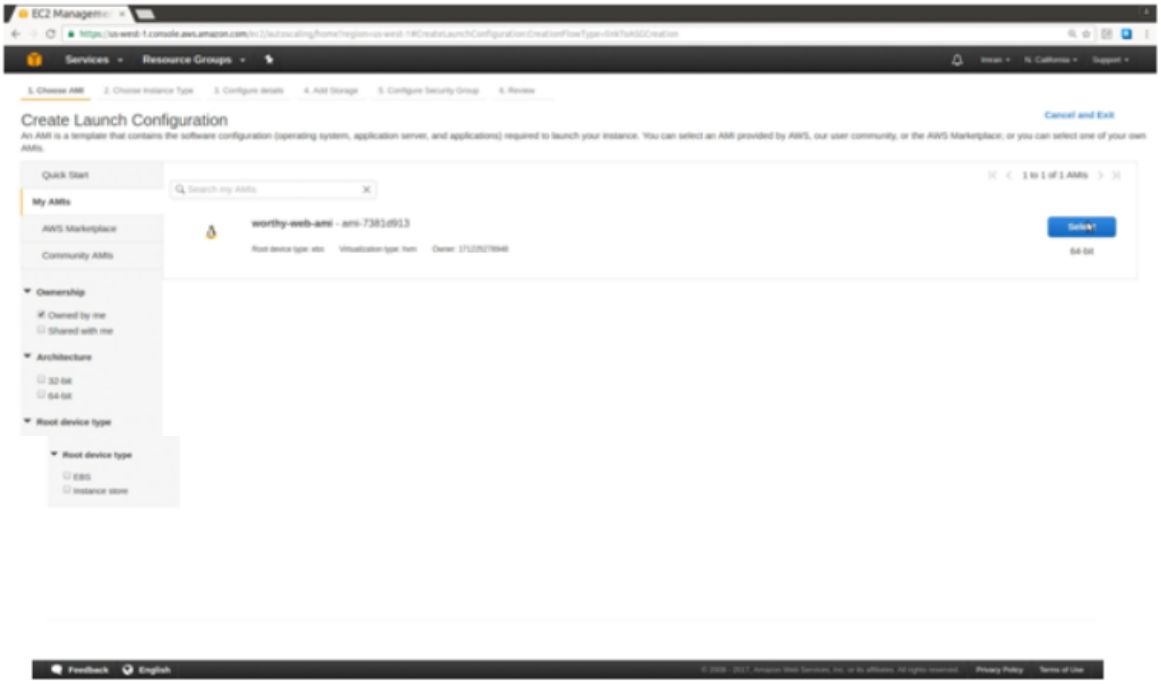
First we need to create and configure a Launch Configuration. From the EC2 Management Dashboard, select the AutoScaling Groups option from the navigation pane as shown in the following screenshot. This will bring up the Auto Scaling Groups dashboard. Next, select the Create Auto Scaling group to bring up the Auto Scaling setup wizard.



Click on Create Auto Scaling group.



Choose the AMI that we created previously.

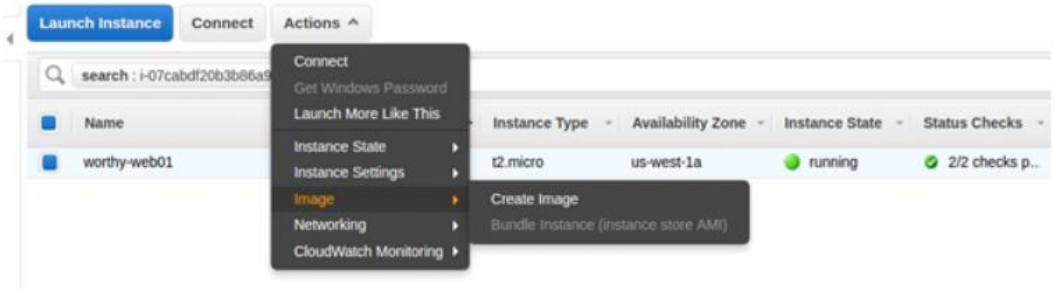


Setting Up AMI:

For more details about AMI refer ELB-exercise. Now once we have our instance ready we will create an AMI for the purpose of autoscaling.

Creating AMI:

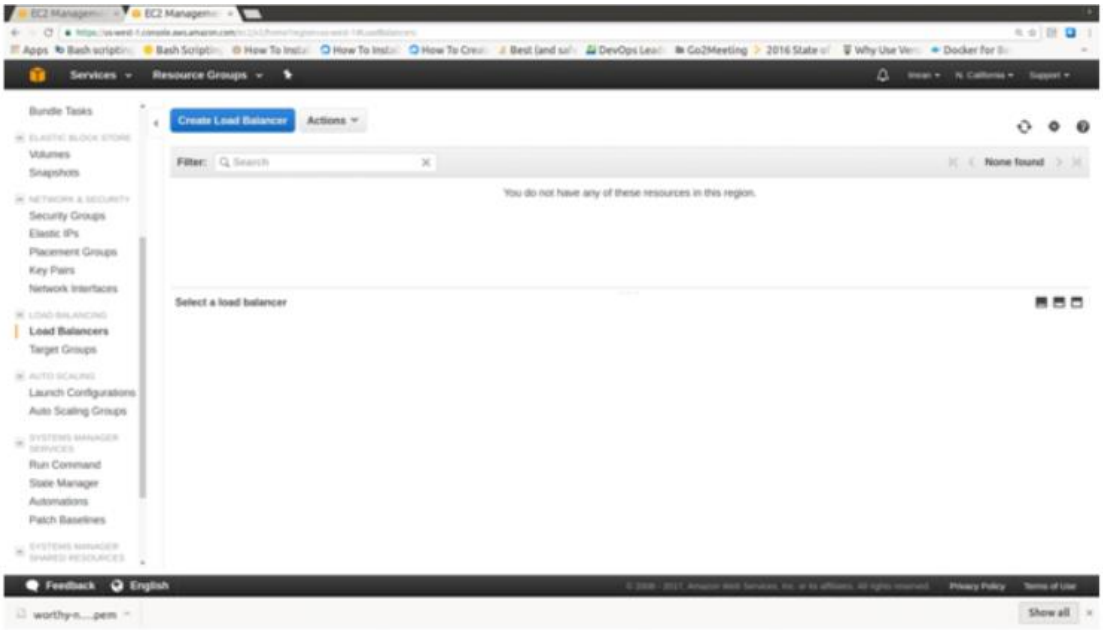
Select an instance, click on Actions and select ImageCreate Image

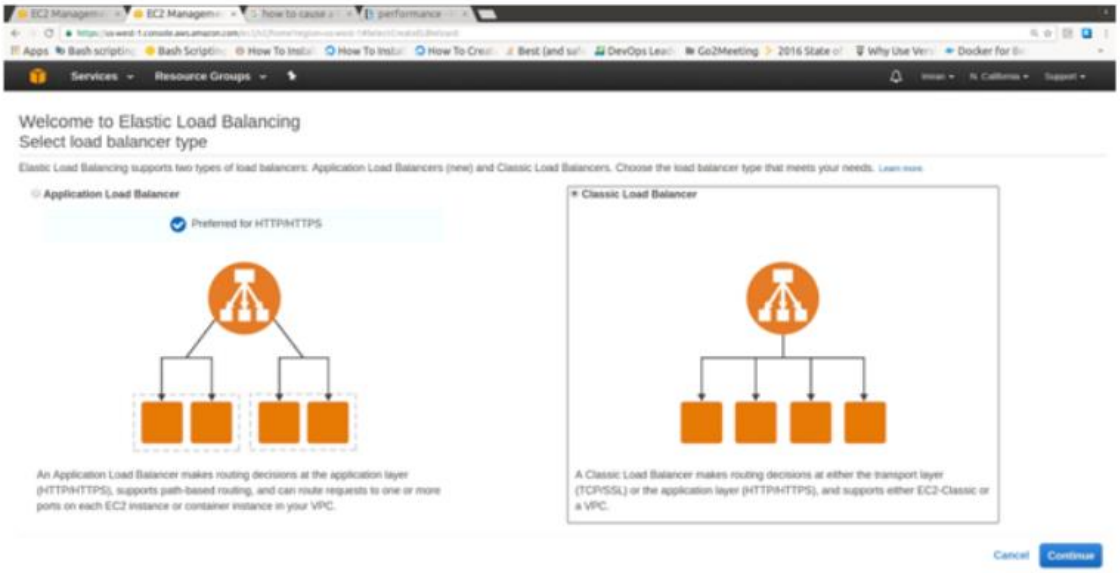


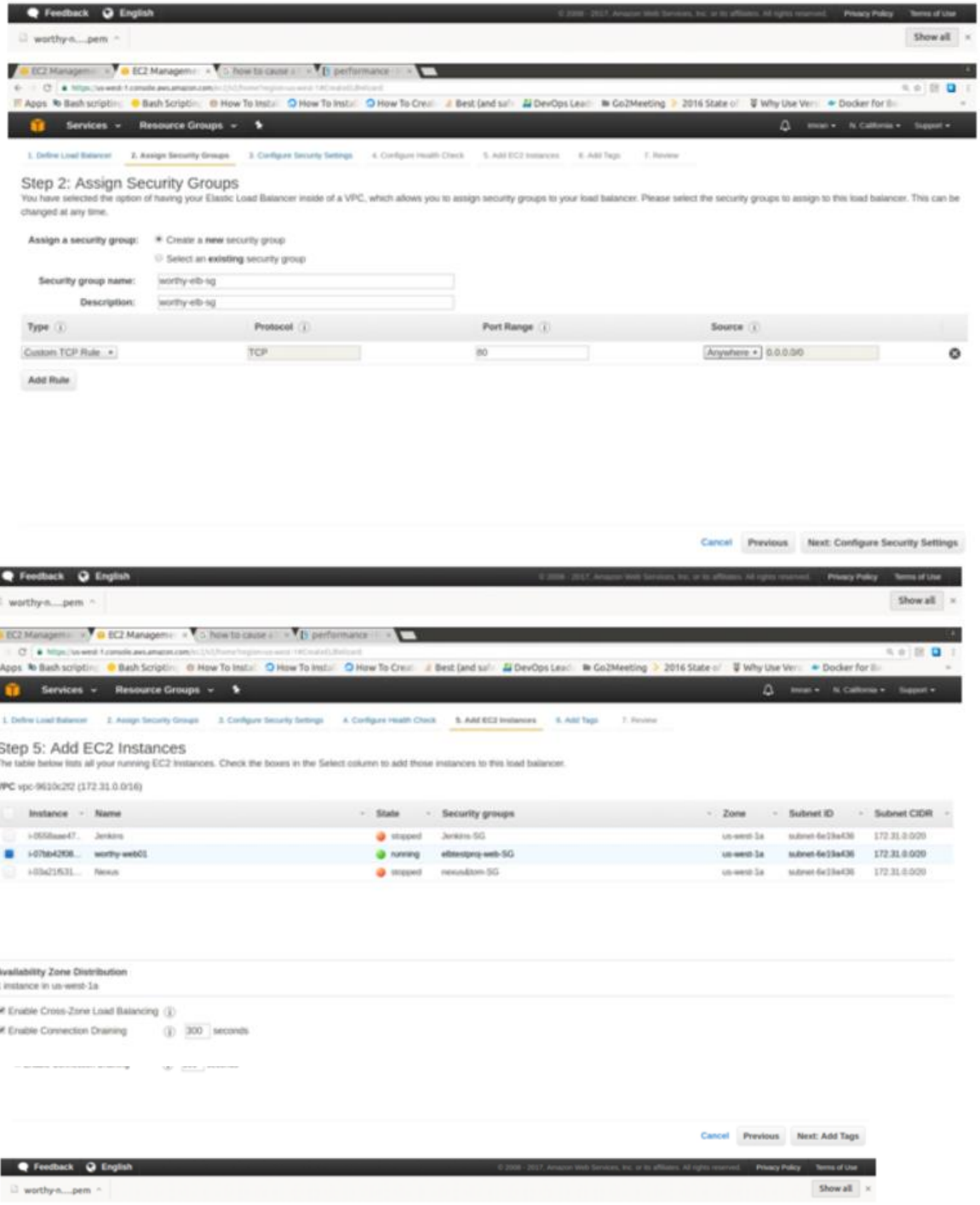
You will see create image dialog box as shown below:

Creating AWS ELB:

Once we have our AMI ready we can start creating ELB, please refer to below screenshots. Select Load Balancer on the left pane of the EC2 Dashboard. To create Load Balancer, follow the steps mentioned in the ELB-Exercise.



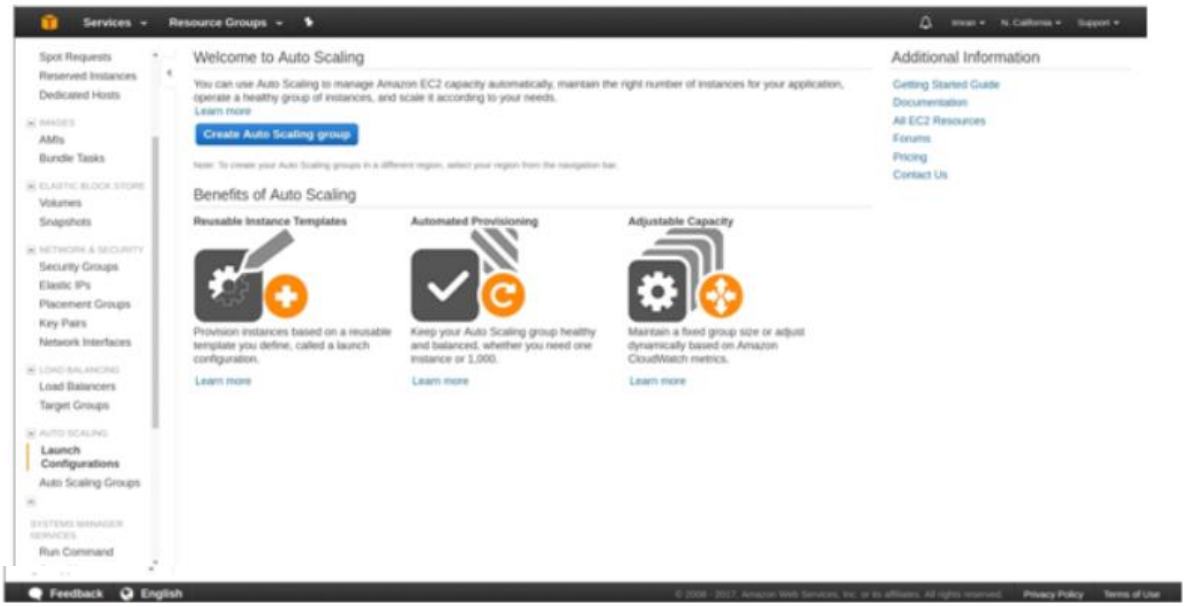




Once we have our ELB ready, wait for the instance in ELB to become healthy and use ELB DNS name to verify if the Worthy website is accessible.

Create Launch Configuration:

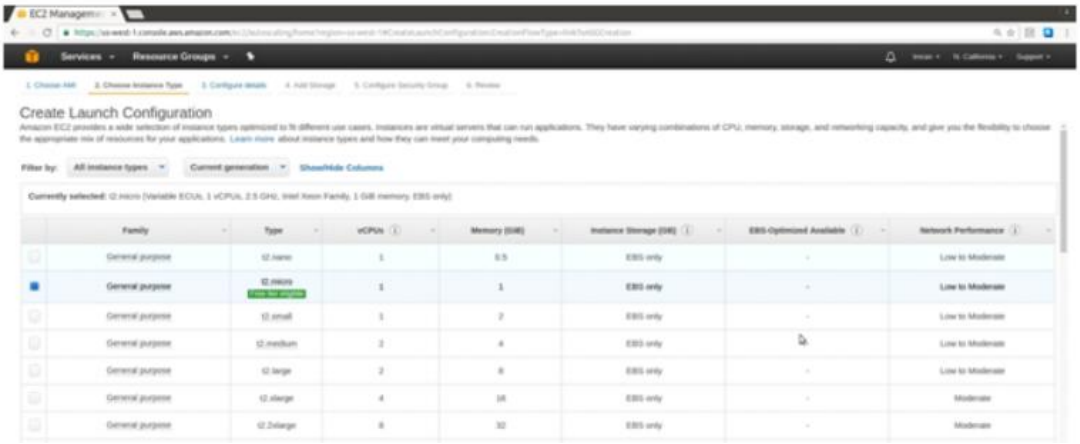
First we need to create and configure a Launch Configuration. From the EC2 Management Dashboard, select the AutoScaling Groups option from the navigation pane as shown in the following screenshot. This will bring up the Auto Scaling Groups dashboard. Next, select the Create Auto Scaling group to bring up the Auto Scaling setup wizard.

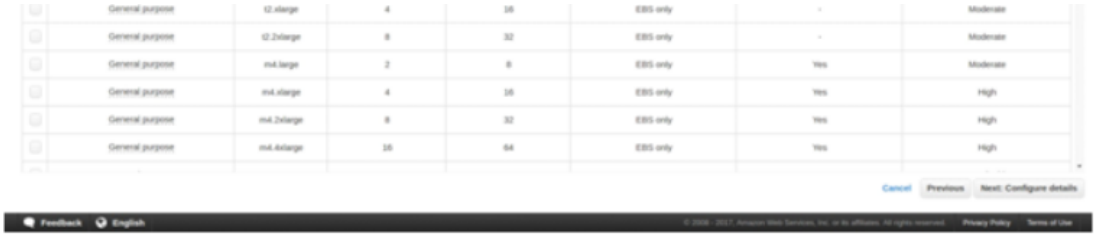


Click on Create Auto Scaling group.

Choose the AMI that we created previously.

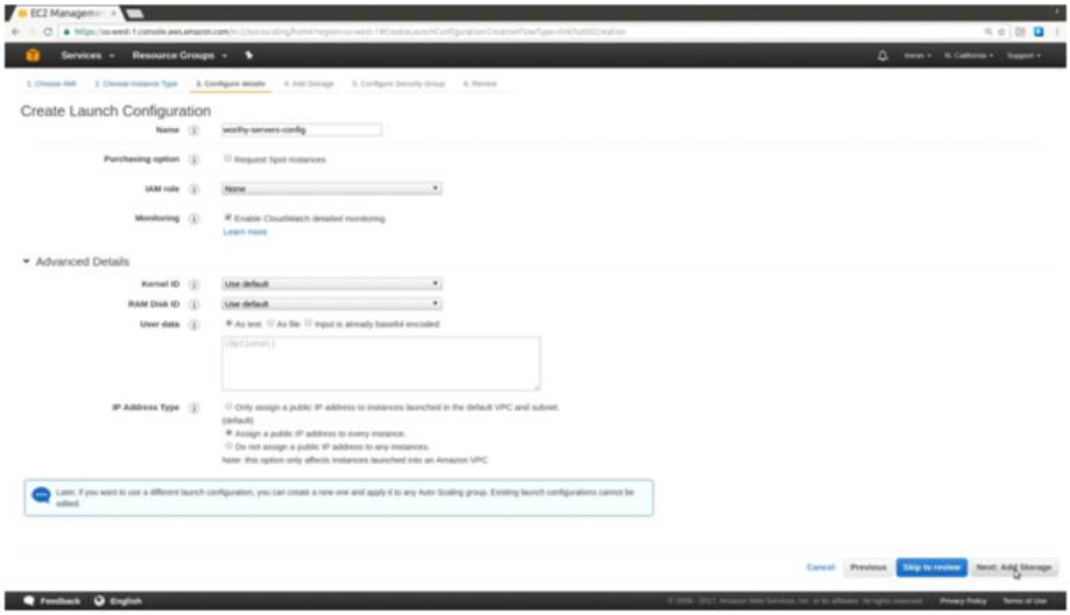
Choose an instance type: Select t2.micro for free tier.





1. Configure Details:

Provide aName for the Launch Configuration. Select the Enable CloudWatch detailed monitoring checkbox if you wish to have your instances monitored for a duration of 60 seconds. By default your instances will be monitored by CloudWatch for a minimum period of 300 seconds (five minutes) for no charge at all. Selecting detailed monitoring will incur additional charges, so use it with caution.



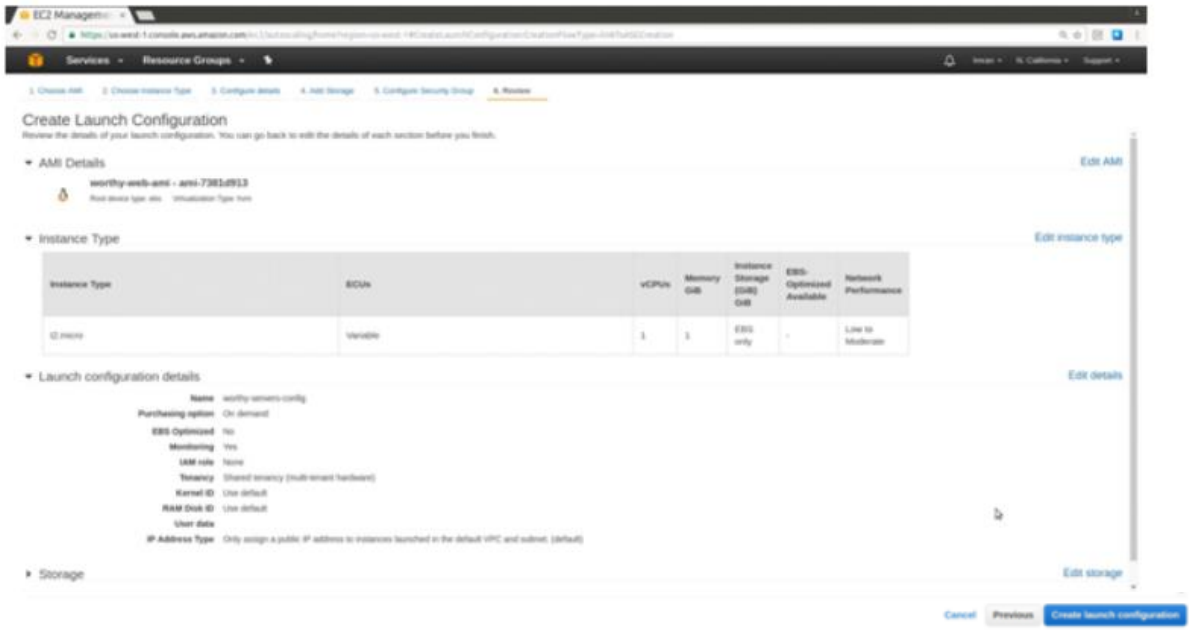
Note: Enabling the CloudWatch detailed monitoring option is highly recommended in case the instances belong to a production environment. Once the configure details are filled out, you can even set the instance's IP addressing scheme by selecting the Assign a public IP address to every instance option from the Advanced Details section.

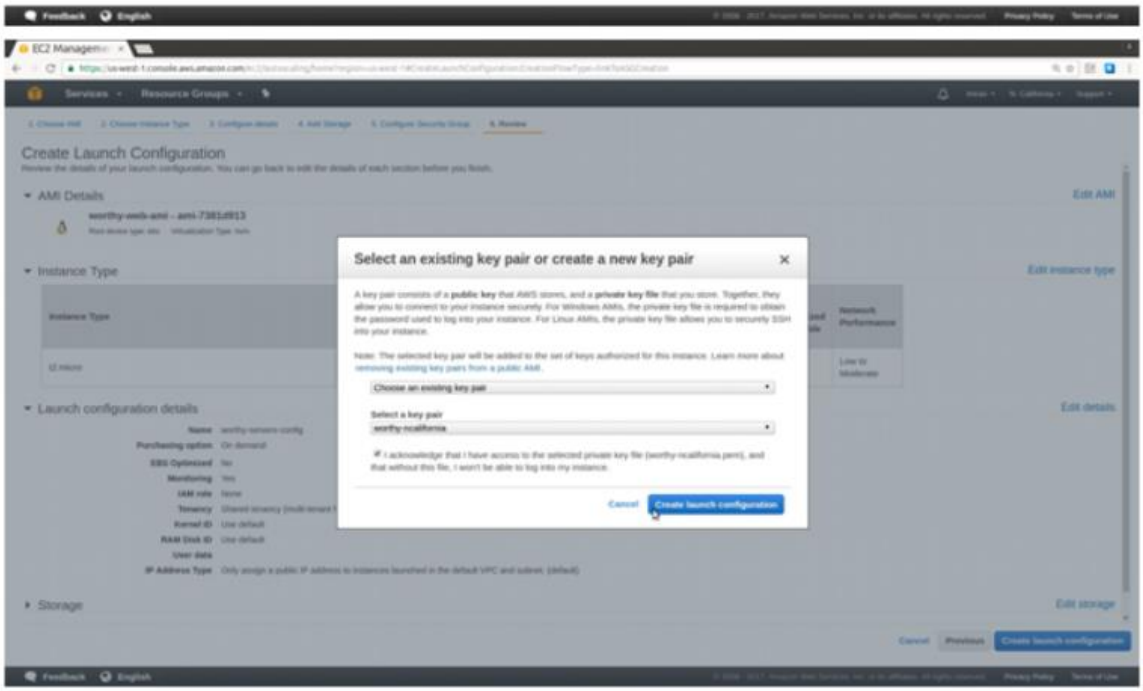
1.Add Storage:

You can add an optional Volume to your instances by selecting the Add New Volume button on the Add Storage page. Here I have not provided any additional volumes to my instances. Click on Next: Configure Security Group to either create or select an existing security group for your auto scaled instances.

Configure Security Group: From the Configure Security Group page, select an appropriate Security Group for your Auto Scaled instances. Since we are working with web server instances we use the following inbound rules: SSH □MyIPand HTTP □ Anywhere

Review: Make sure that all the details are configured correctly. Once verified, click on Create Launch Configuration to complete the process.





Creating the Auto Scaling Group: An Auto Scaling group is a collection of EC2 instances, and the core of the Auto Scaling service. You create an Auto Scaling group by specifying the launch configuration you want to use for launching the instances and the number of instances your group must maintain at all times. You also specify the Availability Zone in which you want the instances to be launched.

Configure Auto Scaling Group Details:

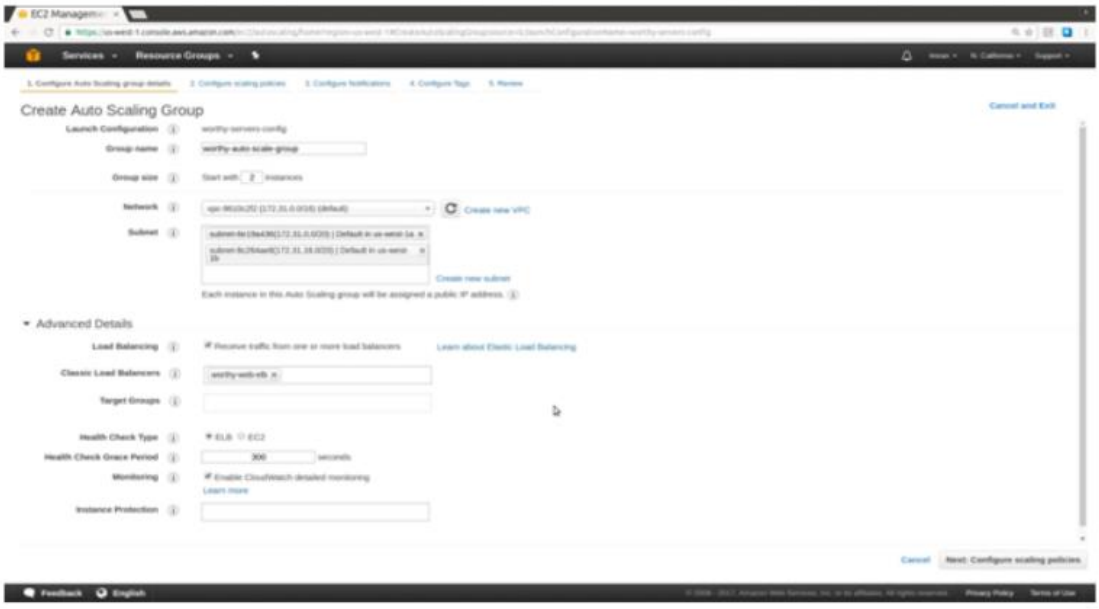
The first step in creating Auto Scaling Group requires providing a suitable name for Auto Scaling Group as well as its Network and Load Balancing details. Fill in the required fields as per your requirements:

1.Group name: Provide a suitable name for your Auto Scaling Group.

2.Group size: Here, enter the desired capacity for your Auto Scaling Group. The value entered here represents the number of instances. Here I have chosen 2 instances just for practice.

3.Network: If you are launching a t2.micro instance, you must select a VPC in Network. Otherwise, if your account supports EC2-Classic and you are launching a type of instance that doesn't require a VPC, you can select either Launch into EC2-Classic or a VPC.

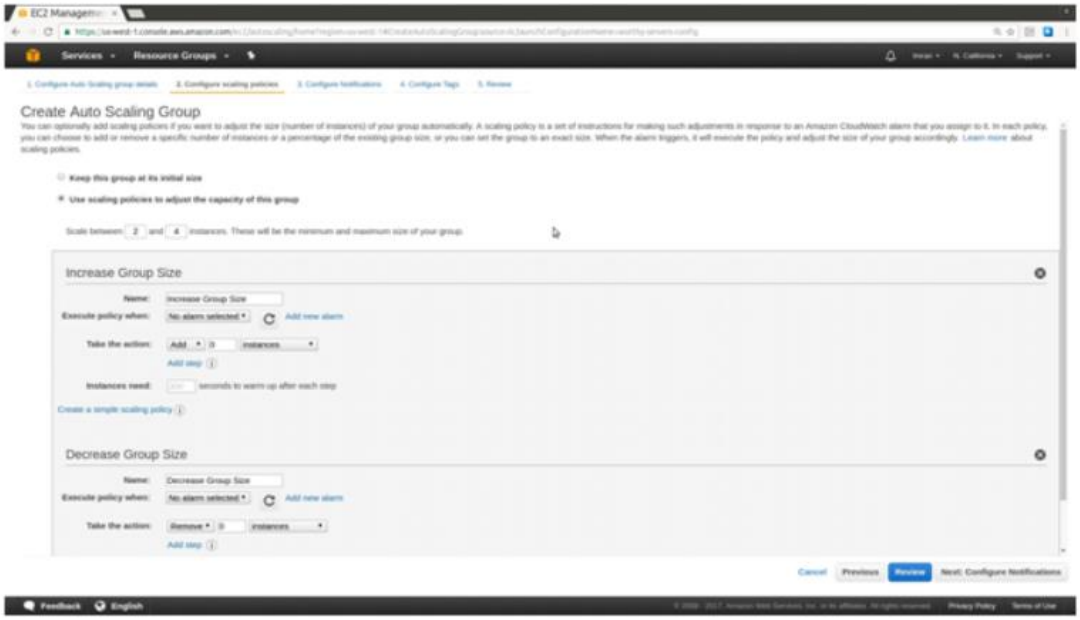
4.Subnet: If you select a VPC in the previous step, select one or more subnets from Subnet. If you select EC2-Classic in the previous step, select one or more Availability Zones from Availability Zones. In my case, I have selected two subnets, each created in a different AZ.



Note: Each instance in this Auto Scaling Group will be provided with a public IP address. After filling the basic details we will configure the Advanced Details section of our Auto Scaling Group: Load Balancing: Since we have already created and configured our ELB, we will be using that to balance out incoming traffic for our instances.

Health Check Type: We can use either our EC2 instances or ELB as a health check mechanism to make sure that your instances are in a healthy state. By default, Auto Scaling will check your EC2 instances periodically for their health status. If an unhealthy instance is found, Auto Scaling will immediately replace that with a healthy one. Here, I have selected ELB as my health check type, so all the instances health checks are now performed by the ELB itself. Health Check Grace Period: By default, this value is set to 300 seconds.

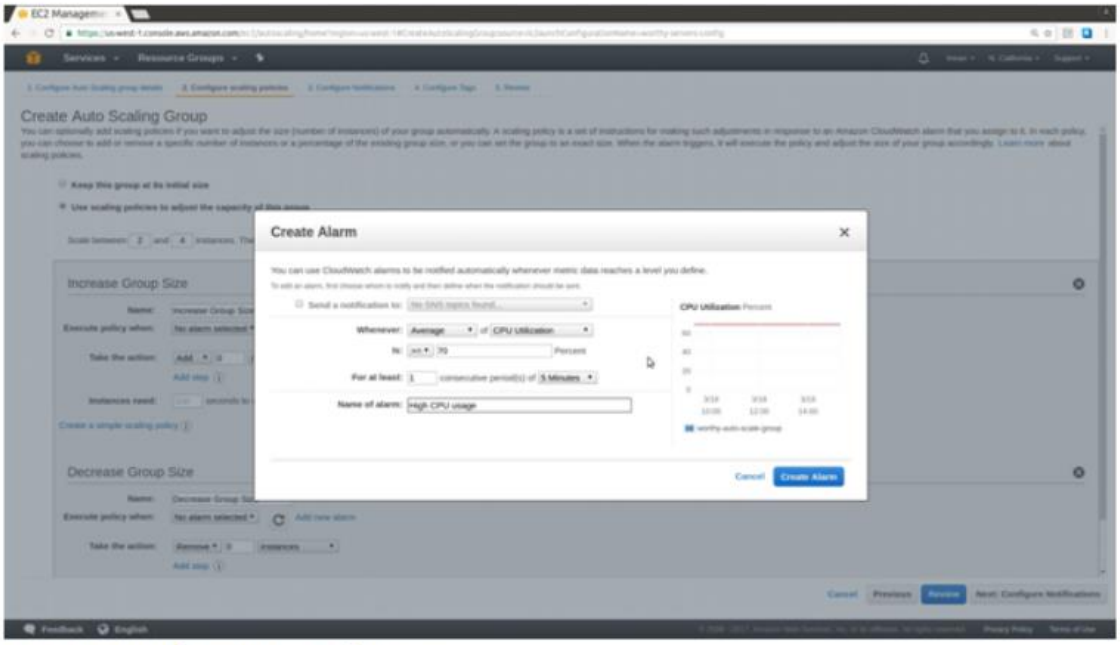
Configure Scaling policies: You can create a scaling policy that uses CloudWatch alarms to determine when your Auto Scaling group should scale out or scale in. Each CloudWatch alarm watches a single metric and sends messages to Auto Scaling when the metric breaches a threshold that you specify in your policy. You can use alarms to monitor any of the metrics that the services in AWS that you're using send to CloudWatch, or you can create and monitor your own custom metrics.



There are two policies used by an Auto Scaling Group: one to increase the instance count based on certain alarms and the other to decrease the instance count.

