Amazon Web Services

1. Create an AWS account

Steps to create free tier access:

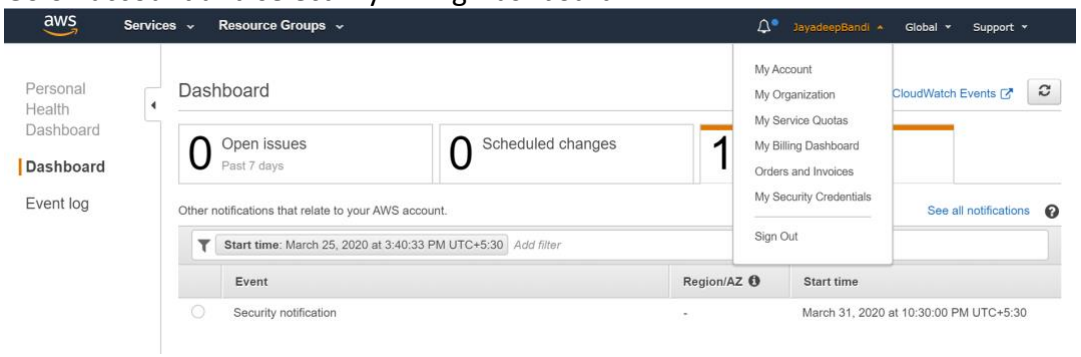
a. Go to the Amazon Web Services home page.
b. Choose Create an AWS Account.
   Note: If you've signed in to AWS recently, the button might say Sign in to the Console.
c. Enter your account information, and then choose Continue.
   Important: Be sure that you enter your account information correctly, especially your email address. If you enter your email address incorrectly, you won't be able to access your account. If Create a new AWS account isn't visible, first choose Sign in to a different account, and then choose Create a new AWS account.
d. Choose Personal or Professional.
   Note: Personal accounts and professional accounts have the same features and functions.
e. Enter your company or personal information.
f. Read and accept the AWS Customer Agreement.
   Note: Be sure that you read and understand the terms of the AWS Customer Agreement.
g. Choose Create Account and Continue.

   You receive an email to confirm that your account is created. You can sign in to your new account using the email address and password you supplied. However, you can't use AWS services until you finish activating your account.

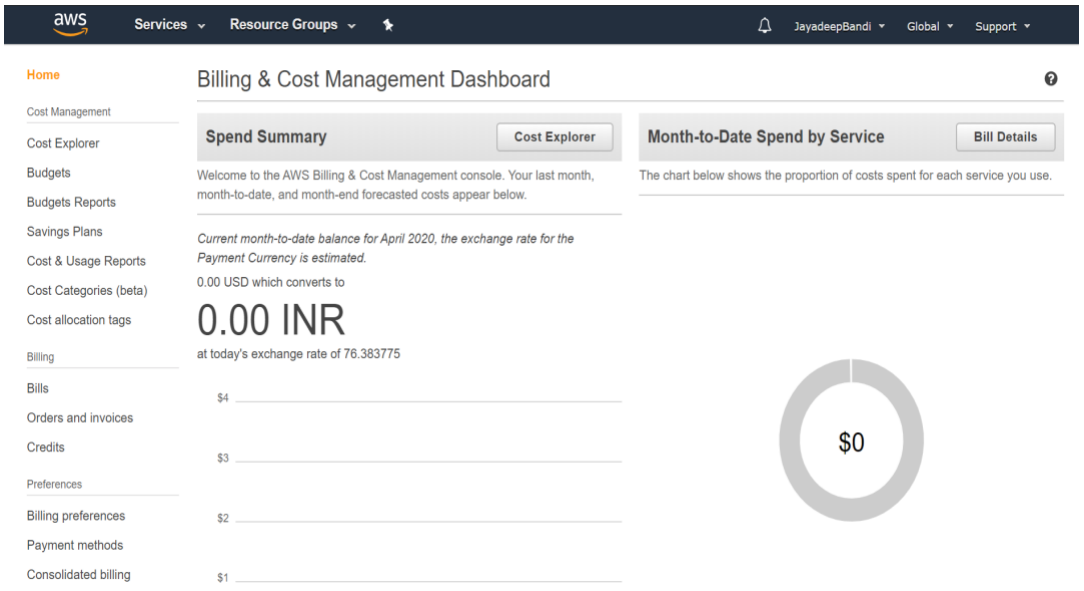
h. Add a payment method
   On the Payment Information page, enter the information about your payment method, and then choose Secure Submit.
   Note: If you want to use a different address for your AWS account, choose Use a new address before you choose Secure Submit.
i. Verify your phone number
j. Choose whether you want to verify your account by Text message (SMS) or a Voice call.
k. Choose your country or region code from the list.
l. Enter a phone number where you can be reached in the next few minutes.
m. Enter the code displayed in the captcha.
n. When you're ready, choose Contact me. In a few moments, an automated system will contact you.
   Note: If you chose to verify your account by SMS, choose Send SMS instead.
o. Enter the PIN you receive by text message or voice call, and then choose Continue.

2. AWS Budget Setup

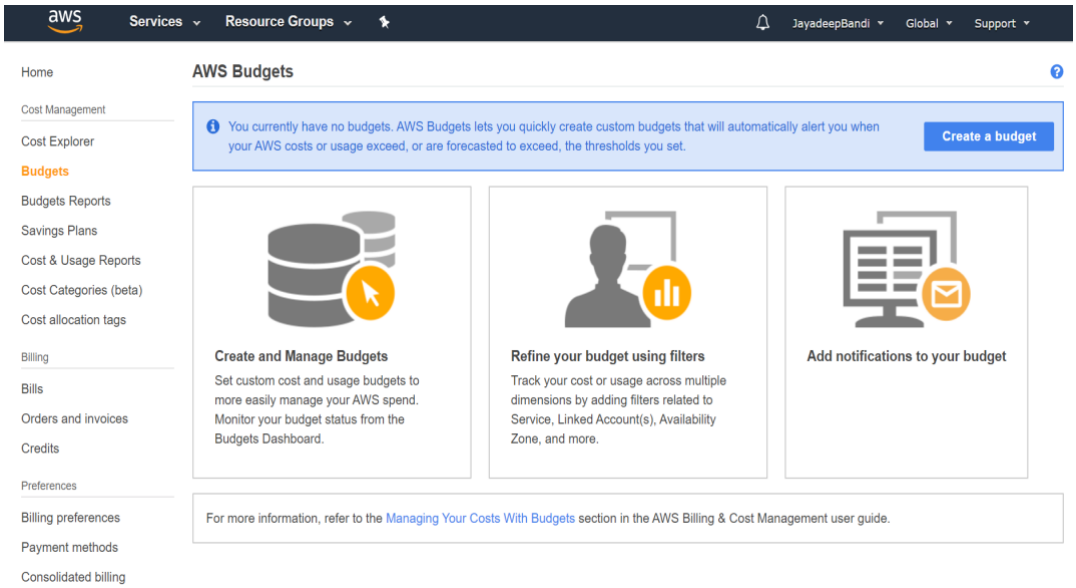
a. Go on account and select My Billing Dashboard.



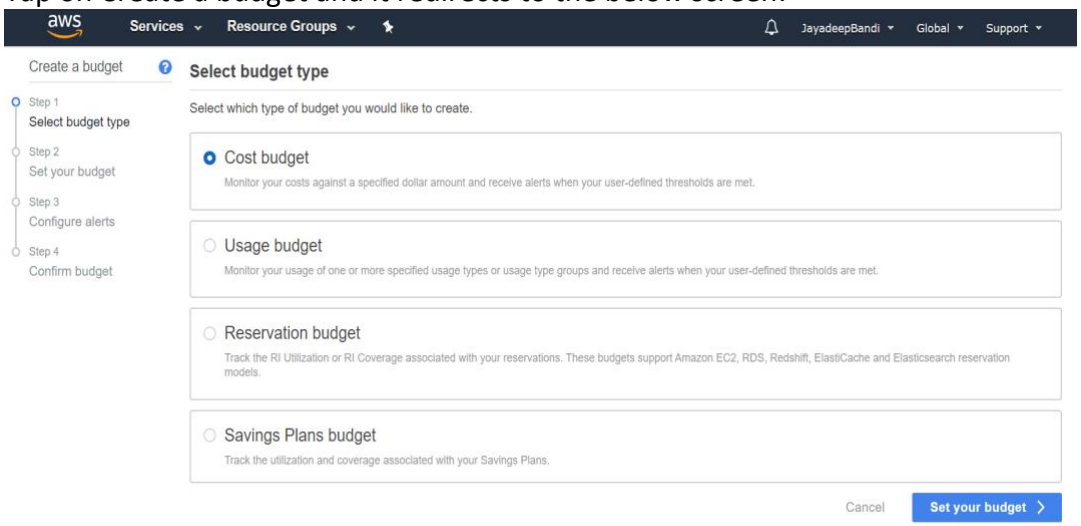
b. It redirects to the below screen.



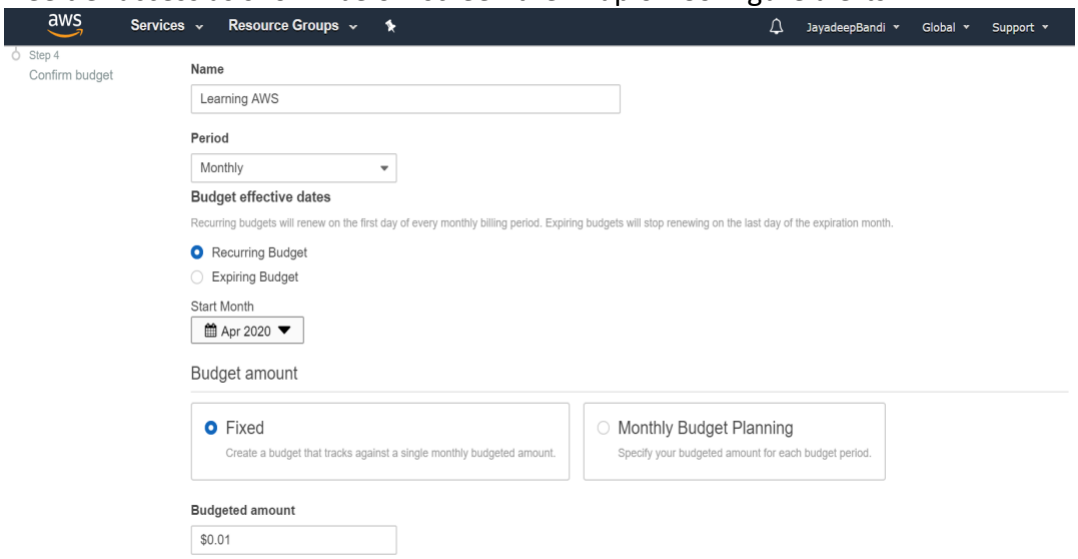
c. Tap on Budgets which is on the left hand side and it redirects to the below screen.



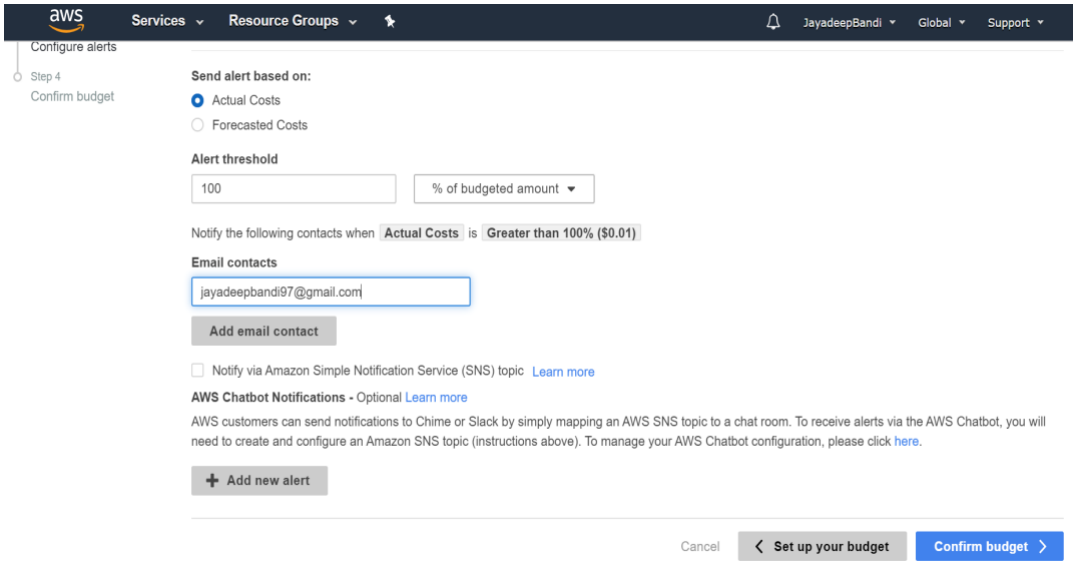
d. Tap on Create a budget and it redirects to the below screen.



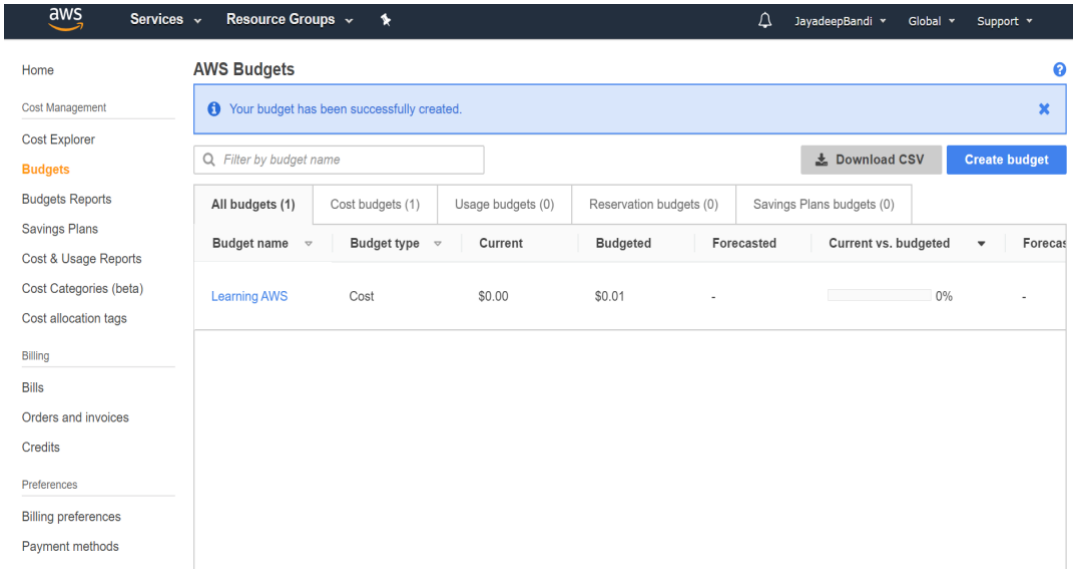
e. Select Cost budget and Tap on Set your budgeted amount to $0.01 for free tier access as shown below screen then Tap on Configure alerts.



f. It redirects to Confirm budget screen as shown in below screen and Tap on Confirm budget and followed by Create.

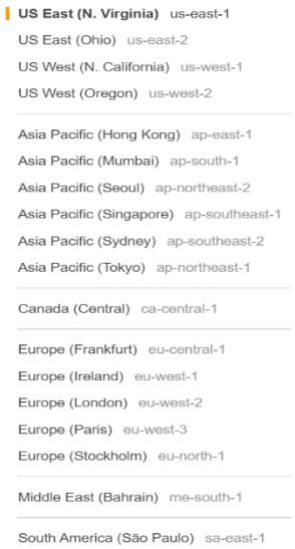


g. It creates Budget of my free tier access successfully as shown below screen.