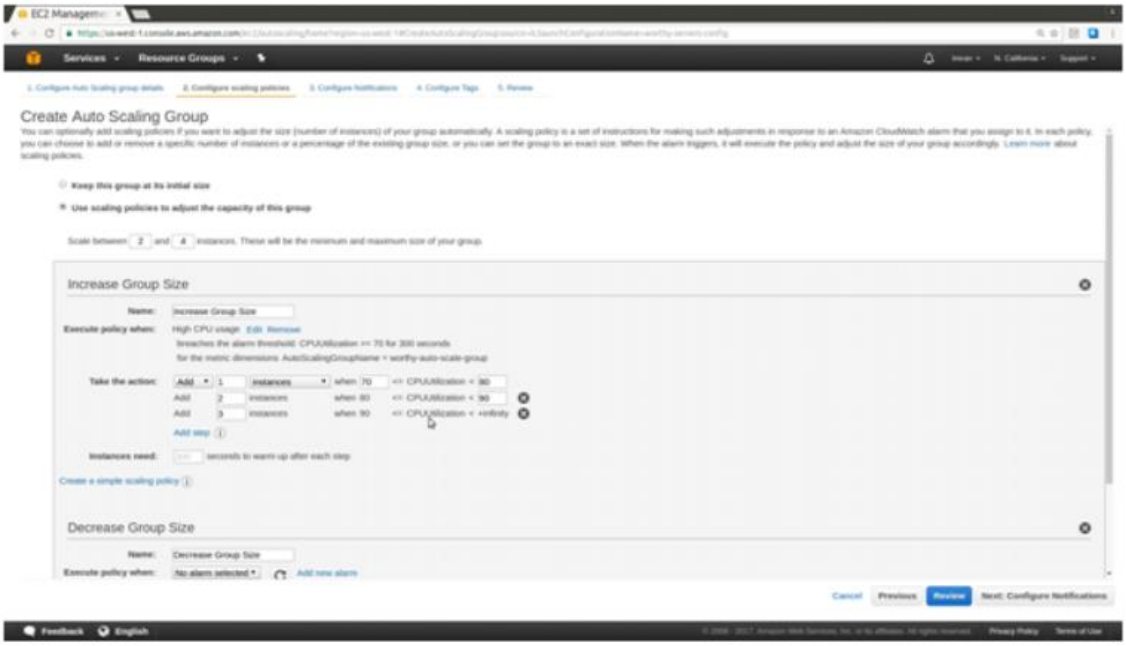
Increase Group Size policy, as shown in the following screenshot:

Name: Provide a suitable name for your scale-out policy. Execute policy when: Here you have to select a pre-configured alarm using which the policy will get triggered. As we are configuring this for the first time, select the Add new alarm option. This will pop up the Create Alarm dialog, as shown in the following screenshot: Scaling as per CPU utilization: We want our Auto Scaling Group to be monitored based on the CPU Utilization metric for an interval of 5 minutes. If the average CPU Utilization is greater than or equal to 50 percent for at least one consecutive period, then send a notification mail.

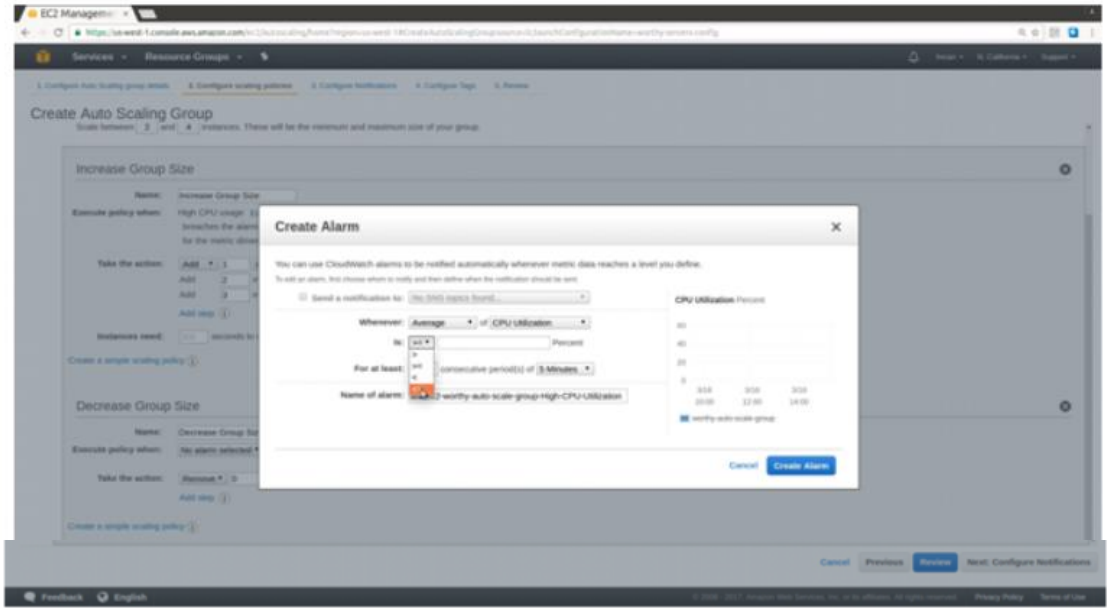


Set The Action After Trigger:

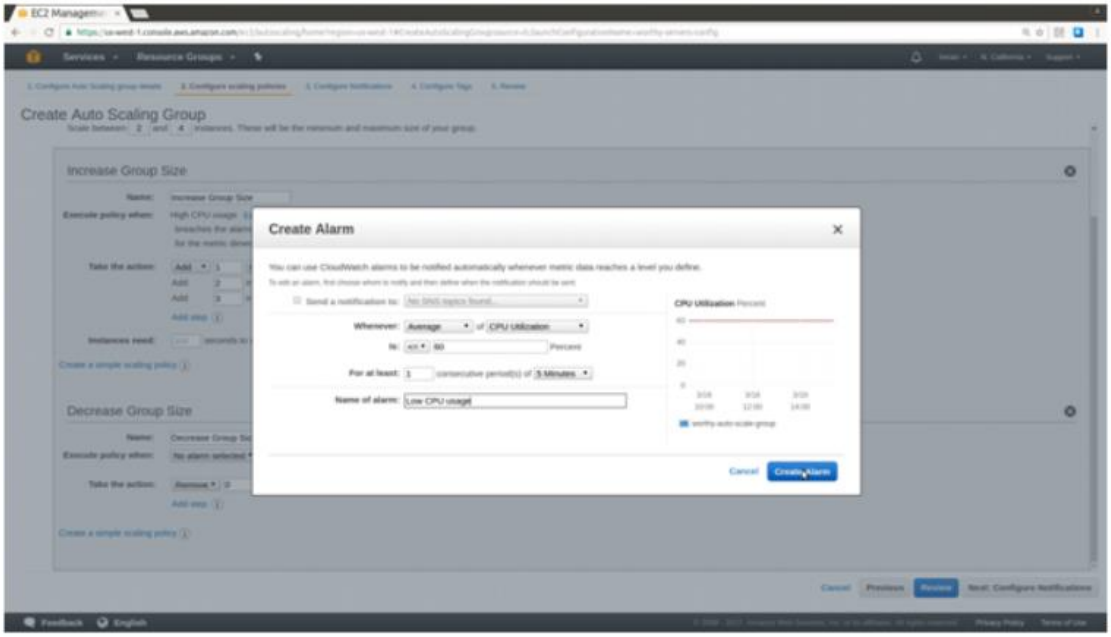
With the basic alarm now set, you can configure your policy what action it has to take if the particular threshold is breached. Select Add from the dropdown list and provide a suitable number of instances that you wish to add when a certain condition matches. Here I have created a four-step scaling policy that first adds one instance to the group when the average CPU utilization is within a particular threshold range, such as 70-80 percent. Next, another two instances are added when the CPU utilization increases to 80-90 percent. You can add multiple such steps by selecting the Add step option, as shown in the following screenshot.



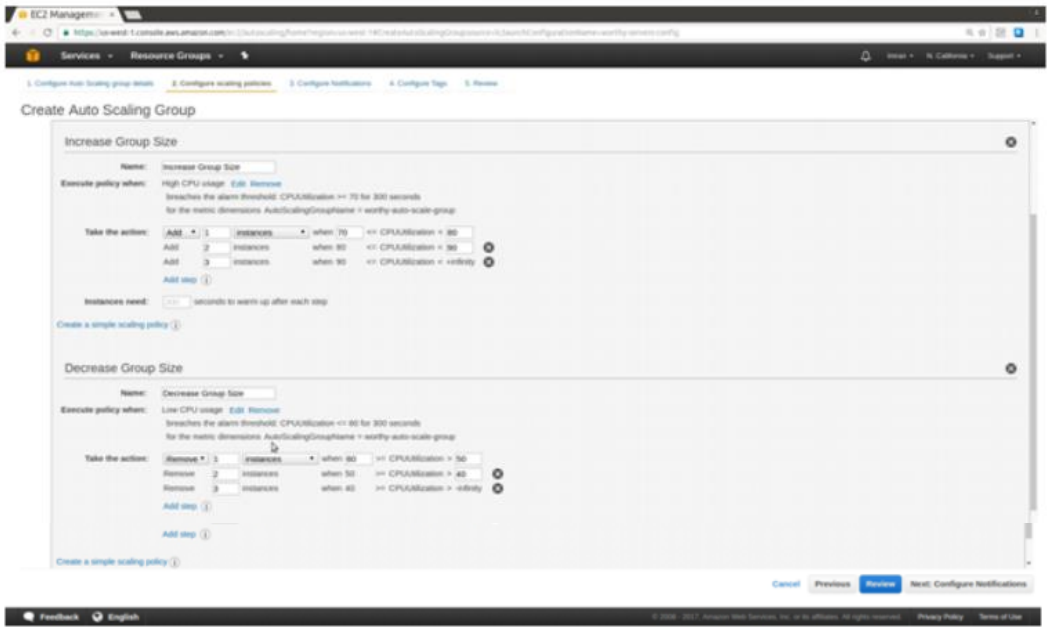
Set scale in alarm trigger: Similarly set the threshold for decreasing group size and terminating the instances.



When the decreasing threshold is reached another instance is removed when the CPU utilization decreases to less than 60.



Below Screenshot gives the review of the increasing and decreasing group size of auto scaling group. Click on Next Configure Notifications



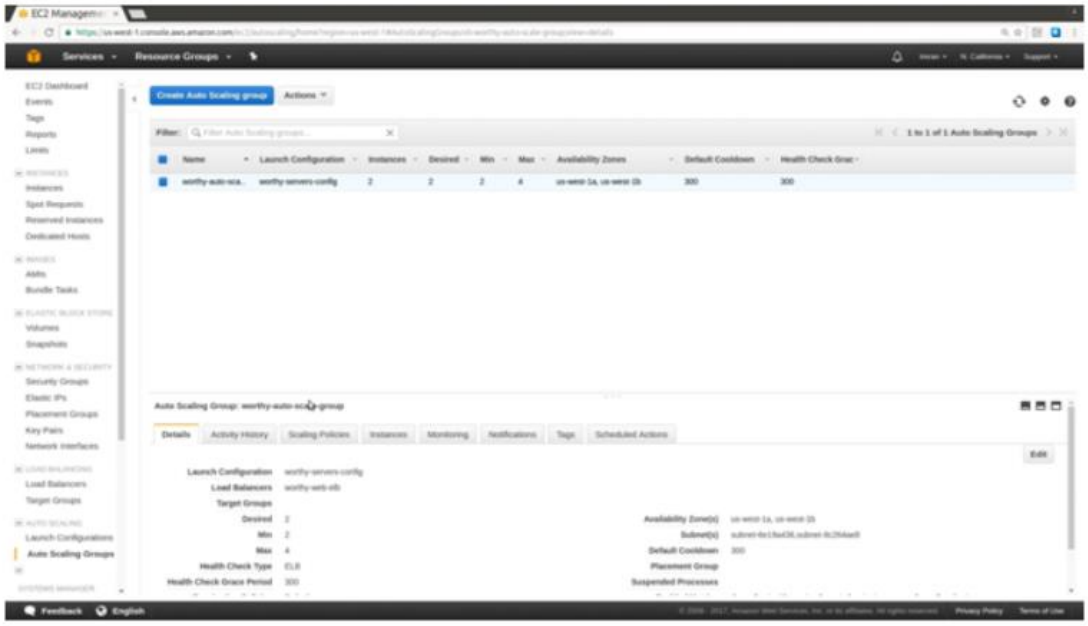
We can configure our Auto Scaling Group to send notifications to any particular endpoint such as an e-mail address whenever a specified event gets triggered, such as the successful launch of an instance, or a failure to launch an instance. Here I have not given any notification. Click on next configure tags

We can tag our instances for organizing, managing and identifying our instances more effectively and efficiently. Next click on review which gives all the details you have selected for auto scaling group and click on create auto scaling group.

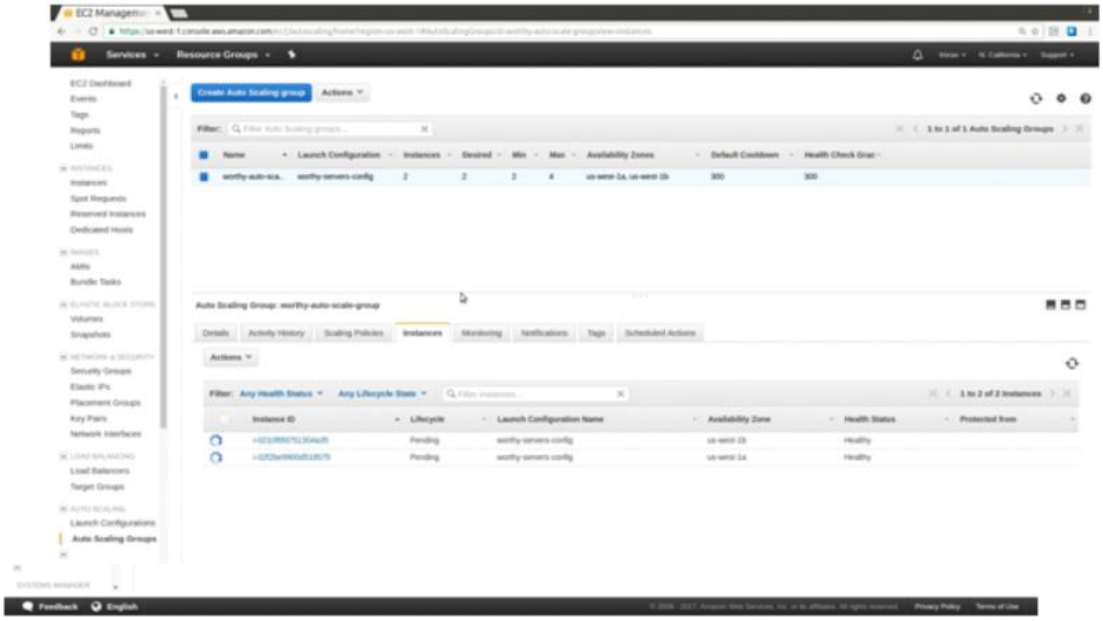
Then you can see a message as Successfully created auto scaling group.

Verify all the details On the Auto Scaling Groups page, select the Auto Scaling group that you just created.

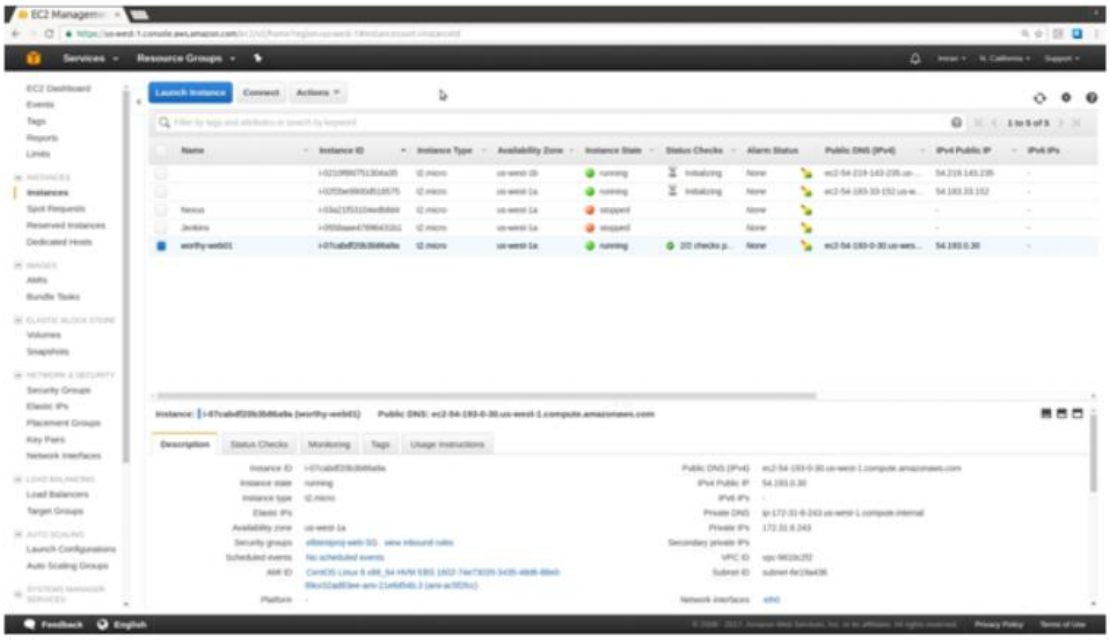
1.The Details tab provides information about the Auto Scaling group.



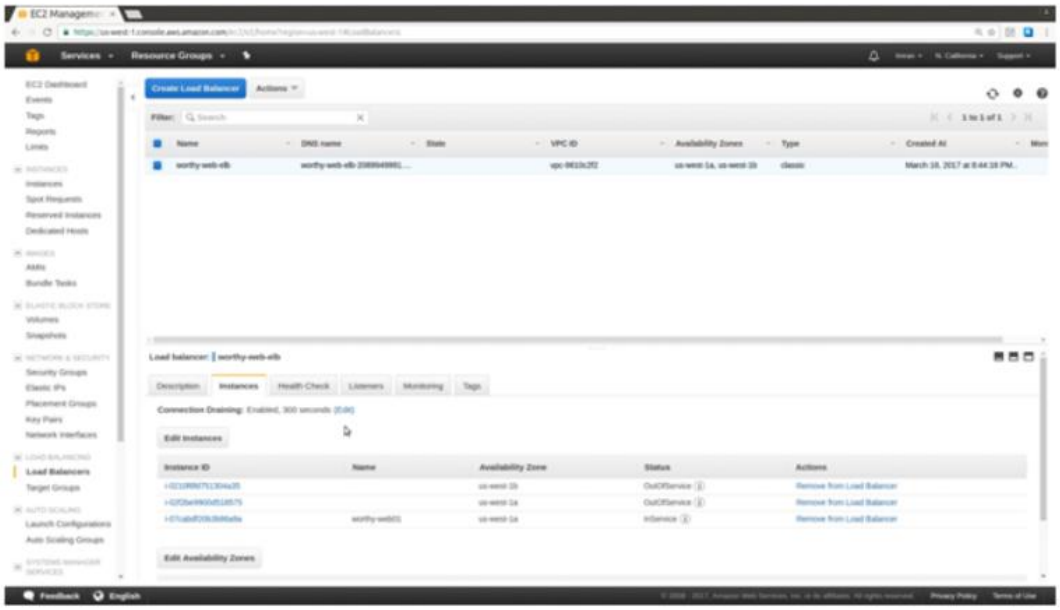
On the Instances tab, the Lifecycle column shows the state of your instance.



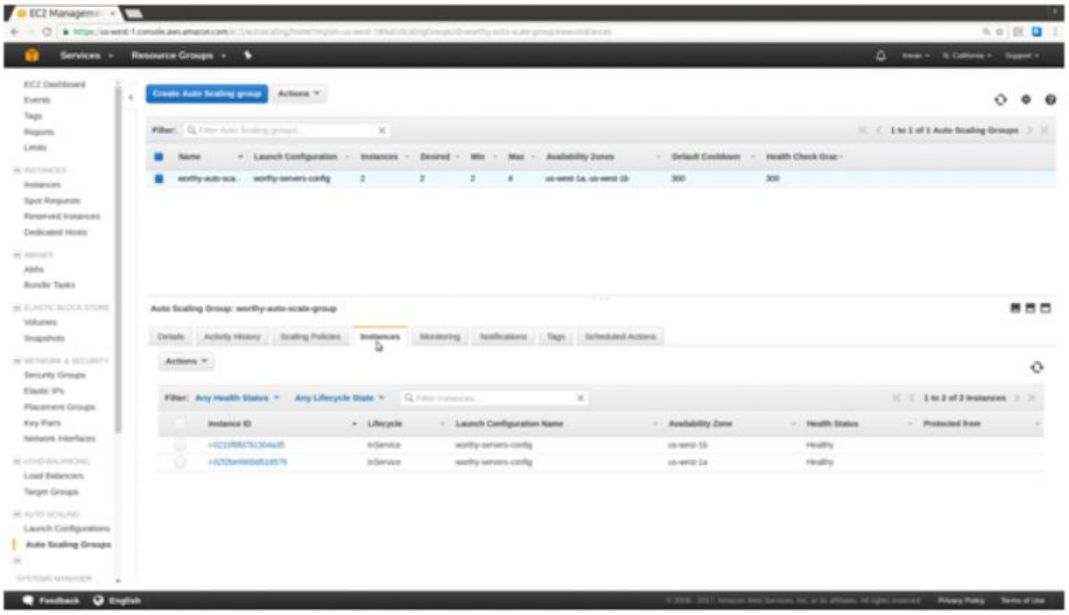
On the EC2 Instances Dashboard you can see the scaled out instances as shown in the below screenshot:



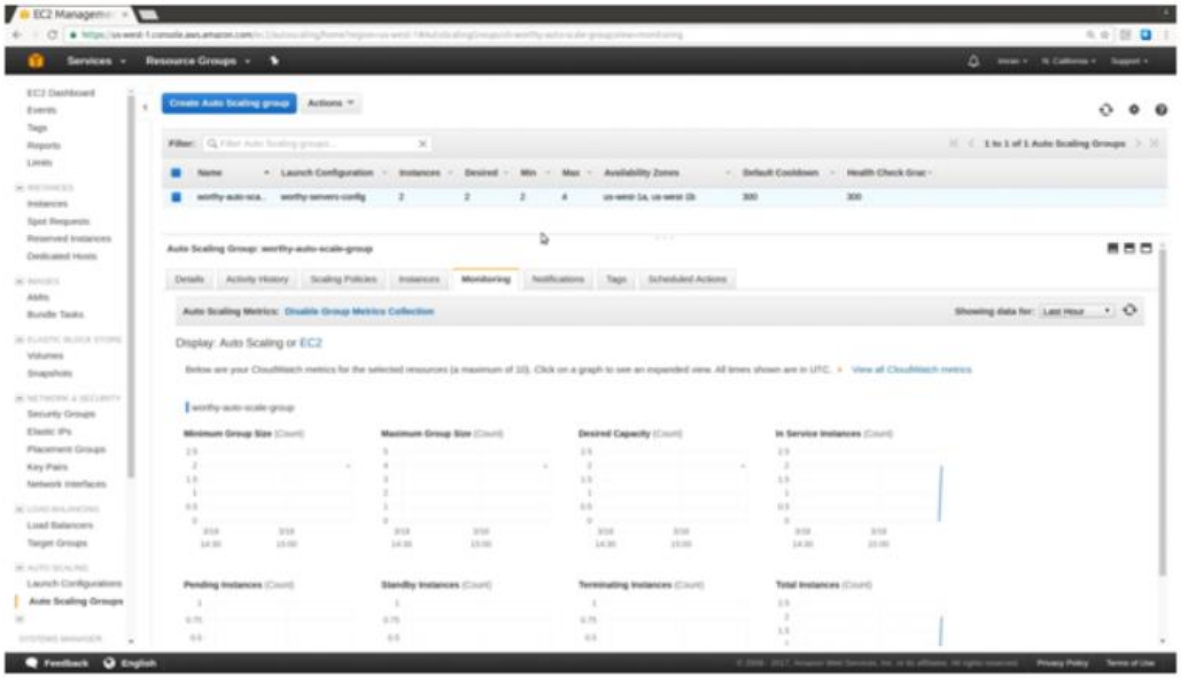
On the load balancer navigation page, select the load balancer you can see the instances attached to the load balancer which are created by auto scaling group.

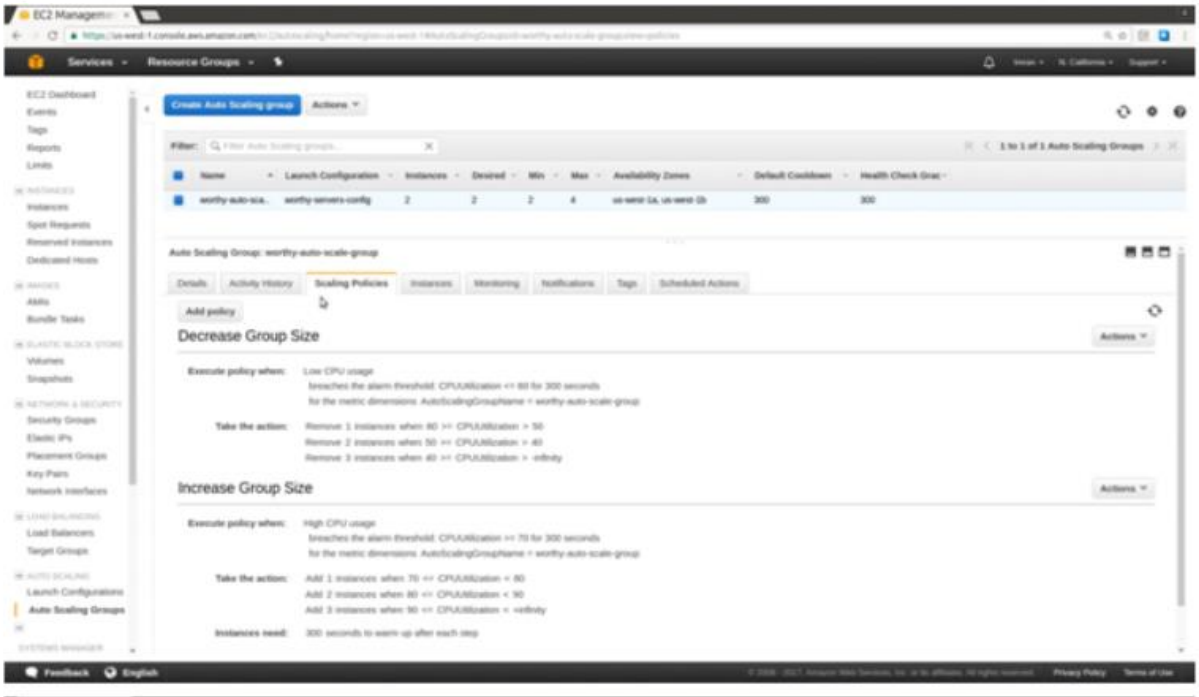


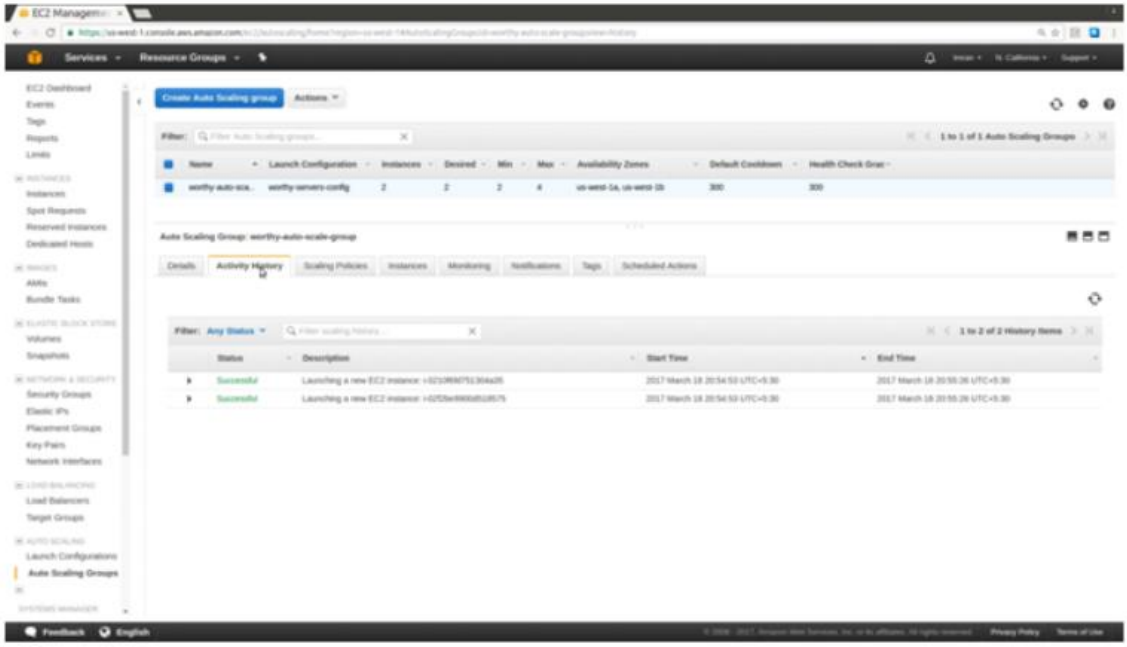
In the below screenshot you can see that your Auto Scaling group has launched your EC2 instance, and that it is in the InService lifecycle state. The Health Status column shows the result of the EC2 instance health check on your instance.

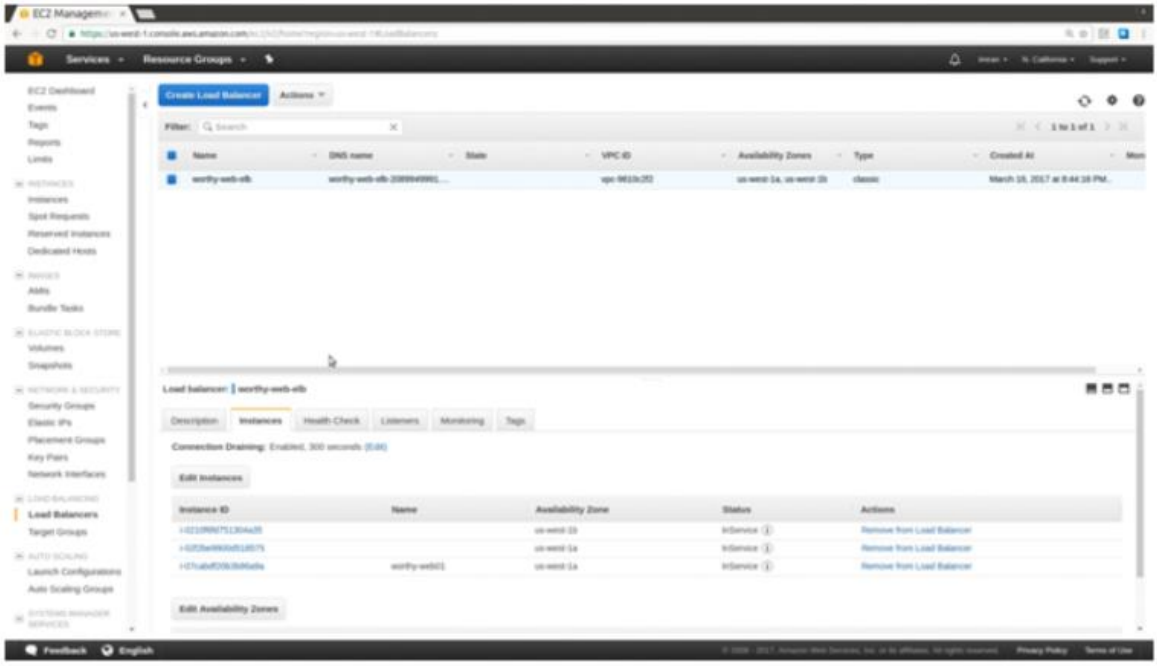


Monitoring tab shows the cloud watch metrics for the selected auto scaling group

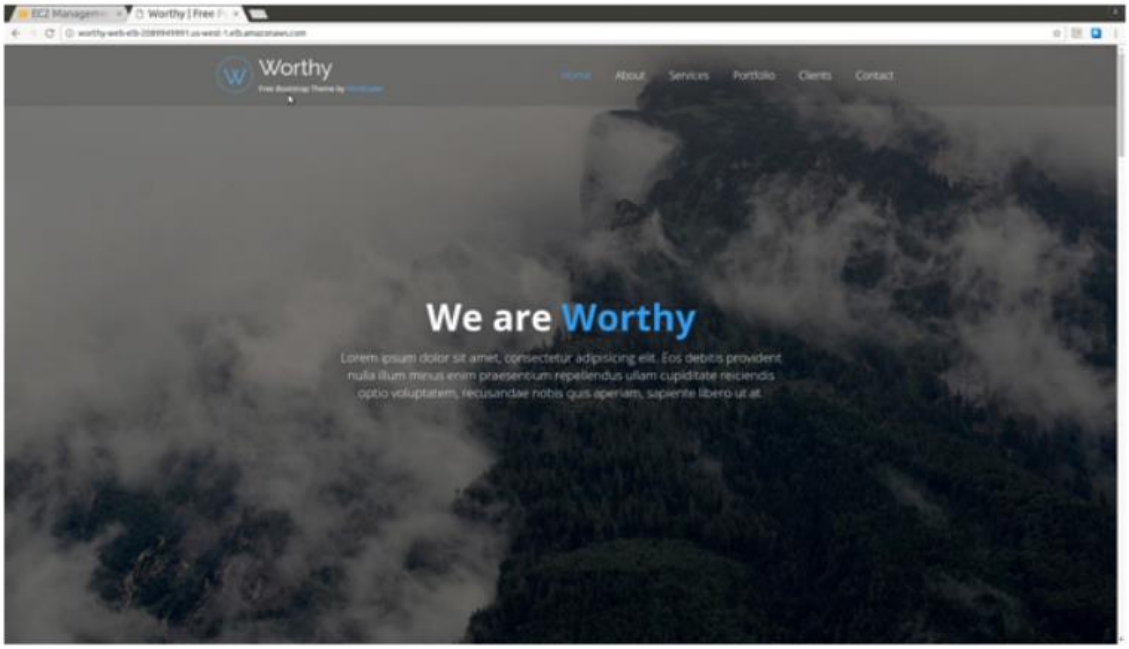








Use ELB DNS name to verify website.



AWS Cloud Watch

Amazon CloudWatch is a monitoring service for AWS cloud resources and the applications we run on AWS. It provides a reliable, scalable, and flexible monitoring solution that we can start using within minutes. We no longer need to set up, manage, and scale our own monitoring systems and infrastructure. We can use CloudWatch to collect and track metrics, which are variables we can measure for our resources and applications. CloudWatch alarms send notifications or automatically make changes to the resources we are monitoring based on rules that you define.

Features & Benefits:

1.Monitor Amazon EC2 - View metrics for CPU utilization, data transfer, and disk usage activity from Amazon EC2 instances for no additional charge. For an additional charge, CloudWatch provides Detailed Monitoring for EC2 instances with higher resolution and metric aggregation. No additional software needs to be installed.

2. Monitor Other AWS Resources - Monitor metrics on Amazon DynamoDB tables, Amazon EBS volumes, Amazon RDS DB instances, Amazon Elastic MapReduce job flows, Elastic Load Balancers, Amazon SQS queues, Amazon SNS topics, and more for no additional charge. No additional software needs to be installed.

3. Monitor Custom Metrics - Submit Custom Metrics generated by your own applications via a simple API request and have them monitored by Amazon CloudWatch.

4. Monitor and Store Logs - You can use CloudWatch Logs to monitor and troubleshoot your systems and applications using your existing system, application, and custom log files. You can send your existing system, application, and custom log files to CloudWatch Logs and monitor these logs in near real-time. This can help you better understand and operate your systems and applications, and you can store your logs using highly durable, low-cost storage for later access.

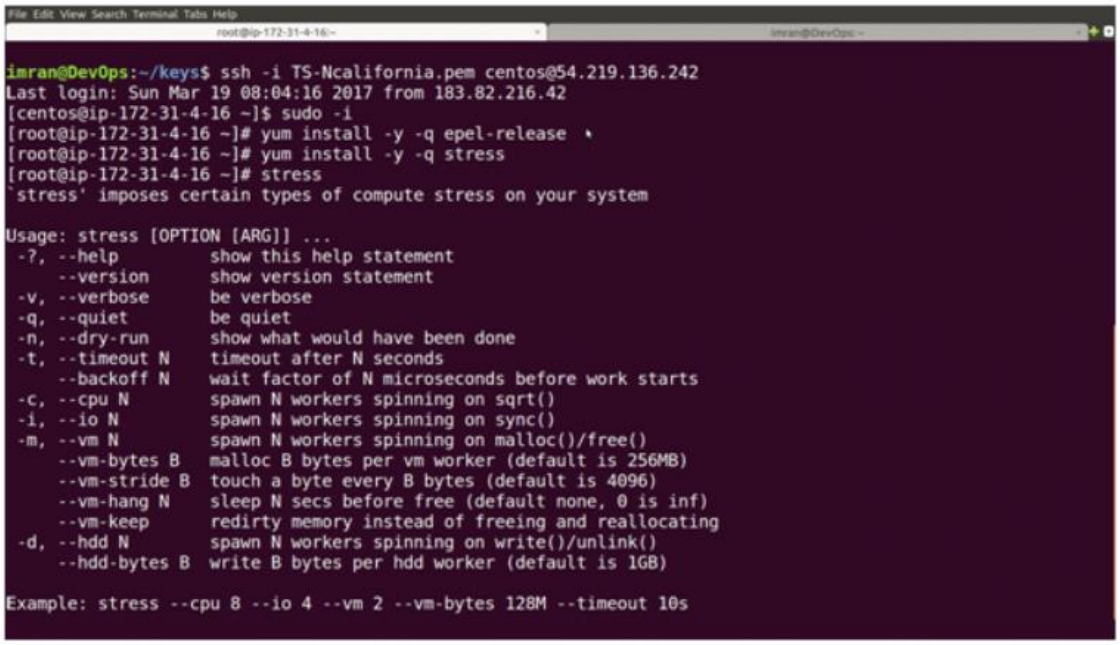
5. Set Alarms - Set alarms on any of your metrics to send you notifications or take other automated actions. For example, when a specific Amazon EC2 metric crosses your alarm threshold, you can use Auto Scaling to dynamically add or remove EC2 instances or send you a notification.