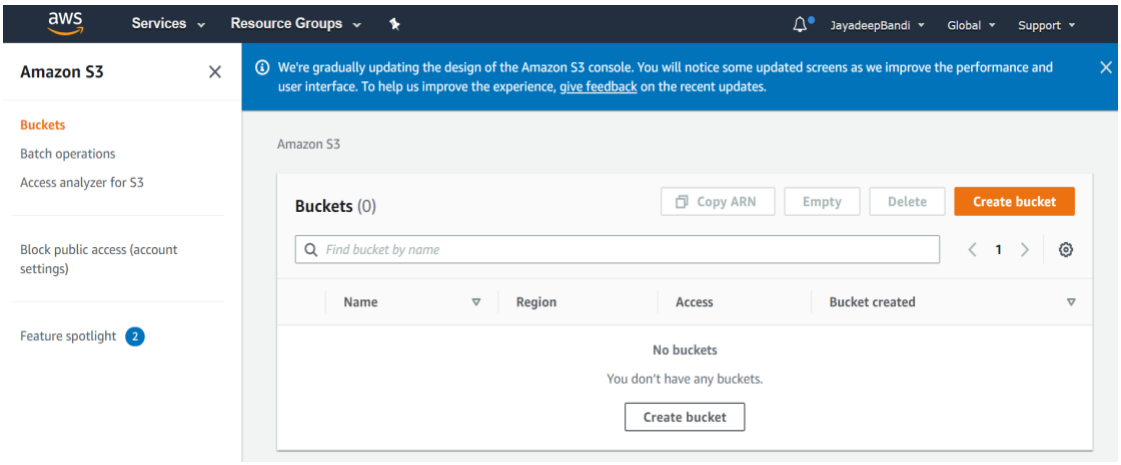
h. Whenever we want to check for the Bills, Tap on Bills option on the left hand side.

3. AWS Regions



- AWS has regions all around the world (us-east-1)
- Each region has availability zones (us-east-1a, us-east-1b…)
- Each availability zone is a physical data center in the region, but separate from the other ones (so that they're isolated from disasters)
- AWS Consoles are region scoped (except IAM and S3)

- IAM & S3 region is Global as shown in below screen.



4. IAM

a. IAM (Identity and Access Management)
b. Your whole AWS security is there:
   - Users: Usually a physical person, Person gets an account and that should not be the root account.
   - Groups: Users can be grouped together; Groups is usually Functions (admins, devops), Teams (engineering, design…). Contains users!
   - Roles: Internal usage within AWS resources, Users is physical person and Roles is for machine
c. Root account should never be used (and shared)
d. Users must be created with proper permissions
e. IAM is at the center of AWS
f. Policies are written in JSON (JavaScript Object Notation), It defines what each of the Users, Groups and Roles can and can't do
g. IAM has a **global** view
h. Permissions are governed by Policies (JSON)
i. MFA (Multi Factor Authentication) can be setup
j. IAM has predefined "managed policies"
k. It's best to give users the minimal amount of permissions they need to perform their job (least privilege principles)
l. For big enterprises they use **IAM Federation**
   - Big enterprises usually integrate their own repository of users with IAM
   - This way, one can login into AWS using their company credentials
   - Identity Federation uses the SAML standard (Active Directory)
m. IAM 101 Brain Dump
   - One IAM User per PHYSICAL PERSON
   - One IAM Role per Application
   - IAM credentials should NEVER BE SHARED
   - Never, ever, ever, ever, write IAM credentials in code. EVER.

- And even less, NEVER EVER COMMIT YOUR IAM credentials
- Never use the ROOT account for initial setup.
- Never use ROOT IAM Credentials

4.1. IAM Hands-On

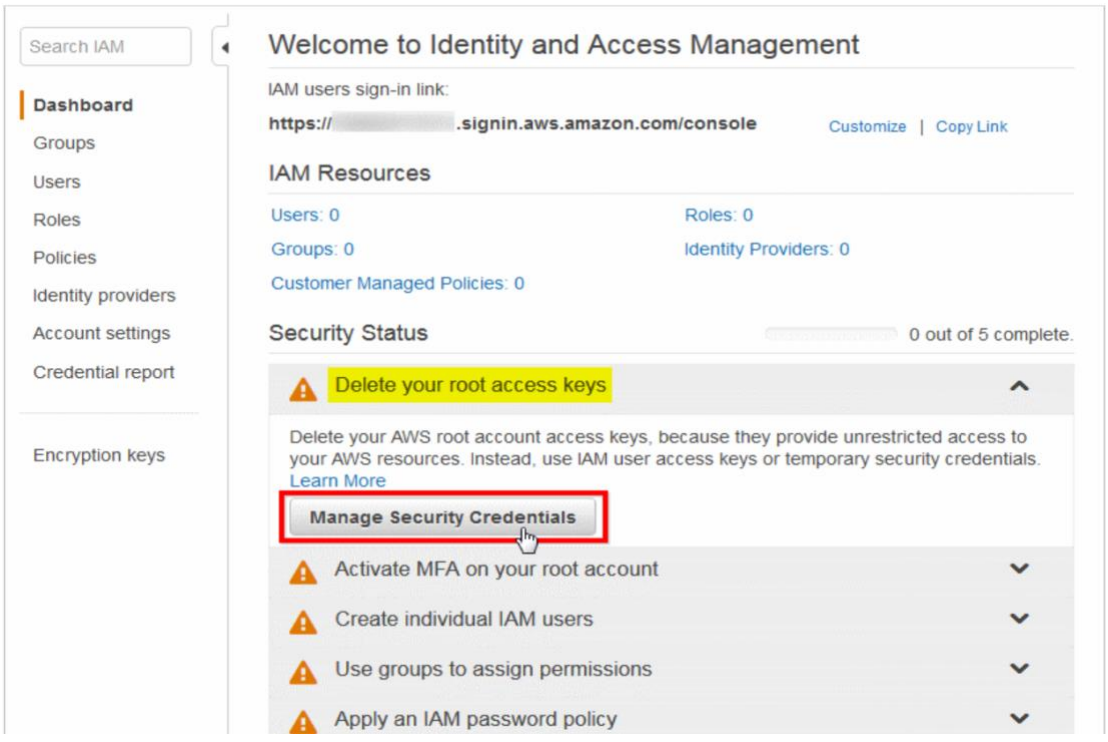
We do have two types of access keys

- Root Access Key
- IAM Access Key

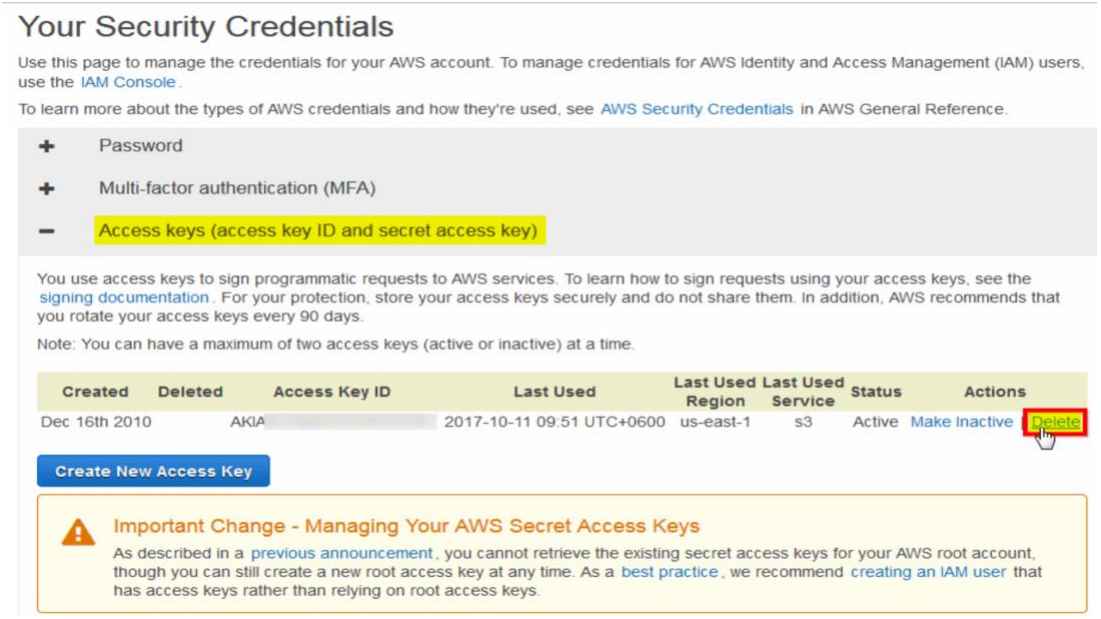
Root Access Keys: Root Access Keys provide unlimited access to your AWS resources. It's not recommended to use them in normal situations. AWS recommends to delete existing Root Access Keys and create IAM user and Access Keys limited to specific service or resource.

To Delete Root Access Keys:

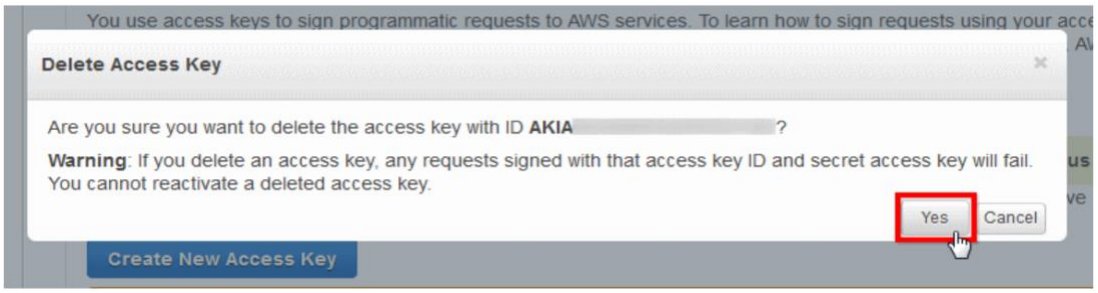
a. Login to AWS with Root Credentials
b. Search for IAM in the AWS Services and it redirects to IAM Dashboard
c. Navigate to **Security Status** and expand the **Delete your root access keys** section and Click **Manage Security Credentials.**



d. Expand the **Access Keys (access key id and secret access key)** section and Click on Delete link next to your access keys row as shown below.



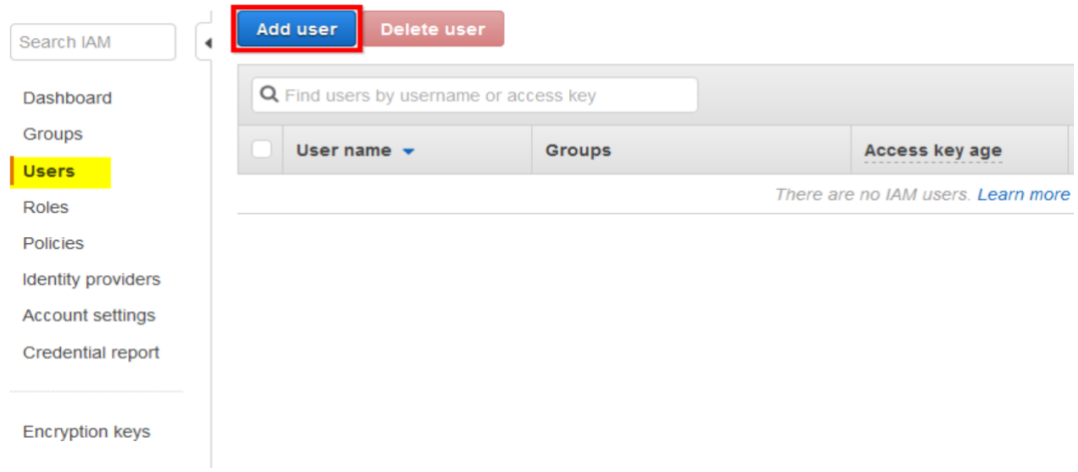
e. Confirm Access Keys deletion.



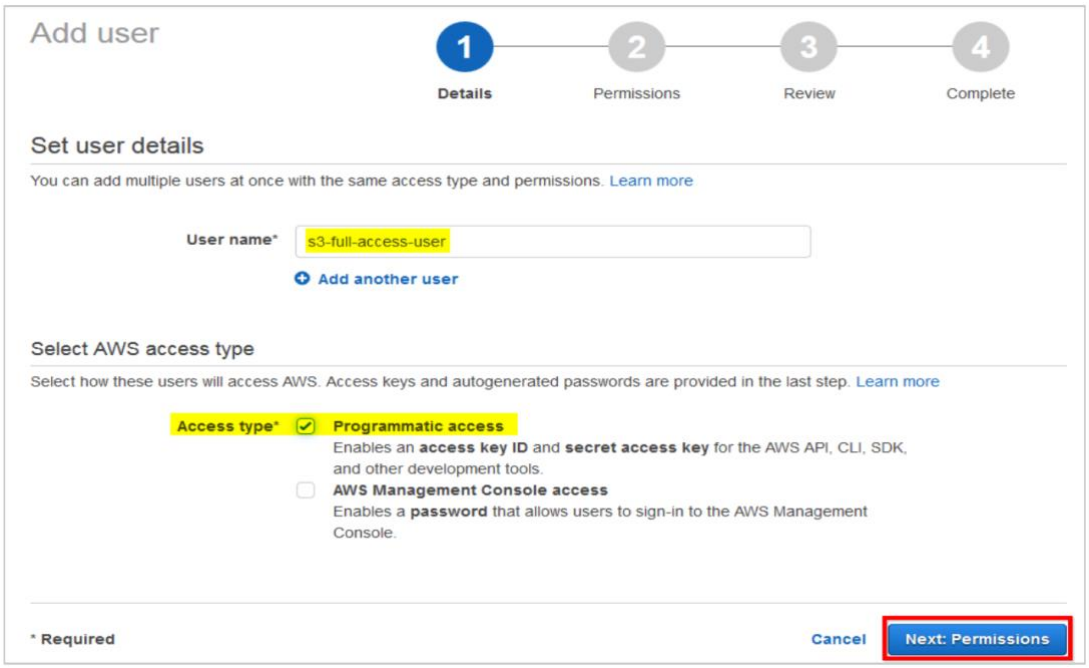
f. Root Access Keys are deleted. Now create IAM user and Access Keys limited to specific service or resource.
Note: Be sure to replace root access keys with IAM access keys in any programs/scripts that are currently using.

IAM Access Keys: To create IAM user and Access Keys follow the steps given below.

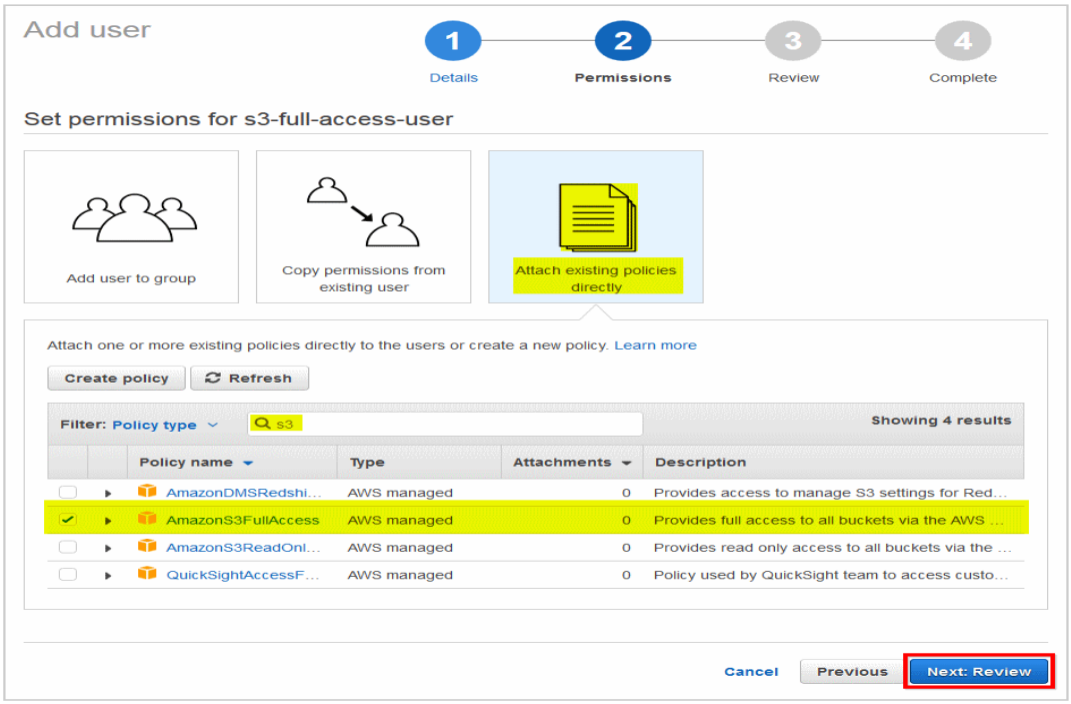
a. Open IAM Dashboard as described above and navigate to Users and Click on Add User



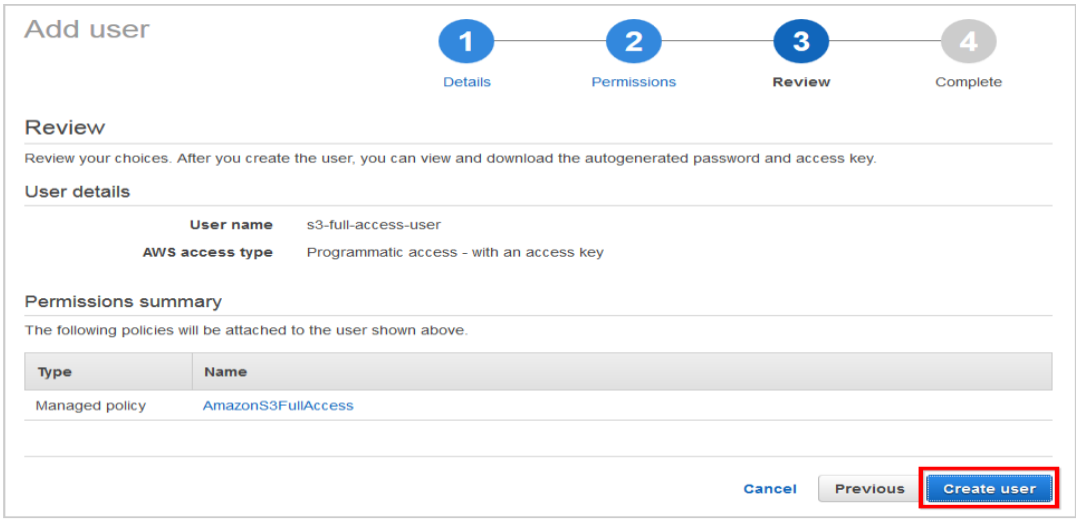
b. It redirects to below screen. Specify **User name**, Check **Programmatic access**, **AWS Management Console access** and Click **Next: Permissions.**



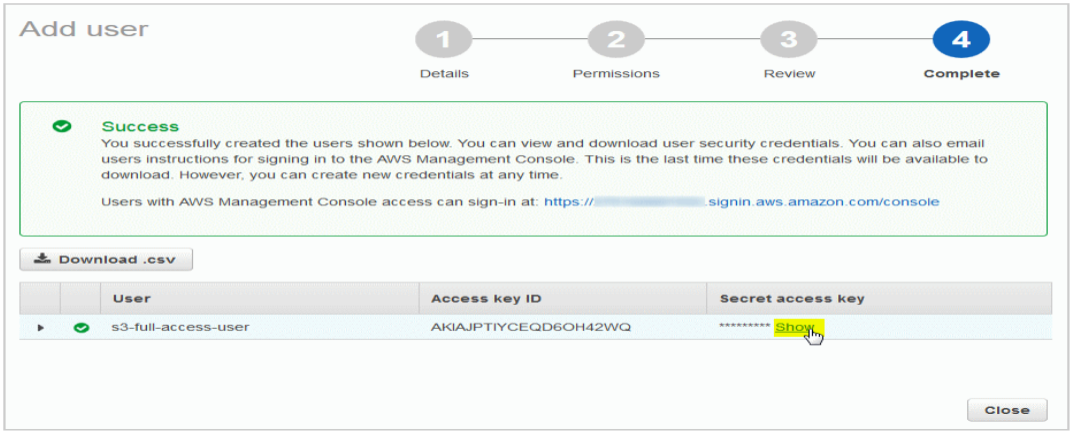
c. It redirects to Permissions Screen and Select **Attach existing policies directly**, select **AmazonAdminAccess** and tap on **Next: Review.**



d. It redirects to Review new user details and click **Create User**



e. The final page is as shown in the below screen.



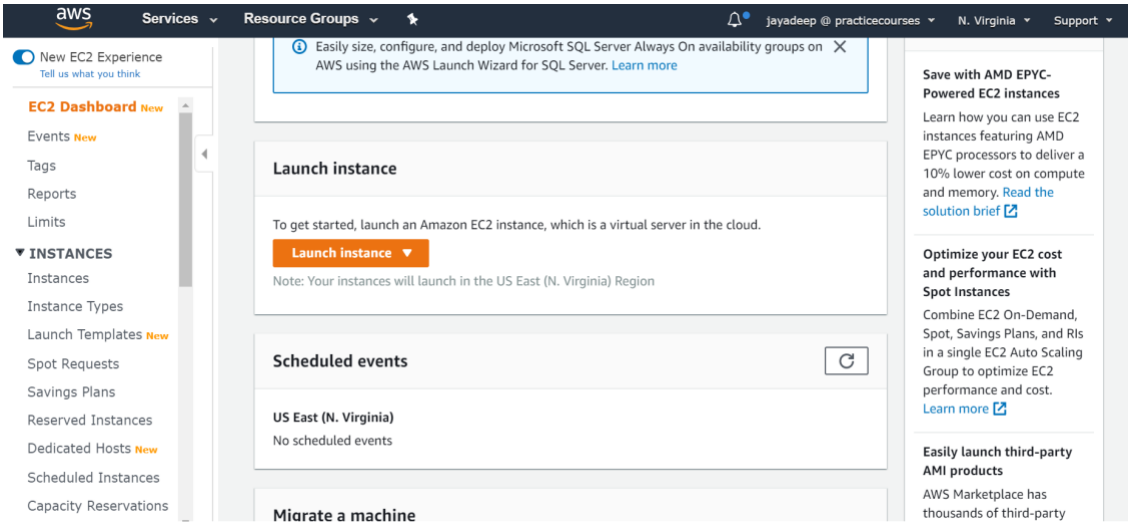
f. Tap on Download.csv file for the credentials of the IAM User.

5. What is EC2?

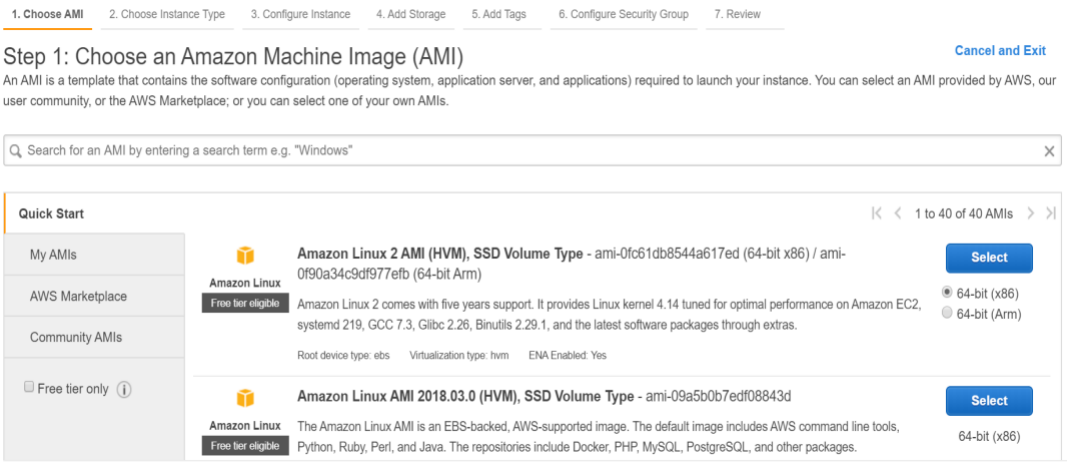
- EC2 is one of the most popular of AWS offering
- It mainly consists in the capability of:
  a. Renting virtual machines (EC2)
  b. Storing data on virtual drives (EBS)
  c. Distributing load across machines (ELB)
  d. Scaling the services using an auto-scaling group (ASG)
- Knowing EC2 is fundamental to understand how the Cloud works

5.1. Hands-On: Launching an EC2 Instance running Linux

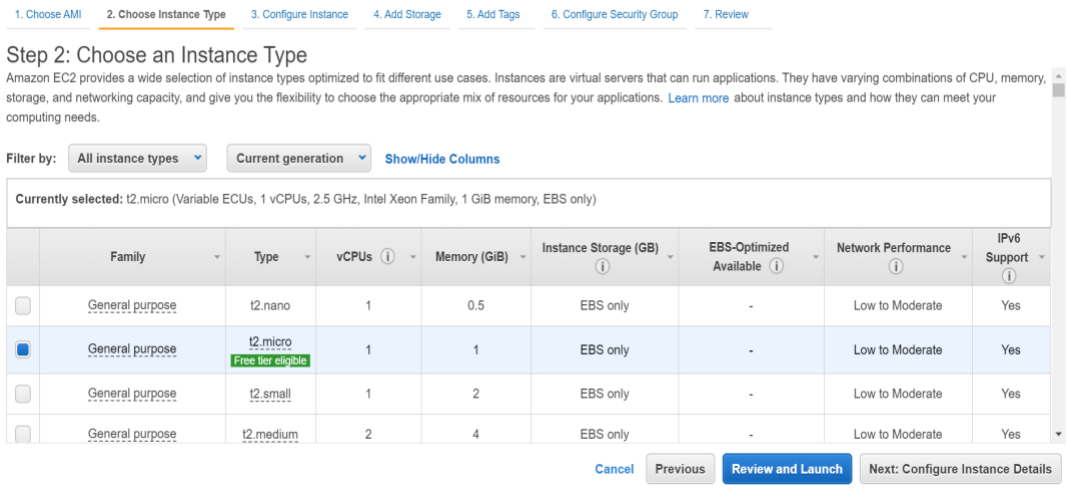
1. To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.



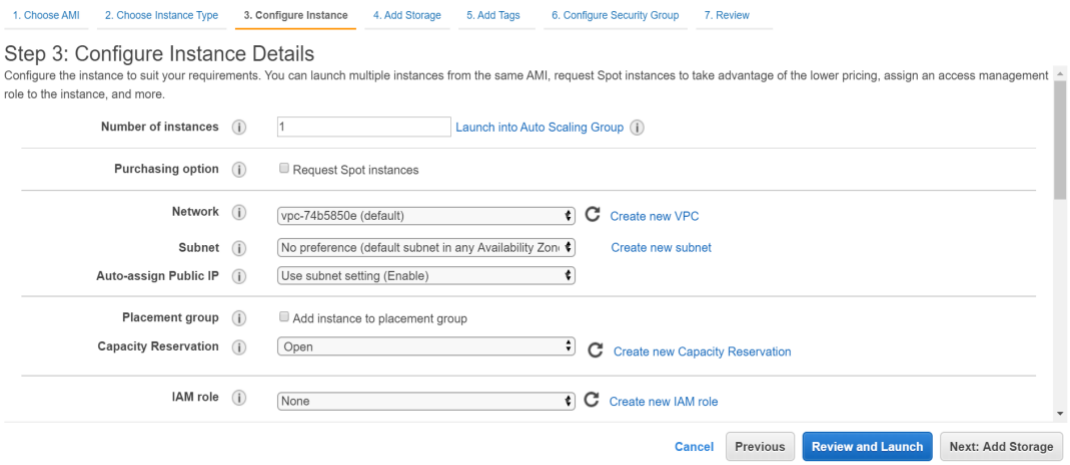
2. Choose an **Amazon Machine Image** (AMI): An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.



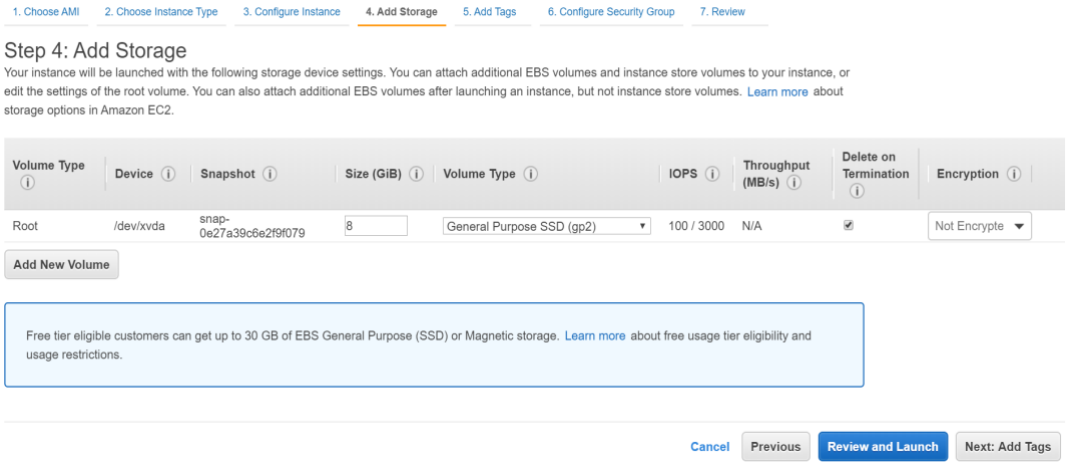
3. We'll work with **Amazon Linux 2 AMI (HVM), SSD Volume Type,** Tap on the Select button on the right side.
4. **Choose an Instance Type:** Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Select Free tier eligible one and Tap on **Next: Configure Instance Details**



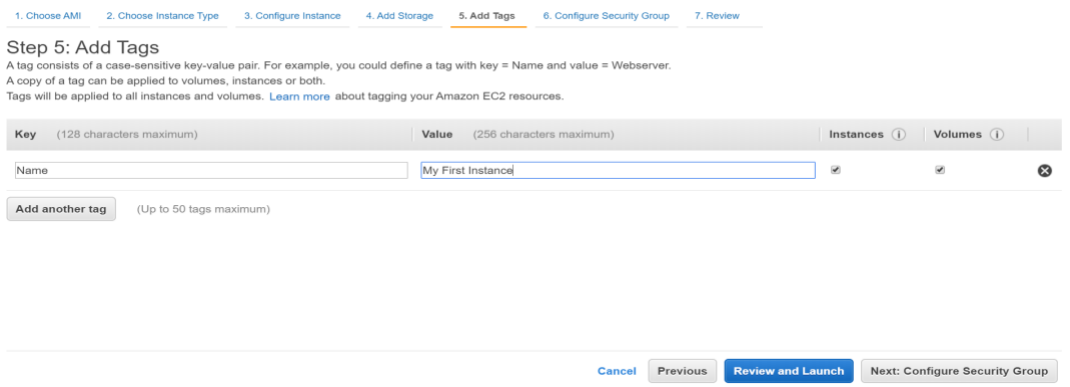
5. **Configure Instance Details:** Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.