6. View Graphs and Statistics - Amazon Cloudwatch Dashboards enable you to create re-usable graphs of AWS resources and custom metrics so you can quickly monitor operational status and identify issues at a glance.

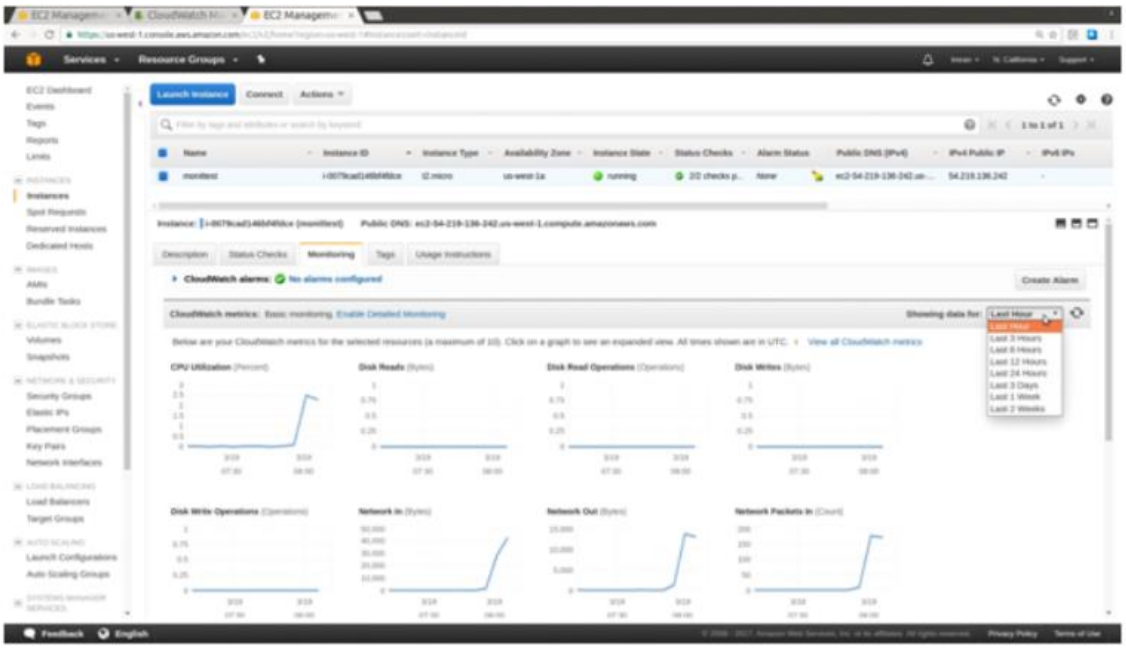
7. Monitor and React to ResourceChanges - CloudWatch Events provides a stream of events describing changes to your AWS resources. You can easily build workflows that automatically take actions you define, such as invoking an AWS Lambda function, when an event of interest occurs.

Monitoring AWS EC2 Instance CPU Utilization:

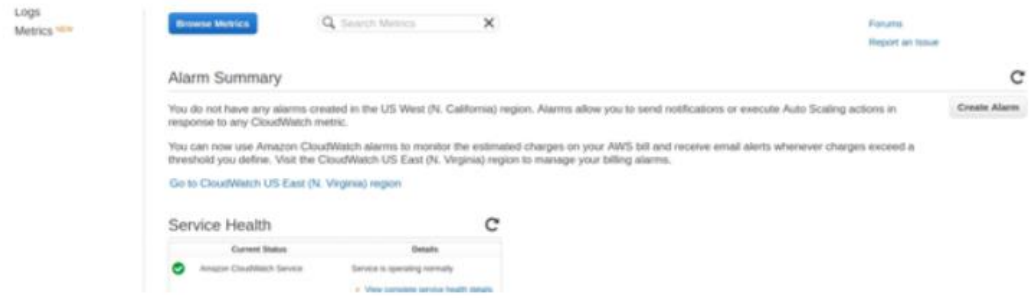
In order to monitor CPU Utilization we need an ec2 instance where we will cause high cpu load and test. I have created a centos 6 ec2 instance with tag Name: monittest and login to the instance. For testing purpose Install stress tool, which can create a high CPU usage.



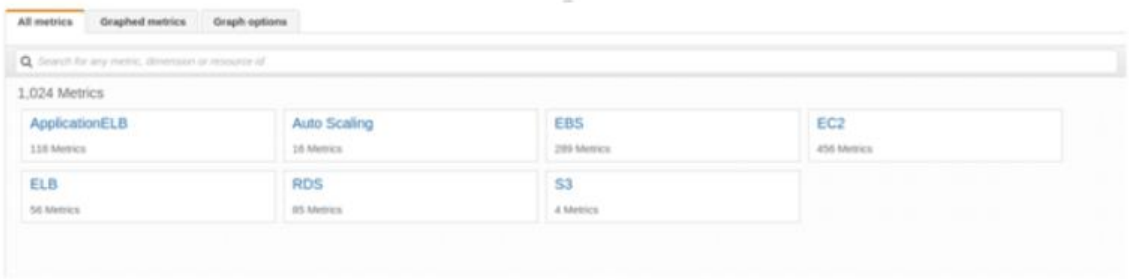
We can check lots of metrics by selecting the instance monitoring tab. All those graphs for CPU, Disk & network usage collected by Cloudwatch monitoring tool by default. You can see data from last hour to last two weeks.



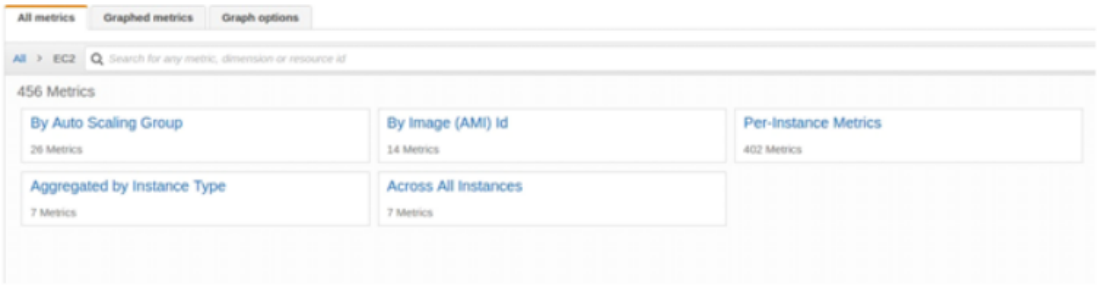
Click on CloudWatch service from AWS main dashboard. Go to Metrics



You can see some AWS services available here. Click on Ec2

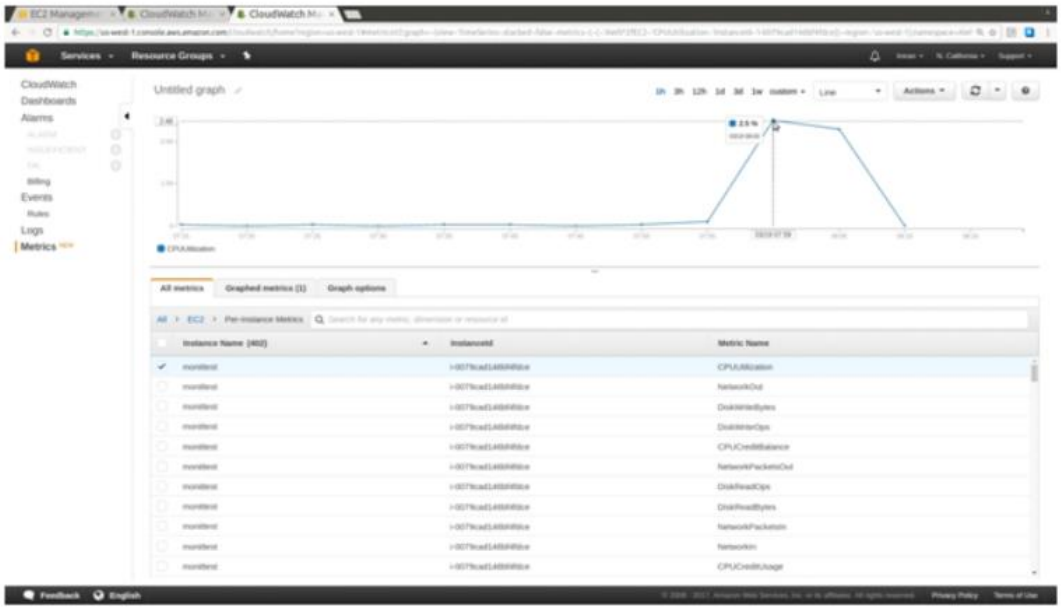


There are some metrics available in EC2. Select Per-Instance metrics.



You can see the instance which we created along with the metrics attached to it.

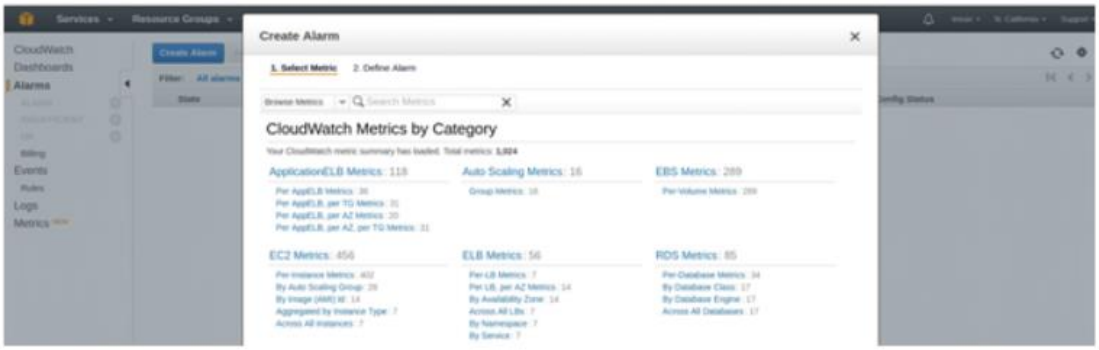
Select the instance monittest with CPU utilization

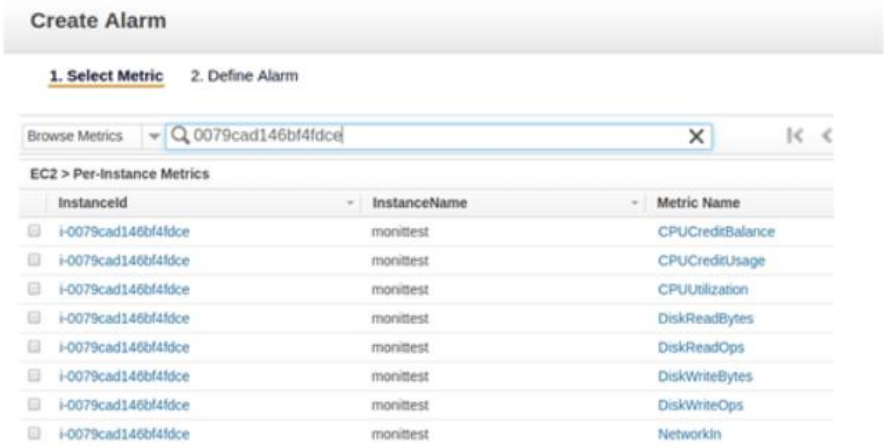


As you can see above for instance monittestits showing cpu utilization graph for last 1 hour. We can customize the graph display. Click custom and select relative or absolute time to see graph details as per your wish.

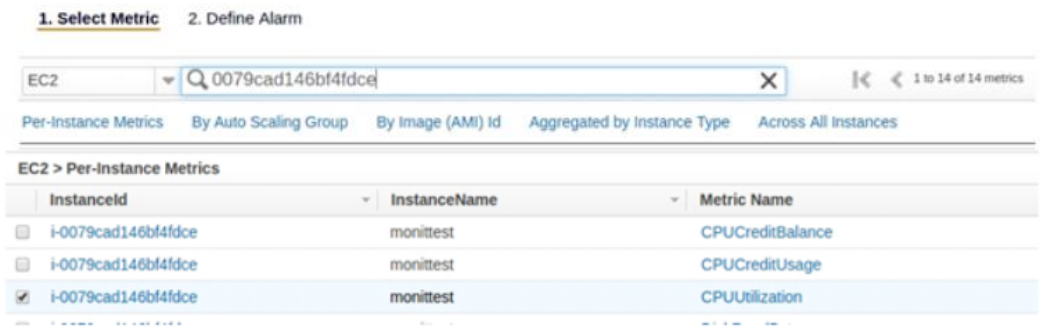
Setting up Alarm.

Click on Alarms and select Create Alarm.

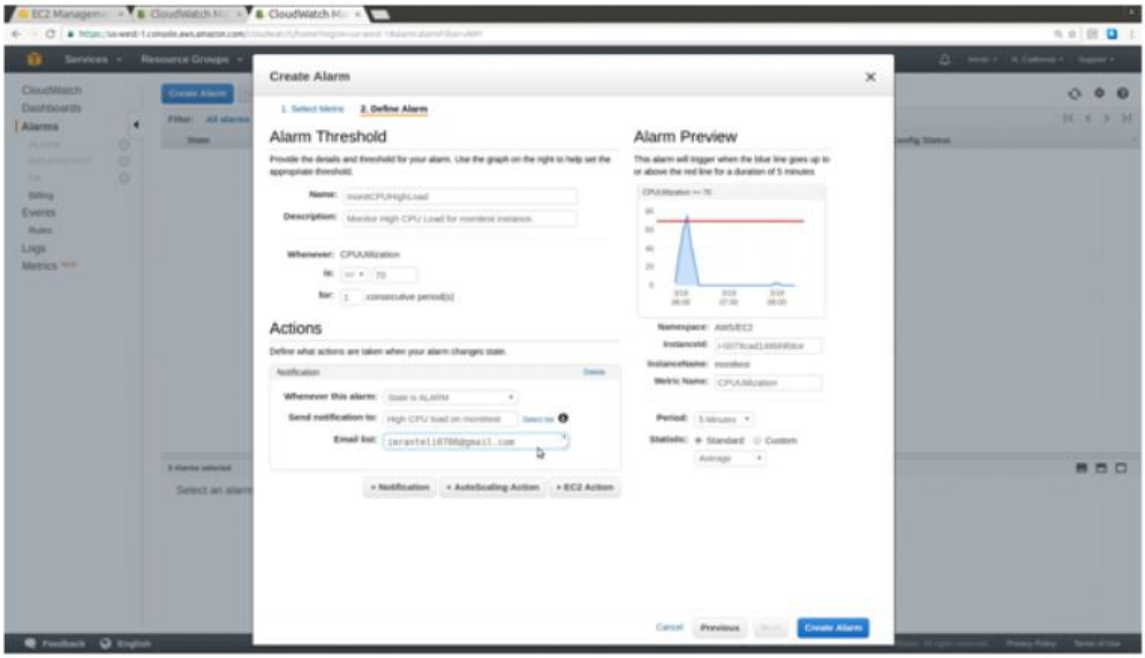




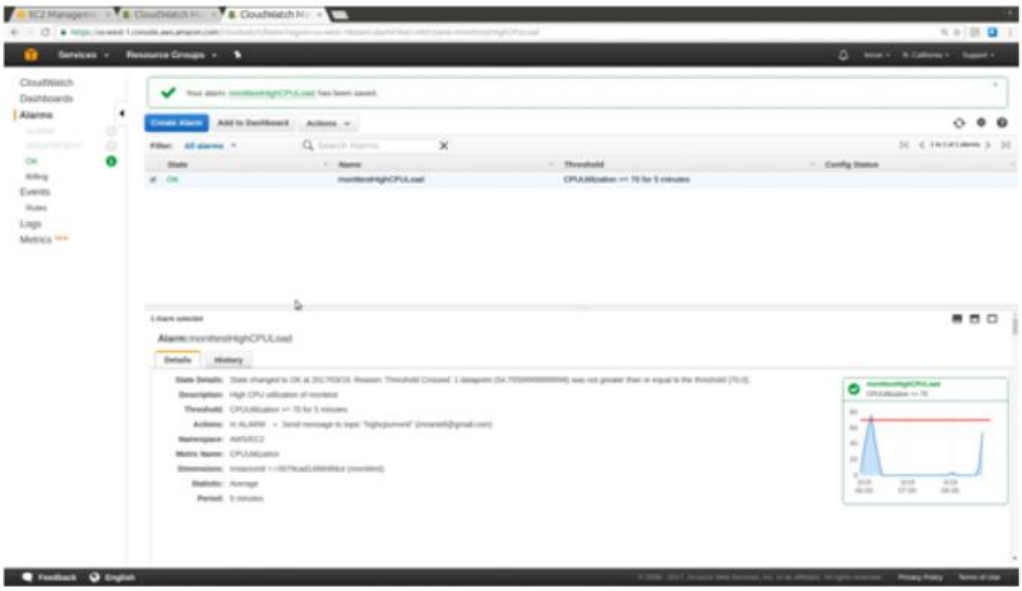
Search with instance ID to find all the metrics related to our instance. Put a check mark on CPU Utilization against monittest



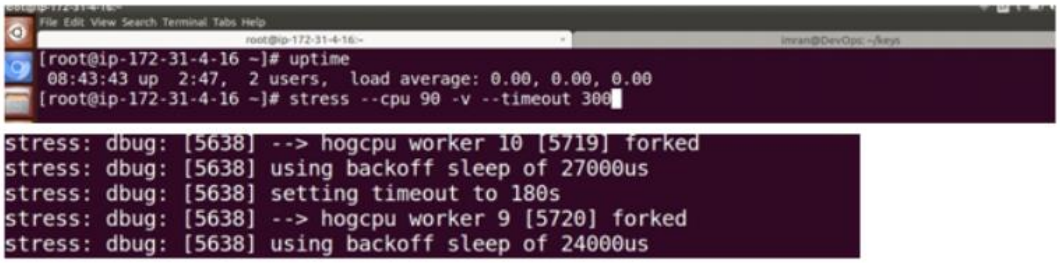
Click on next. You have to give some details related to threshold and provide some name and description to alarm. In the actions section you have to specify the email id to which the notification to be received. Click Create Alarm, once done.



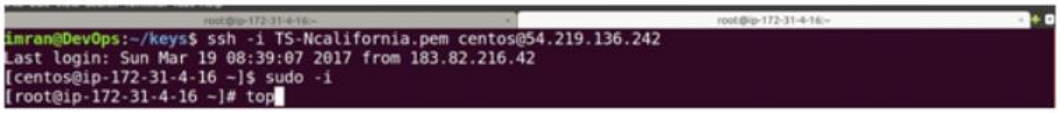
You will get an email from AWS for verifying email address, once its verified you will start receiving email alert whenever the instance cpu load crosses beyond 70 %. Alarm has been created successfully for monittest instance.



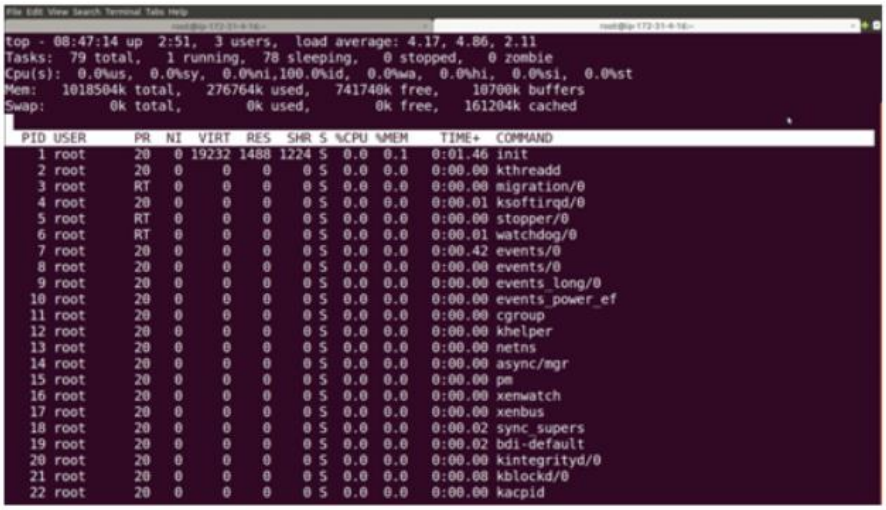
To test that we can use stress utility and cause high cpu on our instance. For this follow the commands shown in the below screenshot.



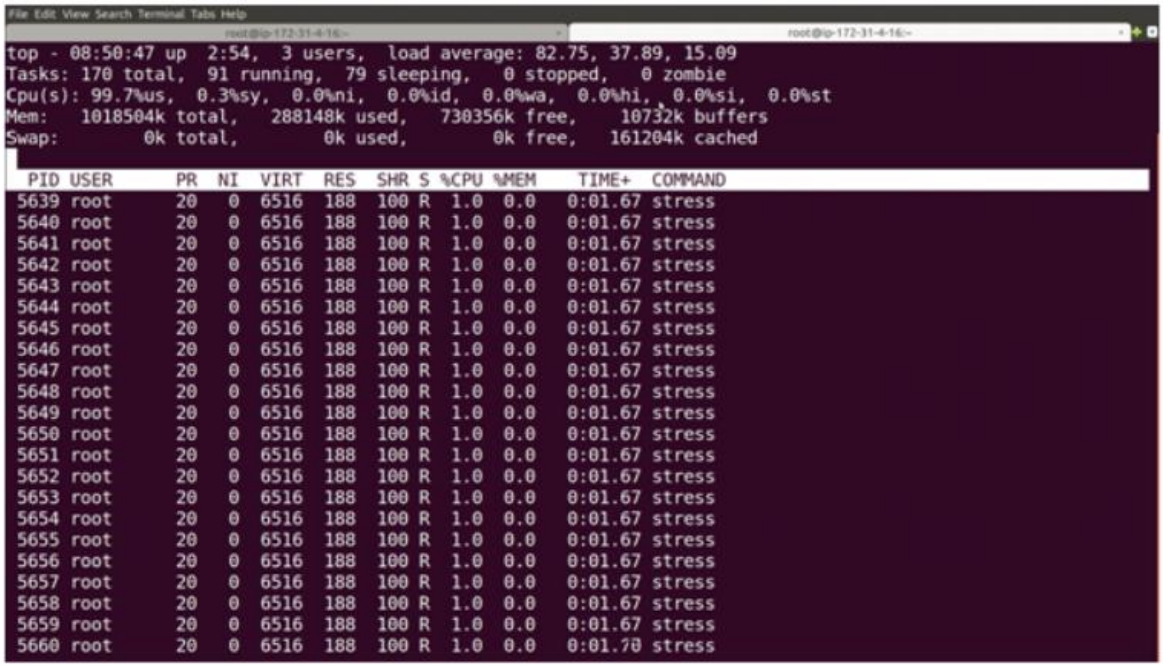
Login from another tab and run top command to see it real time from the system.



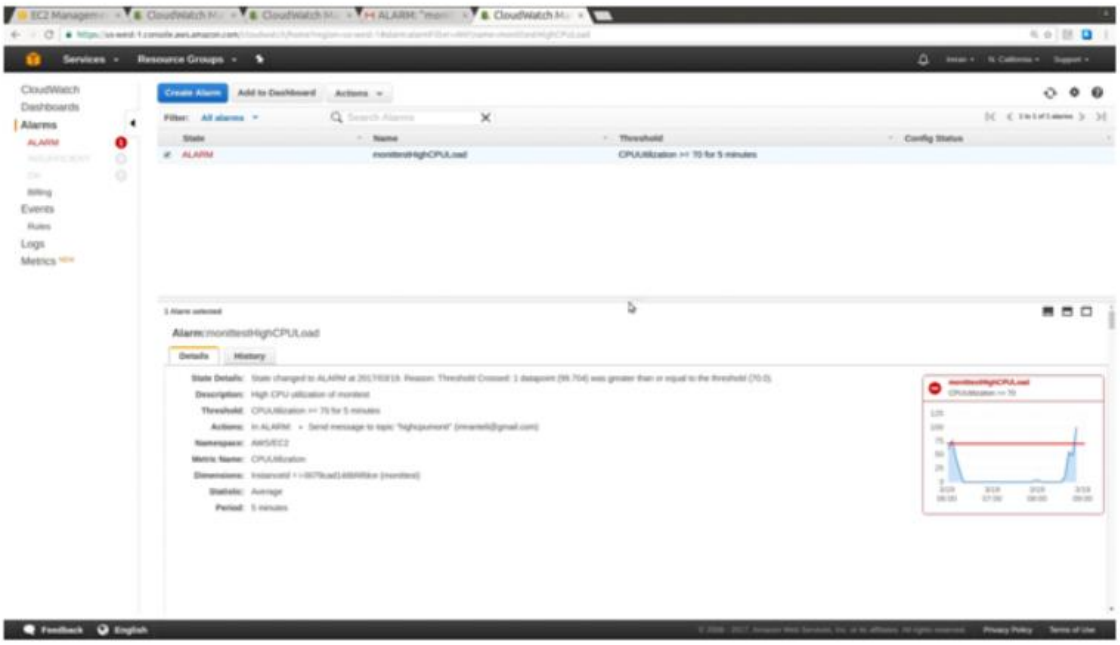
Top command shows the current load average, running, sleeping and stopped processes of the system.



Observe the load average, if it is above 80 then all the processes are stressed.



After few minutes you will receive an email in your inbox which shows Alarm and graph in red colour. It also gives an indication on alarm dashboard that the CPU utilization >=70 for 5 minutes.



Email from AWS

ALARM: "monittestHighCPULoad" in US West - N. California

AWS Notifications no-reply@sns.amazonaws.com via amazonses.com

to me

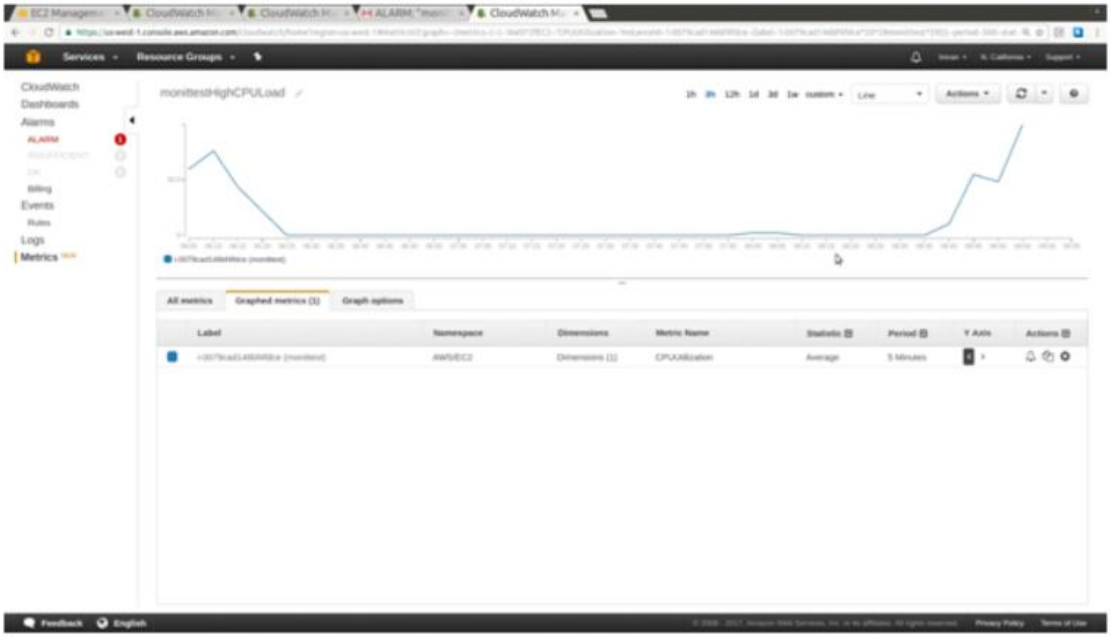
You are receiving this email because your Amazon CloudWatch Alarm "monittes
California region has entered the ALARM state, because "Threshold Crossed: 1
equal to the threshold (70.0)." at "Sunday 19 March, 2017 09:01:21 UTC".

View this alarm in the AWS Management Console:
https://console.aws.amazon.com/cloudwatch/home?region=us-west-1#s=Alarm

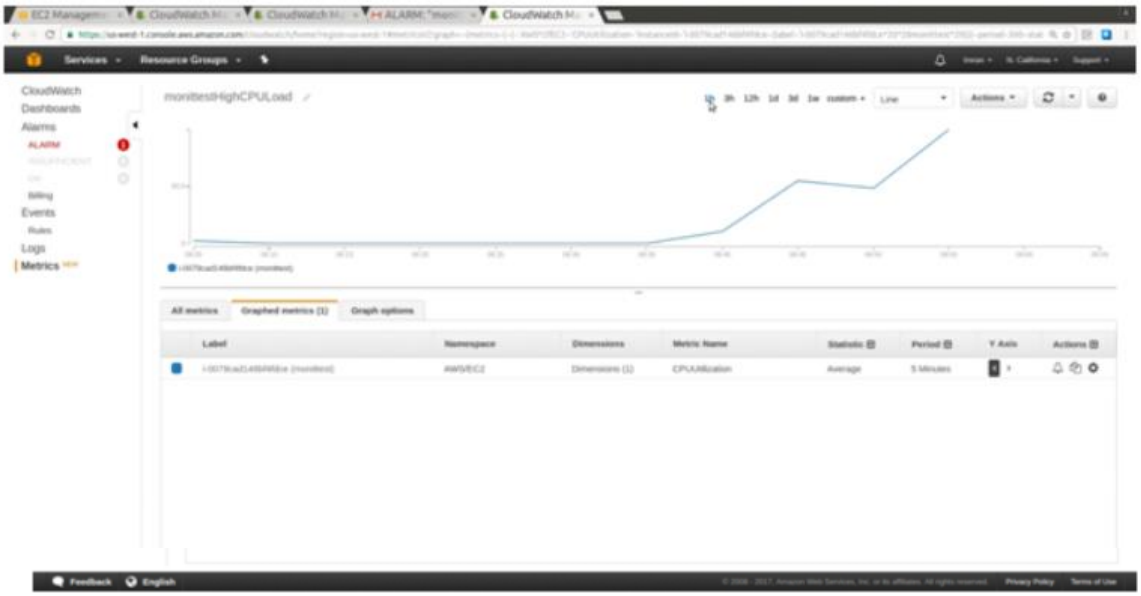
Alarm Details:
- Name: monittestHighCPULoad
- Description: High CPU utilization of monitest
- State Change: OK -> ALARM
- Reason for State Change: Threshold Crossed: 1 datapoint (99.704) was gre
(70.0).
- Timestamp: Sunday 19 March, 2017 09:01:21 UTC
- AWS Account: 171225278948

Threshold:
- The alarm is in the ALARM state when the metric is GreaterThanOrEqualToTh

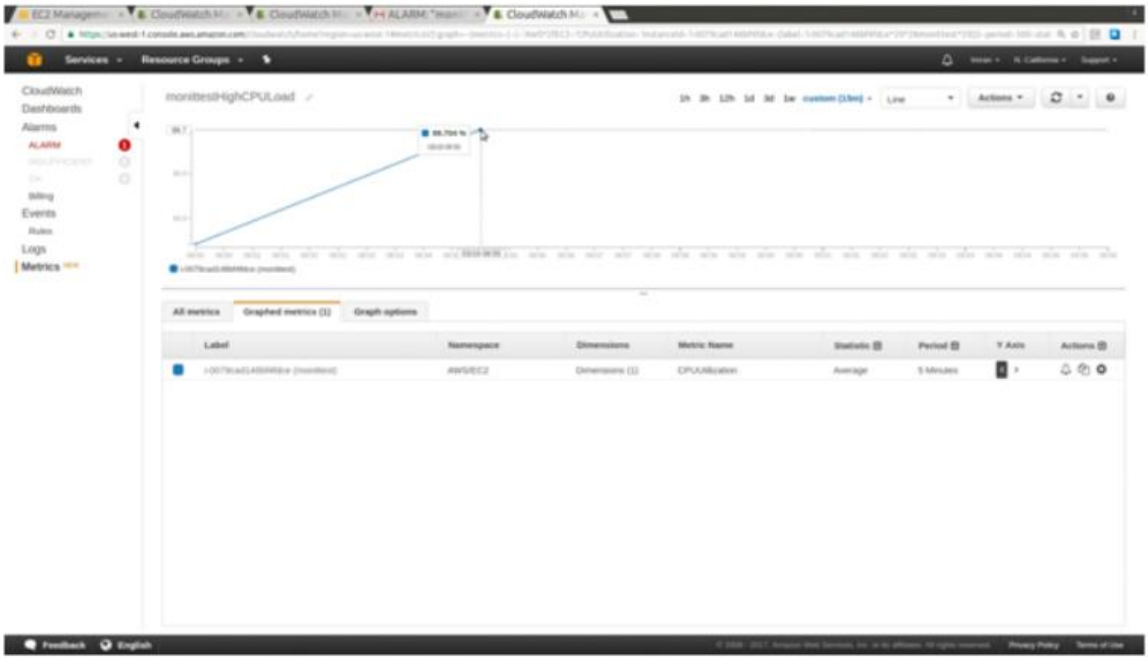
Graph for our CPU Utilization last 3 hours.



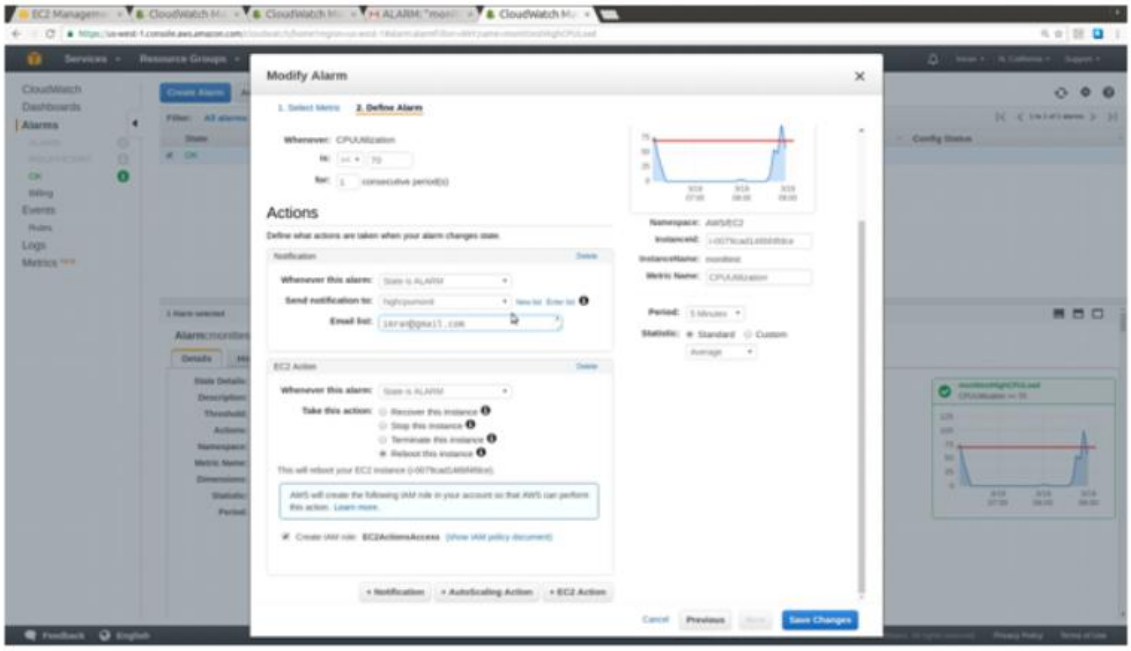
Graph for our CPU Utilization last 1 hour.