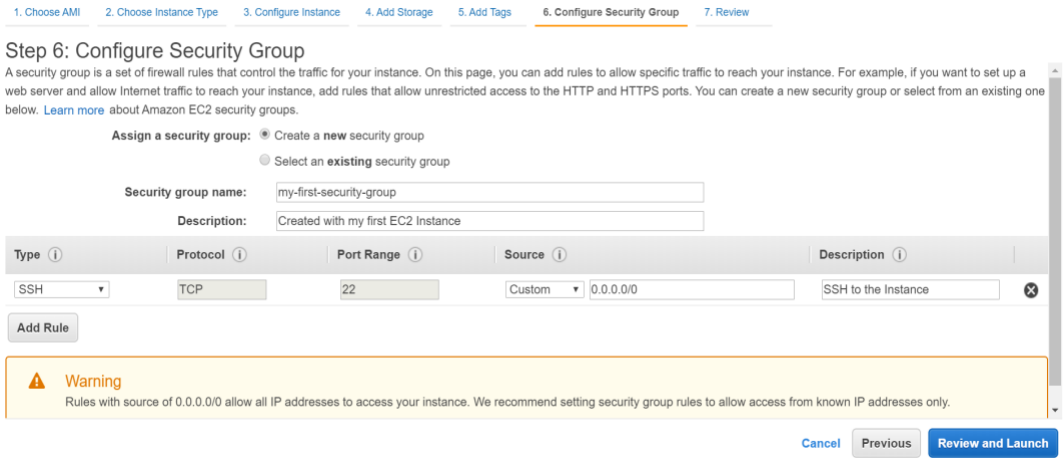
6. **Add Storage:** Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes.



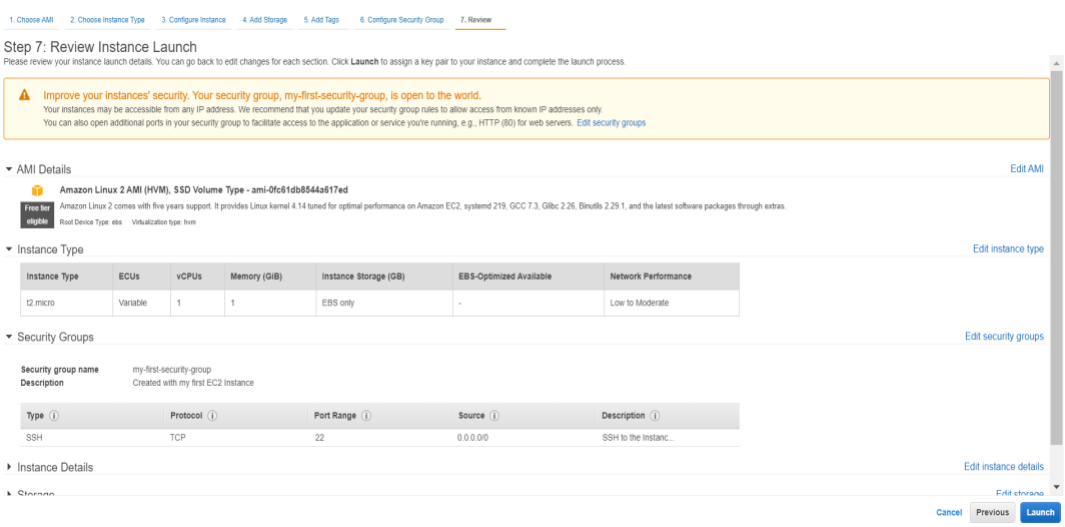
7. **Add Tags:** A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = My First Instance. A copy of a tag can be applied to volumes, instances or both. Tags will be applied to all instances and volumes.



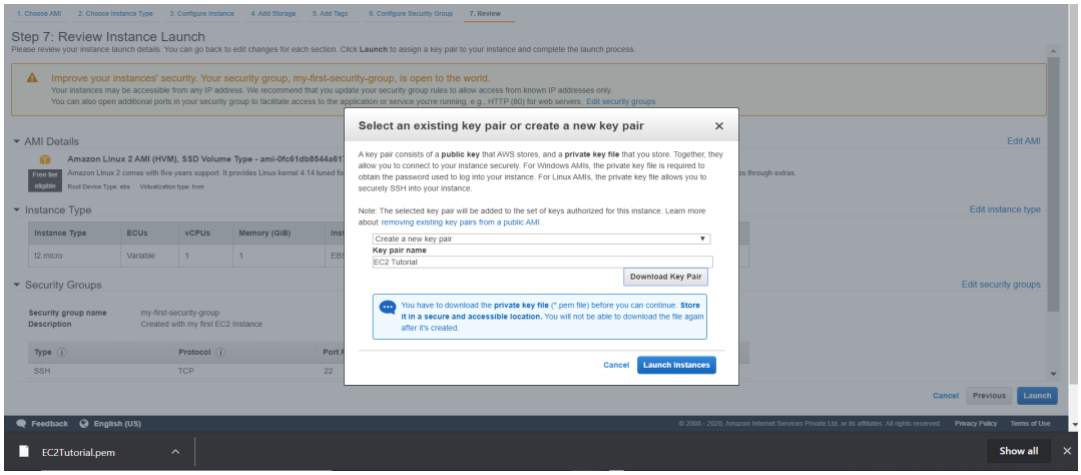
8. **Configure Security Group:** A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below.



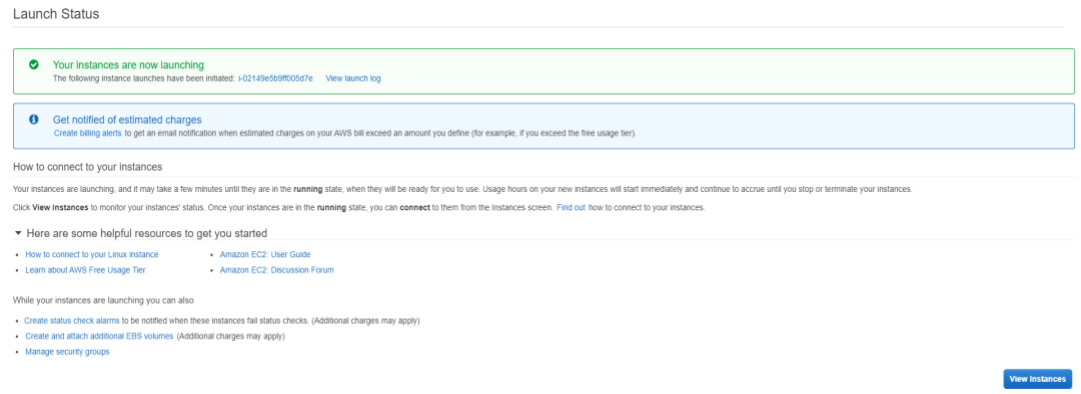
9. **Review Instance Launch:** Please review your instance launch details. You can go back to edit changes for each section. Click Launch to assign a key pair to your instance and complete the launch process.



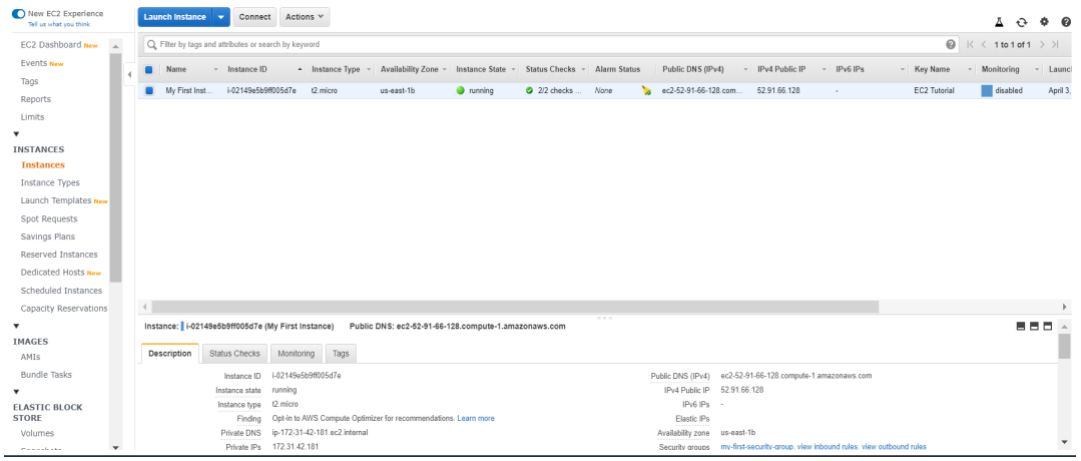
10. Tap on Launch, Pop-up is shown as Select an existing key pair or create a new key pair as shown in below screen. Create a new key pair, Name the new key pair and Download the key pair. Tap on Launch Instances.



11. Tap on Launch Instances



12. Tap on View Instances and My First Instance is created and launched as shown in below screen.



5.2. SSH Overview:

Secure Shell (SSH) SSH, also known as Secure Socket Shell, is a network protocol that provides administrators with a secure way to access a remote

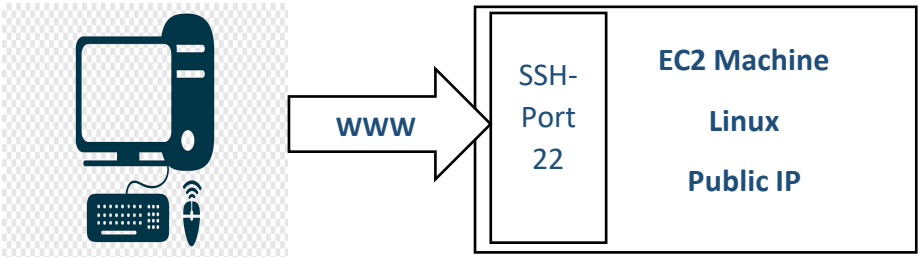
computer. SSH also refers to the suite of utilities that implement the protocol.

SSH Summary Table:

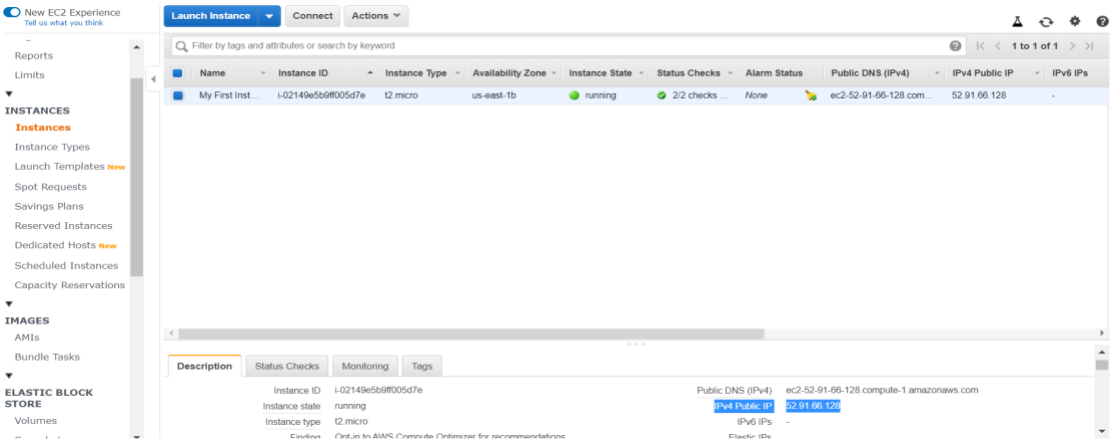
	SSH	PuTTY	EC2 Instance Connect
Mac	1	0	1
Linux	1	0	1
Windows < 10	0	1	1
Windows >= 10	1	1	1

5.2.1. How to SSH using Linux or Mac:

- SSH is one of the most important function. It allows you to control a remote machine, all using the command line.



- Open the EC2 Instances, check for **IPv4 Public IP** with this IP address we'll be able to connect to our Instances.

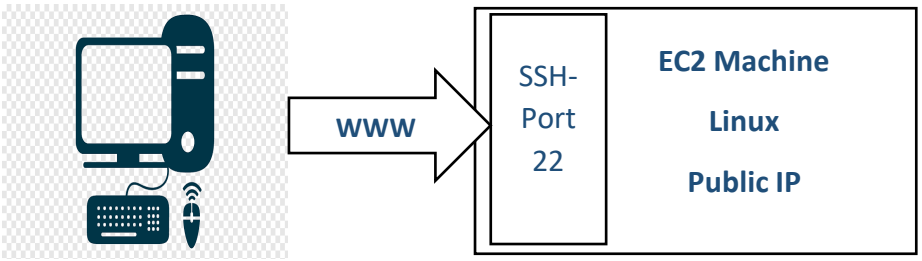


- In Mac: Open the command prompt and type: ssh –i EC2Tutorial.pem ec2-user@52.91.66.128 (type your instances IP address)
- It gives us WARNING: UNPROTECTED PRIVATE KEY FILE! Permissions 0644 for 'EC2Tutorial.pem' are too open. It is required that your private key files are NOT accessible by others. This private key will be ignored. Load key "EC2Tutorial.pem": bad permissions ec2-user@52.91.66.128: Permission denied.

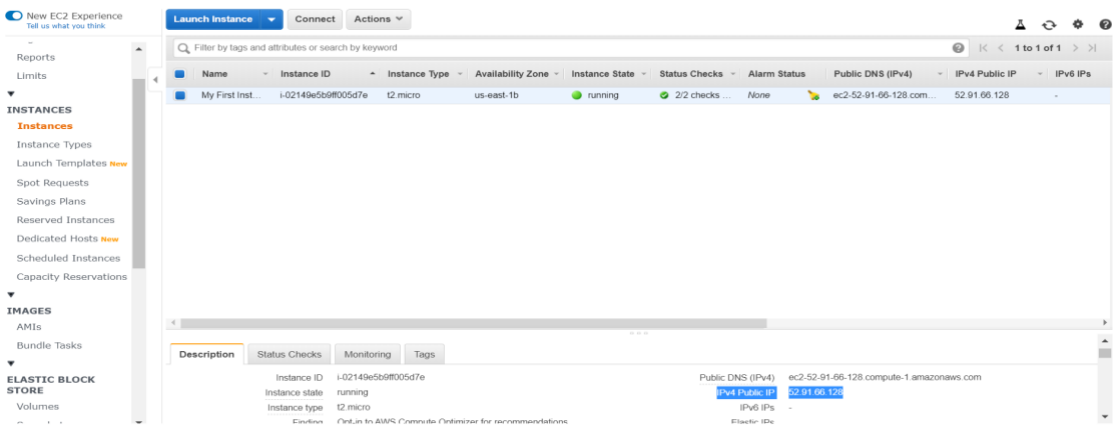
- So to fix the above error we need use this command: chmod 0400 EC2Tutorial.pem, then ssh –i EC2Tutorial.pem ec2-user@52.91.66.128
- It gets connected to our EC2 Instances, in order to check we can write whoami then it reverts as ec2-user
- To close the connection with EC2 Instances, type logout/^C.

5.2.2. How to SSH using Windows

- SSH is one of the most important function. It allows you to control a remote machine, all using the command line.



- We'll configure all the required parameters necessary for doing SSH on Windows using the free tool **PuTTY.**
- Open the EC2 Instances, check for **IPv4 Public IP** with this IP address we'll be able to connect to our Instances.

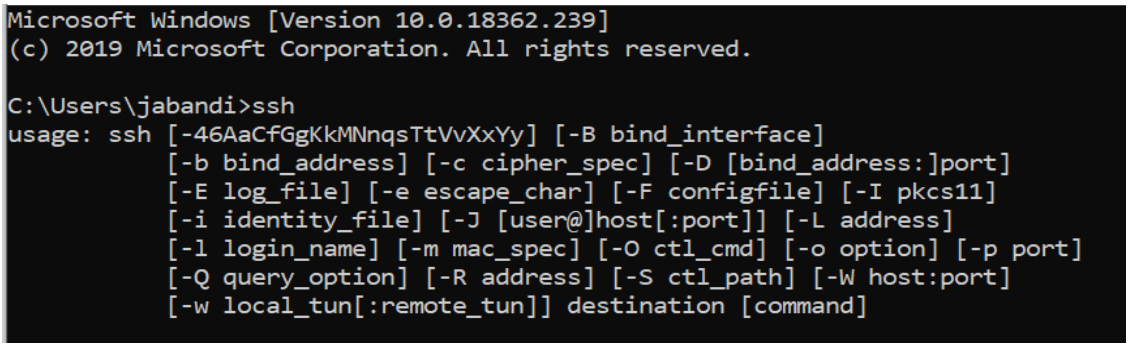


- Download PuTTY, PuTTY is an open source SSH client used to connect to a remote server. It is basically a terminal for windows based operating systems. It supports several network protocols, including SCP, SSH, Telnet, and raw socket connection.
- Go to PuTTYgen, tap on Files and click on Load private key, select EC2Tutorial.ppm then tap on ok. It says successfully imported the foreign key.
- It gives us key for pasting into Open SSH authorized keys file, key fingerprint, key comments and tap on Save private key as EC2Tutorial.ppk file.
- Open PuTTY, enter hostname: ec2-user@52.91.66.128 and port: 22.
- Save this and name it as MyEC2Instance. Tap on MyEC2Instance a pop-up is generated as PuTTY Security Alert, tap on yes then we get PuTTY Fatal Error because we didn't link our private key file.

- Go back to PuTTY, tap on MyEC2Instace and load, go to Connections and Select SSH and Auth, there is Private key file for authentication and browse for EC2Tutorial.ppk file and go back to Session to Save and click on Open.
- It gets connected to our EC2 Instances, in order to check we can write whoami then it reverts as ec2-user
- To close the connection with EC2 Instances, type logout/^C.

5.2.3. How to SSH using Windows 10

- Go to PowerShell/Command Prompt and type ssh, if OS allows ssh then we'll get the response as shown in the below screen.

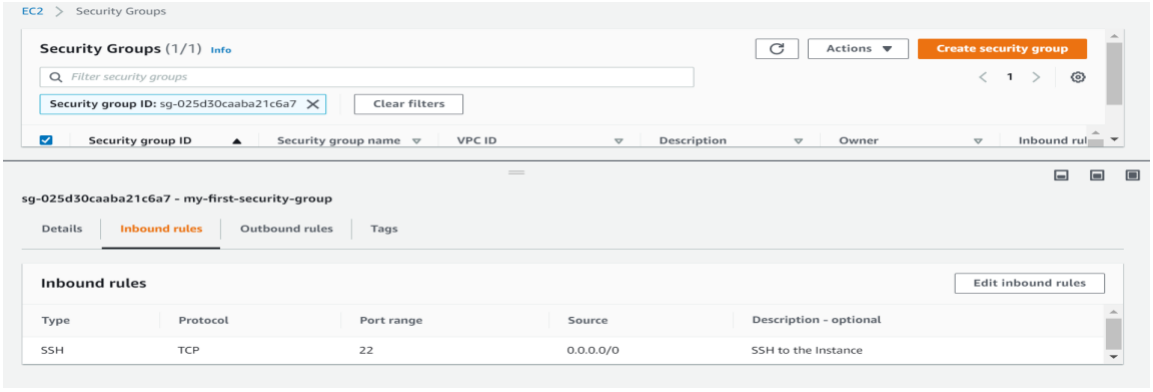


- If it's not gave us any response, then we need to follow steps using PuTTY.
- To connect to EC2 Instance, type this command: ssh –i C:\Users\EC2Tutorial.pem ec2-user@52.91.66.128
- It gives us WARNING: UNPROTECTED PRIVATE KEY FILE! Permissions 0644 for 'EC2Tutorial.pem' are too open. It is required that your private key files are NOT accessible by others. This private key will be ignored. Load key "EC2Tutorial.pem": bad permissions ec2-user@52.91.66.128: Permission denied.
- We need to go to EC2Tutorial.pem file location, right click on file, properties, security then check for the users who has access to the key. Tap on Advanced, we need to make sure that the owner to key should be you and remove other users and disable the inheritance and tap on Apply and Save.
- Try with the command again: ssh –i C:\Users\EC2Tutorial.pem ec2-user@52.91.66.128
- It gets connected to our EC2 Instances, in order to check we can write whoami then it reverts as ec2-user
- To close the connection with EC2 Instances, type logout/^C.

5.2.4. SSH Troubleshooting

1. **There's a connection timeout:**

   This is a security group issue. Any timeout (not just for SSH) is related to security groups or a firewall. Ensure your security group looks like this and correctly assigned to your EC2 instance.



2. **There's still a connection timeout issue:**

   If your security group is properly configured as above, and you still have connection timeout issues, then that means a corporate firewall or a personal firewall is blocking the connection. **Please use EC2 Instance Connect as described in the below.**

3. **SSH does not work on Windows:**

   If it says: ssh command not found, that means you have to use Putty**.** Follow steps again. If things don't work, please use EC2 Instance Connect as described in the below.

4. **There's a connection refused:**

   This means the instance is reachable, but no SSH utility is running on the instance. Try to restart the instance. If it doesn't work, terminate the instance and create a new one. Make sure you're using **Amazon Linux 2.**

5. **Permission denied (publickey, gssapi-keyex, gssapi-with-mic):** This means either two things:

   You are using the wrong security key or not using a security key. Please look at your EC2 instance configuration to make sure you have assigned the correct key to it.

   You are using the wrong user. Make sure you have started an Amazon Linux 2 EC2 instance, and make sure you're using the user ec2-user. This is something you specify when doing ec2-user@<public-ip> (ex: ec2-user@52.91.66.128) in your SSH command or your PuTTY configuration.

6. **Nothing is working - "whoa!!":**

   Don't panic. Use **EC2 Instance** Connect from the below. Make sure you started an **Amazon Linux 2** and you will be able to follow along with the tutorial ☺

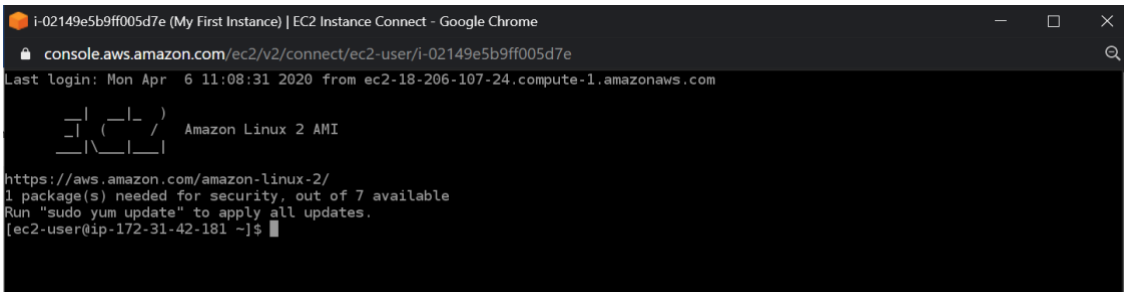
7. **I was able to connect yesterday, but today I can't:**

   This is probably because you have stopped your EC2 instance and then started it again today. **When you do so, the public IP of your EC2 instance will change**. Therefore, in your command, or Putty configuration, please make sure to edit and save the new public IP.

5.3. EC2 Instance Connect

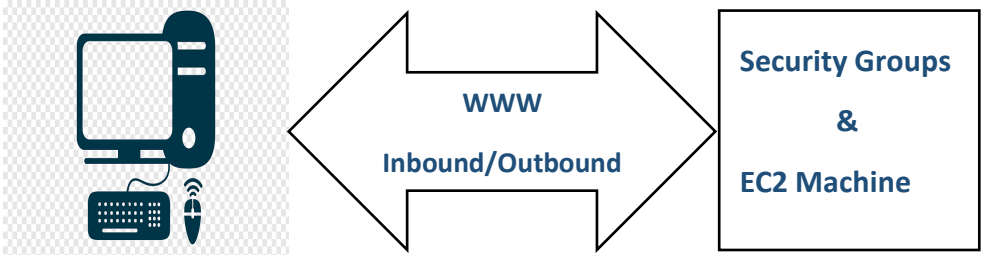
- Tap on Connect and select EC2 Instance Connect (browser-based SSH connection) and User name (ex. ec2-user) needs to be entered and tap on connect. Make sure that for the EC2 Instance, inbound security groups should have port number and source.

- The response from the browser will be as shown below. This confirms us that we've connected to EC2 Instances from the browser.



5.4. Security Groups

- Security Groups are the fundamental of network security in AWS
- They control how traffic is allowed into or out of our EC2 Machines.

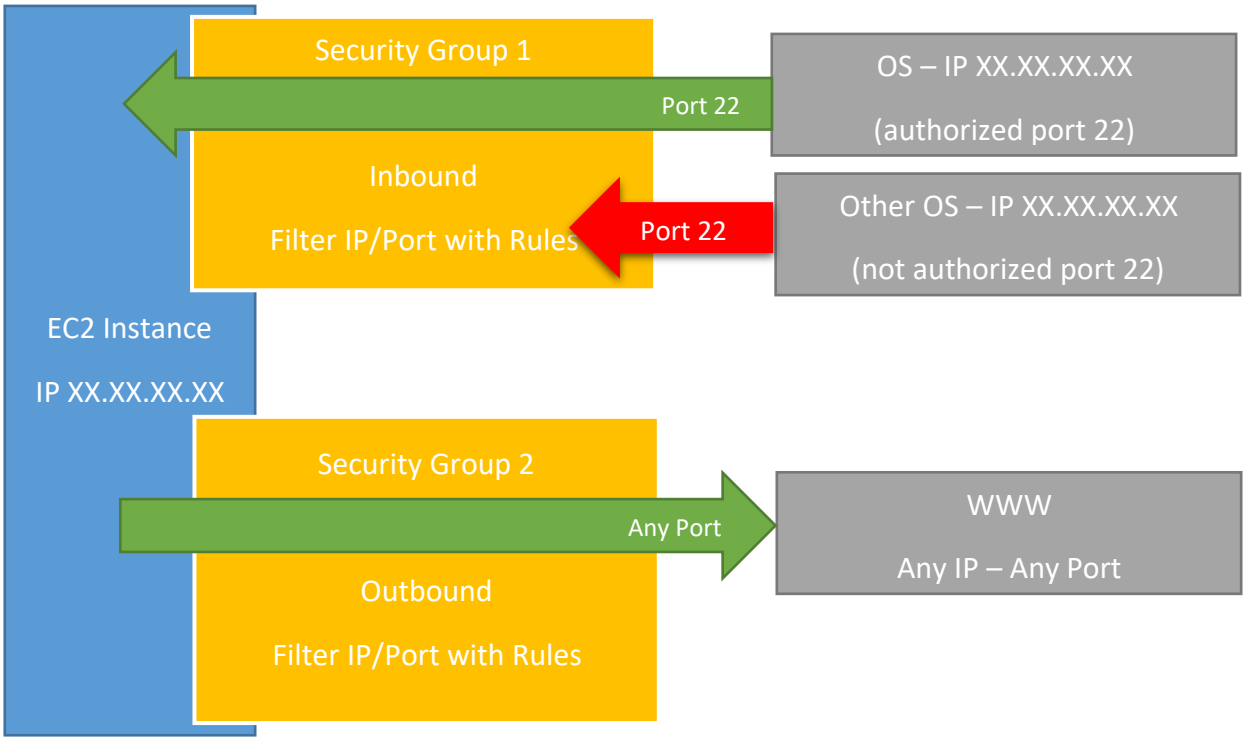


- It is the most fundamental skill to learn to troubleshoot networking issues, allow, inbound and outbound ports.
- Security Groups are acting as a “firewall” on EC2 instances
- They regulate:

    1. Access to Ports
    2. Authorised IP ranges – IPv4 and IPv6
    3. Control of inbound network (from other to the instance)
    4. Control of outbound network (from the instance to other)

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	TEST HTTP PAGE
SSH	TCP	22	122.149.196.85/32	
Custom TCP Rule	TCP	4567	0.0.0.0/0	JAVA APP

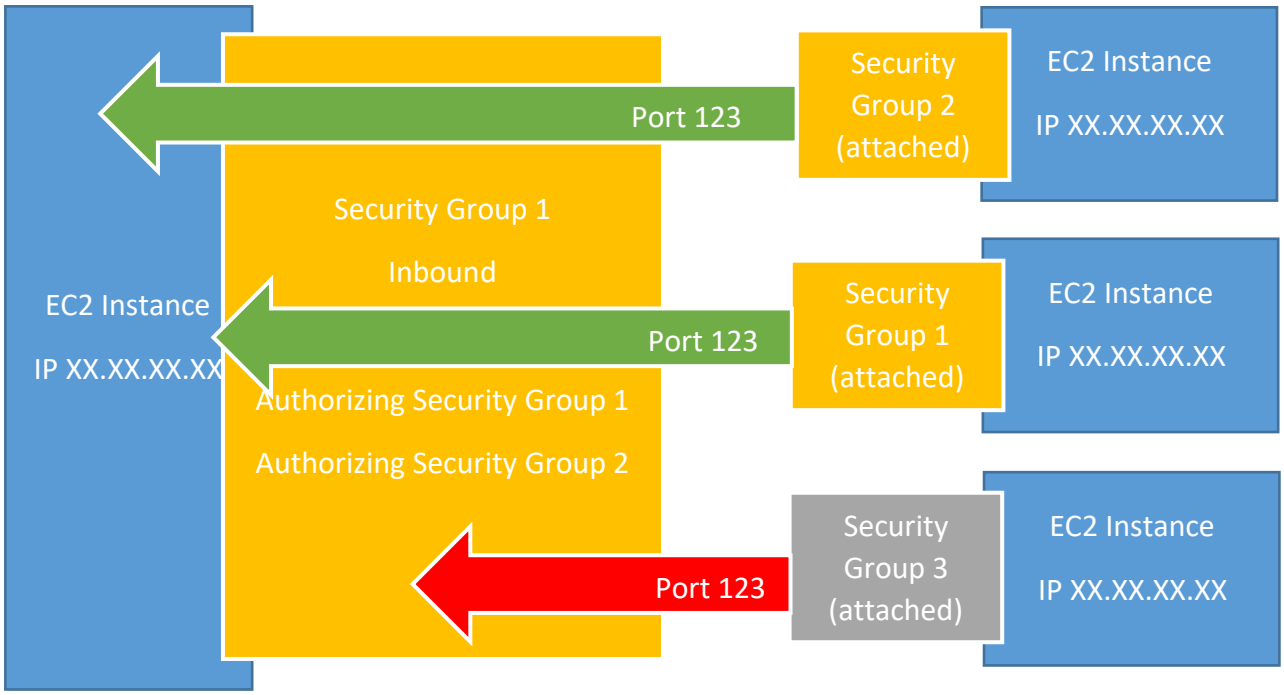
Security Groups Diagram



- Security Groups can be attached to multiple instances
- Locked down to a region / VPC combination
- Does live “outside” the EC2 – if traffic is blocked the EC2 instance won’t see it
- It’s good to maintain one separate security group for SSH access
- If your application is not accessible (time out), then it’s a security group issue
- If your application gives a “connection refused” error, then it’s an application error or it’s not launched