Graph for our CPU Utilization last 15 minues.

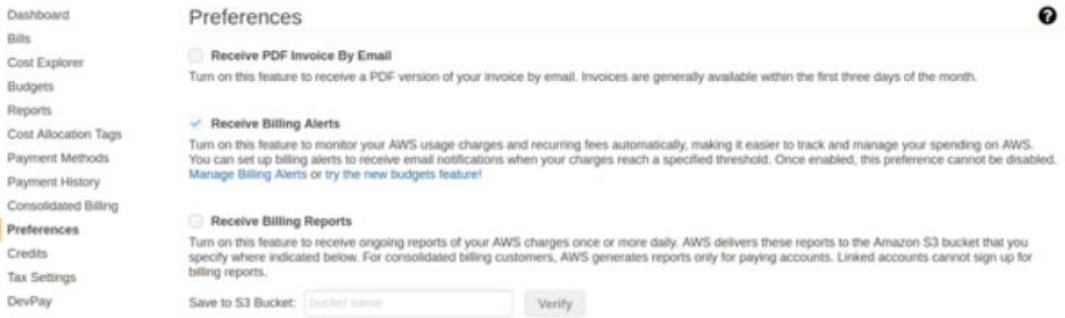


You can also change/add the actions of alarms like EC2 actions to stop/reboot/terminate instance.



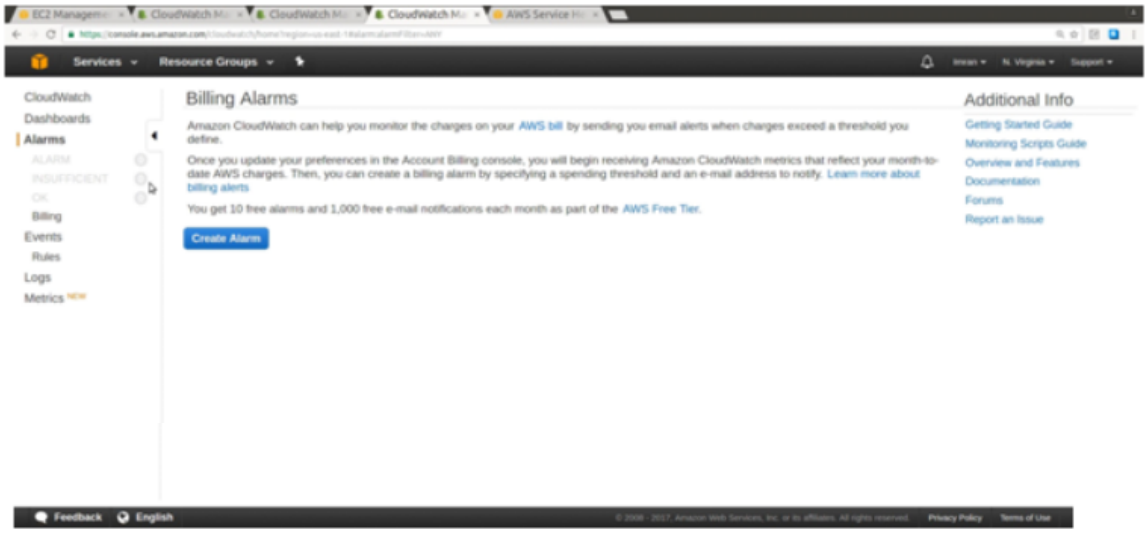
Monitoring AWS Billing:

We can also monitor our account's estimated costs and usage by setting up an alarm. Go to My Billing Dashboard ==> Preferences ==> Put a check mark on Receive billing alerts. Click on the Save Preferences

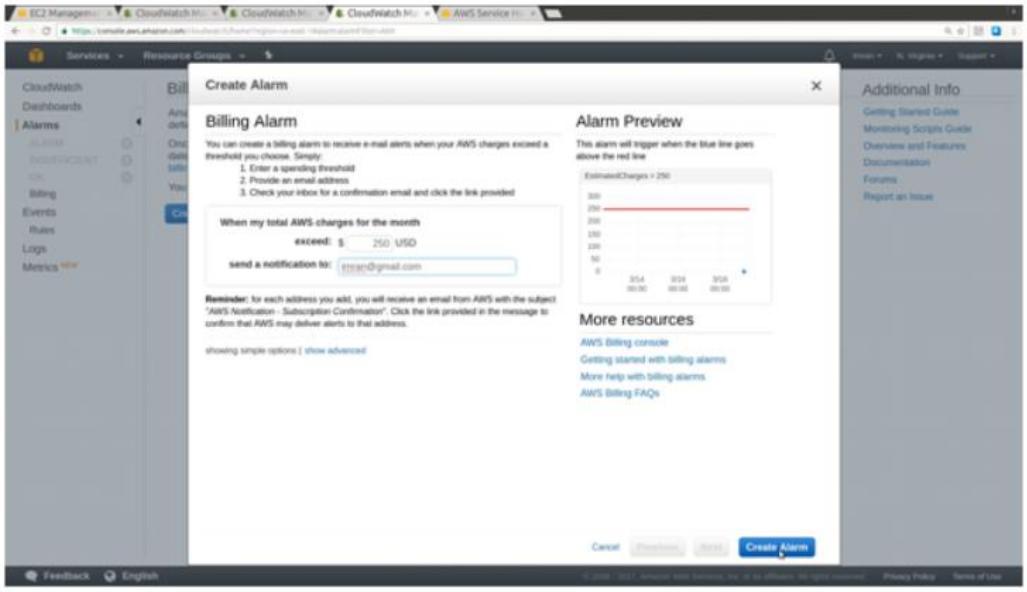


Save preferences

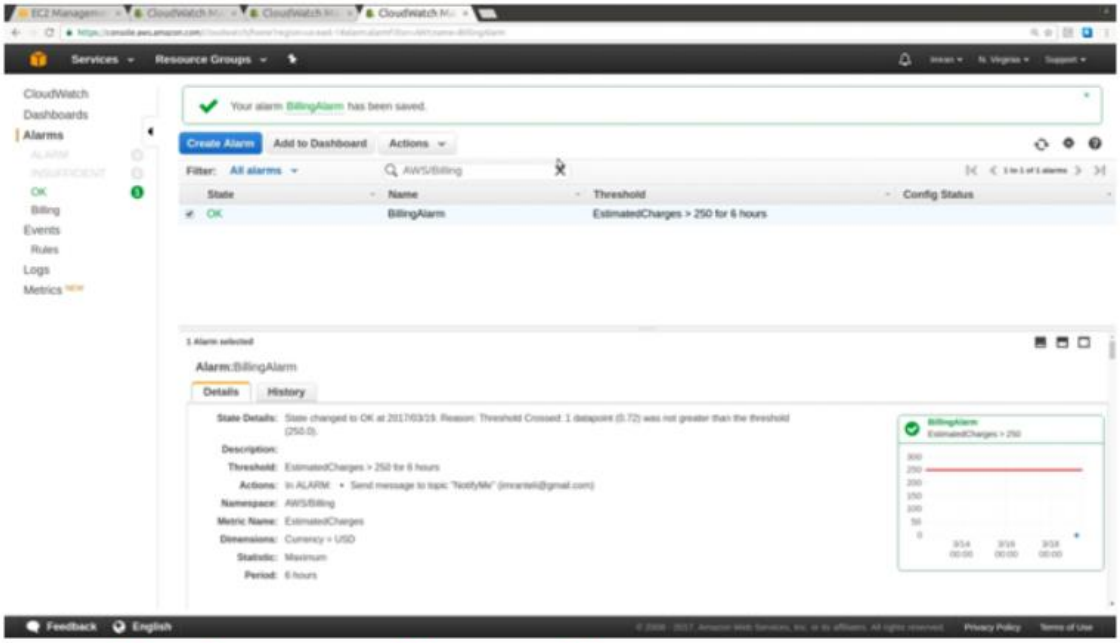
Go to CloudWatch service, Click on Billing and select create alarm.



Enter the threshold, if your AWS account charges exceeds than the threshold value you will receive an email which you will provide in the send notification. Click on create alarm.



View Alarm: your alarm has been successfully created for billing section of AWS.

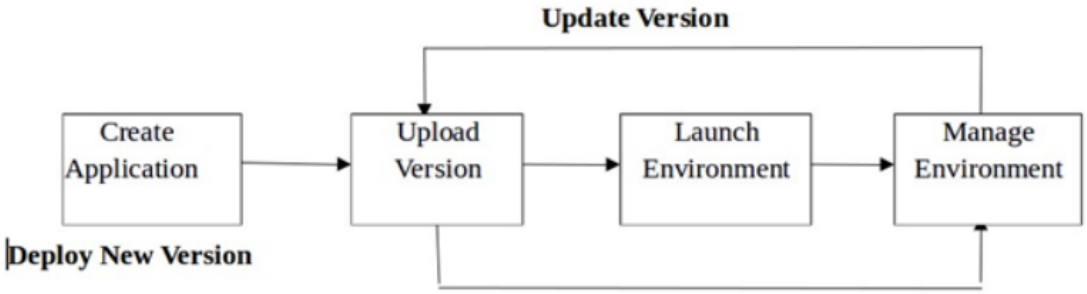


AWS Elastic Beanstalk + Jenkins

AWS Elastic Beanstalk is a cloud deployment and provisioning service that automates the process of getting applications set up on the Amazon Web Services (AWS) infrastructure. It is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS.

Workflow Of Beanstalk:

Create an application, upload an application version in the form of an application source bundle (for example, a Java .war file) to Elastic Beanstalk, and then provide some information about the application. Elastic Beanstalk automatically launches an environment and creates and configures the AWS resources needed to run your code. After your environment is launched, you can then manage your environment and deploy new application versions. The following diagram illustrates the workflow of Elastic Beanstalk Create Tomcat platform in Beanstalk.



Benefits Of Elastic Beanstalk:

1.Fast and simple to begin

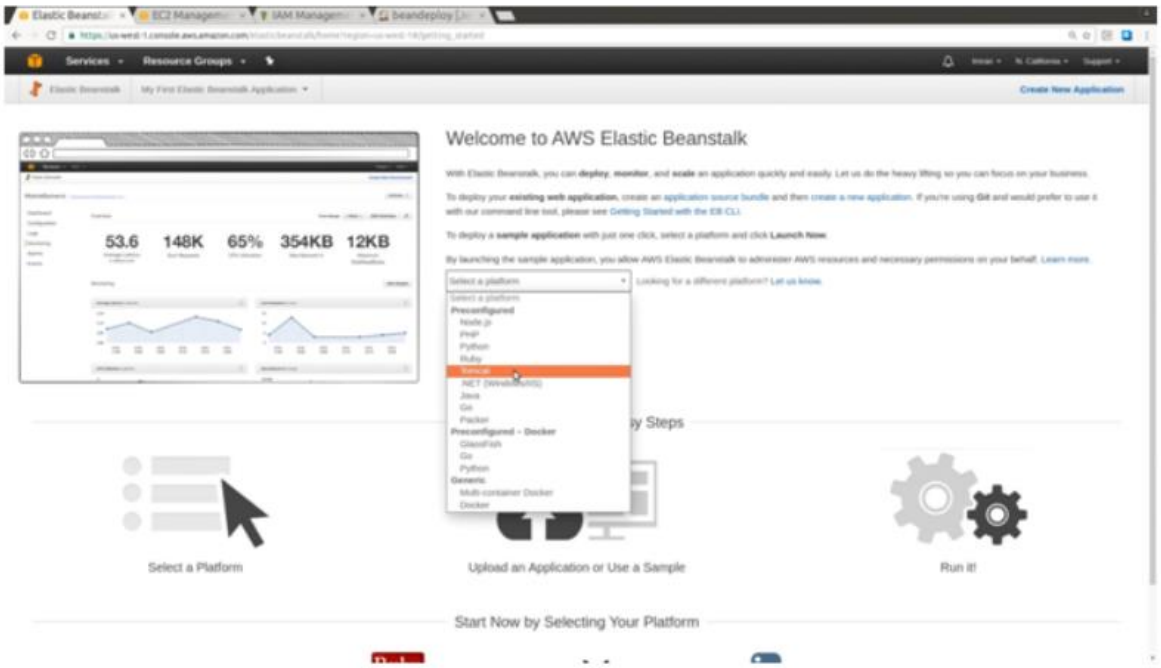
2.Developer productivity

3.Impossible to outgrow

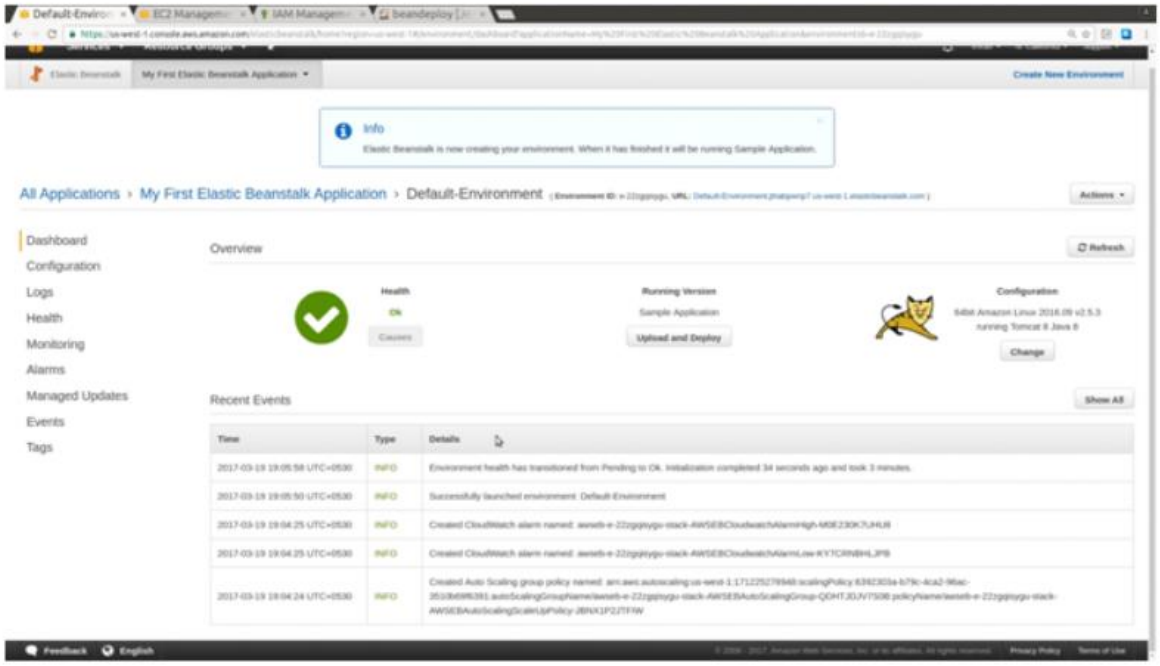
4.Complete resource control

Create An Application:

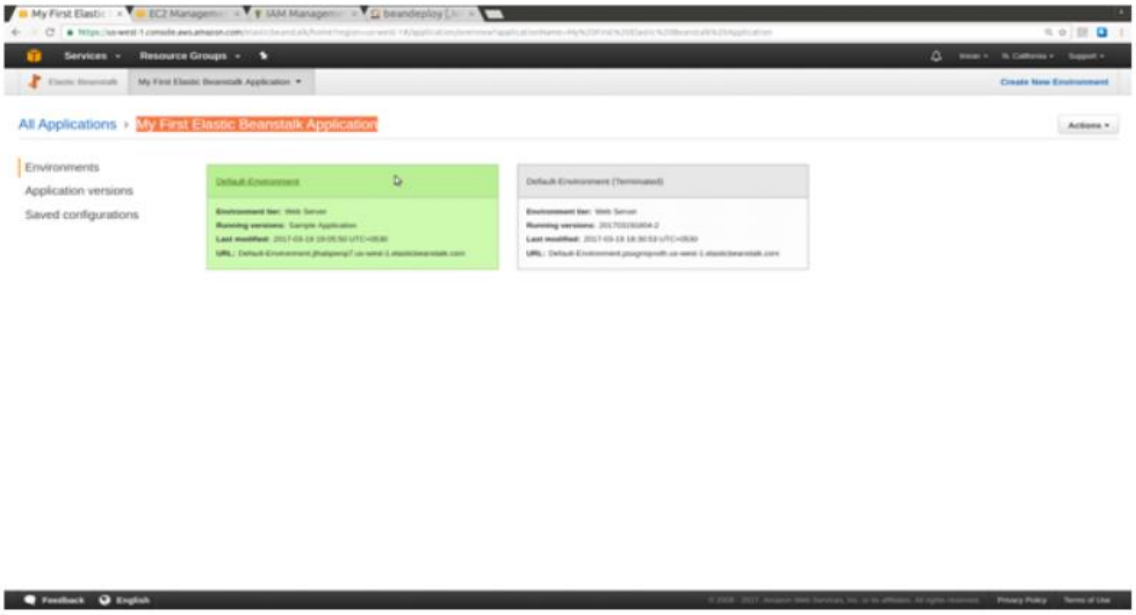
Open Elastic Beanstalk Management console from AWS main page. Select the application in which you want to deploy.



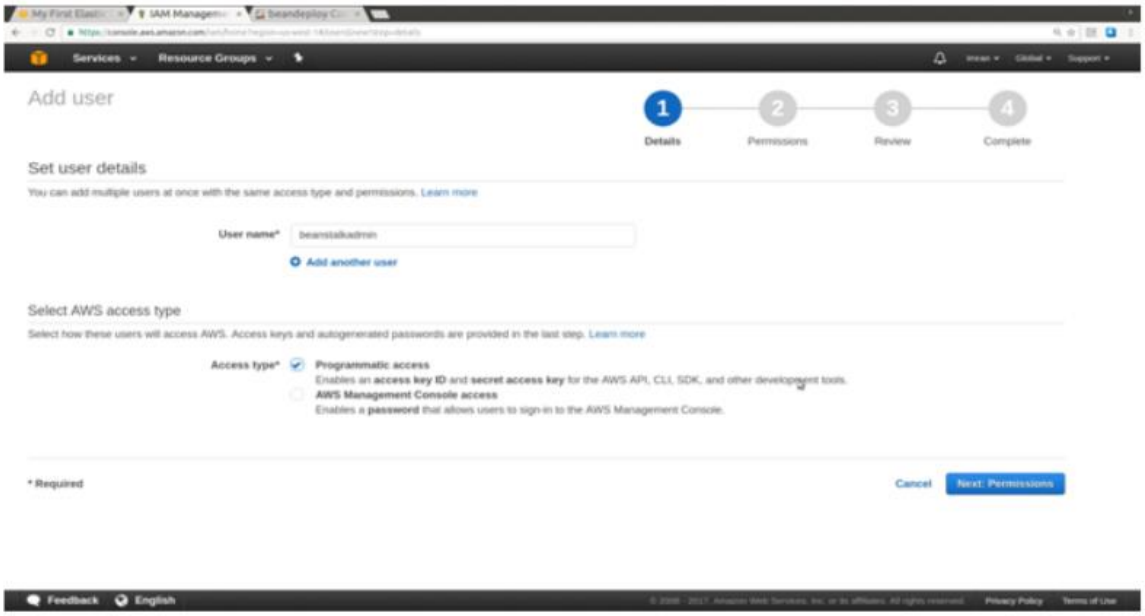
Here I selected tomcat application and click on Launch now. After creating the Elastic Beanstalk application, you can view information about the application you deployed and its provisioned resources by going to the environment dashboard in the AWS Management Console. The dashboard shows the health of your application's environment, the running version, and the environment configuration. While Elastic Beanstalk creates your AWS resources and launches your application, the environment will be in a Pending state.



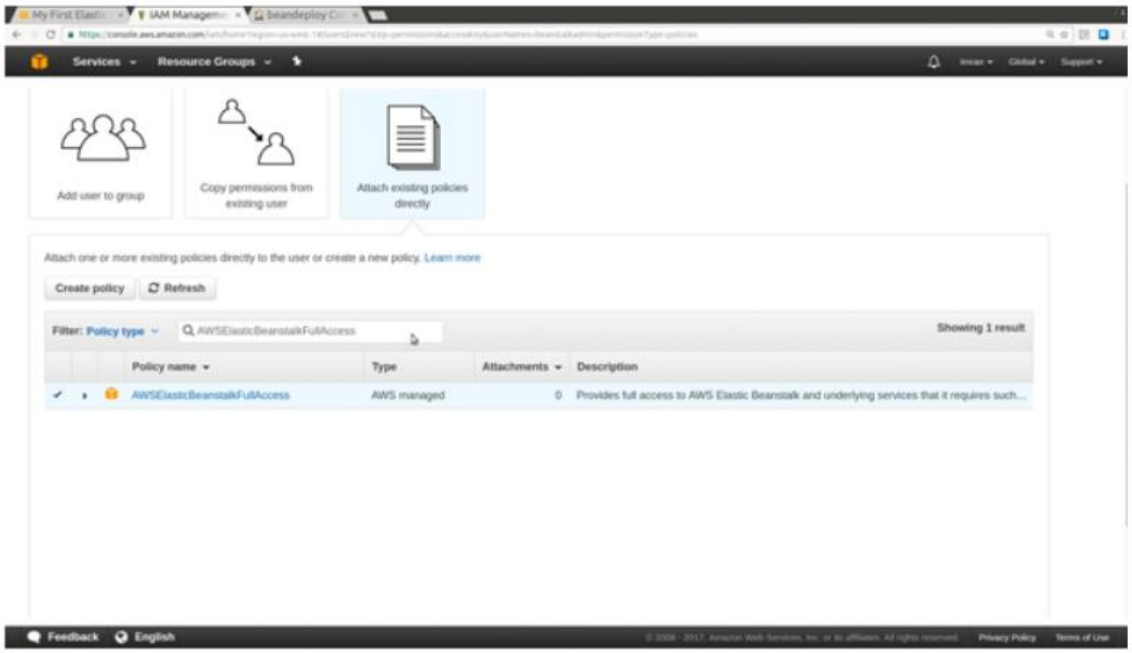
Click on My First Elastic Beanstalk Application in the environment's dashboard to see the status messages about launch events.



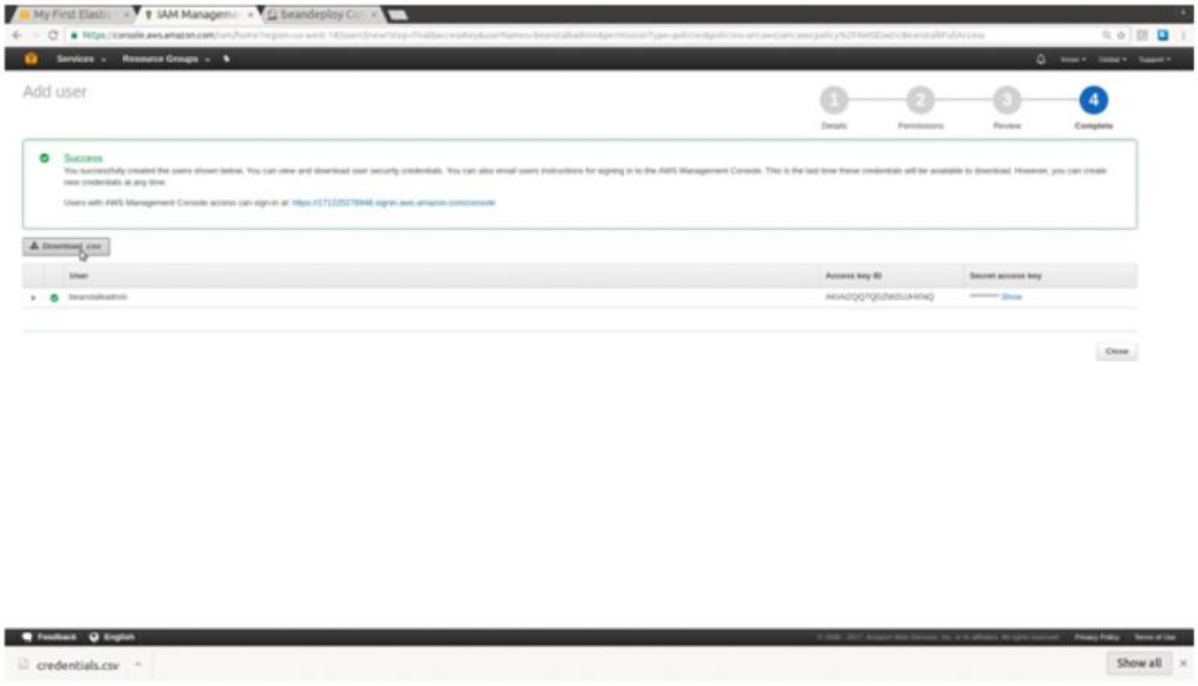
Every Elastic Beanstalk application can have multiple environments. As in real time we have Dev, QA, Staging and Prod etc.



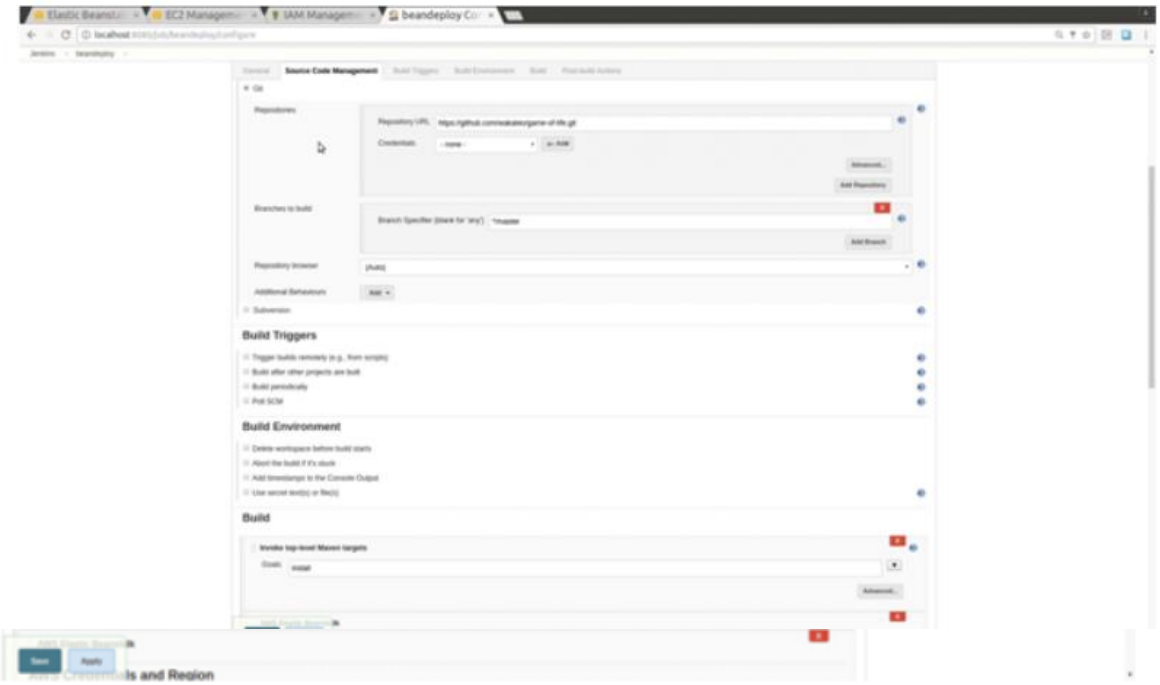
You will get an access key and secret access Ids key to access the account. Click on Next to attach permissions to the user. Attach AWSElasticBeanstalkFullAccess policy.



Download Access Keys Csv File:

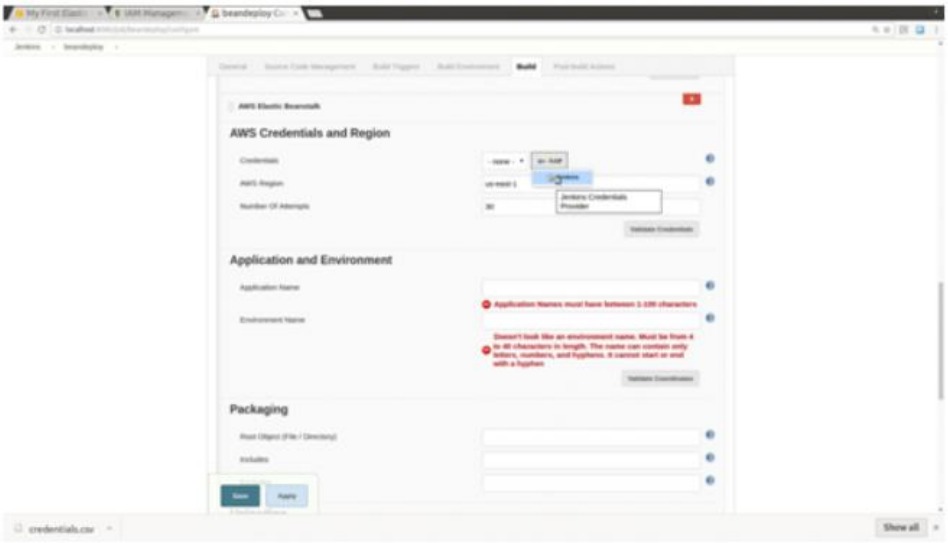


Configure Jenkins Job:



Install AWS Beanstalk Plugin In Jenkins And Configure Beanstalk Plugin In Add Build Step

Add credentials(keys) of the IAM user.



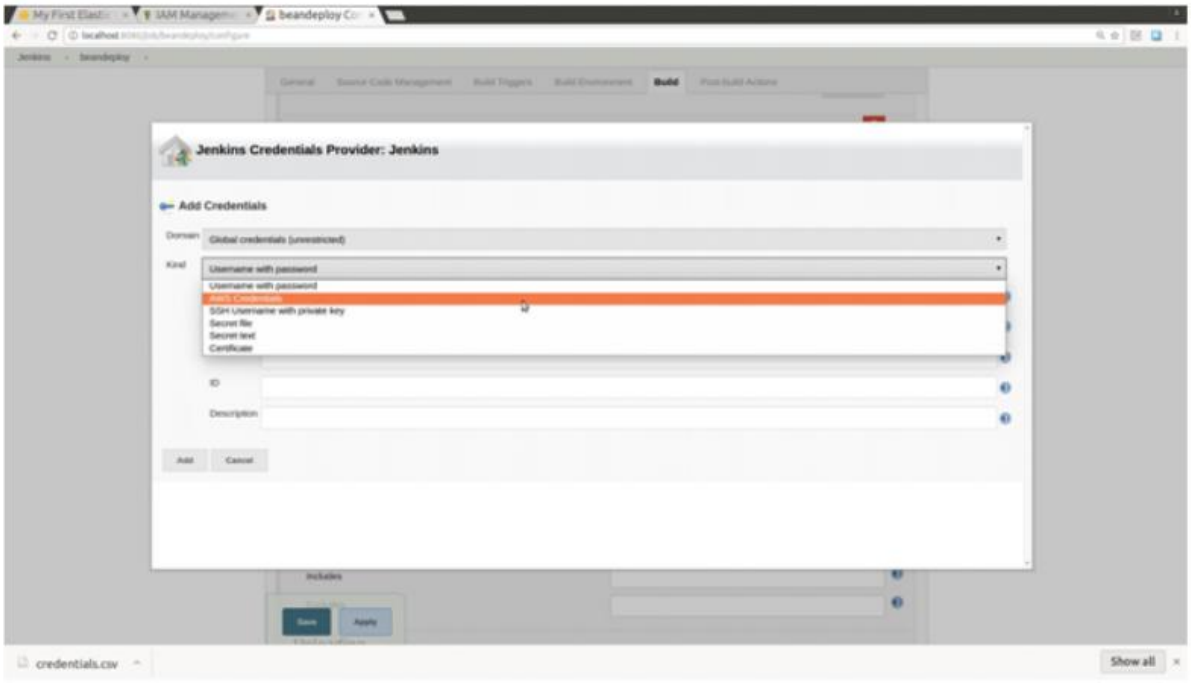
Check the credentials.csv file from the command line as shown below Access key highlighted below.

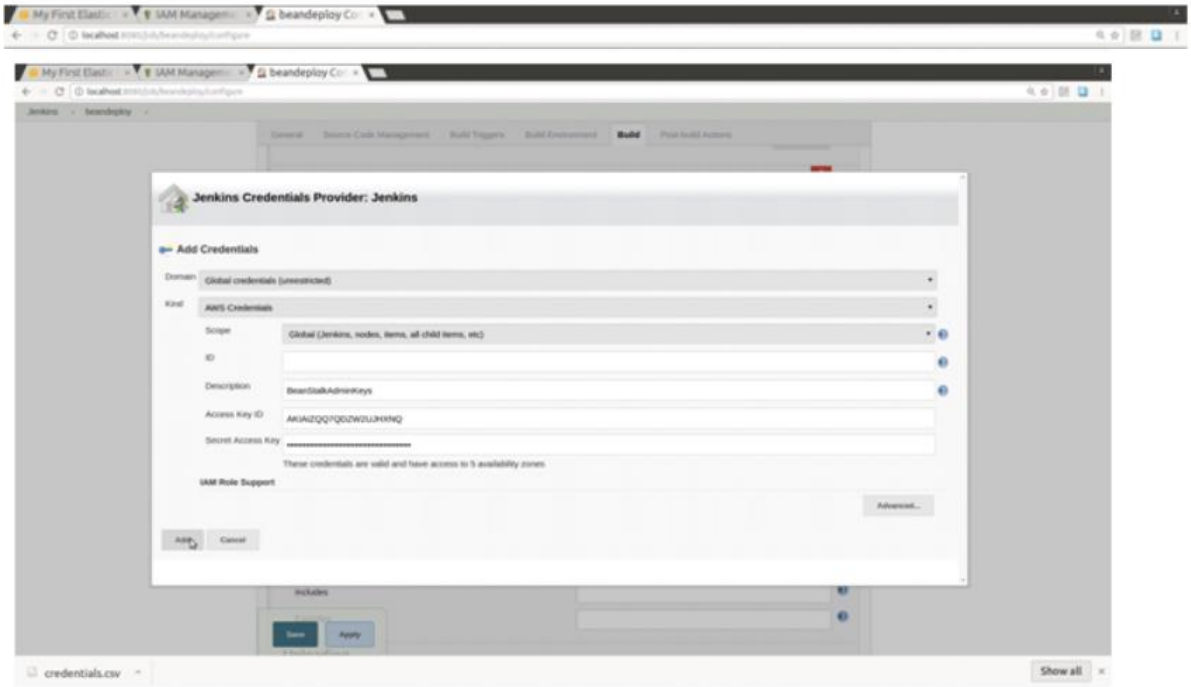
```
imran@DevOps:~$ cd Downloads/
imran@DevOps:~/Downloads$ cat credentials.csv
User name,Password,Access key ID,Secret access key,Console login link
beanstalkadmin,,AKIAIZQQ7QDZW2UJHXNQ,KXzgqn0L76qc4siJoRDPMT5Lxrw0dGeA5fbz9QuA,https://171225278948.signin.aw
s.amazon.com/console
```

Security Key hightlighted below.