- All inbound traffic is blocked by default
- All outbound traffic is authorised by default

Referencing other security groups Diagram



5.4.1. Private Vs Public Vs Elastic IP

Private vs Public IP (IPv4)

- Networking has two sorts of IPs. IPv4 and IPv6:
  1. IPv4: 1.160.10.240
  2. IPv6: 3ffe: 1900: 4545: 3: 200: f8ff: fe21: 67cf
- We'll be using IPv4
- IPv4 is still the most common format used online.
- IPv6 is newer and solves problems for the Internet of Things (IoT).
- IPv4 allows for 3.7 billion different addresses in the public space
- IPv4: [0-255]. [0-255]. [0-255]. [0-255].
- Example: We can access Public IP with WWW

Fundamental Differences

Public IP

1. Public IP means the machine can be identified on the internet (WWW)
2. Must be unique across the whole web (not two machines can have the same public IP).
3. Can be geo-located easily

Private IP

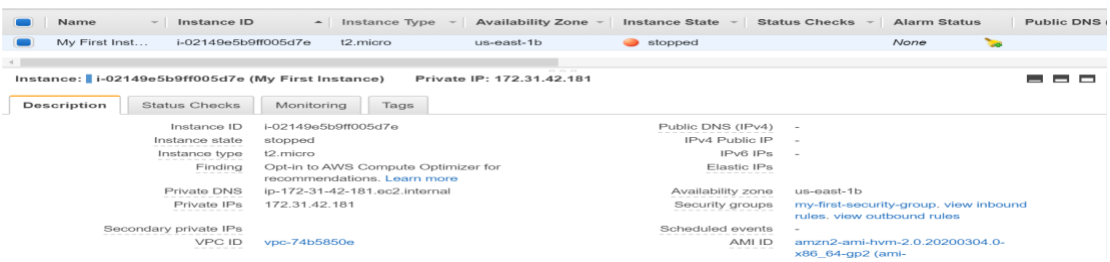
1. Private IP means the machine can only be identified on a private network only.
2. The IP must be unique across the private network
3. BUT two different private networks (two companies) can have same IPs.
4. Machines connect to WWW using an internet gateway (a proxy)
5. Only a specified range of IPs can be used as private IP

Elastic IPs

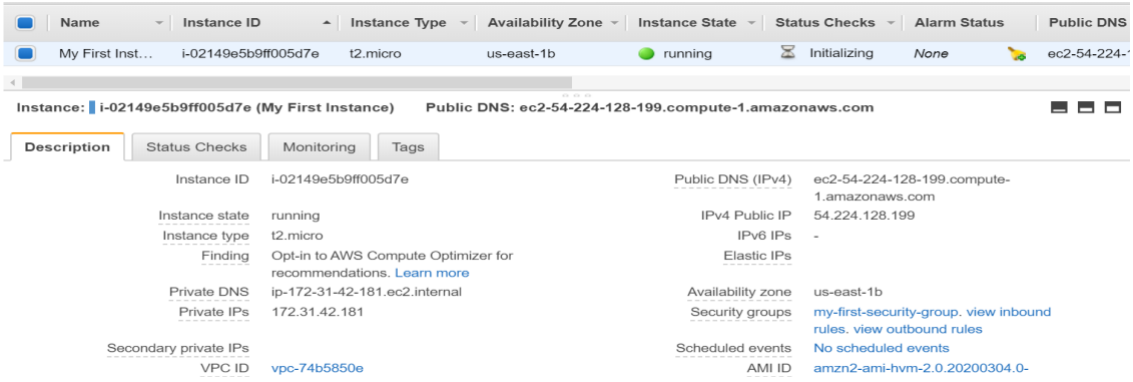
1. With an Elastic IP address, you can mask the failure of an instance or software by rapidly remapping the address to another instance in your account.
2. You can only have 5 Elastic IP in your account (you can ask AWS to increase that).
3. Overall, try to avoid using Elastic IP:
   a. They often reflect poor architectural decisions
   b. Instead, use a random public IP and register a DNS name to it
   c. Or, use a Load Balancer and don't use a public IP.

In AWS EC2 – Hands On

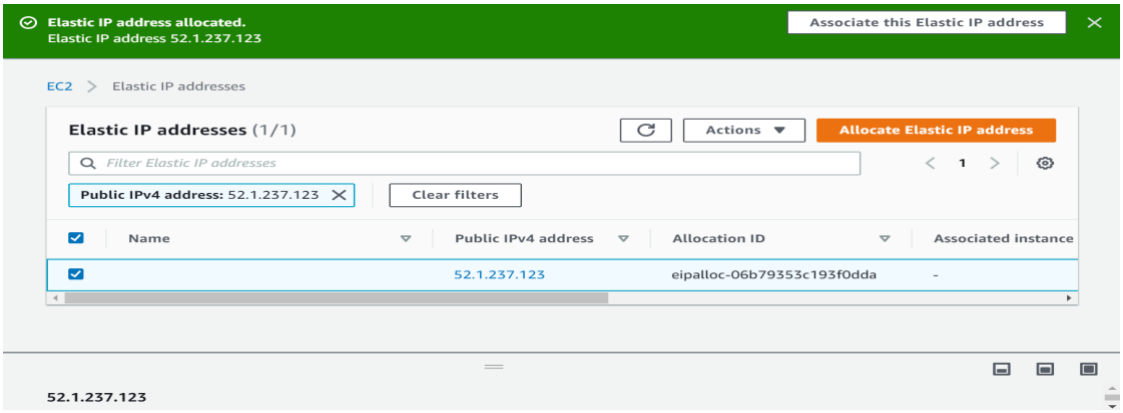
- By default, your EC2 machine comes with:
  a. A private IP for the internal AWS Network
  b. A public IP, for the WWW.
- When we are doing SSH into our EC2 machines:
  a. We can't use a private IP, because we are not in the same network
  b. We can only use the public IP.
- If your machine is stopped and then started, the public IP can change
- If we stop our EC2 Instance, then the IPv4 Public IP gets disappeared as shown in below screen.



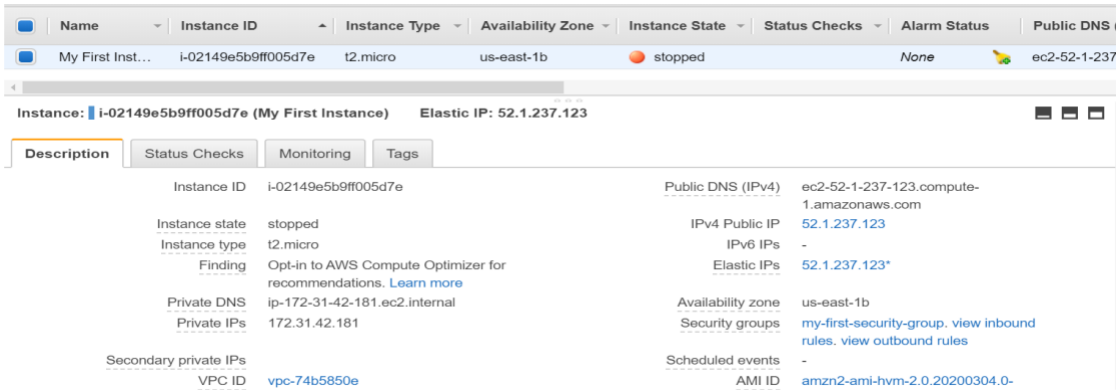
- If we restart the EC2 Instance, then we'll be having different IPv4 Public IP as shown in the below screen.



- For Elastic IP, tap on it which is on the left side of the screen
- Select Allocate Elastic IP Address, then it allocates IP address as shown in below screen.



- Now we need to Associate Elastic IP address with EC2 Instance, then stop the EC2 Instance. So that IPv4 Public IP won't get disappeared because Elastic IP will be the IPv4 Public IP as shown in the below screen.

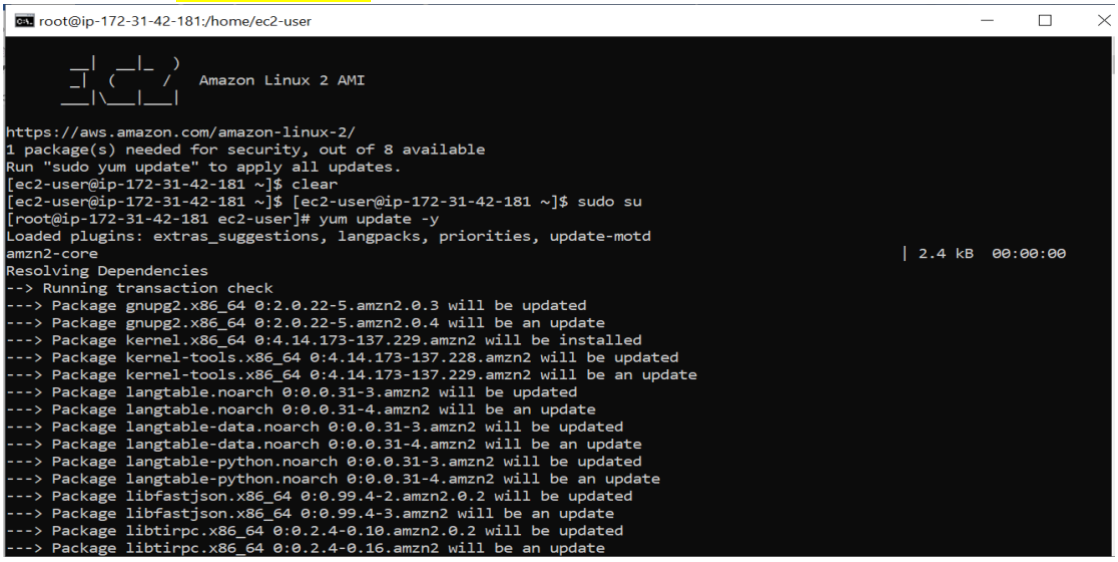


- At the end, we need to dissociate the Elastic IP and release the Allocation of the Elastic IP because it costs us.

5.5. Launching an Apache Server on EC2

- We'll install an Apache Web Server to display a web page
- We'll create an index.html that shows the hostname of our machine

- Run the EC2 Instance in the command prompt of the local machine, and type sudo su, this will elevate the root user rights of the machine towards EC2 Instance and yum update –y will forcefully update all the packages.



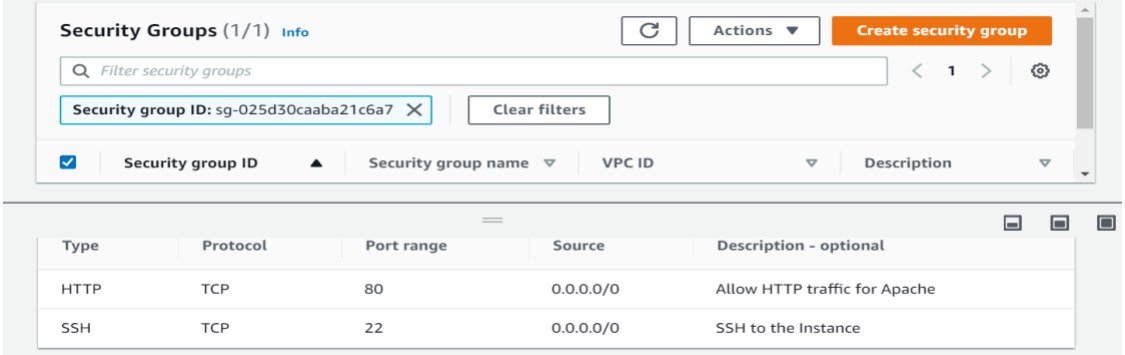
- Now we can install httpd (Apache Server) with the following command: yum install –y httpd.x86_64 in the local machine.

```
[root@ip-172-31-42-181 ec2-user]# yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                                                | 2.4 kB  00:00:00
No packages marked for update
[root@ip-172-31-42-181 ec2-user]# yum install y httpd.x86_64
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
No package y available.
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.4.41-1.amzn2.0.1 will be installed
--> Processing Dependency: httpd-tools = 2.4.41-1.amzn2.0.1 for package: httpd-2.4.41-1.amzn2.0.1.x86_64
--> Processing Dependency: httpd-filesystem = 2.4.41-1.amzn2.0.1 for package: httpd-2.4.41-1.amzn2.0.1.x86_64
--> Processing Dependency: system-logos-httpd for package: httpd-2.4.41-1.amzn2.0.1.x86_64
--> Processing Dependency: mod_http2 for package: httpd-2.4.41-1.amzn2.0.1.x86_64
--> Processing Dependency: httpd-filesystem for package: httpd-2.4.41-1.amzn2.0.1.x86_64
--> Processing Dependency: /etc/mime.types for package: httpd-2.4.41-1.amzn2.0.1.x86_64
--> Processing Dependency: libaprutil-1.so.0()(64bit) for package: httpd-2.4.41-1.amzn2.0.1.x86_64
--> Processing Dependency: libapr-1.so.0()(64bit) for package: httpd-2.4.41-1.amzn2.0.1.x86_64
--> Running transaction check
---> Package apr.x86_64 0:1.6.3-5.amzn2.0.2 will be installed
---> Package apr-util.x86_64 0:1.6.1-5.amzn2.0.2 will be installed
--> Processing Dependency: apr-util-bdb(x86-64) = 1.6.1-5.amzn2.0.2 for package: apr-util-1.6.1-5.amzn2.0.2.x86_64
---> Package generic-logos-httpd.noarch 0:18.0.0-4.amzn2 will be installed
---> Package httpd-filesystem.noarch 0:2.4.41-1.amzn2.0.1 will be installed
---> Package httpd-tools.x86_64 0:2.4.41-1.amzn2.0.1 will be installed
---> Package mailcap.noarch 0:2.1.41-2.amzn2 will be installed
---> Package mod_http2.x86_64 0:1.15.3-2.amzn2 will be installed
--> Running transaction check
---> Package apr-util-bdb.x86_64 0:1.6.1-5.amzn2.0.2 will be installed
--> Finished Dependency Resolution
```

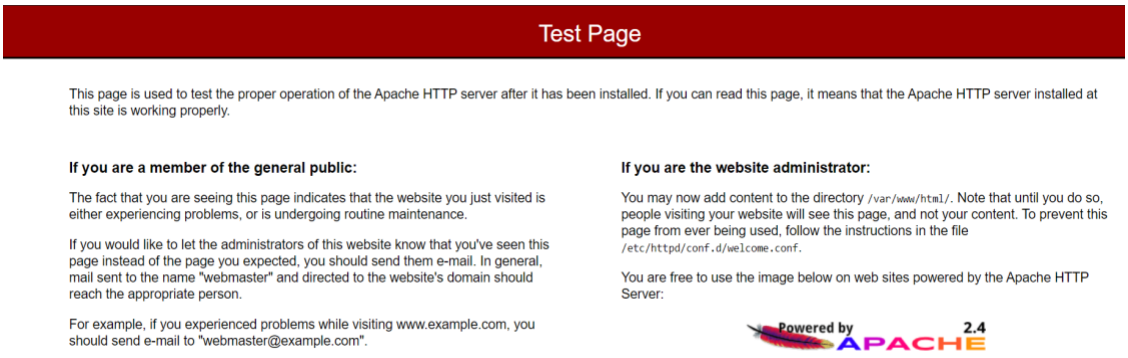
- After the installation of httpd, systemctl start httpd.service and systemctl enable httpd.service commands will help us to start and enable the httpd server and curl localhost:80 is the port number for httpd server. This reverts with the html scripts on the command prompt as shown in the below screen.

```
  Verifying  : httpd-2.4.41-1.amzn2.0.1.x86_64                                  3/9
  Verifying  : httpd-filesystem-2.4.41-1.amzn2.0.1.noarch                       4/9
  Verifying  : mod_http2-1.15.3-2.amzn2.x86_64                                  5/9
  Verifying  : apr-1.6.3-5.amzn2.0.2.x86_64                                     6/9
  Verifying  : mailcap-2.1.41-2.amzn2.noarch                                    7/9
  Verifying  : generic-logos-httpd-18.0.0-4.amzn2.noarch                        8/9
  Verifying  : httpd-tools-2.4.41-1.amzn2.0.1.x86_64                            9/9

Installed:
  httpd.x86_64 0:2.4.41-1.amzn2.0.1

Dependency Installed:
  apr.x86_64 0:1.6.3-5.amzn2.0.2                  apr-util.x86_64 0:1.6.1-5.amzn2.0.2
  apr-util-bdb.x86_64 0:1.6.1-5.amzn2.0.2         generic-logos-httpd.noarch 0:18.0.0-4.amzn2
  httpd-filesystem.noarch 0:2.4.41-1.amzn2.0.1    httpd-tools.x86_64 0:2.4.41-1.amzn2.0.1
  mailcap.noarch 0:2.1.41-2.amzn2                 mod_http2.x86_64 0:1.15.3-2.amzn2

Complete!
[root@ip-172-31-42-181 ec2-user]# systemctl start httpd.service
[root@ip-172-31-42-181 ec2-user]# systemctl enable httpd.service
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to /usr/lib/systemd/system/httpd.service.
[root@ip-172-31-42-181 ec2-user]# curl localhost:80
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
        <head>
                <title>Test Page for the Apache HTTP Server</title>
                <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
                <style type="text/css">
                        /*<![CDATA[*/
```

- We need to configure the Inbound Security Groups with Type: HTTP and Port: 80 as shown in the below screen.