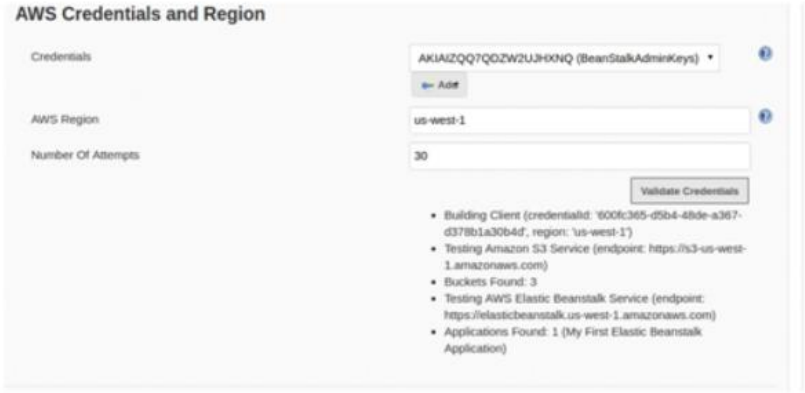
```
imran@DevOps:~$ cd Downloads/
imran@DevOps:~/Downloads$ cat credentials.csv
User name,Password,Access key ID,Secret access key,Console login link
beanstalkadmin,,AKIAIZQQ7QDZW2UJHXNQ,KXzgqn0L76qc4siJoRDPMT5Lxrw0dGeA5fbz9QuA,https://171225278948.signin.aw
s.amazon.com/console
```

When you click on add for the credentials, you will see the Kind option. Select AWS Credentials then add the access & secret key and click on Add.


Jenkins Credentials Provider: Jenkins
Add Credentials
Domain
Global credentials (unrestricted)
Kind
Username with password
Username with password
AWS Credentials
SSH Username with private key
Secret file
Secret text
Certificate
ID
Description
Add
Cancel
credentials.csv
Show all

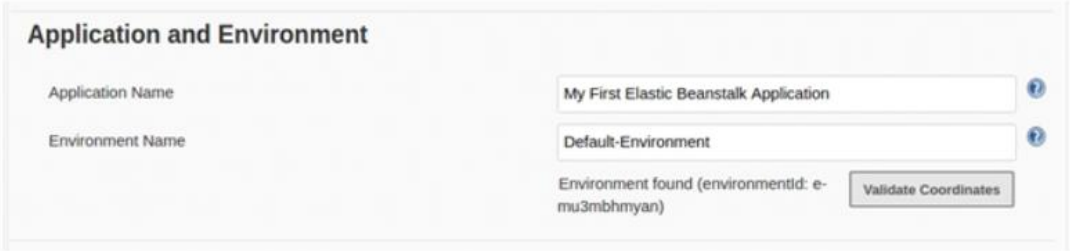

Jenkins Credentials Provider: Jenkins
Add Credentials
Domain
Global credentials (unrestricted)
Kind
AWS Credentials
Scope
Global (Jenkins, nodes, items, all child items, etc)
ID
Description
BeanStalkAdminKeys
Access Key ID
Secret Access Key
These credentials are valid and have access to 5 availability zones
IAM Role Support
Advanced...
Add
Cancel
credentials.csv
Show all

Validate the given credentials by selecting validate credentials while configuring AWS beanstalk plugin..

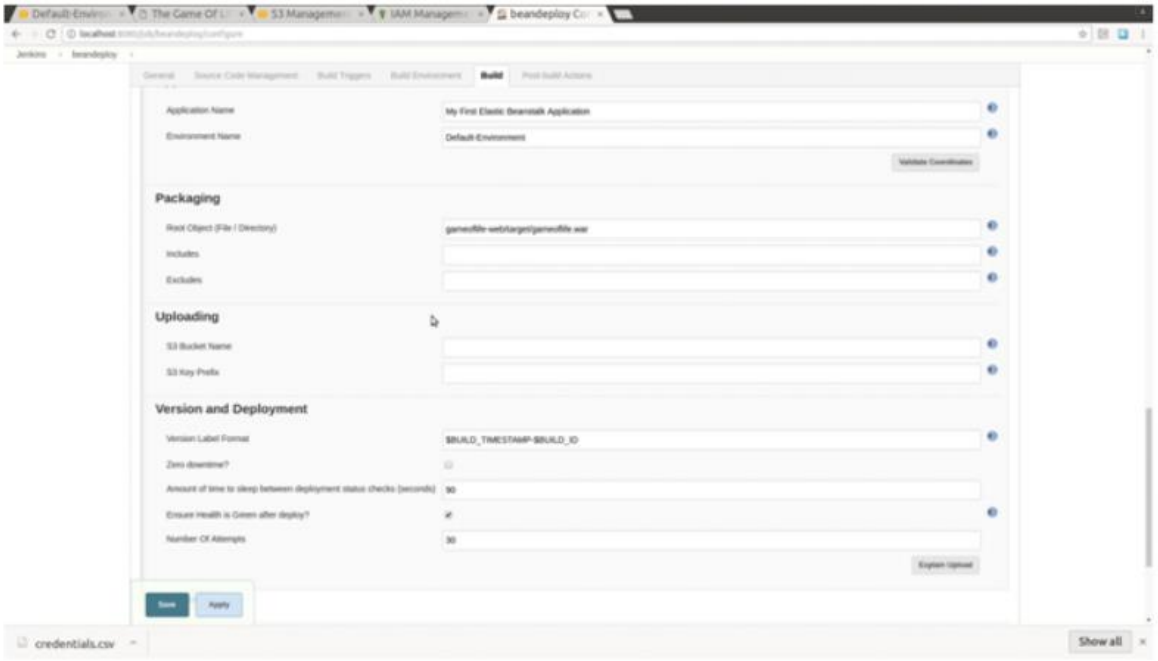


Give Application and environment name.

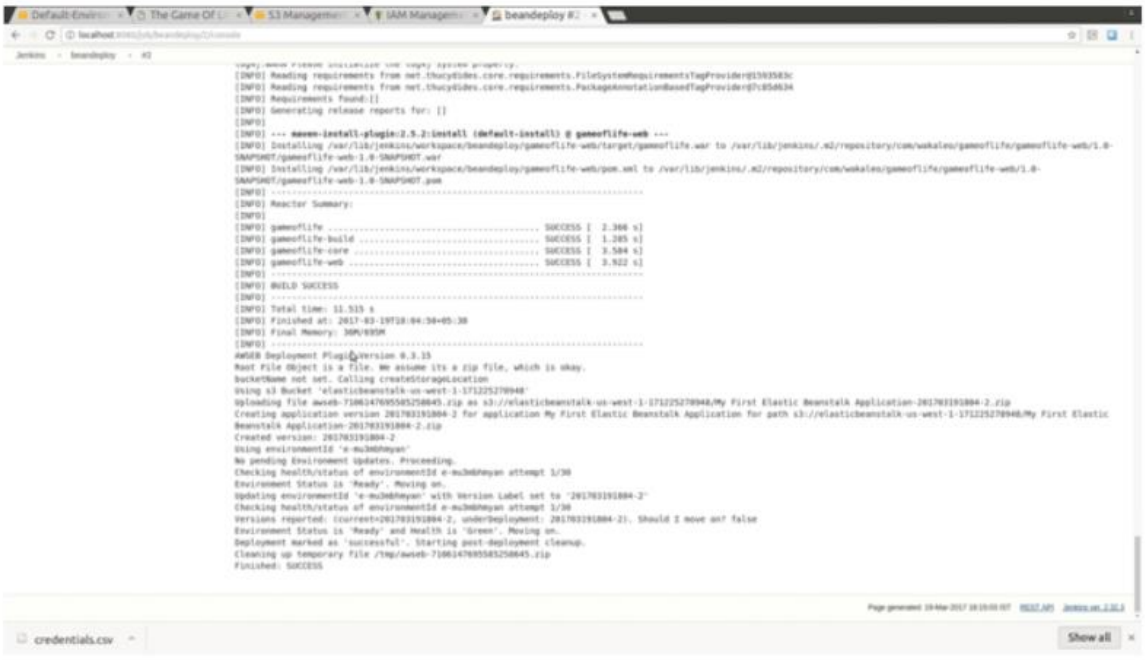
Check that information in Beanstalk.



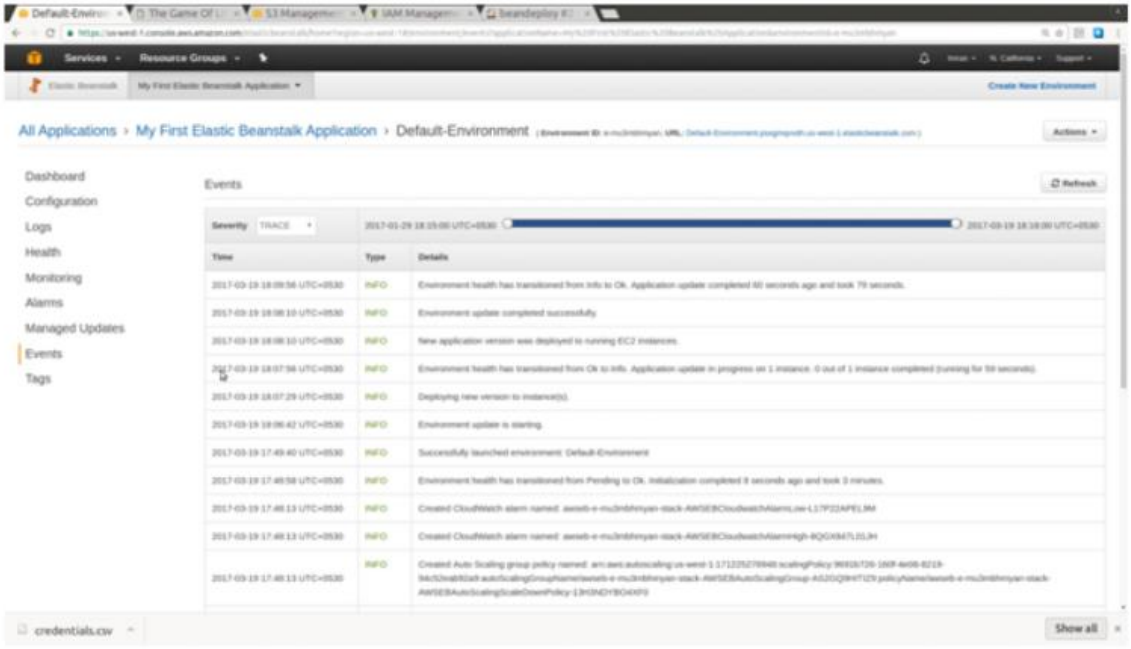
Root object should be the path of artifact located in workspace. Version we will keep BUILD_ID AND BUILD_TIMESTAMP same as the CD project.

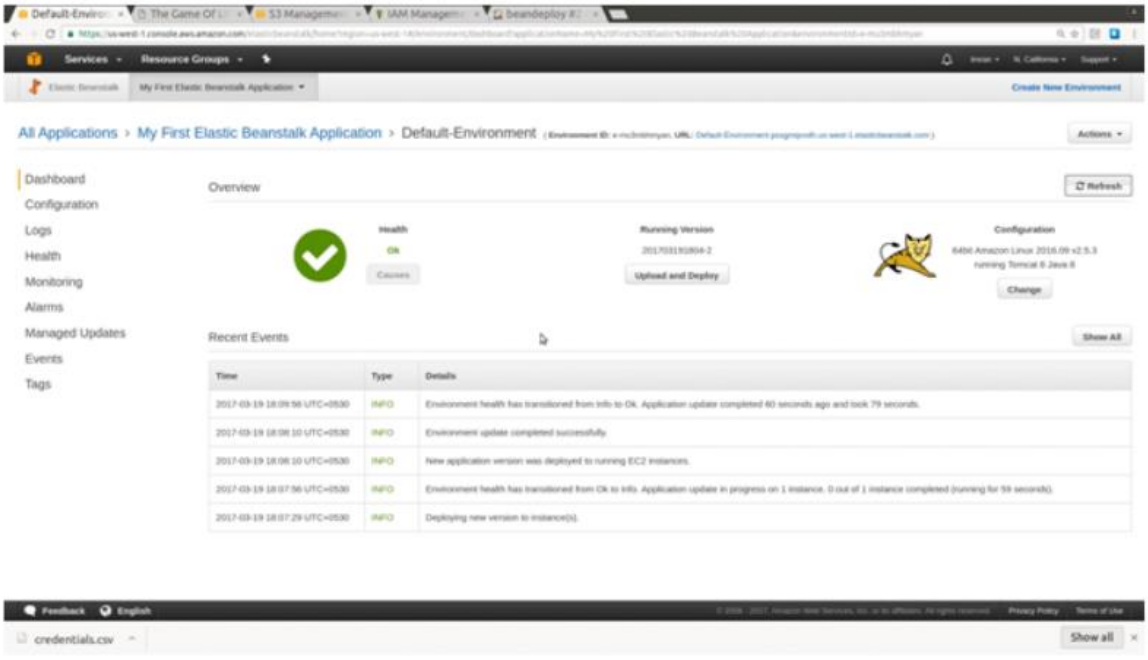


Run the job and check the console output.

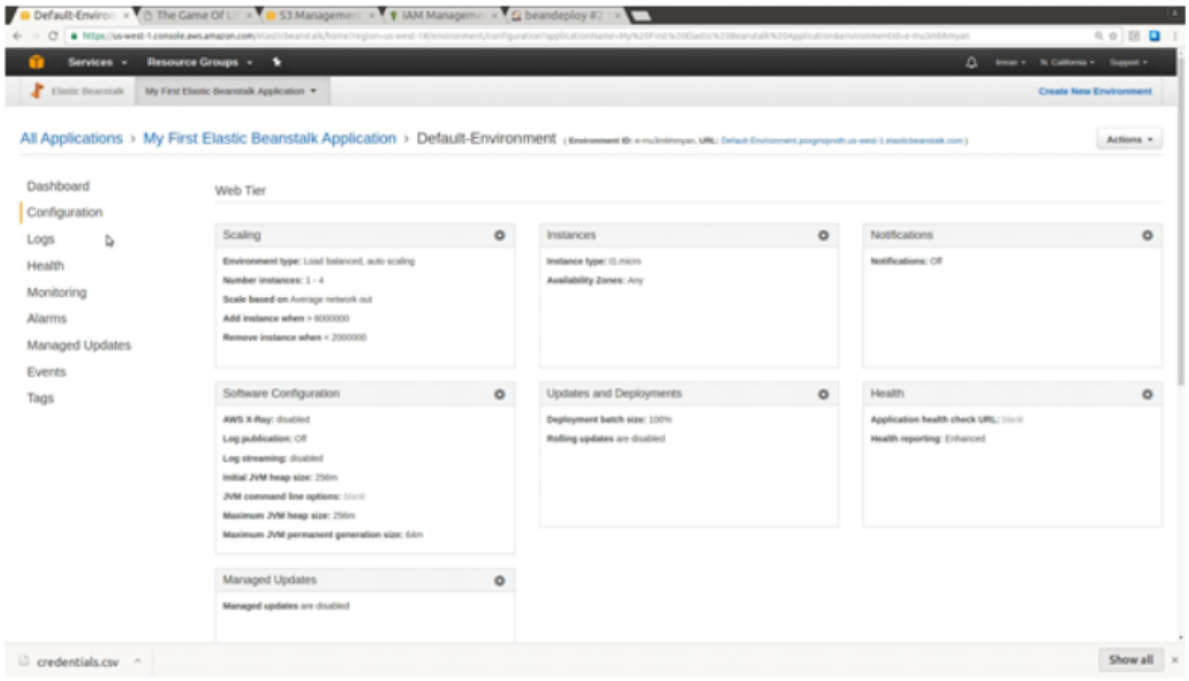


Verify the Environments Events.

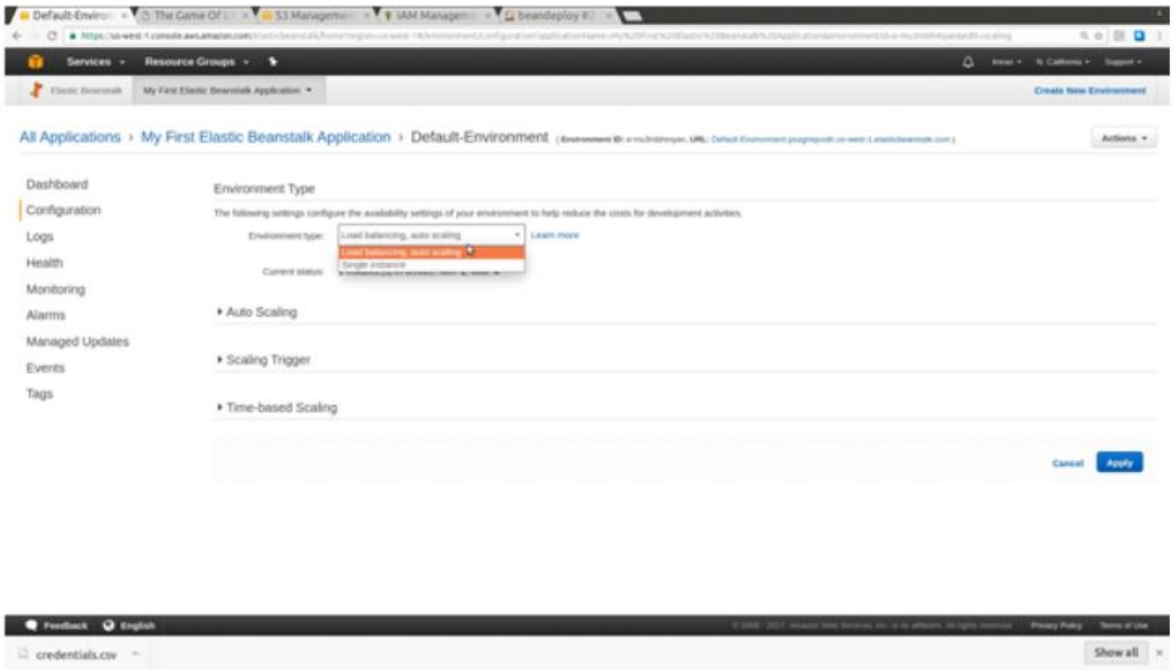




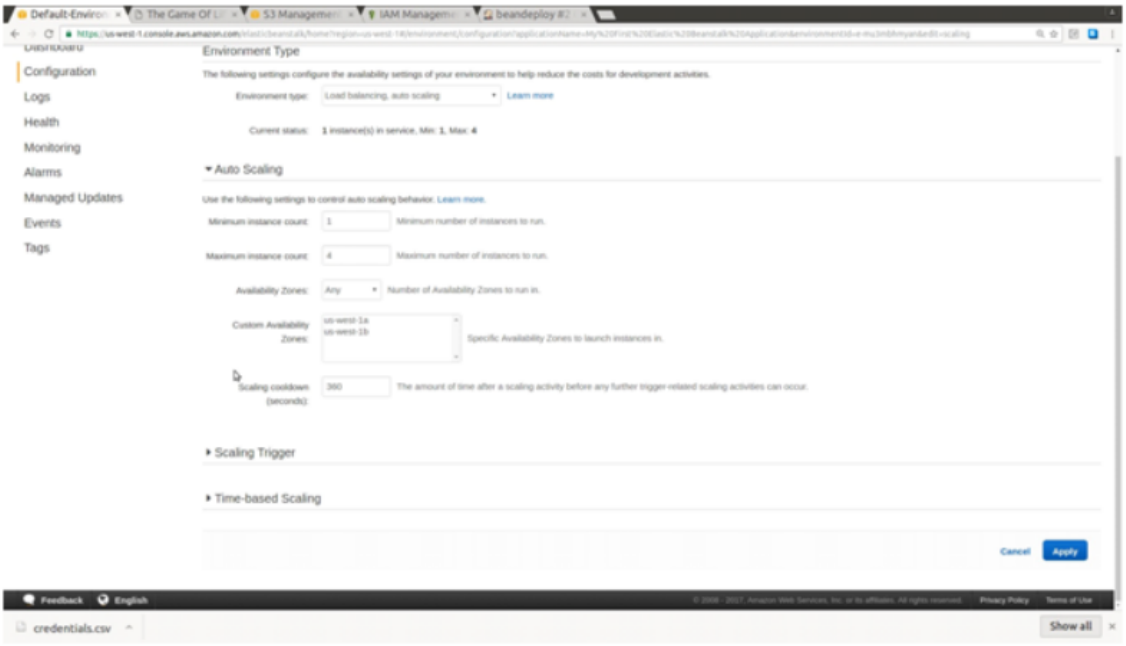
Click on the environment URL to verify if the application is accessible. We can change the configuration of the Beanstalk Environment as per our need. We can improvise the settings in the configuration tab of My First Beanstalk Application. Select scaling option in the configuration and follow the below steps:



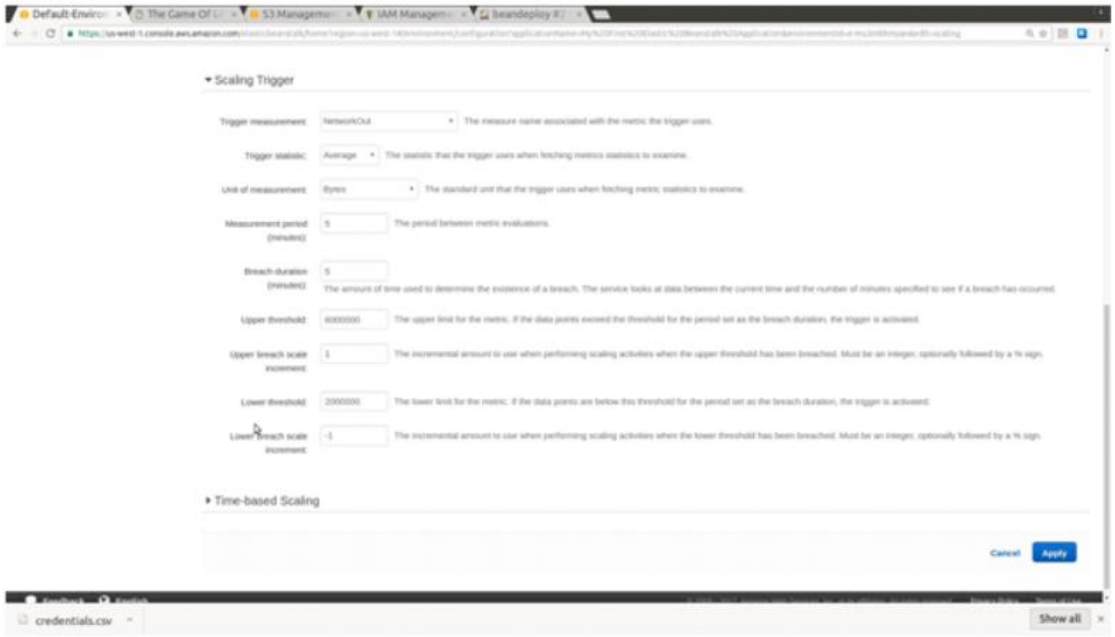
Select the Environment-Type as Load balancing, auto scaling or single instance.



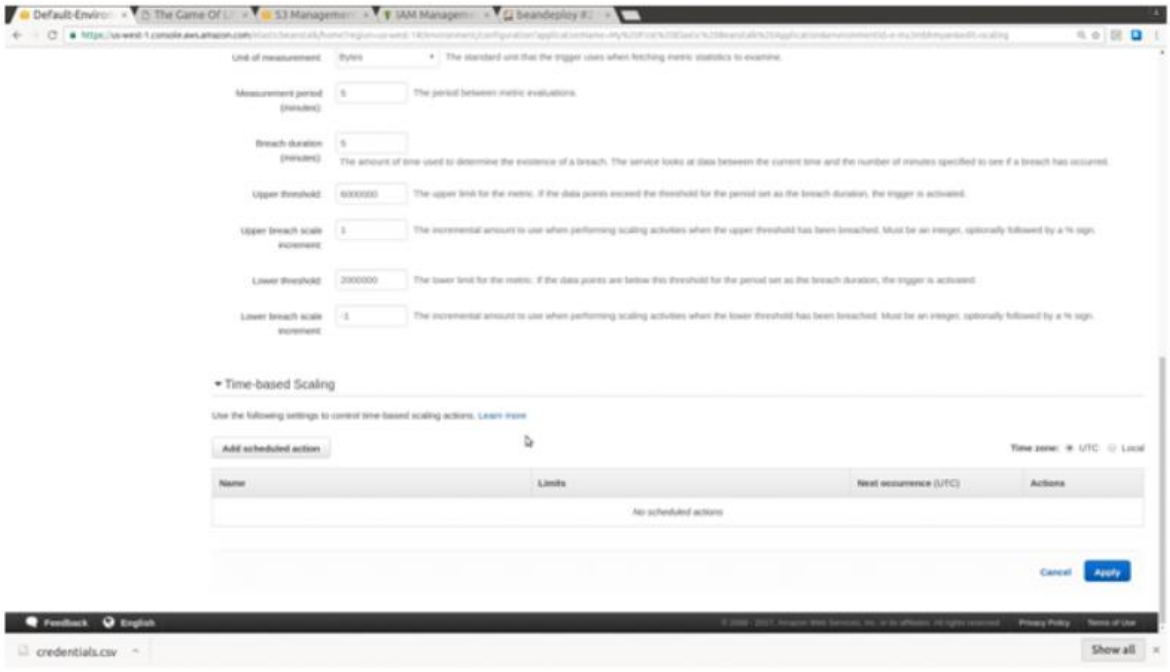
This is helpful to balance the load to available webservers and also performs autoscaling. While configuring Auto scaling behaviour we can mention the parameters like minimum & maximum instances to be scaled, Availability Zones, and cooldown period.



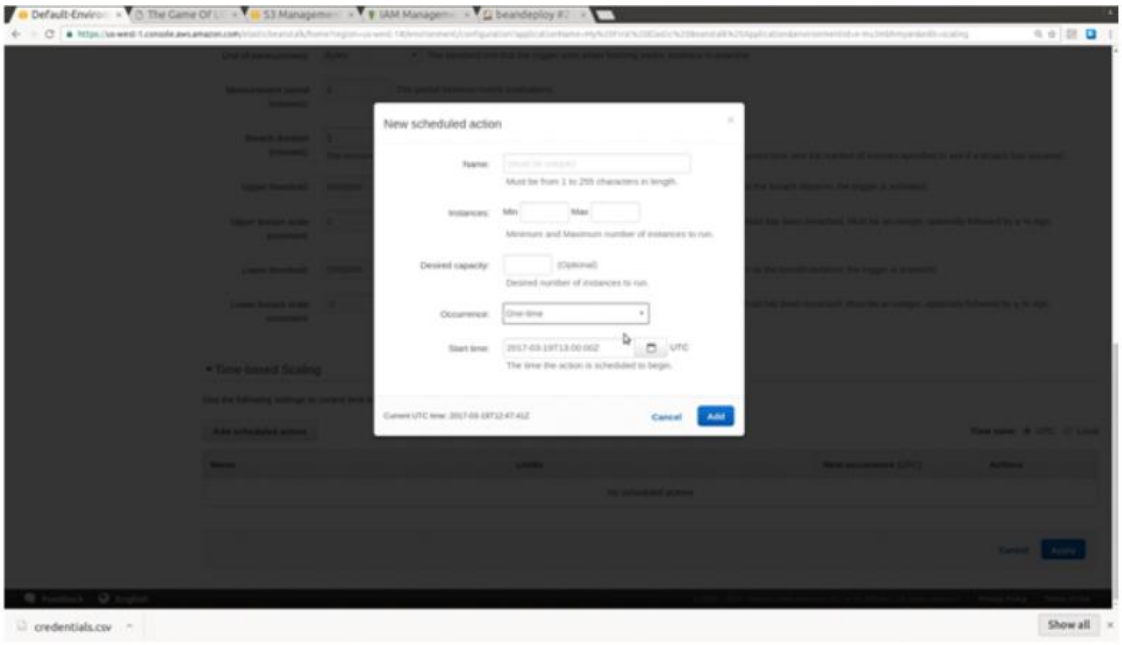
In Scaling triggers, we mention Network, CPU, threshold etc.

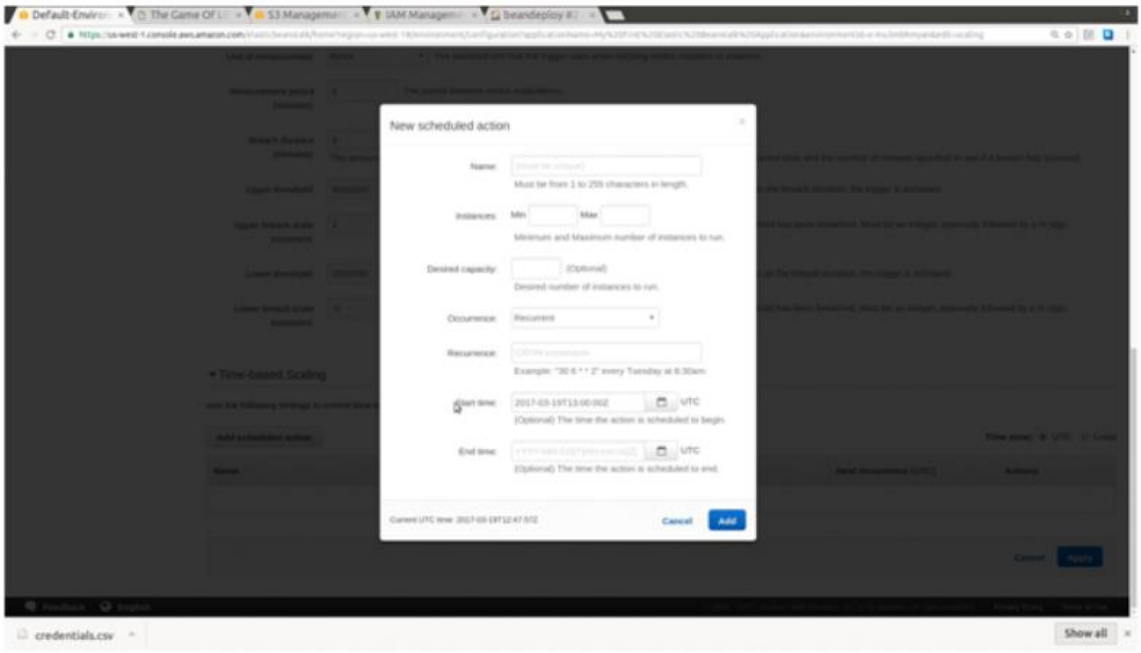


In Time based scaling we mention specific time in a day when the instance count should be raised.



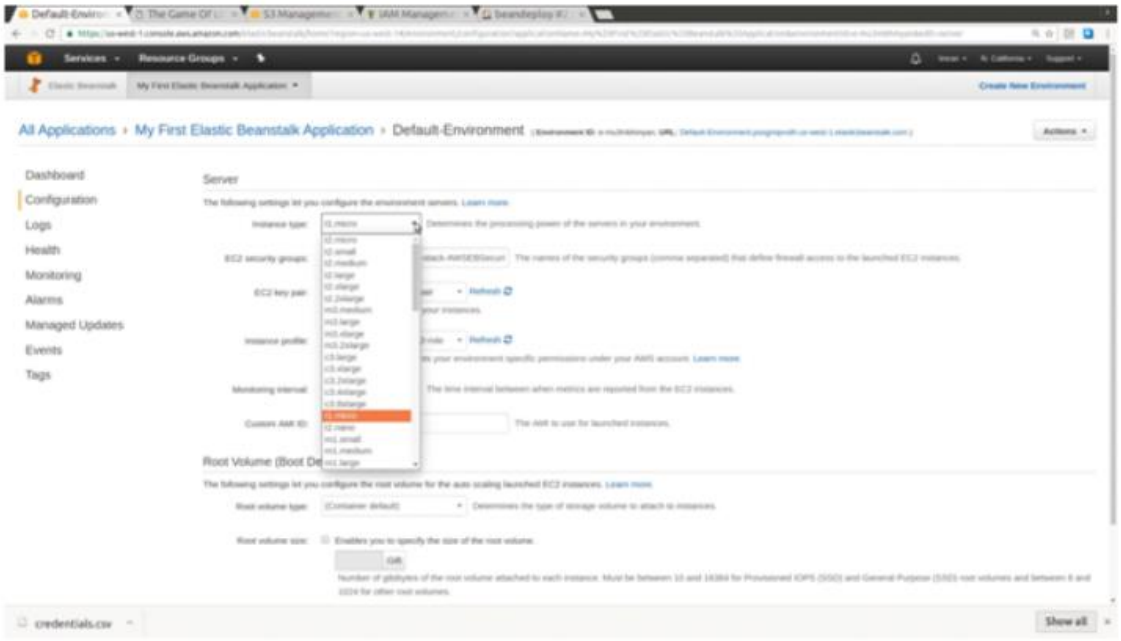
Below screenshot gives the details of how to configure Time-Based scaling



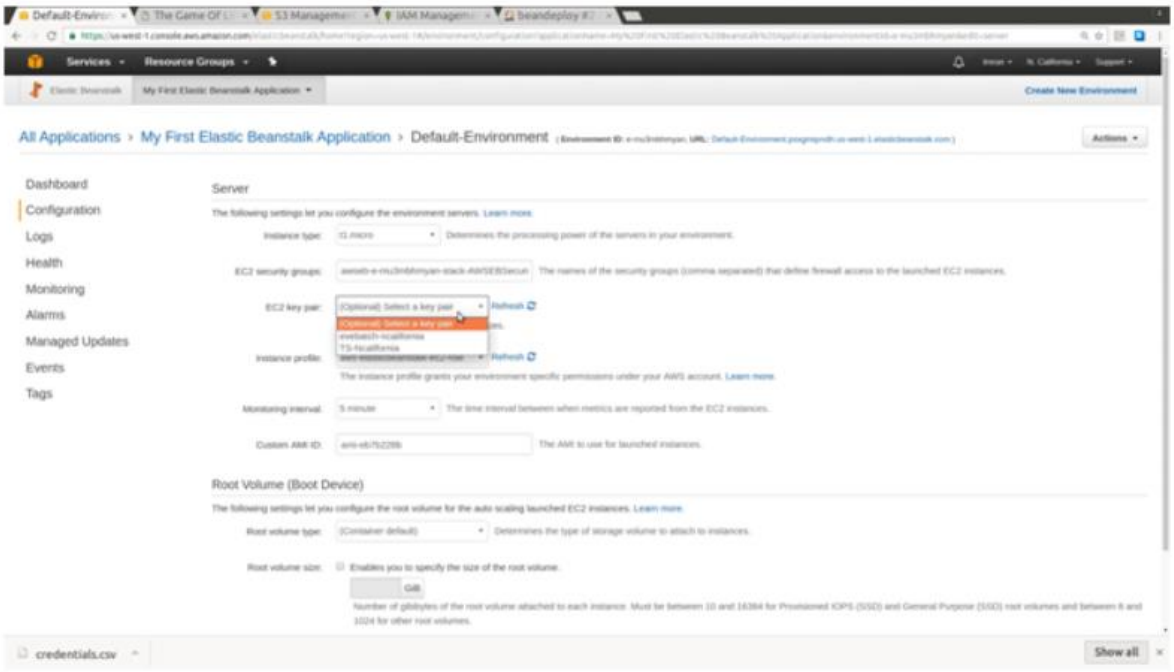


Select instances from configuration tab and follow below steps:

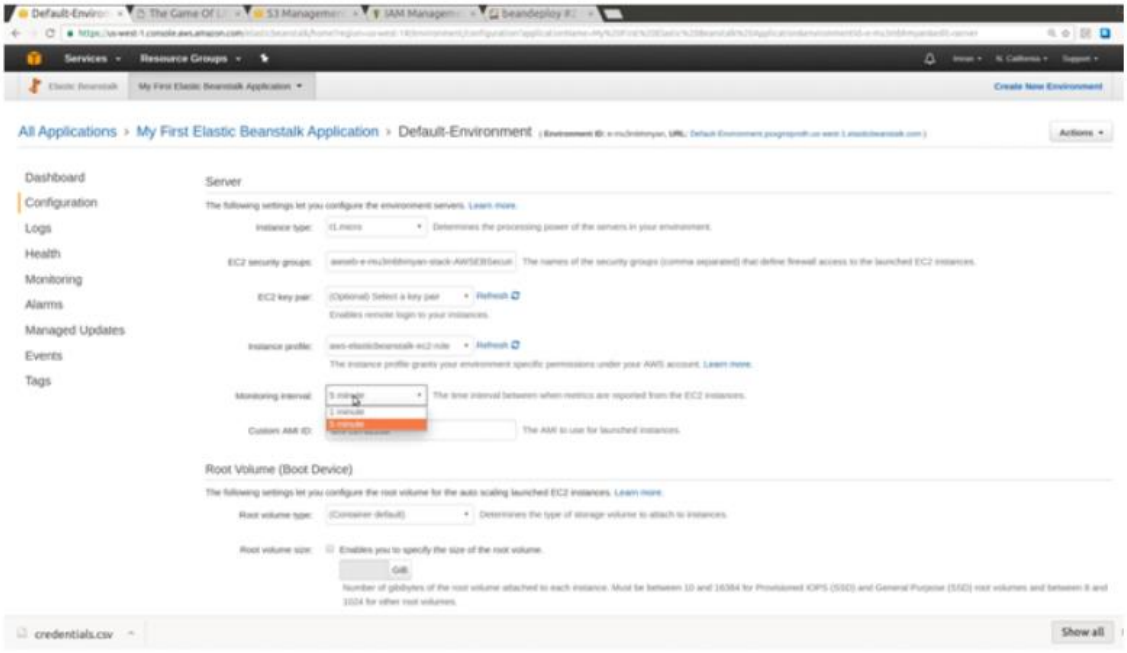
In Server, select the Instance type as t1.micro