- After the configuration we can use the url: http://52.206.43.243(IPv4 Public IP):80(Port) in the browser. It reverts us as success response with Apache Server screen.



- With echo “Hello World” > /var/www/html/index.html command, browser gives us Hello World as a response. For Example: echo “Hello World from $(hostname -f)” > /var/www/html/index.html command, browser gives us Hello World from ip-172-31-42-181.ec2.internal as shown in the below screens.

5.6. EC2 User Data

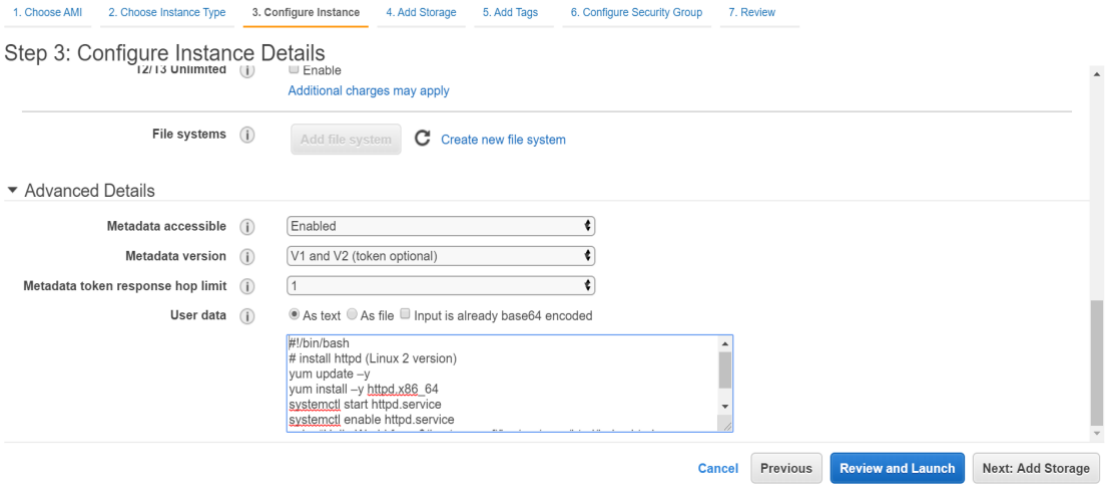
- It’s possible to bootstrap our instances using an EC2 User data script.
- Bootstrapping means launching commands when a machine starts
- That script is only run once at the instance first start
- EC2 user data is used to automate boot tasks such as:
    a. Installing updates

b. Installing software
c. Downloading common files from the internet
d. Anything you can think of

- The EC2 User Data Script runs with the root user

EC2 User Data Hand-On

- We want to make sure that this EC2 instance has an Apache HTTP server installed on it – to display a simple web page
- For it, we are going to write a user-data script.
- This script will be executed at the first boot of the instance.
- We need to follow the steps of EC2 Instance Creation and User Data is at the 3rd step i.e., Configure Instance Details. We will paste all the commands that needs to run at the start of Apache Server.
  For example:
  1. #!/bin/bash
  2. # install httpd (Linux 2 version)
  3. yum update –y
  4. yum install –y httpd.x86_64
  5. systemctl start httpd.service
  6. systemctl enable httpd.service
  7. echo “Hello World from $(hostname -f)” > /var/www/html/index.html
- Paste all the above commands in the User Data of the Configure Instance Details and automatically run the instance with sudo command as shown in the below screen.



- In the Configure Security Group step, use the existing security group details and EC2Tutorial.pem file, then Launch the Instance.

5.7. EC2 Instance Launch Types

- On Demand Instances: short workload, predictable pricing
- Reserved Instances: long workloads (>= 1 year)

- Convertible Reserved Instances: long workloads with flexible instances
- Scheduled Reserved Instances: launch within time window you reserve
- Spot Instances: short workloads, for cheap, can lose instances
- Dedicated Instances: no other customers will share your hardware
- Dedicated Hosts: book an entire physical server, control instance placement

EC2 On Demand

- Pay for what you use (billing per second, after the first minute)
- Has the highest cost but no upfront payment!
- No long term commitment
- Recommended for short-term and un-interrupted workloads, where you can't predict how the application will behave.

EC2 Reserved Instances

- Up to 75% discount compared to On-demand
- Pay upfront for what you use with long term commitment
- Reservation period can be 1 or 3 years
- Reserve a specific instance type
- Recommended for steady stage usage application (think database)
- Convertible Reserved Instance
    - Can change the EC2 instance type
    - Up to 54% discount
- Scheduled Reserved Instances
    - Launch within time window you reserve

EC2 Spot Instances

- Can get a discount up to 90% compared to On-demand
- You bid a price and get the instance as long as it's under the price
- Price varies based on offer and demand
- Spot instances are reclaimed with a 2-minute notification warning when the spot price goes above your bid
- Used for batch jobs, Big Data analysis, or workloads that are resilient to failures.
- Not great for critical jobs or databases

EC2 Dedicated Hosts

- Physical dedicated EC2 server for your use
- Full control of EC2 Instance placement
- Visibility into the underlying sockets / physical cores of the hardware
- Allocated for your account for a 3-year period reservation
- More expensive
- Useful for software that have complicated licensing model (BYOL – Bring Your Own License)

- Or for companies that have strong regulatory or compliance needs

EC2 Dedicated Instances

- Instances running on hardware that's dedicated to you
- May share hardware with other instances in same account
- No control over instance placement (can move hardware after Stop / Start)

Characteristics	Dedicated Instances	Dedicated Hosts
Enabled the use of dedicated physical servers	X	X
Per instance billing (subject to a $2 per region fee)	X	
Per host billing		X
Visibility of sockets, cores, host ID		X
Affinity between a host and instance		X
Targeted instance placement		X
Automatic instance placement	X	X
Add capacity using an allocation request		X

EC2 Pricing

- EC2 instances pricing (per hour) varies based on these parameters:
  - Region you are in
  - Instance type you are using
  - On-Demand Vs Spot Vs Reserved Vs Dedicated Host
  - Linux Vs Windows Vs Private OS (RHEL, SLES, Windows SQL)
- You are billed by the second, with a minimum of 60 seconds.
- You also pay for other factors such as storage, data transfer, fixed IP public addresses, load balancing
- You don't pay for the instance if the instance is stopped
- For Example: t2.small in US-EAST-1 (N. Virginia), cost $0.023 per Hour
- If used for:
  - 6 seconds, it costs $0.023/60 = $0.000383 (minimum of 60 seconds)
  - 60 seconds, it costs $0.023/60 = $0.000383 (minimum of 60 seconds)
  - 30 minutes, it costs $0.023/2 = $0.0115
  - 1 month, it costs $0.023 * 24 * 30 = $16.56 (assuming a month is 30 days)
  - X seconds (X > 60), it costs $0.023 * X/3600
- The best way to know the pricing is to consult the pricing page: https://aws.amazon.com/ec2/pricing/on-demand/

What's an AMI?

- As we saw, AWS comes with base images such as:
  - Ubuntu
  - Fedora
  - RedHat
  - Windows
  - Etc…
- These images can be customized at runtime using EC2 User data
- But what if we could create our own image, ready to go?
- That's an AMI – an image to use to create our instances
- AMIs can be built for Linux or Windows machine

Why would you use a custom AMI?

- Using a custom built AMI can provide the following advantages:
  - Pre-installed packages needed
  - Faster boot time (no need for long ec2 user data at boot time)
  - Machine comes configured with monitoring / enterprise software
  - Security concerns – control over the machines in the network
  - Control of maintenance and updates of AMIs over time
  - Active directory integration out of the box
  - Installing your app ahead of time (for faster deploys when auto-scaling)
  - Using someone else's AMI that is optimized for running an app, DB, etc.
- AMI are built for a specific AWS region (!)

EC2 Instances Overview

- Instances have 5 distinct characteristics advertised on the website:
  - The RAM (type, amount, generation)
  - The CPU (type, make, frequency, generation, number of cores)
  - The I/O (disk performance, EBS optimizations)
  - The Network (network bandwidth, network latency)
  - The Graphical Processing Unit (GPU)
- It may be daunting to choose the right instance type (there are over 50 of them) – https://aws.amazon.com/ec2/instance-types/
- https://ec2instances.info/ can help with summarizing the types of instances
- R/C/P/G/H/X/I/F/Z/CR are specialized in RAM, CPU, I/O, Network, GPU
- M instance types are balanced
- T2/T3 instance types are "burstable"

Burstable Instances (T2)

- Burstable instances can be amazing to handle unexpected traffic and getting the insurance that it will be handled correctly
- If your instance consistently runs low on credits, you need to move to a different kind of non-burstable instance (all the ones described before).

CPU Credits

Instance type	Launch credits	vCPUs	CPU credits earned per hour	Maximum earned CPU credit balance	vCPUs	Baseline performance (% CPU utilization)
t2.nano	30	1	3	72	1	5%
t2.micro	30	1	6	144	1	10%
t2.small	30	1	12	288	1	20%
t2.medium	60	2	24	576	2	40% (of 200% max)*
t2.large	60	2	36	864	2	60% (of 200% max)*
t2.xlarge	120	4	54	1296	4	90% (of 400% max)*
t2.2xlarge	240	8	81	1944	8	135% (of 800% max)*

T2 Unlimited

- Nov 2017: It is possible to have an "unlimited burst credit balance"
- You pay extra money if you go over your credit balance, but you don't lose in performance
- Overall, it is a new offering, so be careful, costs could go high if you're not monitoring the health of your instances
- Read more here: https://aws.amazon.com/blogs/aws/new-t2-unlimited-going-beyond-the-burst-with-high-performance/

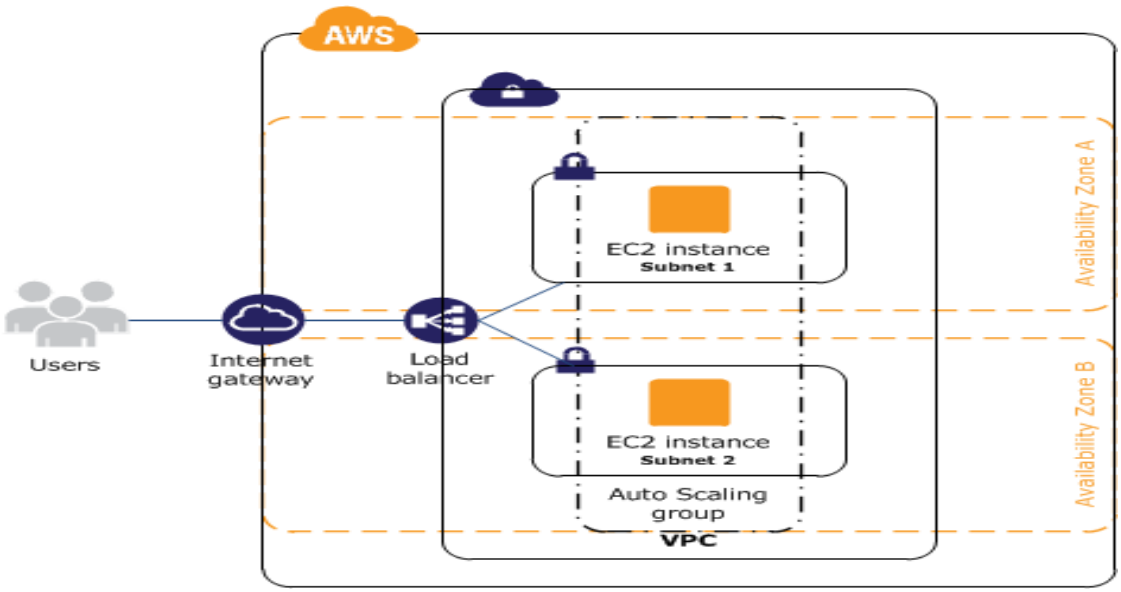
EC2 Checklist (Exam Perspective)

- Know how to SSH into EC2 (and change .pem file permissions)
- Know how to properly use security groups
- Know the fundamental differences between Private Vs Public Vs Elastic IP
- Know how to use User Data to customize your instance at boot time
- Know that you can build custom AMI to enhance your OS
- EC2 instances are billed by the second and can be easily created and thrown away, welcome to the cloud!

6. EC2 Load Balancer

What is Load Balancing?

- Load balancers are servers that forward internet traffic to multiple servers (EC2 Instances) downstream.



Why use a load balancer?

- Spread load across multiple downstream instances
- Expose a single point of access (DNS) to your application
- Seamlessly handle failures of downstream instances
- Does regular health checks to your instances
- Provide SSL termination (HTTPS) for your websites
- Enforce stickiness with cookies
- High availability across zones
- Separate public traffic from private traffic

Why use an EC2 Load Balancer?