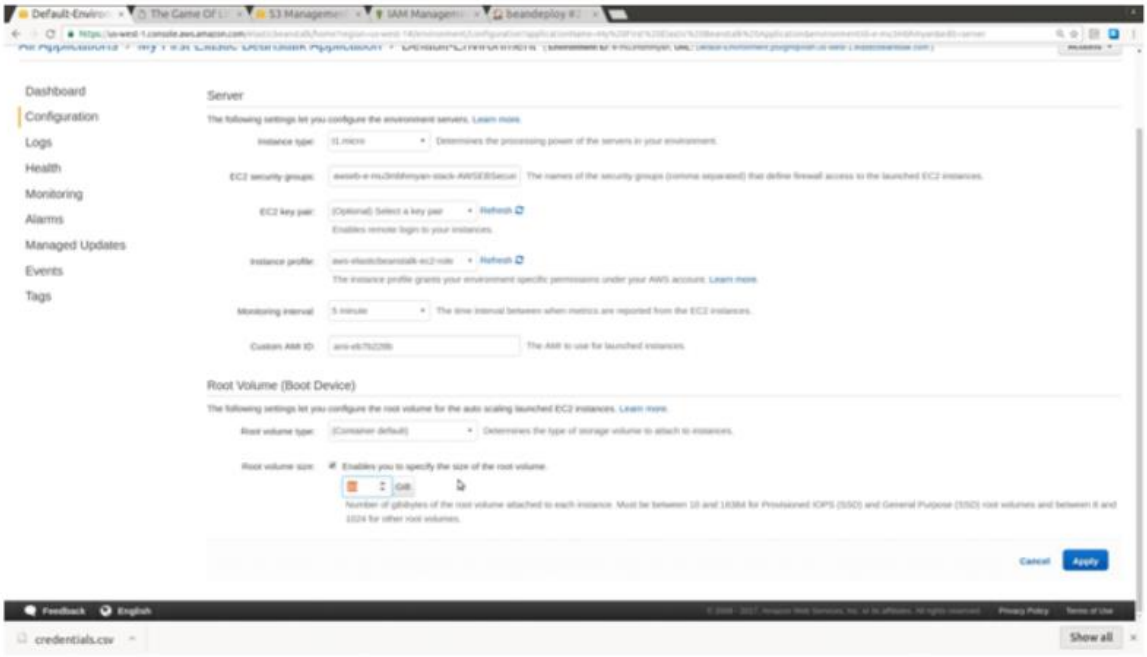
Select the EC2 key pair



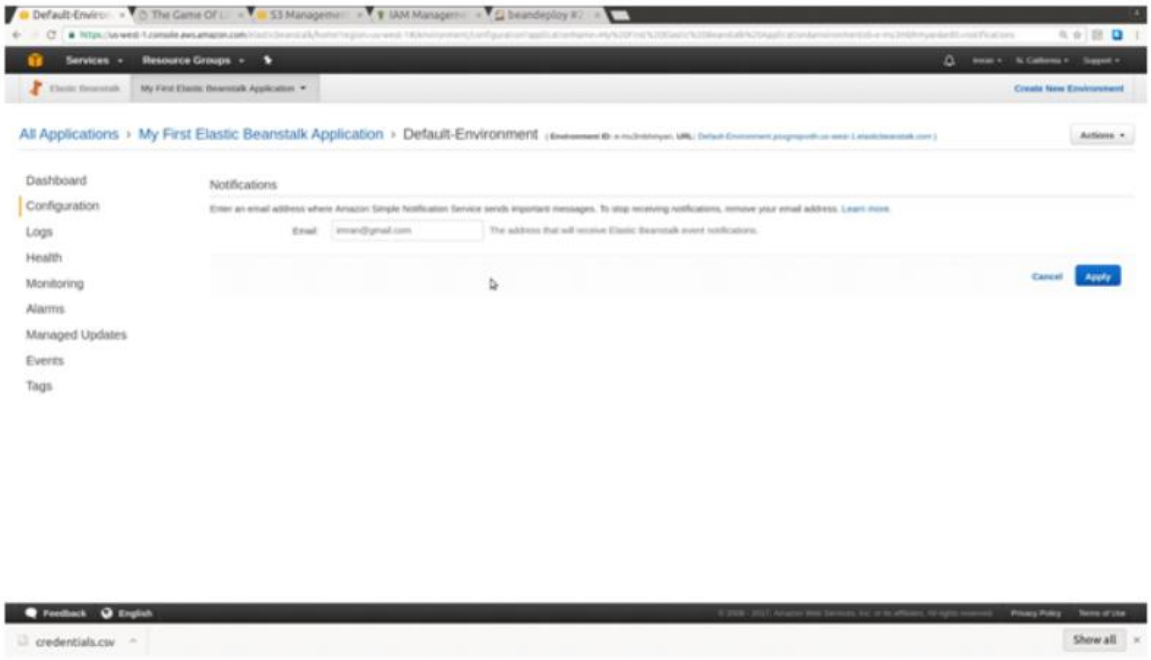
Monitoring interval



Root Volume size for the instances

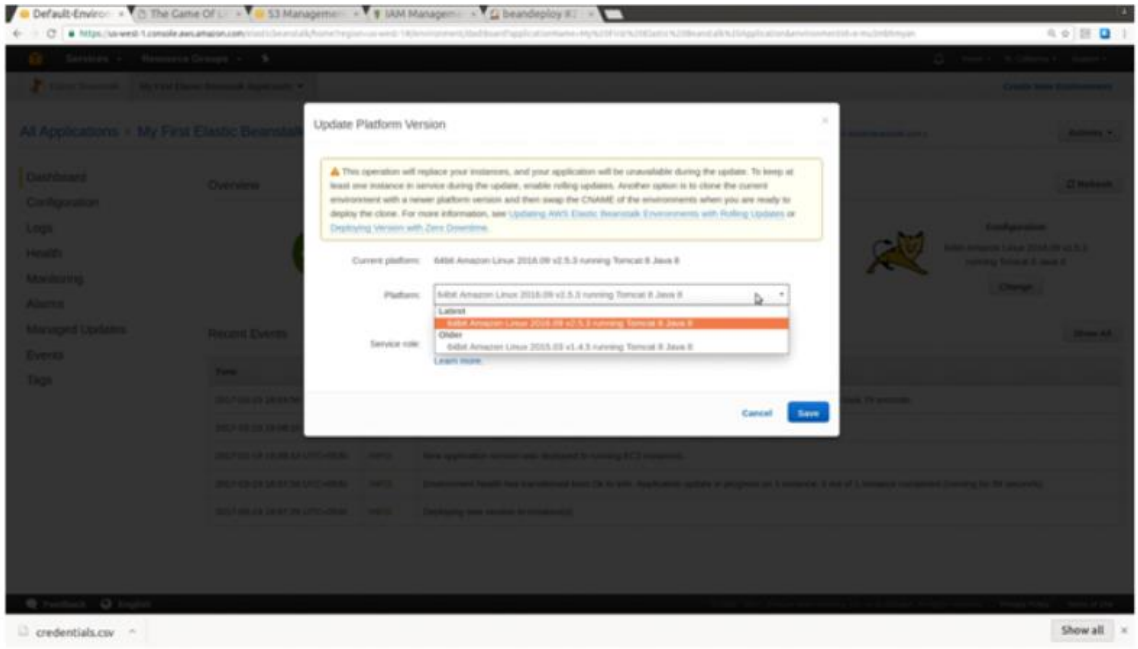


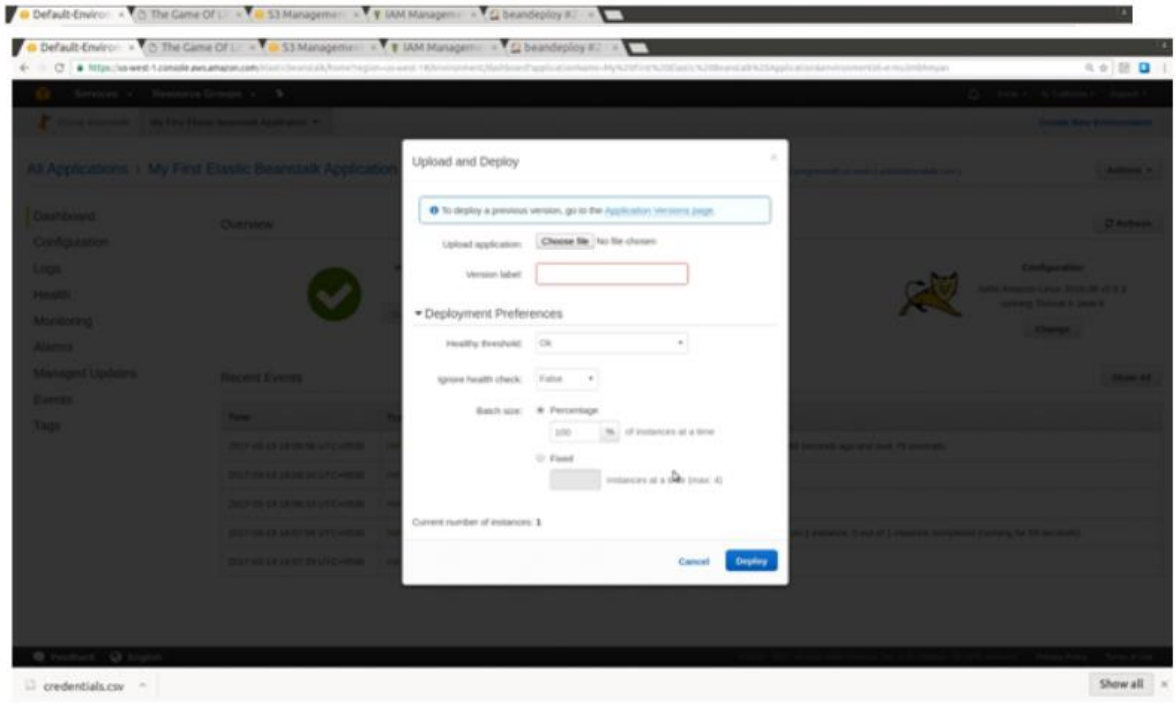
Provide the Email in Notifications page in order to receive the important messages from the Amazon Simple Notification service. Click on Apply



At the application level changing the platform version is allowed.

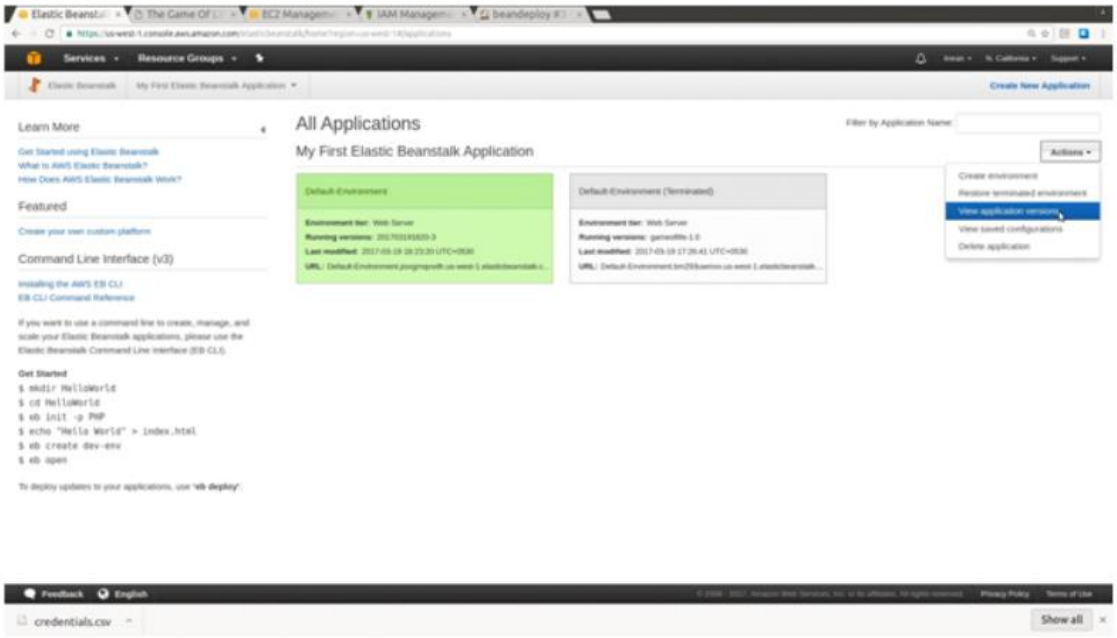
In the application page click on Change to change the platform version.



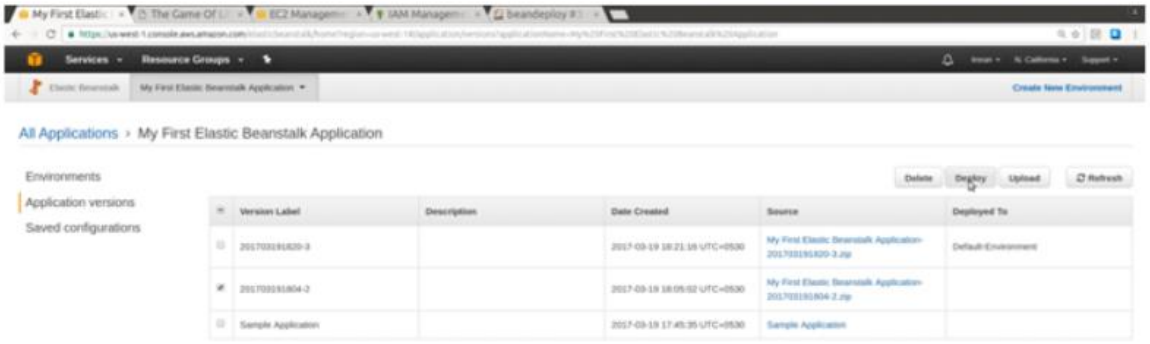


Run Jenkins job few more times to deploy new versions of softwares.

Rollback the application. Click on Actions and select view application versions

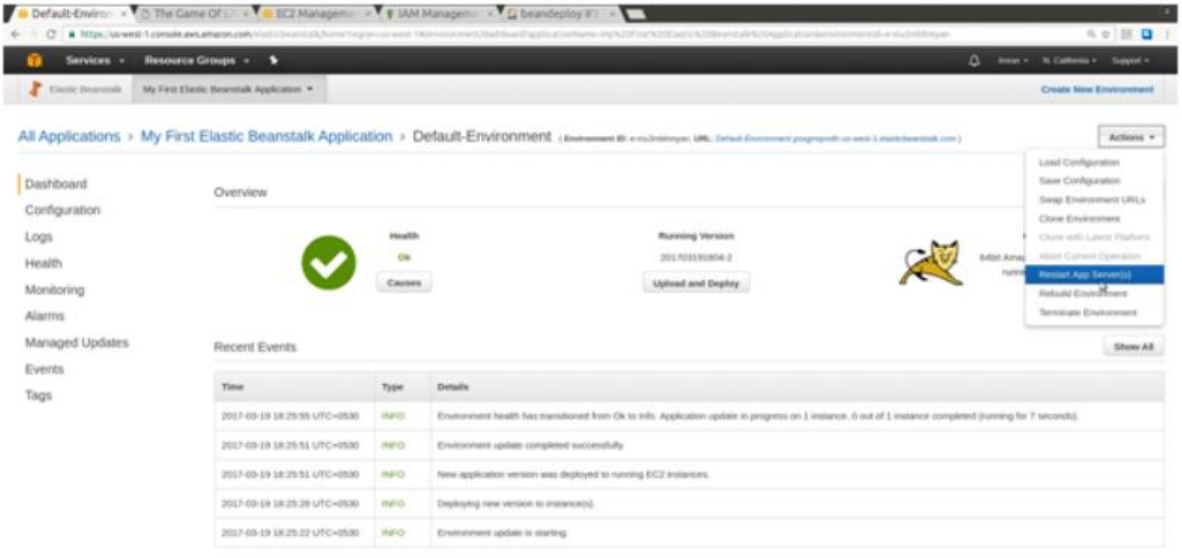


In Application Versions, select an older version and click on Deploy.



Restarting app servers.

Click on Actions and select Restart App Service



Amazon S3 (Simple Storage Service)

Amazon Simple Storage Service (Amazon S3) is storage for the Internet. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web. You can accomplish these tasks using the simple and intuitive web interface of the AWS Management Console.

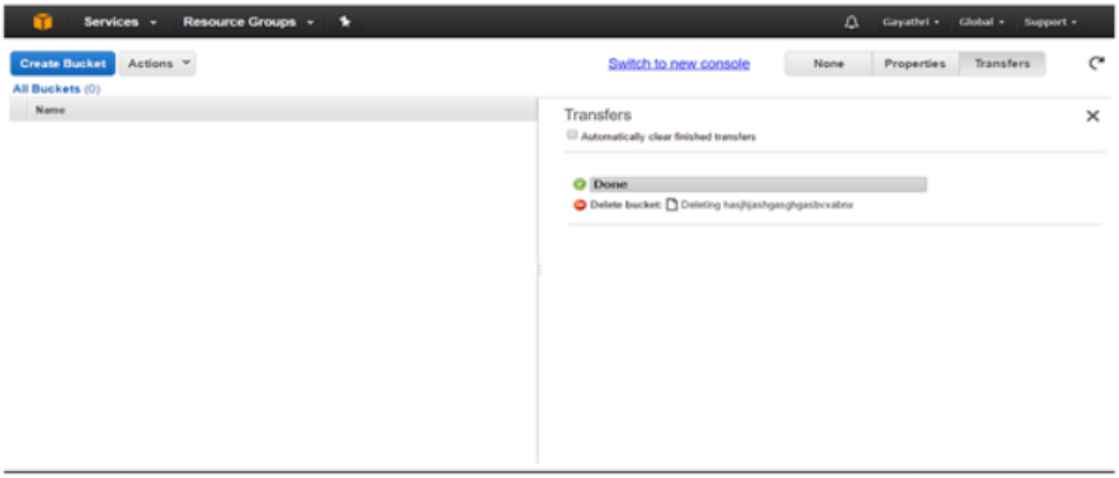
Following are some of the advantages of Amazon S3 Service:

1.Create Buckets – Create and name a bucket that stores data. Buckets are the fundamental container in Amazon S3 for data storage.
2.Store data in Buckets – Store an infinite amount of data in a bucket. Upload as many objects as you like into an Amazon S3 bucket. Each object can contain up to 5 TB of data. Each object is stored and retrieved using a unique developer-assigned key.
3.Download data – Download your data or enable others to do so. Download your data any time you like or allow others to do the same.
4.Permissions – Grant or deny access to others who want to upload or download data into your Amazon S3 bucket. Grant upload and download permissions to three types of users. Authentication mechanisms can help keep data secure from unauthorized access.
5.Standard interfaces – Use standards-based REST and SOAP interfaces designed to work with any Internet-development toolkit.

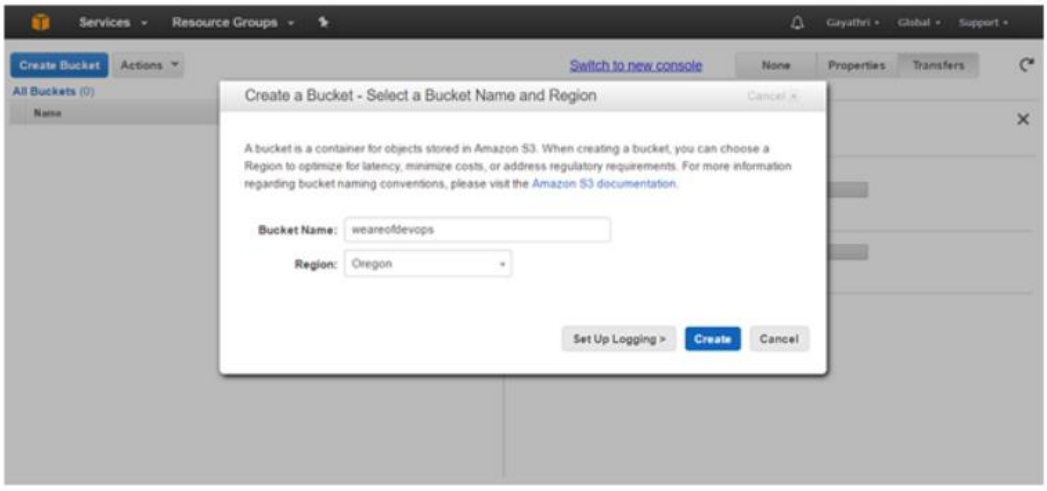
Amazon S3 stores data as objects within buckets. An object consists of a file and optionally any metadata that describes that file. To store an object in Amazon S3, you upload the file you want to store to a bucket. When you upload a file, you can set permissions on the object as well as any metadata. Buckets are the containers for objects. You can have one or more buckets. For each bucket, you can control access to it (who can create, delete, and list objects in the bucket), view access logs for it and its objects, and choose the geographical region where Amazon S3 will store the bucket and its contents.

Creating Amazon S3 Bucket:

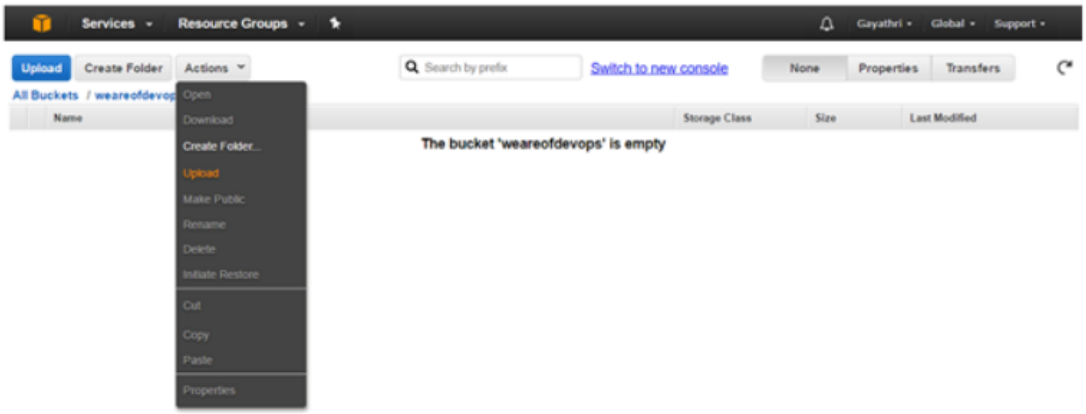
Go to Amazon S3 Service page from main Dashboard, Click on Create Bucket



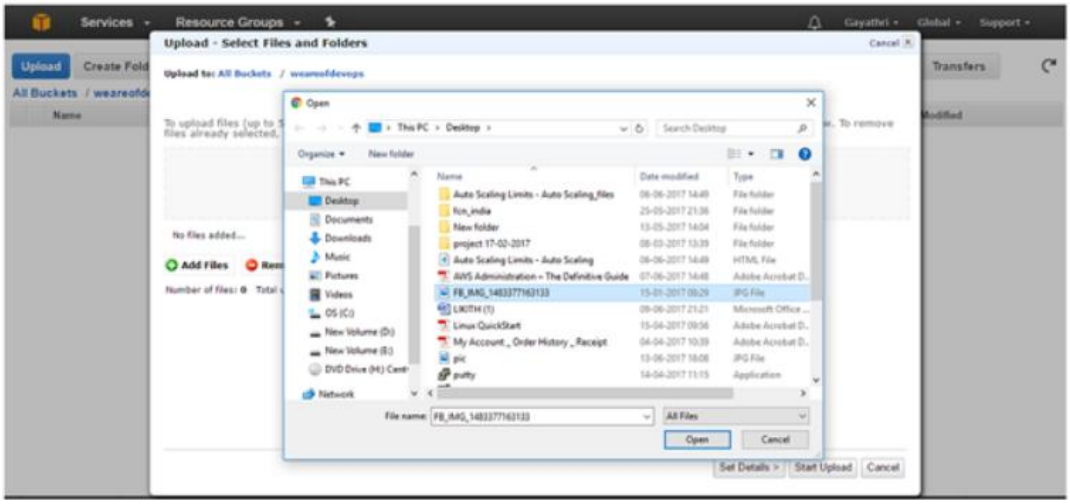
You will see create bucket dialog box in which you have to give Bucket Name and Region. Click on Create



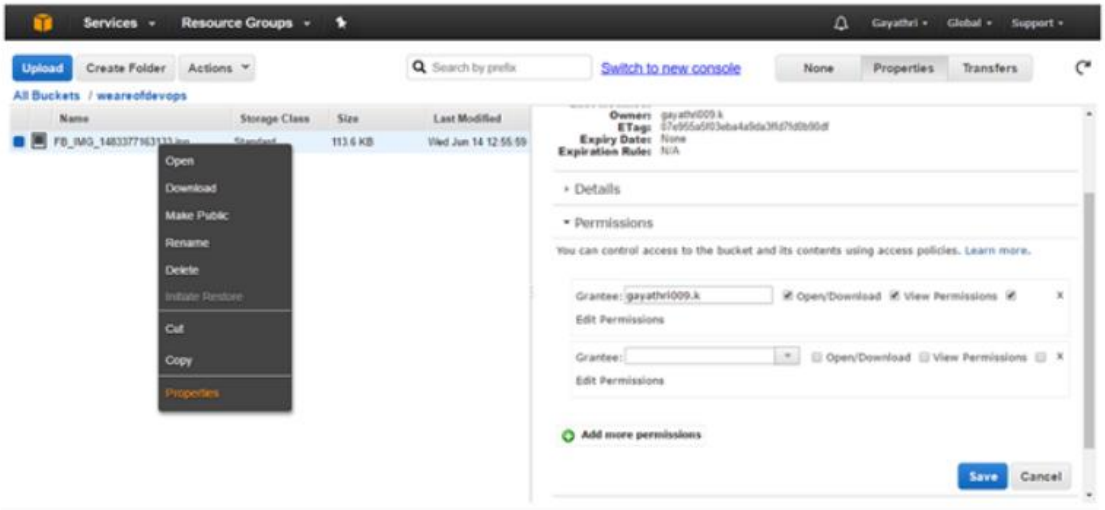
Bucket is created and now you can upload files into that which you want to store. Click on Upload and choose Add files to choose the file to be upload.



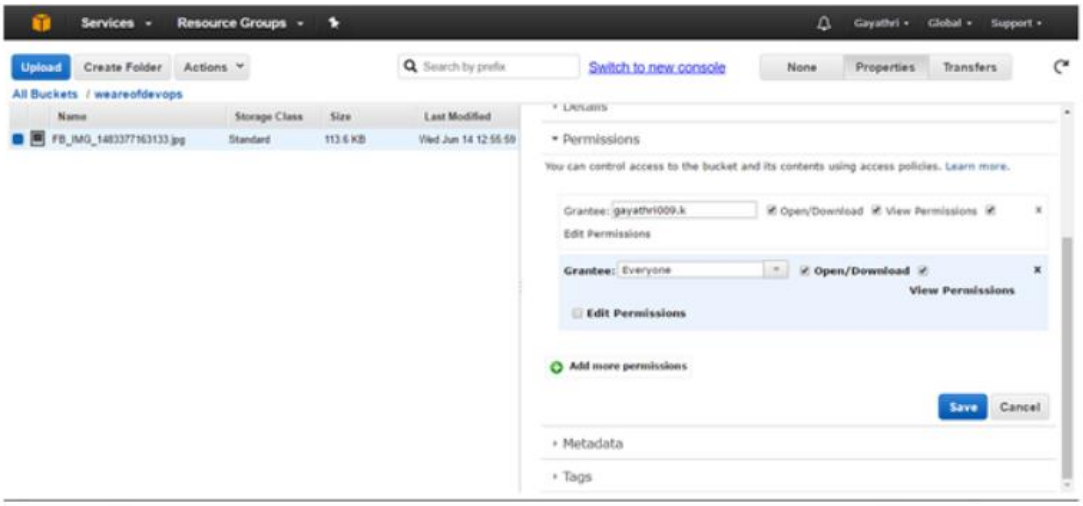
Choose a file to upload, and then choose Open.



The file is uploaded, now you have to give permissions to the file, who has to open/download, view and edit permissions etc. So select the file right click on it you can see properties where you can change/give permissions.



In the permissions tab you can add permissions as required. Here I have given permissions to everyone. Once done click on save. Now you can see the details of the bucket, it gives the url for the file which you can open it through browser.

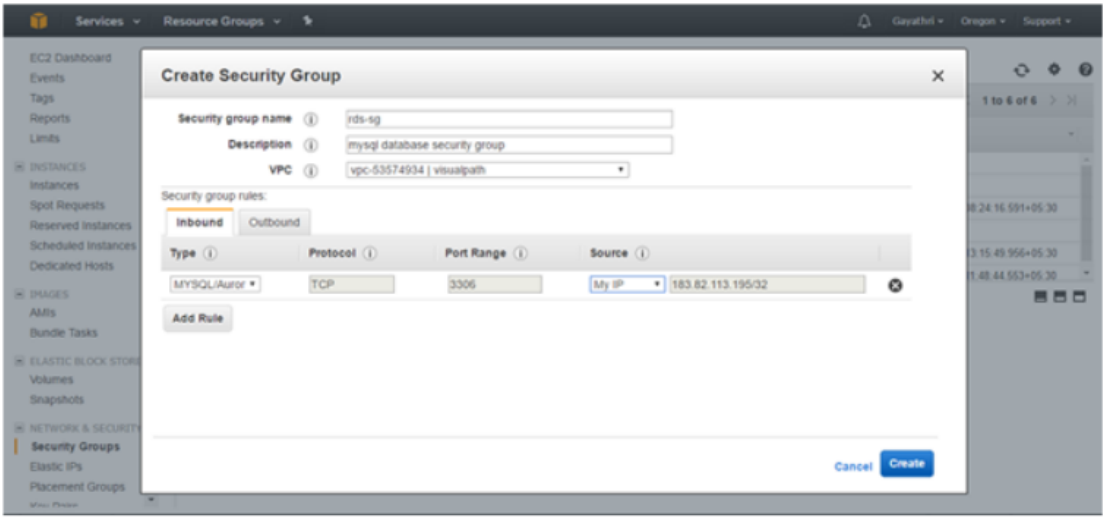


Similarly, we can create folders in bucket, move the objects and delete buckets to prevent further charges if you no longer need to store the objects that you have uploaded.

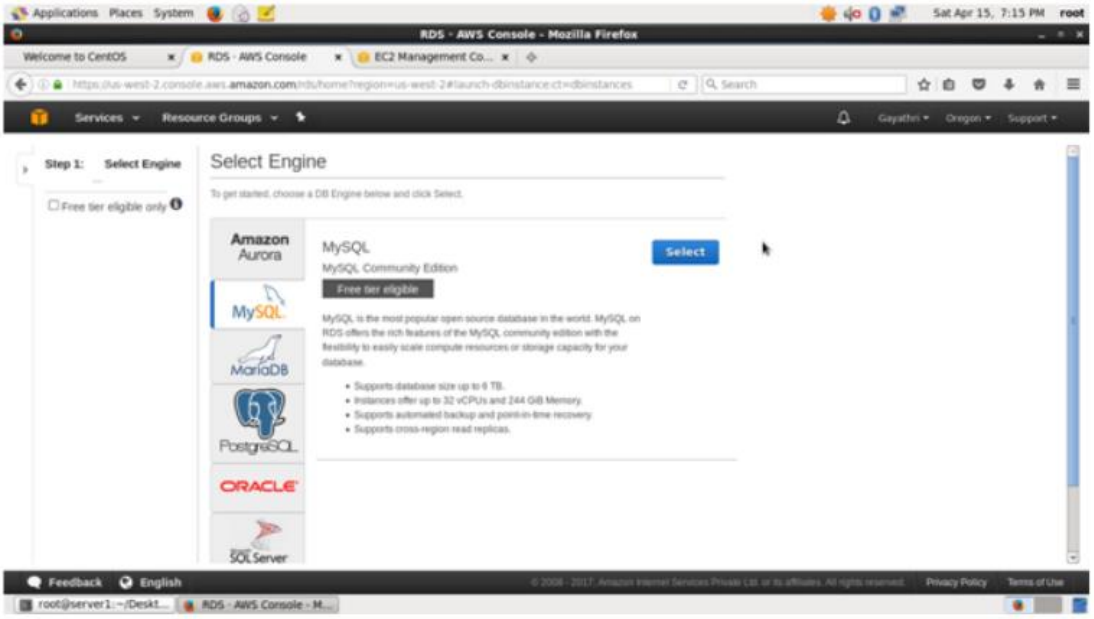
AWS RDS

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks. The basic building block of Amazon RDS is the DB instance. A DB instance is an isolated database environment in the cloud. A DB instance can contain multiple user-created databases, and you can access it by using the same tools and applications that you use with a stand-alone database instance. You can create and modify a DB instance by using the AWS Command Line Interface, the Amazon RDS API, or the AWS Management Console. Each DB instance runs a DB engine. Amazon RDS currently supports the MySQL, MariaDB, PostgreSQL, Oracle, and Microsoft SQL Server DB engines. Each DB engine has its own supported features, and each version of a DB engine may include specific features. Additionally, each DB engine has a set of parameters in a DB parameter group that control the behaviour of the databases that it manages.

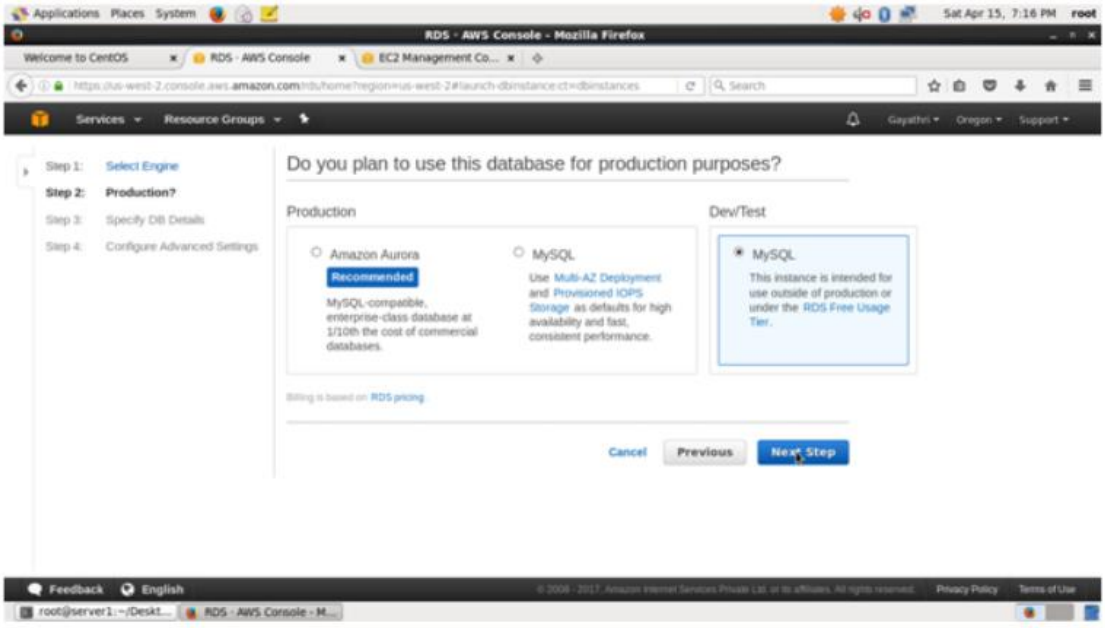
Creating MySQL DB Instance:
Create A Security Group Of MySQL Port (3306) For RDS Instance.



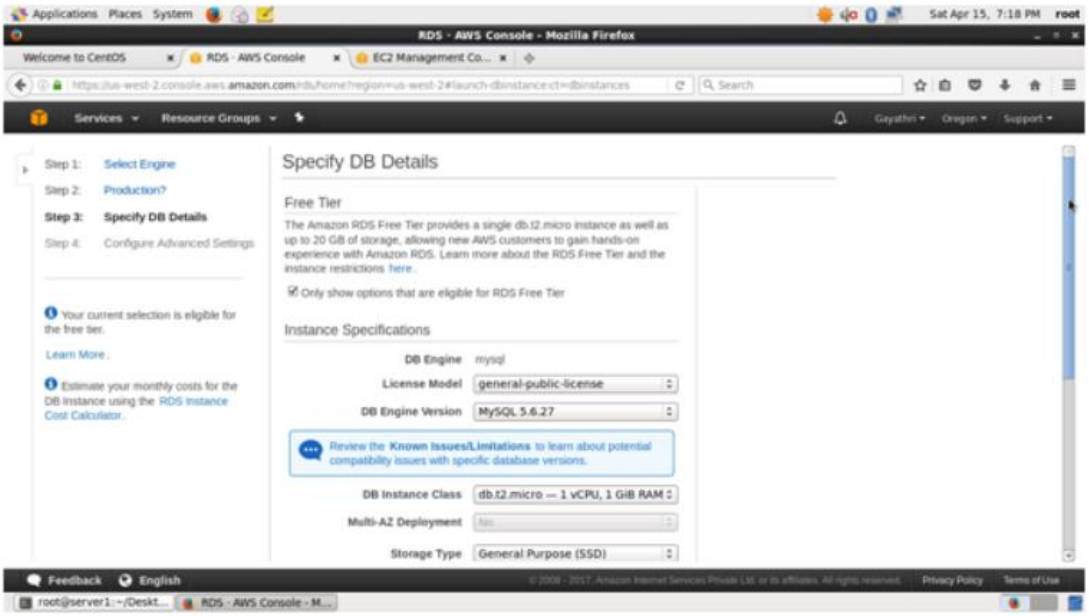
Create an AWS RDS instance with MySQL database.



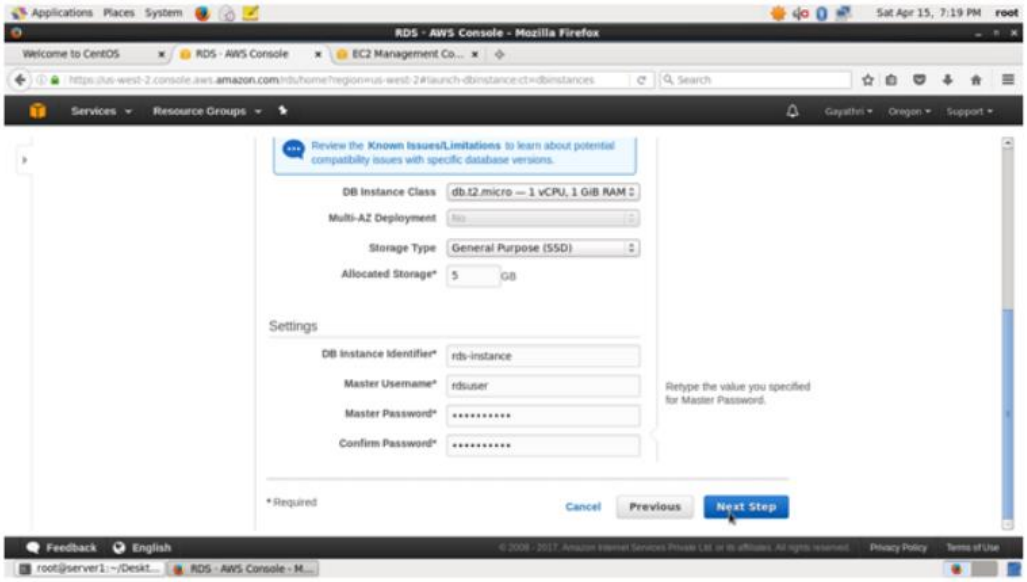
Select MySQL RDS free usage Tier for practice and click on next step



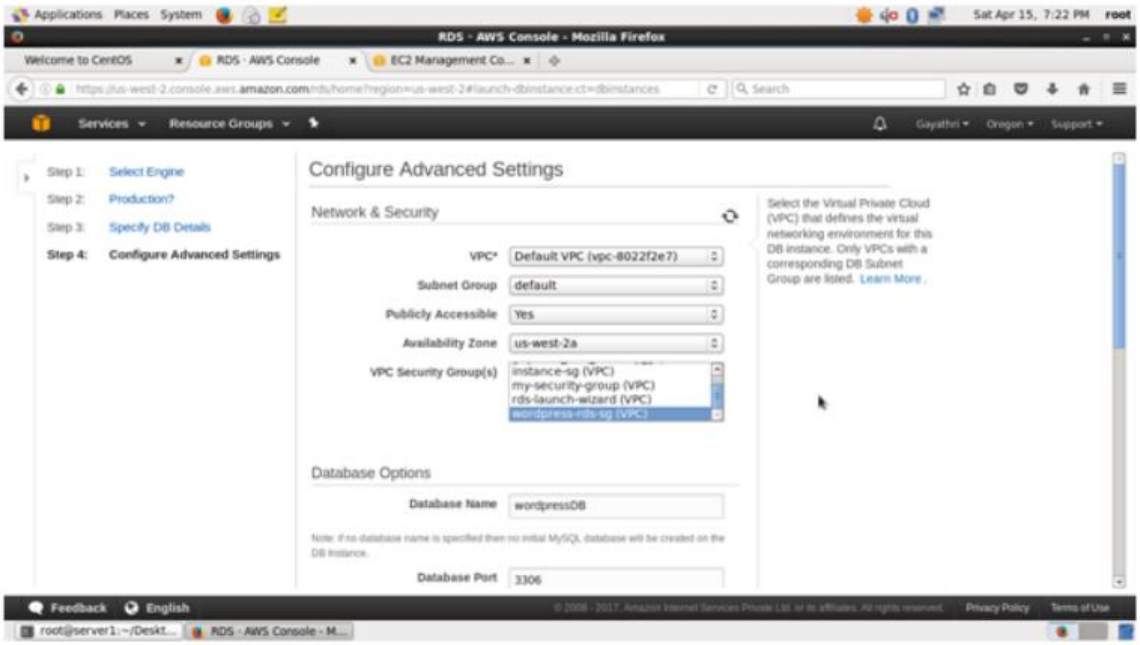
Specify the DB Details like License Model, DB Engine version, DB instance class, storage type, allocated size etc. Here I have chosen the default settings.



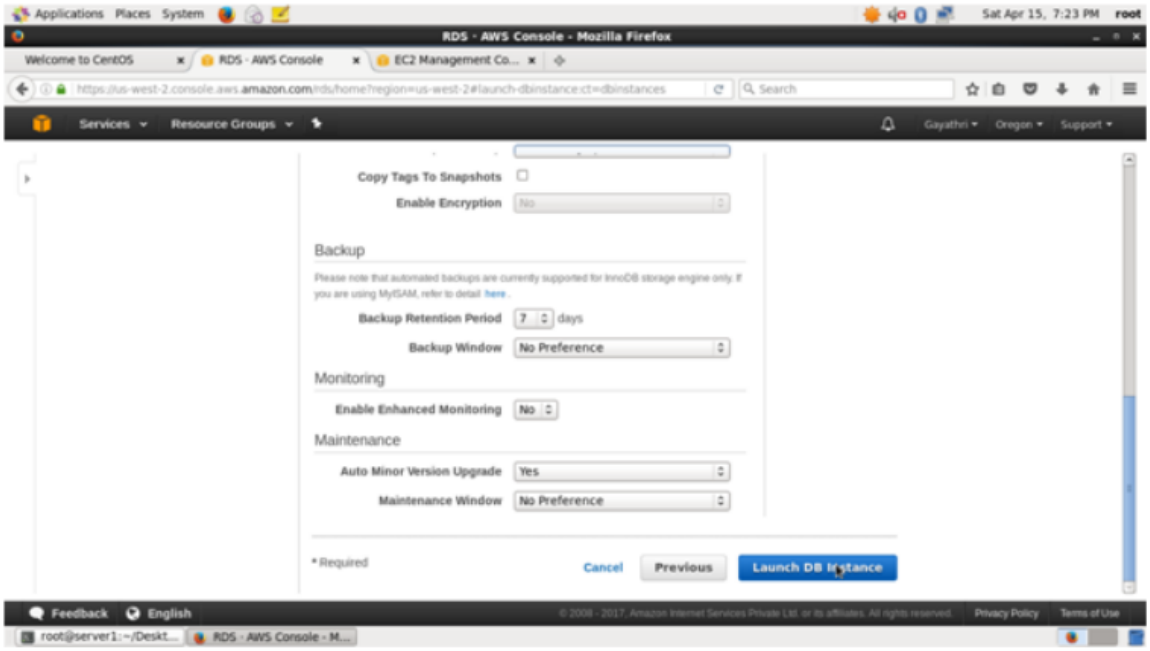
Create User for Database with Username and Password as shown below, click on next step.



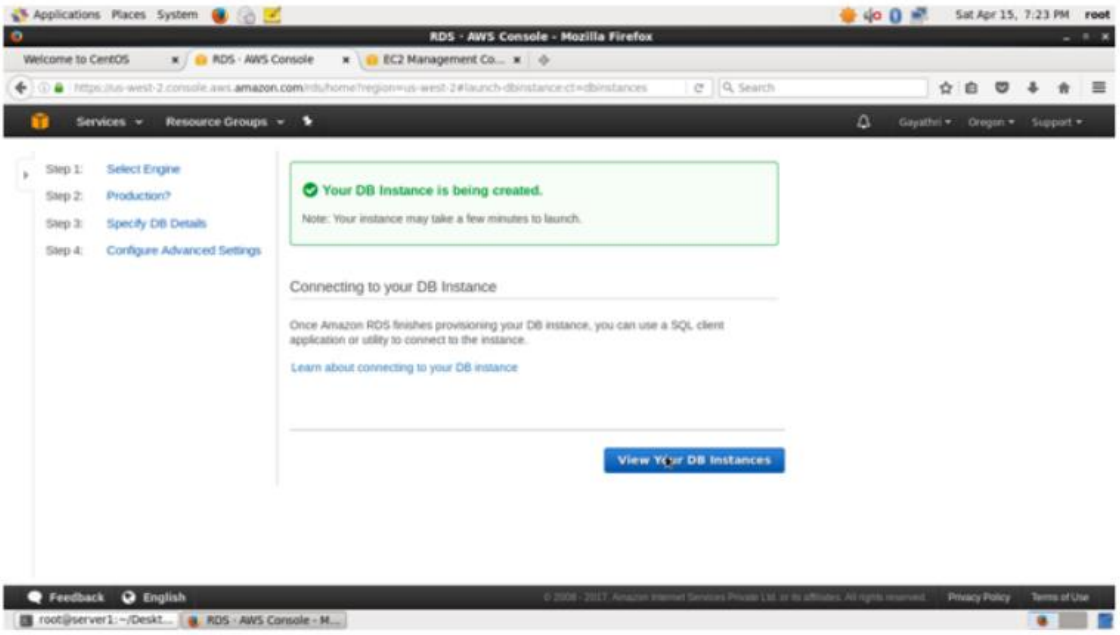
Configure the Advanced Settings like VPC, Subnet Group, AZ and security group for DB instance.



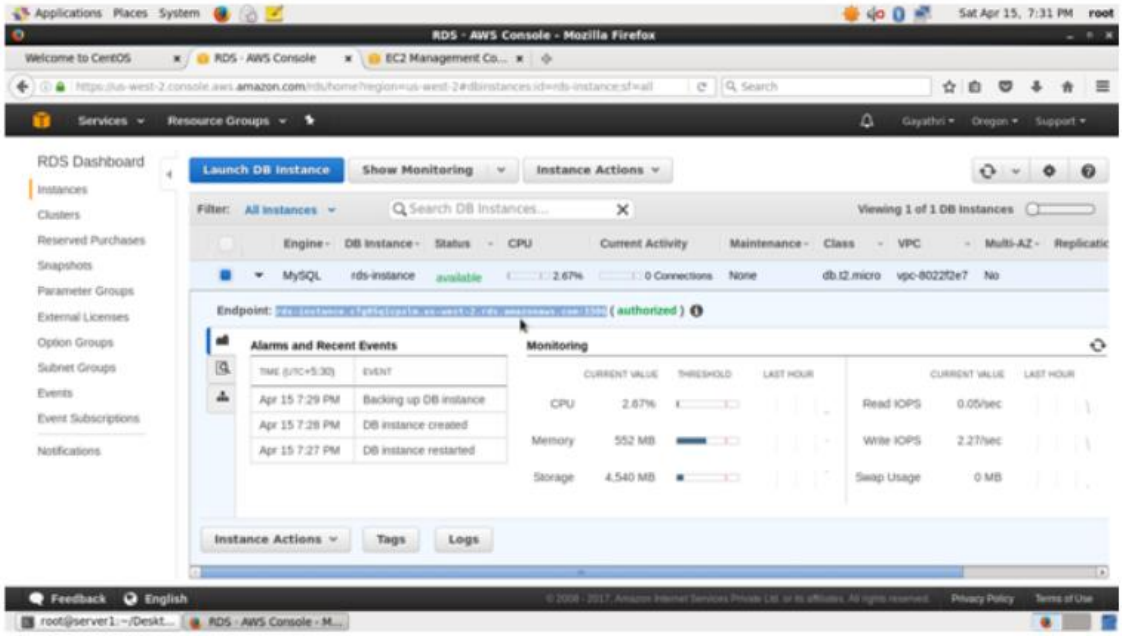
You can also take backup for DB instance if you want by specifying Backup Retention Period. Click on Launch DB instance.



RDS DB instance is successfully created. You can also connect to the DB instance by using End-point.



The highlighted link is the endpoint for DB instance which youhhaveto connect.



Login to the RDS instance from EC2 centos.

mysql -h rds-instance.c7g05q1cpxlm.us-west-2.rds.amazonaws.com -P 3306 -u rdsuser -p

You will get a prompt then enter your password.

We will enter into the mysql shell.

ROUTE53

Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service. It is designed to give developers and businesses an extremely reliable and cost effective way to route end users to Internet applications by translating names like www.example.com into the numeric IP addresses like 192.0.2.1 that computers use to connect to each other. Amazon Route 53 is fully compliant with IPv6 as well.

Amazon Route 53 effectively connects user requests to infrastructure running in AWS such as Amazon EC2 instances, Elastic Load Balancing load balancers, or Amazon S3 buckets and can also be used to route users to infrastructure outside of AWS. You can use Amazon Route 53 to configure DNS health checks to route traffic to healthy endpoints or to independently monitor the health of your application and its endpoints.

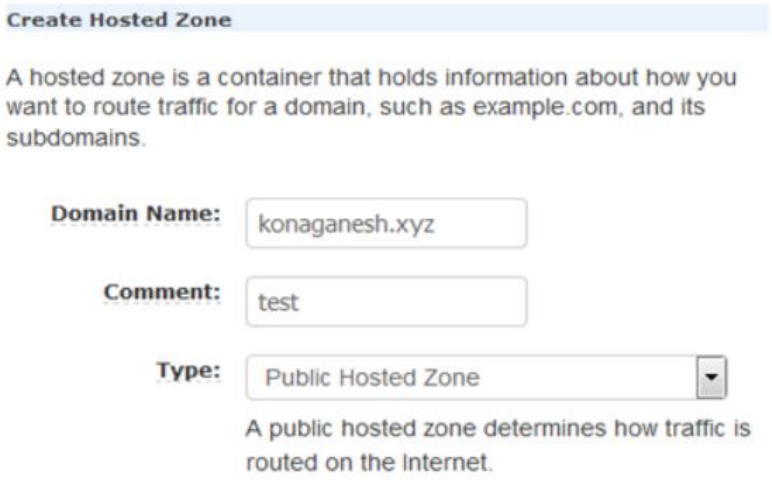
STEPS TO CREATE HOSTED ZONE:

1.Buy a website from any DNS providers like GODADDY, etc..

2.Click on route53 service in AWS console

3.Click on create Hosted Zone

4.Enter your Domain name in the box mentioned like below

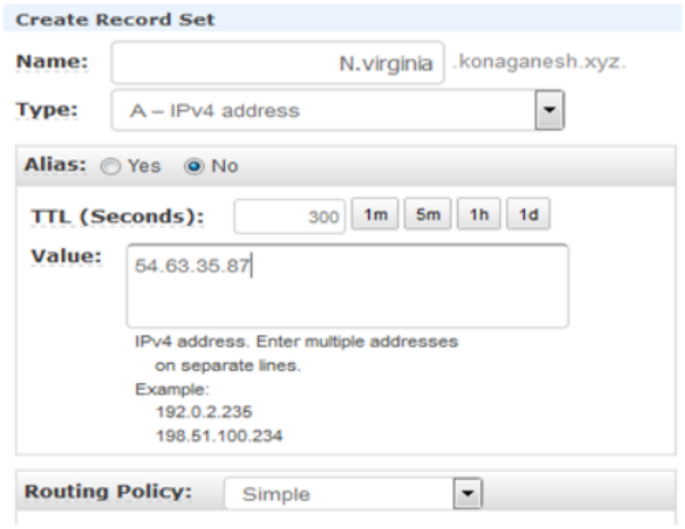


5.Click on create button to create hosted zone

6.After click on create button automatically some NS(Name Server) and SOA(Start Of Authority) records are created like as shown as below

Name	Type	Value	Evaluate
konaganesh.xyz.	NS	ns-1296.awsdns-34.org. ns-1572.awsdns-04.co.uk. ns-851.awsdns-42.net. ns-12.awsdns-01.com.	-
konaganesh.xyz.	SOA	ns-1296.awsdns-34.org. awsdns-hostmaster.amazo	-

7.Copy the NS entities and paste it on your domain manager(GODADDY) 8.Now you create two EC2 instances in two different regions . And deploy an application to each instance 9.Now back on to route53 console and create record set as shown as below



1.In name box write any name to access your webserver

2.In type box there are several options like A,CNAME etc..,we have to select one of the option based on our requirement

1.A------ipv4 address of the webserver

2.CNAME------ to typing the CNAME we can give directly domain name of the webserver instead of Alias name

3.Mostly these two type of options are preferred

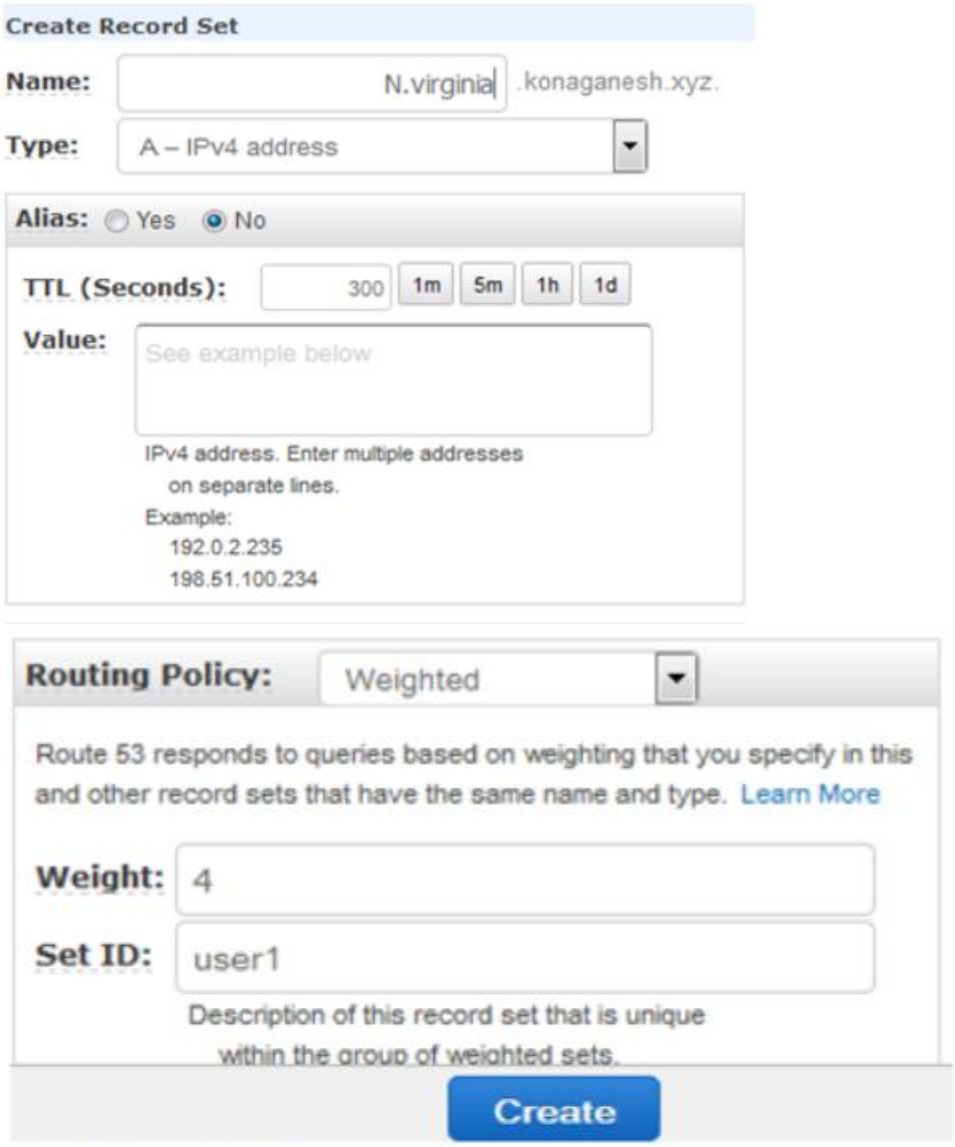
1.In the value box we have to type ip address or dns name of webserver 2.In routing policy we have select any one of the option based on your requirement

1.Simple: it is a default routing policy single alias name with single webserver.

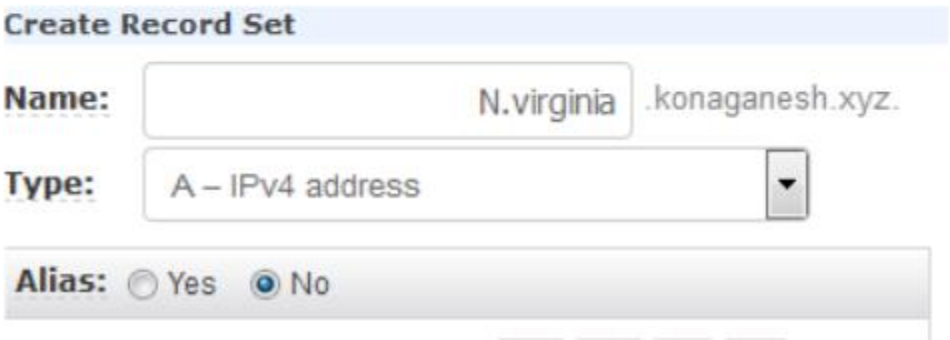
2.Weighted: Based on the weight we have to mention in the weight box how many no.of requests to go particular instances in the different regions.

For weighted we have to specify same name to different record set in the name box

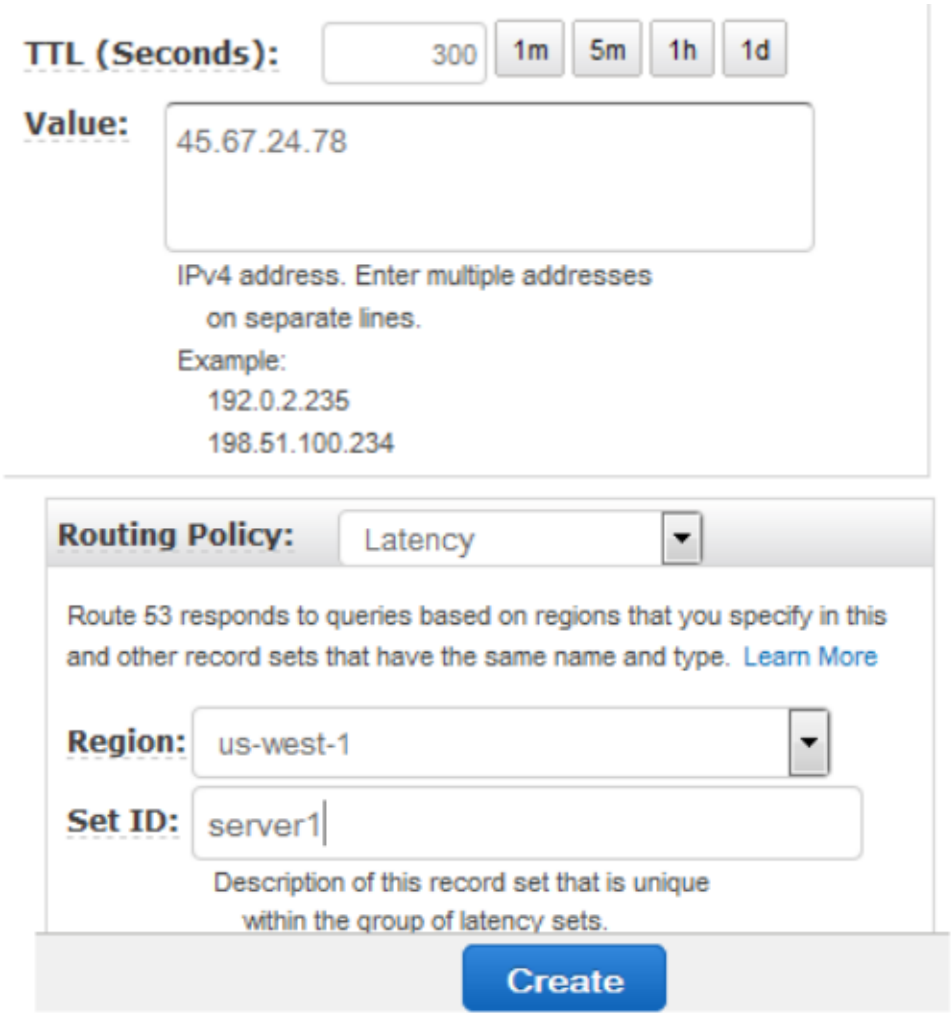
1st Record Set:



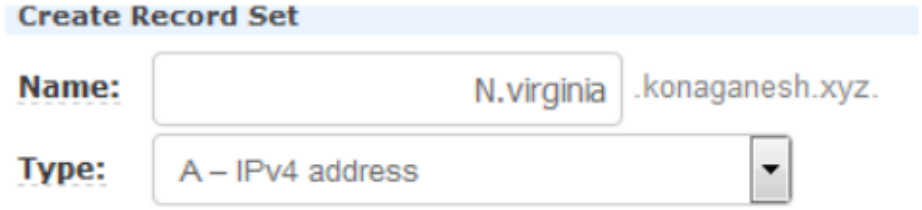
2nd Record Set:

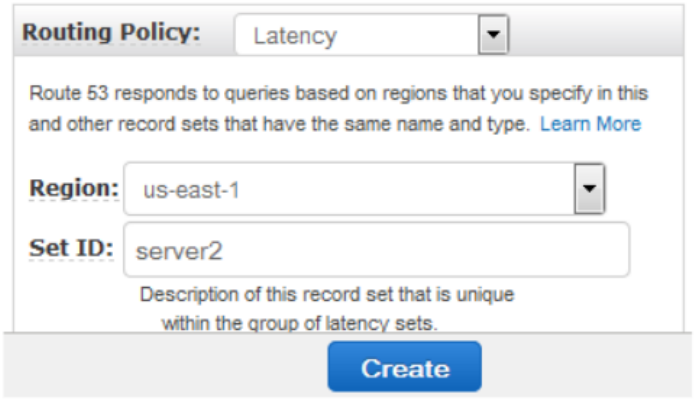


Please do not copy text



2nd Record Set:

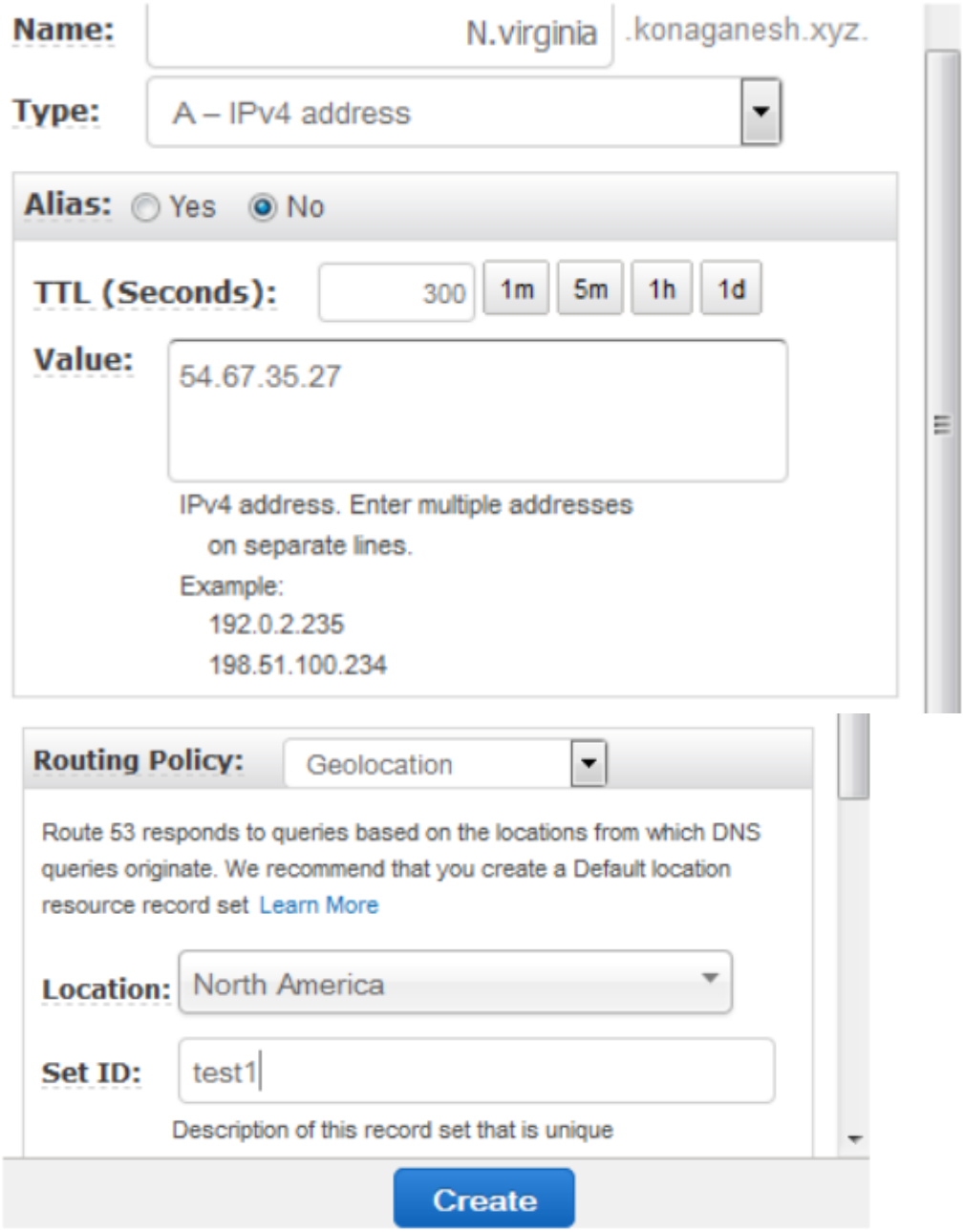


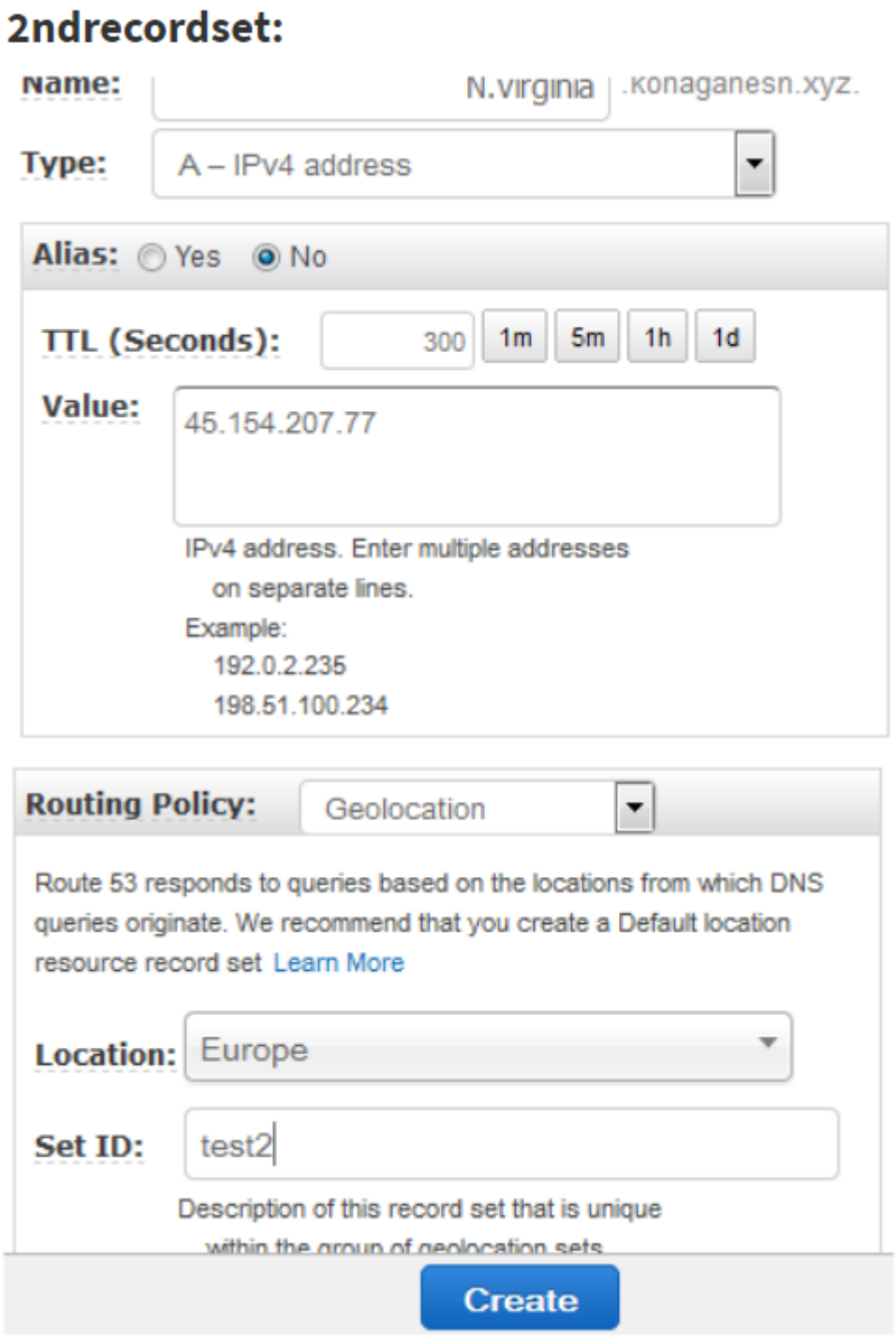


2.Fail over: It acts like ELB (region based) with different regions . if one instance (primary)failed in one region then it will automatically hit the another instance(secondary) where it is in another region.

3.Geolocation: its choose where route53 will send your traffic based on geographic location of your users

1st Record Set:




2ndrecordset:
Name:
N.virginia
.konaganesn.xyz.
Type:
A – IPv4 address
Alias:
Yes
No
TTL (Seconds):
300
1m
5m
1h
1d
Value:
45.154.207.77
IPv4 address. Enter multiple addresses on separate lines.
Example:
192.0.2.235
198.51.100.234
Routing Policy:
Geolocation
Route 53 responds to queries based on the locations from which DNS queries originate. We recommend that you create a Default location resource record set
Learn More
Location:
Europe
Set ID:
test2
Description of this record set that is unique within the group of geolocation sets
Create

AWS CLI

[Amazon Command Line Interface]

Installing & Configuration AWS-CLI

1.Create IAM User From AWS Console

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources for your users.

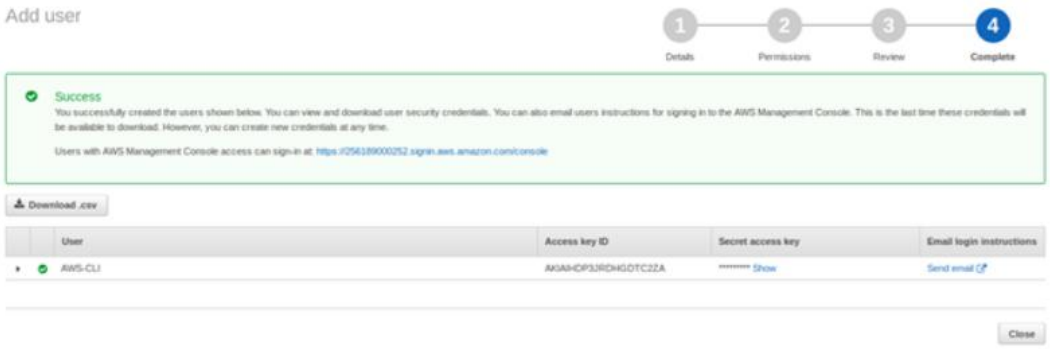
1.Open IAM dashboard

2.Creat user with programmatic access.

3.Select Attach Existing Policies Directly

Check Policy Name - AdminstrativeAccess& Click Next

Click Create user



4.After creating the user, Download Credentials file for Access Key and Security Key for AWS configuration for CLI

1.Setup AWS CLI

```
@devops:~$ sudo apt-get install awscli
```

AWS Access Setup

```
satish@devops:~$ aws configure
AWS Access Key ID [None]: AKIAIARHYZPBDZYS3LEA
AWS Secret Access Key [None]: 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
Default region name [None]: us-west-2
Default output format [None]: json
```

The aws configure set command can be used to set a single configuration value in the AWS config file. Default region is the name of the region you want to make calls against by default. This is usually the region closest to you, but it can be any region. Default Output format can be either json, test or table. If you dont give any value, default value is json.

Configuration And Credential Files

We can add additional configure profile by adding entries to config and credential files. The CLI stores credentials specified with aws configure in a local file named credentials in a folder named. aws in your home directory. Home directory location varies but can be referred to using the environment variables %UserProfile% in Windows and $HOME or ~ (tilde) in Unix-like systems. For example, the following commands list the contents of the. aws folder:

Linux, macOS, or Unix

$ ls ~/.aws

Windows

>dir "%UserProfile%\.aws"

```
satish@devops:~$ cd ~/.aws/
satish@devops:~/.aws$ ls
config  credentials
satish@devops:~/.aws$
```

In order to separate credentials from less sensitive options, region and output format are stored in a separate file named config in the same folder

```
~/.aws/credentials
[default]
aws_access_key_id=XXXXXXXXXXXXXXXXXXXX
aws_secret_access_key=XXXXXXXXXXXXXXXXXXXXXHUGGXXXXX
```

```
[user2]
aws_access_key_id=AKIAIXXXXXXXXXDHBXXXX
aws_secret_access_key=je7MtXXXXXXXBF/XXXXXXX/XXXXXXXXKEY
```

```
~/.aws/config
[default]
region=us-west-2
output=json
[profile user2]
region=us-east-1
output=text
```

Note: The AWS credentials file uses a different naming format than the CLI config file for named profiles. Do not include the 'profile ' prefix when configuring a named profile in the AWS credentials file.

The following settings are supported.

aws_access_key_id – AWS access key.

aws_secret_access_key – AWS secret key.

aws_session_token – AWS session token. A session token is only required if you are using temporary security credentials.

region – AWS region.

output – output format (json, text, or table)

AWS CLI NAMED PROFILE

You can configure additional, named profiles by using the --profile option. The AWS CLI supports named profiles stored in the config and credentials files. You can configure additional profiles by using aws configure with the --profile option or by adding entries to the config and credentials files.

```
satish@devops:~/.aws$ aws configure --profile testuser
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:
```

Precedence For AWS CLI

The above configuration values have the following precedence:

1.Command line options

2.Environment variables

3.Configuration file

AWS CLI SUB COMANDS

1. LIST------> List common configuration sources

2. GET-------> get the value of single configure value

3. SET------> set the value of single configure var to see the list of configuration data

:~/.aws$ aws configure list ------> It will give output default aws profile

:~/.aws$ aws configure list --profile testuser

```
satish@devops:~/.aws$ aws iam list-users
{
    "Users": [
        {
            "Path": "/",
            "UserName": "AWS-CLI",
            "Arn": "arn:aws:iam::256189000252:user/AWS-CLI",
            "CreateDate": "2017-06-06T10:25:21Z",
            "UserId": "AIDAJOJDFDEFX26CC3HMK"
        },
        {
            "Path": "/",
            "UserName": "cli_user",
            "Arn": "arn:aws:iam::256189000252:user/cli_user",
            "CreateDate": "2017-06-06T05:49:13Z",
            "UserId": "AIDAIZ7UTLVMTED4JUDRK"
        },
        {
            "Path": "/",
            "UserName": "devadmin",
            "Arn": "arn:aws:iam::256189000252:user/devadmin",
            "CreateDate": "2017-04-24T18:27:06Z",
            "UserId": "AIDAIX3EYXXE6JHOGZTIA"
        },
        {
            "Path": "/",
            "UserName": "testuser",
            "Arn": "arn:aws:iam::256189000252:user/testuser",
            "CreateDate": "2017-06-09T05:12:37Z",
            "UserId": "AIDAIM2I7PFLZCQDXOH6A"
        }
    ]
}
```

arn-->The Amazon Resource Name (ARN) specifying the group.

arn:partition:service:region:account:resource

Manage Users & Security Groups from AWS CLI

User Creation

Syntax

```
Devops: ~/.aws$ awsiam create-user –user-name testuser2
```

```
satish@devops:~/.aws$ aws iam create-user --user-name testuser2
{
    "User": {
        "CreateDate": "2017-06-09T10:19:22.508Z",
        "UserId": "AIDAISIF7O7H4WARZ5RCY",
        "UserName": "testuser2",
        "Path": "/",
        "Arn": "arn:aws:iam::256189000252:user/testuser2"
    }
}
```

Creating User & Password with Profile

```
Devops: ~/.aws$ awsiam create-login-profile –user-name testuser –password
*******
```

```
satish@devops:~/.aws$ aws iam create-login-profile --user-name testuser --passwo
rd devops@2468#
{
    "LoginProfile": {
        "PasswordResetRequired": false,
        "UserName": "testuser",
        "CreateDate": "2017-06-09T10:17:12.926Z"
    }
}
```

Creating Access key &SecretAccess key for User

```
satish@devops:~/.aws$ aws iam create-access-key --user-name testuser
{
    "AccessKey": {
        "SecretAccessKey": "hKgqu4BNCsaDjFMaepnDQd77Z4X4DZeanThLkga2",
        "AccessKeyId": "AKIAIM3MCXZV72K7DYBQ",
        "Status": "Active",
        "CreateDate": "2017-06-09T10:21:39.644Z",
        "UserName": "testuser"
    }
}
```

Security Group Creation

```
satish@devops:~/.aws$ aws ec2 create-security-group --group-name testgroup --description "testgroups"
{
    "GroupId": "sg-160c376d"
}
```

Inbound Rules for Security Group

:~/.aws$ aws ec2 authorize-security-group-ingress --group-name testgroup --protocol tcp --port 8080 --port 22 --cidr 0.0.0.0/0

```
:~/.aws$ aws ec2 authorize-security-group-ingress --group-name testgroup --protocol tcp --port 8080 --port 22 --cidr 0.0.0.0/0
```

Creating Key Pair

```
devops:~/.aws$ aws ec2 create-key-pair --key-name test-key --query 'KeyMaterial' --output text > test.pem
```

```
~/.aws$ aws ec2 create-key-pair --key-name test-key --query 'KeyMaterial' --output text >test.pem
```

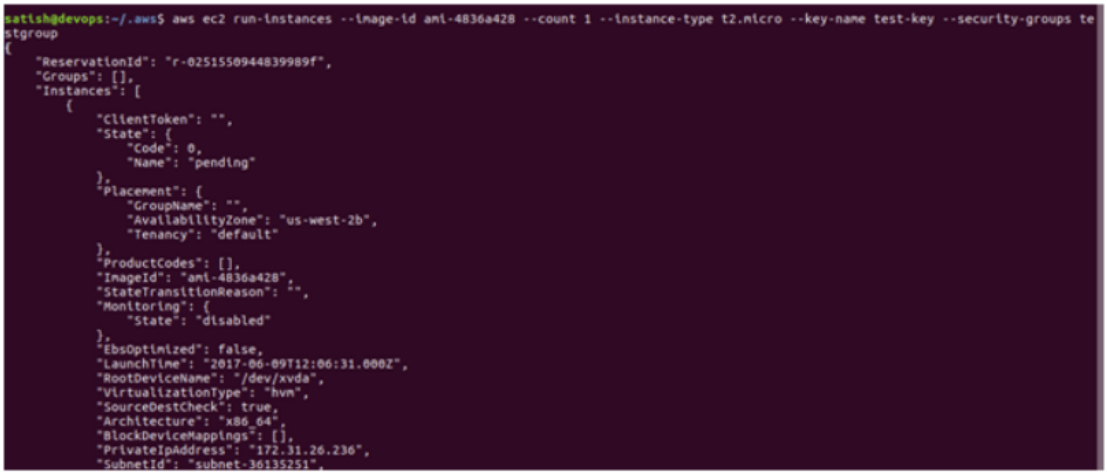
we can also create key pair by giving for only Region also

```
~/.aws$ aws –region us-west-2 ec2 create-key-pair --key-name test-key3 --query 'KeyMaterial' --output text >test.pem
```

Setup & Manage EC2

Create Instances

```
:~/.aws$ aws ec2 run-instances --image-id ami-4836a428 --count 1 --instance-type t2.micro --key-name test-key --security-groups testgroup
```




```
@devops:~$ aws ec2 start-instances --instance-ids i-06f132d68ec51c93f
```

```
satish@devops:~$ aws ec2 start-instances --instance-ids i-06f132d68ec51c93f
{
    "StartingInstances": [
        {
            "InstanceId": "i-06f132d68ec51c93f",
            "CurrentState": {
                "Name": "pending",
                "Code": 0
            },
            "PreviousState": {
                "Name": "pending",
                "Code": 0
            }
        }
    ]
}
```

Stop instances

```
devops:~$ aws ec2 stop-instances --instance-ids i-06f132d68ec51c93f
```

```
satish@devops:~$ aws ec2 stop-instances --instance-ids i-06f132d68ec51c93f
{
    "StoppingInstances": [
        {
            "CurrentState": {
                "Name": "stopping",
                "Code": 64
            },
            "InstanceId": "i-06f132d68ec51c93f",
            "PreviousState": {
                "Name": "stopping",
                "Code": 64
            }
        }
    ]
}
```

Code -> (integer)

The low byte represents the state. The high byte is an opaque internal value and should be ignored.

1.0 : pending
2.16 : running
3.32 : shutting-down
4.48 : terminated
5.64 : stopping
6.80 : stopped

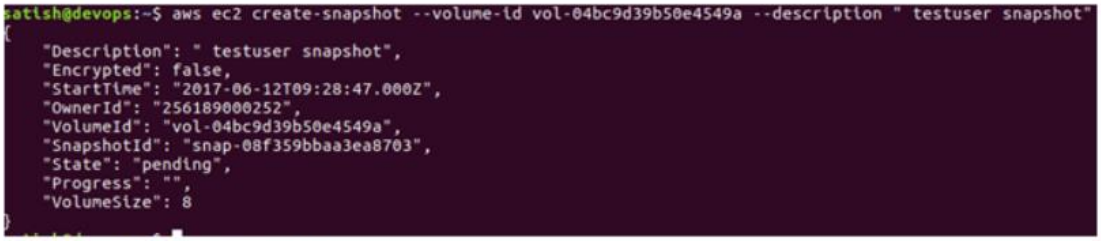
1.Terminate Instances

```
$ aws ec2 terminate-instances –instances-ids i-564845gt655
```

Snapshot & Volume

Snapshot Creation

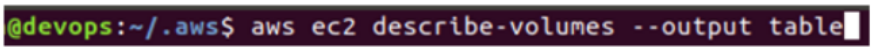
```
@devops:~/.aws$ aws ec2 create-snapshot --volume-id vol-04bc9d39b50e4549a
--description " testuser snapshot"
```



Manage Volume Creation

Describe Volumes

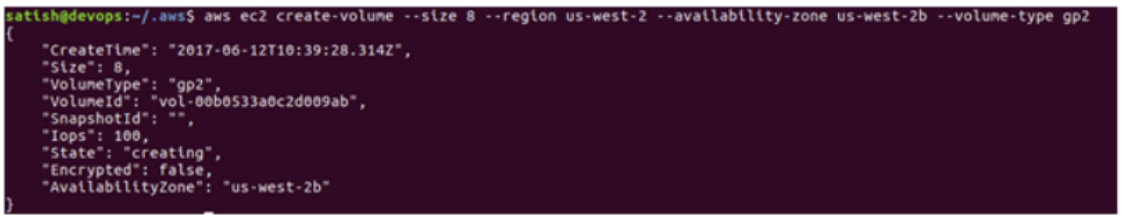
```
@devops:~/.aws$ aws ec2 describe-volumes --output table
```




```
@devops:~/.aws$ aws ec2 describe-volumes --filters Name=volume-id,Values=vol-04bc9d39b50e4549a --output table
```

Creating Volume

```
@devops:~/.aws$ aws ec2 create-volume --size 8 --region us-west-2 --availability-zone us-west-2b --volume-type gp2
```



--volume-type (string)

Possible values:

1.standard

2.io1 -- Provisioned IOPS SSD

3.gp2 -- General Purpose SSD

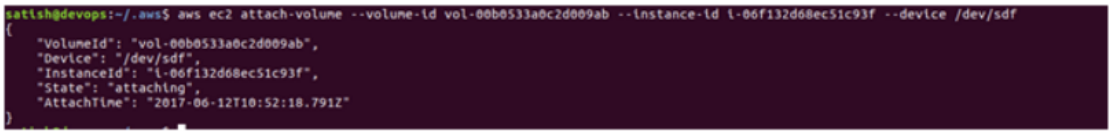
4.sc1 --Cold HDD

5.st1 --Throughput Optimized HDD

Default value: standard (Magnetic Volumes)

Attach Volume

devops:~/.aws$ aws ec2 attach-volume --volume-id vol-00b0533a0c2d009ab --instance-id i-06f132d68ec51c93f --device /dev/sdf



De-Attach the volume

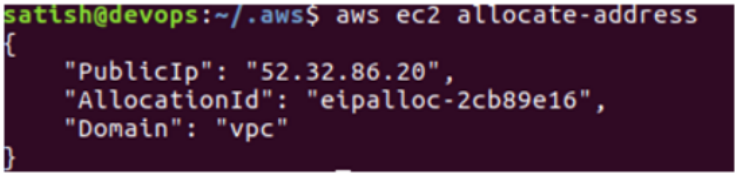
```
devops:~/.aws$ aws ec2 detach-volume --volume-id vol-1234567890abcdef0 --
force
```

Note: Before detach the volume we should be unmount the volume within your Operating System.

Elastic IP

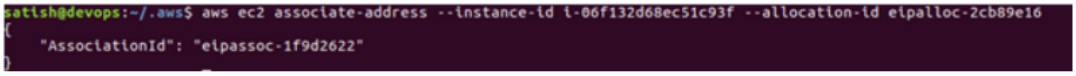
Elastic IP Address Allocation

```
@Devops:~/.Aws$ Aws Ec2 Allocate-Address
```



Attaching Elastic IP To Instances

```
@Devops:~/.Aws$ Aws Ec2 Associate-Address --Instance-Id I-
06f132d68ec51c93f --Allocation-Id Eipalloc-2cb89e16
```

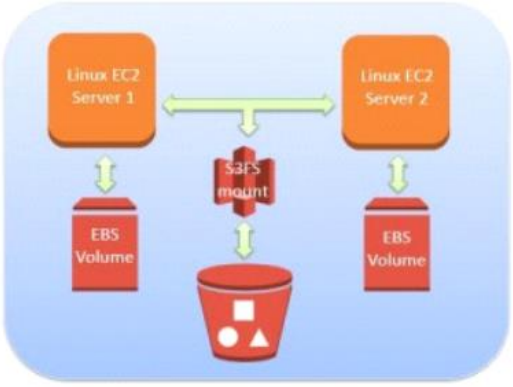


Get List Of Elastic IP's

```
$ Aws Ec2 Describe-Addresses
```

Disassociate Elastic IP

```
@Devops:~/.Aws$ Aws Ec2 Disassociate-Address --Association-Id Eipassoc-Cf962df2
```



Delete Elastic IP

Devops:~/.Aws$ Aws Ec2 Release-Address --Allocation-Id Eipalloc-2cb89e16

Controlling Multiple Machines

The moment more than one machine is defined within a Vagrantfile, the usage of the various vagrant commands changes slightly. The change should be mostly intuitive. Commands that only make sense to target a single machine, such as vagrant ssh, now require the name of the machine to control. Using the example above, you would say vagrant ssh web or vagrant ssh db.

Other commands, such as vagrant up, operate on every machine by default. So if you ran vagrant up, Vagrant would bring up both the web and DB machine. You could also optionally be specific and say vagrant up web or vagrant up db.

S3cmd

Documented By VenuRupani.

s3md is a command line utility used for creating s3 buckets, uploading, retrieving and managing data to Amazon s3 storage.

Install S3cmd Package

s3cmd is available in default rpm repositories for CentOS, RHEL and Ubuntu systems, you can install it using simply executing following commands on your system.

On Ubuntu/Debian:

```
$ sudo apt-get install s3cmd
```

Install Latest S3cmd From Source: -

If you are not getting latest version of s3cmd using above package managers, You can install last s3cmd version on your system using source code.
Visit this url (http://ufpr.dl.sourceforge.net/project/s3tools/s3cmd/)

or
Use below command to download latest version of s3cmd.

```
$ wgethttp://ufpr.dl.sourceforge.net/project/s3tools/s3cmd/1.6.1/s3cmd-1.6.1.tar.gz
$ tar xzf s3cmd-1.6.1.tar.gz
```

Now install it using below command with source files.

```
$ cd s3cmd-1.6.1
$ sudo python setup.py install
```

Configure S3cmd Environment: -

In order to configure s3cmd we would require Access key and Secret key of your S3 Amazon account. After getting key files, use below command to configure s3cmd.

NOTE: - Before executing below command create a user with full permissions of s3 access in your aws account.

```
$ s3cmd – configure
```

After press, the above command please enter the key file information like given below. Enter new values or accept defaults in brackets with Enter. Refer to user manual for detailed description of all options.

```
Access key and Secret key are your identifiers for Amazon S3
Access Key: xxxxxxxxxxxxxxxxxxxxx
Secret Key: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Encryption password is used to protect your files from reading
by unauthorized persons while in transfer to S3

```
Encryption password: xxxxxxxxxx Path to GPG program [/usr/bin/gpg]:
```

When using secure HTTPS protocol all communication with Amazon S3 servers is protected from 3rd party eavesdropping. This method is slower than plain HTTP and can't be used if you're behind a proxy

```
Use HTTPS protocol [No]: Yes
```

New settings:

```
Access Key: xxxxxxxxxxxxxxxxxxxxx
Secret Key: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Encryption password: xxxxxxxxxx
Path to GPG program: /usr/bin/gpg
Use HTTPS protocol: True
HTTP Proxy server name:
HTTP Proxy server port: 0
Test access with supplied credentials? [Y/n] Y
Please wait, attempting to list all buckets...
Success. Your access key and secret key worked fine :-)
Now verifying that encryption works...
Success. Encryption and decryption worked fine :-)
Save settings? [y/N] y
Configuration saved to '/root/.s3cfg'
```

Now setup is ready to create s3 buckets, uploading, retrieving and managing data to Amazon s3 storage.

Uses Of S3cmd Command Line: -

Once configuration is successfully completed. Now find below command details to how to manage s3 buckets using commands.

1.List All S3 Bucket

Use following command to list all s3 buckets in your aws account.

```
$ s3cmd ls
```

2.Creating New Bucket

To create a new bucket in Amazon s3 use below command. It will create bucket named venudevops(create your bucket name as your wish)in aws account.

```
$ s3cmd mb s3://venudevops
Bucket 's3://venudevops/' created
```

3.Uploading File In Bucket

Below command will upload file devops.txt to s3 bucket using s3cmd command.

```
$ s3cmd put devops.txt s3://venudevops/ devops.txt ->
s3://venudevops/devops.txt [1 of 1] 190216 of 190216 100% in 0s 1668.35kB/s
done
```

4.Uploading Directory In Bucket

If we need to upload entire directory use -r to upload it recursively like below.

```
$ s3cmd put -r backup s3://venudevops/
backup/devops1.txt -> s3://venudevops/backup/devops1.txt [1 of 2]
9984 of 9984 100% in 0s 18.78 kB/s done
backup/devops2.txt -> s3://venudevops/backup/devops2.txt [2 of 2]
0 of 0 0% in 0s 0.00 B/s done
```

>

Make sure you are not adding trailing slash in upload directory named backup (eg: backup/), else it will upload only content of backup directory only.

```
$ s3cmd put -r backup/ s3://venudevops/
backup/devops1.txt -> s3://venudevops/devops1.txt [1 of 2]
9984 of 9984 100% in 0s 21.78 kB/s done
backup/devops2.txt -> s3://venudevops/devops2.txt [2 of 2]
0 of 0 0% in 0s 0.00 B/s done
```

5.List Data Of S3 Bucket

List the objects of s3 bucket using ls switch with s3cmd.

```
$ s3cmd ls s3://venudevops/
DIR s3://venudevops/backup/
2017-06-03 10:58 190216 s3://venudevops/file.txt
```

6.Download Files From Bucket

Sometimes if we need to download files from s3 bucket, Use following commands to download it.

```
s3://venudevops/devops.txt -> ./devops.txt [1 of 1]
4 of 4 100% in 0s 10.84 B/s done
```

7.Remove Data Of S3 Bucket

To remove files are folder from s3 bucket use following commands. Removing file from s3 bucket

```
$ s3cmd del s3://venudevops/devops.txt
File s3://venudevops/devops.txt deleted
Removing directory from s3 bucket
```

```
$ s3cmd del s3://venudevops/backup
File s3://venudevops/backup deleted
```

8.Remove S3 Bucket

If we don't need s3 bucket any more, we can simply delete it using following command. Before removing bucket make sure its empty.

```
$ s3cmd rb s3://venudevops
ERROR: S3 error: 409 (BucketNotEmpty): The bucket you tried to delete is not empty
```

s3cmd-sync

Documented By VenuRupani.

1.Syncing Files From Local => S3 Bucket

For example I want to sync my local directory /root/mydir/ to S3 bucket directory s3://venudevops/mydir/ where venudevops is bucket name. I have created some new files in /root/mydir/ and sync to s3 bucket using following command.

```
$ s3cmd sync /root/mydir/ s3://venudevops/mydir/
```

[Sample Output]

```
/root/mydir/index.php -> s3://venudevops/mydir/index.php [1 of 2]
397 of 397 100% in 0s 4.02 kB/s done
/root/mydir/readme.html -> s3://venudevops/mydir/readme.html [2 of 2]
9202 of 9202 100% in 0s 103.62 kB/s done
Done. Uploaded 9599 bytes in 0.3 seconds, 27.92 kB/s
```

Note: Do not forgot to add trailing slash (/) in local directory path when specifying s3 bucket with full directory path.

To keep preserve file attributes like date/time etc use -p or –preserve parameter like below

```
$ s3cmd sync /root/mydir/ --preserve s3://venudevops/mydir/
```

If we want to sync only newly created file on source use –skip-existing parameter. It will skip all files which already exists on destination either its modified on source.

```
# s3cmd sync /root/mydir/ --skip-existing s3://venudevops/mydir/
```

If you want to delete all files from s3 bucket which has removed from local use –delete-removed parameter.

```
$ s3cmd sync /root/mydir/ --delete-removed s3://venudevops/mydir/
```

2. Syncing Files From S3 Bucket => Local Directory

For this example I am again using same folder and bucket used above. To test this i have put some extra files in s3 bucket (s3://venudevops/mydir/) and executed following command to sync all files to local directory.

```
$ s3cmd sync s3://venudevops/mydir/ /root/mydir/
```

[Sample Output]

```
s3://venudevops/mydir/logo.jpg -> /root/mydir/logo.jpg [2 of 3]
7219 of 7219 100% in 0s 125.28 kB/s done
s3://venudevops/mydir/user.php -> /root/mydir/user.php [3 of 3]
40380 of 40380 100% in 0s 596.33 kB/s done
Done. Downloaded 47599 bytes in 0.3 seconds, 184.40 kB/s
```

We can also used–preserve, –skip-existing and –delete-removed parameters during syncing files from S3 bucket to Local directory as followings.

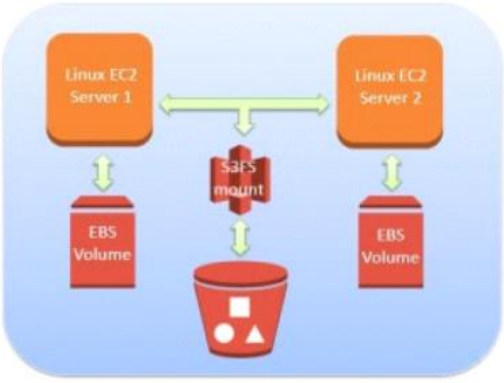
```
$ s3cmd sync s3://venudevops/mydir/ --preserve /root/mydir/ $ s3cmd sync
s3://venudevops/mydir/ --skip-existing /root/mydir/ $ s3cmd sync
s3://venudevops/mydir/ --delete-removed /root/mydir/
```

Read more about s3cmd sync http://s3tools.org/s3cmd-sync

Mount an Amazon S3 Bucket to a Local Linux File System

Documented By Praveen.

S3FS is a FUSE (File System in User Space) will mount Amazon S3 as a local file system. S3FS has an ability to manipulate Amazon S3 bucket in many useful ways. If you wish to access your Amazon S3 bucket without mounting it on your server, you can use s3cmd command line utility to manage S3 bucket..



Install Prerequisite Packages

```
yum install automake fuse fuse-develgcc-c++ git libcurl-devel libxml2-devel
make openssl-devel
```

Download, Compile And Install The Latest Version Of FUSE Module

```
cd /usr/src/
yum install wget unzip
wgethttp://www.techmarksolutions.co.uk/downloads/fuse-2.8.4.tar.gz
tarxzvf fuse-2.8.4.tar.gz
cd fuse-2.8.4
./configure --prefix=/usr/
make
make install
export PKG_CONFIG_PATH=/usr/lib/pkgconfig:/usr/lib64/pkgconfig/
ldconfig
```

#Manually load the FUSE module for the first time

```
modprobe fuse
```

Verify FUSE Installation

```
# pkg-config --modversion fuse
```

Download, Compile And Install The Latest Version Of S3FS

```
cd /usr/src/
wgethttps://github.com/s3fs-fuse/s3fs-fuse/archive/master.zip
unzip master.zip
cd s3fs-fuse-master
./autogen.sh
./configure --prefix=/usr --with-openssl
make
make install
```

Verify S3FS Installation

```
# s3fs –version
```

Create An S3fs Password File For Storing Your AWS Access Key ID And Secret Access Key.

[*Create IAM User and Create S3 Bucket steps are given below*]

The default location for the s3fs password file can be created:

1.using a .passwd-s3fs file in the users home directory (i.e. ~/.passwd-s3fs)

2.using the system-wide /etc/passwd-s3fs file

3.(AccessKeyId:SecretAccessKey)

```
# cd # vi /etc/passwd-s3fs (AccessKeyId:SecretAccessKey)
# cd /etc
```

Change The Permissions Of Passward File

```
# chmod 400 passwd-s3fs
# cd /usr/src
```

Create A Directory For Mounting The S3fs File System

```
# mkdir s3mnt
```

Mount Bucket To Specific Directory

```
# s3fs s3fs-aws /usr/src/s3mnt/ -o passwd_file=/etc/passwd-s3fs # cd s3mnt/
```

Create Directories And Files In S3mnt

```
# mkdir 1 2 3 4 5 6
```

Check S3 Bucket In Aws
Verify The S3fs Mounted File System.

```
grep s3fs /etc/mtab
(*s3fs /usr/src/s3mnt fuse.s3fs rw,nosuid,nodev 0 0*)
df -Th /usr/src/s3mnt/
(*Filesystem Type Size Used Avail Use% Mounted on
s3fs fuse.s3fs 256T 0 256T 0% /usr/src/s3mnt*)
```

Manually Unmount The Virtual Drive Using The Umount Command

```
# umount /usr/src/s3mnt/
```

Create IAM User

Create an IAM user that will be used to access S3 from your EC2 instances.

1.Login to AWS web console
2.Open IAM | Users
3.Select Create New User
4.Enter a user name
1. s3user
5.Copy generated access keys
6.Set a complicated/long password for the user

Create S3 Bucket

7. Login to AWS web console
8. Select Services S3
9. Create a S3 bucket
10.s3fs-aws
11. Add bucket policy to allow IAM User and VPC Endpoint access to all files inside the bucket.

Commands And Details

1.Modprobe

modprobe is a Linux program originally written by Rusty Russell and used to add a loadable kernel module (LKM) to the Linux kernel or to remove a LKM from the kernel. It is commonly used indirectly: udev relies upon modprobe to load drivers for automatically detected hardware.

2.Ldconfig

ldconfig is a program that is used to maintain the shared library cache. This cache is typically stored in the file /etc/ld.so.cache and is used by the system to map a shared library name to the location of the corresponding shared library file

3. Make And Make Install

make connects the libs to the source and creates the required links and sets it up for the final phase it also parses the human readable to machine readable make install This is the final phase where the compiler will create the binary files and moves all required files / executable and associated libs to their appropriate directories.

4../Autogen.Sh

provides automatic build system preparation auto tools for preparing a build for compilation, verifying functionality, and overcoming common build preparation issues.

5. Pkg-Config --Modversion

Requests that the version information of the libraries specified on the command line be displayed. If pkg-configcan find all the libraries on the command line, each library's version string is printed to stdout, one version per line. In this case pkg-configexits successfully. If one or more libraries is unknown, pkg-configexits with a nonzero code, and the contents of stdout are undefined.

S3FS Description

S3fsis a direct mapping of S3 to a filesystem paradigm. Files are mapped to objects. Filesystem metadata (e.g., ownership and file modes) are stored inside the object's meta data. Filenames are keys, with "/" as the delimiter to make listing more efficient, etc.

That's significant because it means there is nothing terribly magical about a bucket being read/written to by s3fs, and in fact you can mount any bucket with s3fs to explore it as a filesystem.

s3fs's main advantage is its simplicity. There are however a few gotchas:

If you're using s3fs to access a bucket it didn't create and have objects in it that have directory-like components in their names (e.g., mypath/myfile), you'll need to create a dummy directory in order to see them (e.g., mkdirmypath).

The project seems to be "regretware". The last open source release was in August 2008. Since then the author seems to have continued all development of new features (e.g., encryption, compression, multi-user access) as a commercial license (subcloud), and with that inherent conflict of interest the future of the GPLed licensed open source version is uncertain.

Advantages Of S3FS

If you're using s3fs to access a bucket it didn't create and have objects in it that have directory-like components in their names (e.g., mypath/myfile), you'll need to create a dummy directory in order to see them (e.g., mkdirmypath).

No embedded documentation. Probably another side-effect of the proprietary version, though the available options are documented no the web site.

Inherits S3's limitations: no file can be over 5GB, and you can't partially update a file so changing a single byte will re-upload the entire file.

Inherits S3's performance characteristics: operation on many small files are very efficient (each is a separate S3 object after all)

Though S3 supports partial/chunked downloads, s3fs doesn't take advantage of this so if you want to read just one byte of a 1GB file, you'll have to download the entire GB.

s3fs supports a disk cache, which can be used to mitigate this limitation

Features

- large subset of POSIX including reading/writing files, directories, symlinks, mode, uid/gid, and extended attributes
- compatible with Amazon S3, Google Cloud Storage, and other S3-based object stores
- large files via multi-part upload
- renames via server-side copy
- optional server-side encryption
- data integrity via MD5 hashes
- in-memory metadata caching
- local disk data caching
- user-specified regions, including Amazon GovCloud
- authenticate via v2 or v4 signatures

S3FS Limitations

UID/GID

Now, there is no full UID/GID support. All files will be owned by root. Additionally, if you allow others to access the bucket (using the -o allow_other option), others can also remove files.

Excessive Time-Outs

Currently s3fs can hang the CPU if you have lots of time-outs. This is not a fault of s3fs but rather libcurl. This happens when you try to copy thousands of files in 1 session, it doesn't happen when you upload hundreds of files or less.

Moving Large Files

Moving, renaming, or erasing files may take considerable time since the whole file needs to be accessed first. A workaround could be to use s3fs's cache support with the -o use_cache= mount option.

File Size

S3FS has a file size limit of 64GB for the current version (limited by s3fs, not Amazon).

Directory Support

Prior to configuring s3fs on my test system, I used the AWS Management Console to create several directories and to upload some files to the target S3 bucket I intended to mount (idevelopment-software). After mounting the bucket to the local file system, I was unable to see any of the directories. I could see the files I uploaded to the root directory of the bucket, but was unable to see any of the directories. I went back to the AWS Management Console and even to S3Fox and verified that the directories did indeed exist.

it turns out that S3 does not have a native concept of a folder (i.e. directory). It is up to each S3 client tool (AWS Console, S3Fox, s3cmd, s3fs) to implement their own strategy for handling folders. Without a common specification in place for storing folders, certain S3 client tools will build directory structures that are not compatible with one another. In my example, the folders I created in the AWS Management Console were not compatible with s3fs.

Unfortunately, the only solution for now is to adopt a single S3 tool exclusively to modify the contents of a bucket. For the S3 buckets I intend to mount, I will only be using s3fs against the contents of those buckets. (server side copies are not possible - due to how FUSE orchestrates the low level instructions, the file must first be downloaded to the client and then uploaded to the new location)

Conclusions

s3fs: safe, efficient storage of medium-large files. Perfect for backup / archiving purposes.

Summary:

- Cloud computing is the way forward in IT industry, there is no going back now. Majority of the people in IT industry some or the other way are consuming Cloud based services.
- Amazon Web Services is the market leader ind Public domain. That is because its very innovative and every now and then comes up with new services and features.
- If you are using AWS you should be checking its news all the time.
- https://aws.amazon.com/new/
- AWS services can be broadly divied into three categoris, SysOps, Developers and DevOps services. There major push is in Developers services now and majoriy of AWS services are Developers services, currently.
- But Most widely used services are SysOps services though, like Ec2, RDS, Route53, VPC etc.
- AWS services can be accessed by Commad lines, scripts and configuration management tools. This is called as programatic access and is used for automation.
- S3cmd, s3fs, AWSCli, are some command line tools to manage AWS services.

Conclusions:

In DevOps field you would be using lots cloud services and AWS is most widely used among them. There are other Cloud Providers like Microsoft Azure, Google Cloud &Rackspace.

After learning AWS services you should work on automating those services through your scripts or tools which we will see in later chapters.