- An ELB (EC2 Load Balancer) is managed load balancer
  - AWS guarantees that it will be working
  - AWS takes care of upgrades, maintenance, high availability
  - AWS provides only a few configuration knobs
- It costs less to setup your own load balancer but it will be a lot more effort on your end.
- It is integrated with many AWS offerings / services

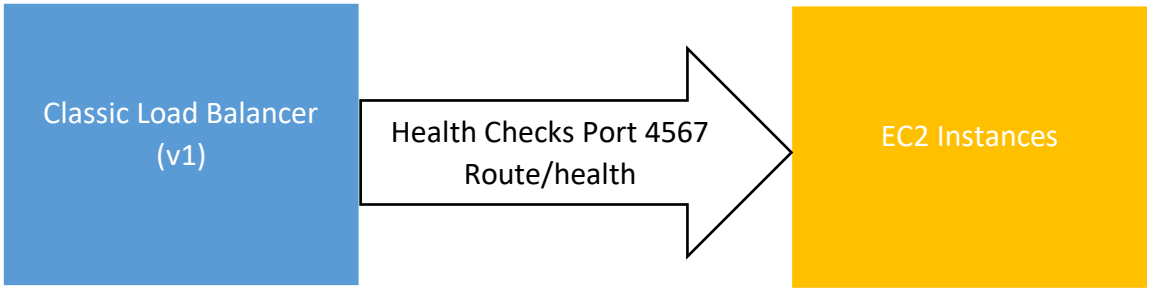
Types of load balancer on AWS

- AWS has 3 kinds of Load Balancers
- Classic Load Balancer (v1 – old generation) – 2009
- Application Load Balancer (v2 – new generation) – 2016

- Network Load Balancer (v2 – new generation) -2017
- Overall, it is recommended to use the newer / v2 generation load balancers as they provide more features
- You can setup internal (private) or external (public) ELBs

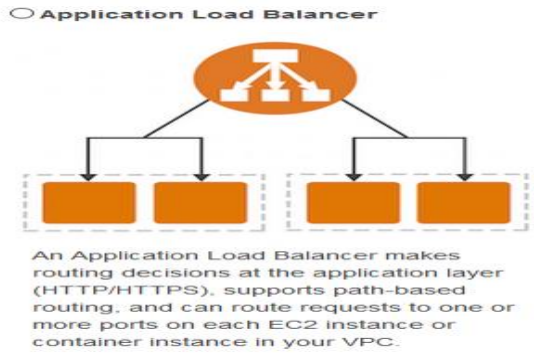
Health Checks

- Health Checks are crucial for Load Balancers
- They enable the load balancer to know if instances it forwards traffic to are available to reply to requests
- The health check is done on a port and a route (/health is common)
- If the response is not 200 (OK), then the instance is unhealthy

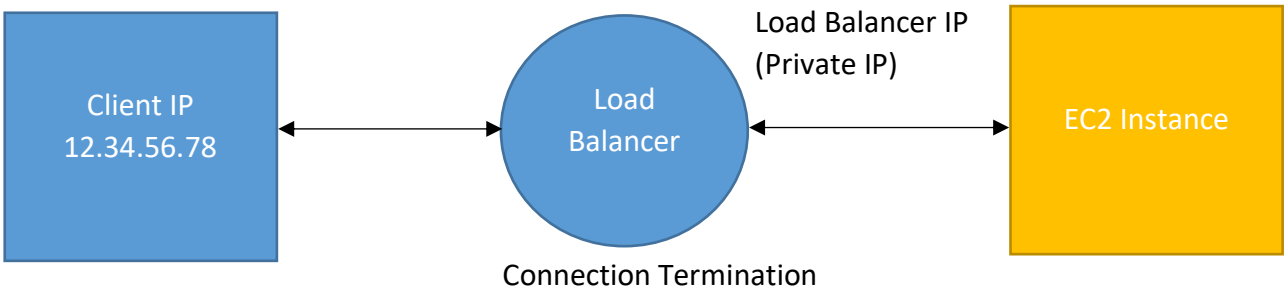


Application Load Balancer (v2)

- Application load balancers (Layer 7) allow to do:
  - Load balancing to multiple HTTP applications across machines (target groups)
  - Load balancing to multiple applications on the same machine (ex. containers)
  - Load balancing based on route in URL
  - Load balancing based on hostname in URL
- Basically, they're awesome for micro services & container-based application (example: Docker & Amazon ECS)
- Has a port mapping feature to redirect to a dynamic port
- In comparison, we would need to create one Classic Load Balancer per application before. That was very expensive and inefficient!

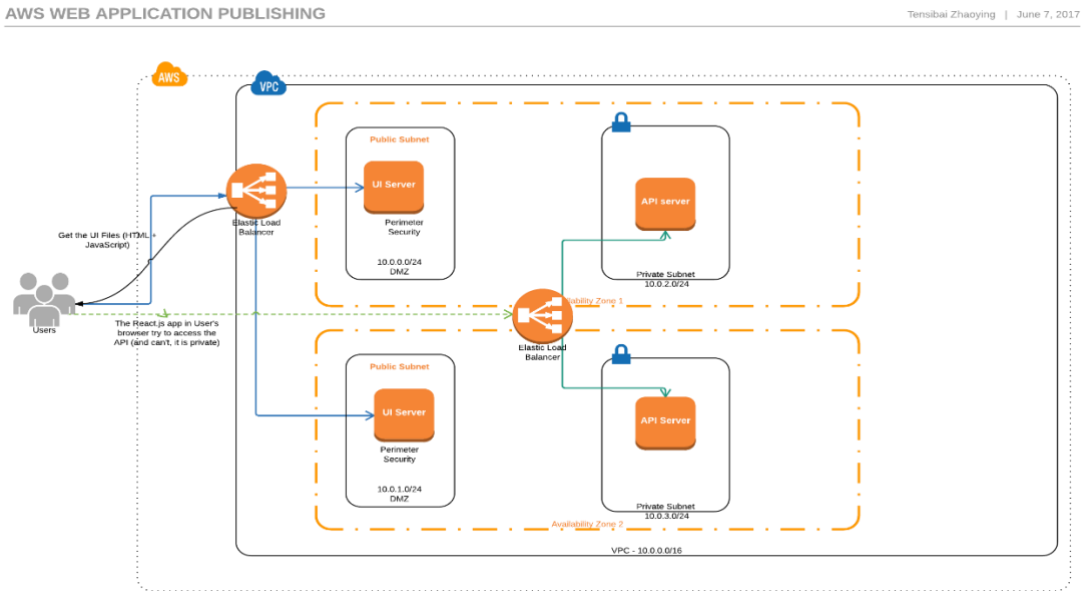


- Stickiness can be enabled at the target group level
  - Same request goes to the same instance
  - Stickiness is directly generated by the ALB (not the application)
- ALB support HTTP/HTTPS & Web sockets protocols
- The application servers don't see the IP of the client directly
  - The true IP of the client is inserted in the header X-Forwarded-For
  - We can also get Port (X-Forwarded-Port) and proto (X-Forwarded-Proto)



Network Load Balancer (v2)

- Network load balancers (Layer 4) allow to do:
  - Forward TCP traffic to your instances
  - Handle millions of requests per seconds
  - Support for static IP or elastic IP
  - Less latency ~ 100ms (vs 400ms for ALB)
- Network Load Balancers are mostly used for extreme performance and should not be the default load balancer you choose
- Overall, the creation process is the same as Application Load Balancers

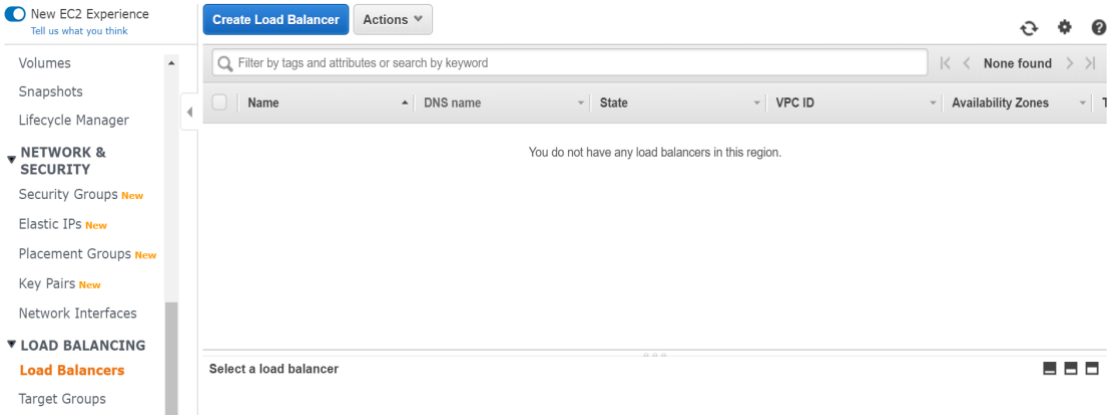


- Classic Load Balancers are deprecated
  - Application Load Balancers for HTTP/HTTPs & Web socket

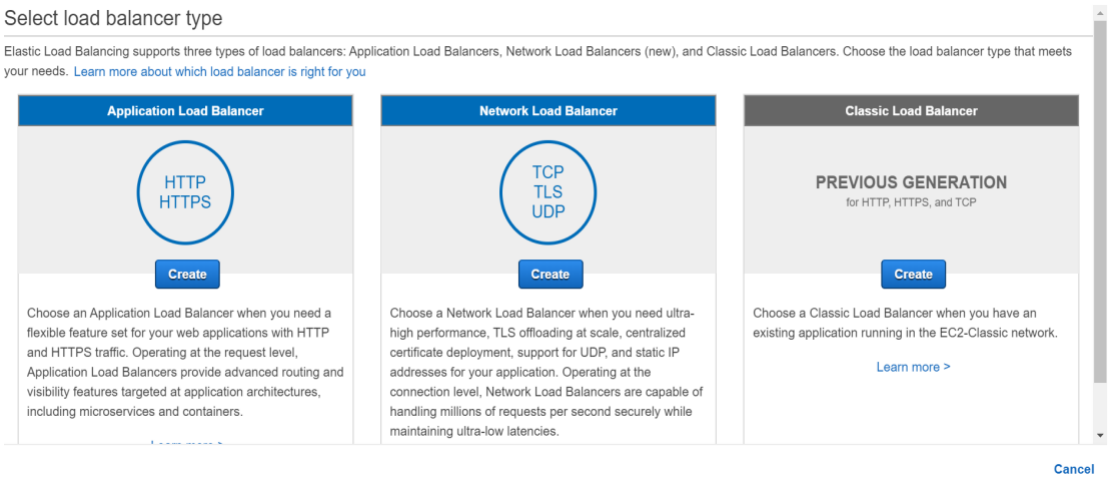
    - Network Load Balancer for TCP
- CLB and ALB support SSL certificates and provide SSL termination
- All Load Balancers have health check capability
- ALB can route on based on hostname / path
- ALB is a great fit with ECS (Docker)
- Any Load Balancer (CLB, ALB, NLB) has a static host name. Do not resolve and use underlying IP
- LBs can scale but not instantaneously – contact AWS for a “warm-up”
- NLB directly see the client IP
- 4xx errors are client induced errors
- 5xx errors are application induced errors
    - Load Balancer Errors 503 means at capacity or no registered target
- If the LB can’t connect to your application, check your security groups!

Load Balancer Hands-On

- After creating EC2 Instance, Load Balancer is created by selecting it on the left side of the screen and Tap on Create Load Balancer as shown in the below screen.

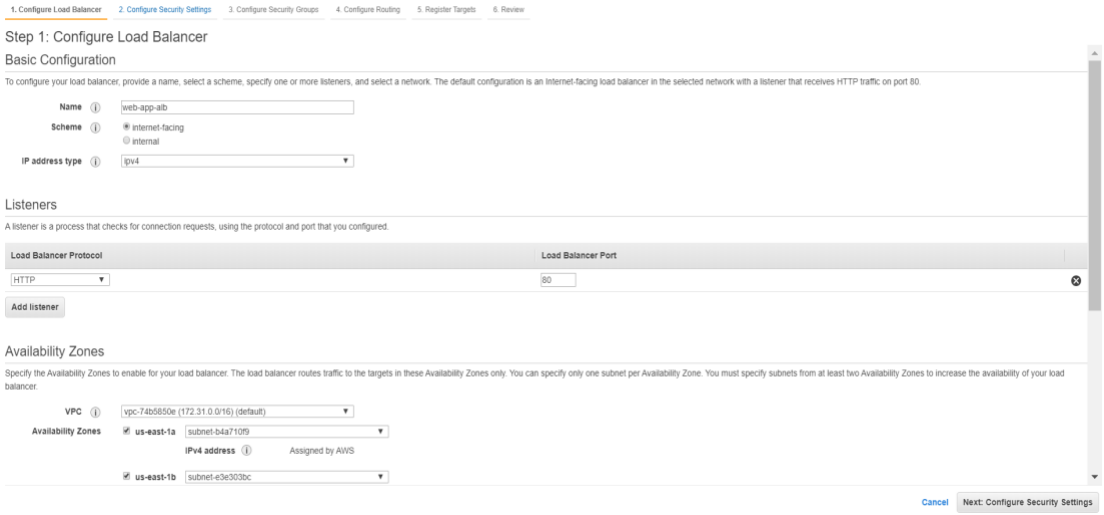


- Select the appropriate LB from the below screen for example: Application Load Balancer.

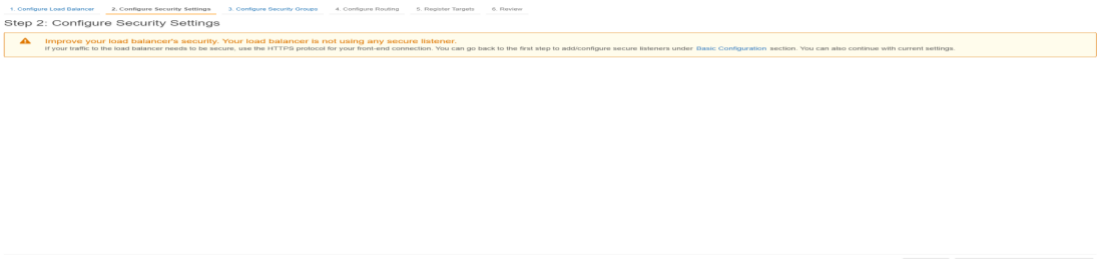


- After tapping Create option of Application Load Balancer. It redirects to Configure Load Balancer, to configure your load balancer, provide a name,

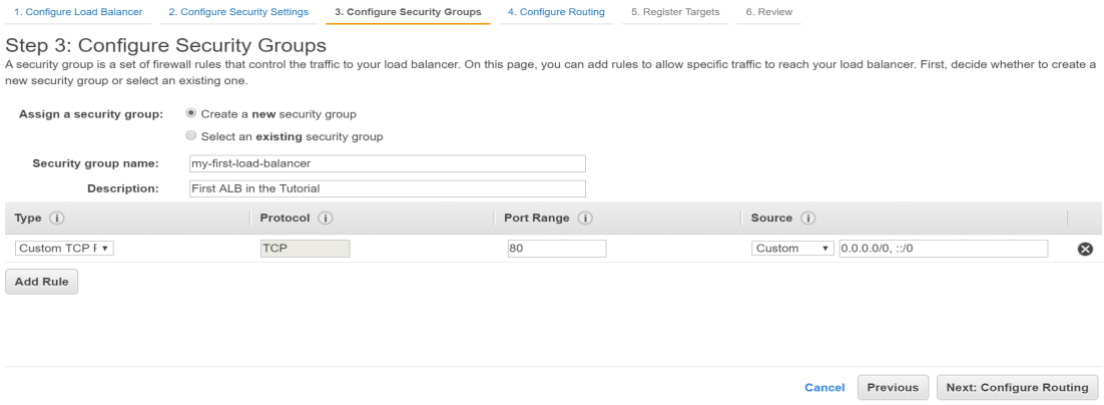
select a scheme, specify one or more listeners, and select a network. The default configuration is an Internet-facing load balancer in the selected network with a listener that receives HTTP traffic on port 80 and tap on Configure Security Settings as shown in the below screen.



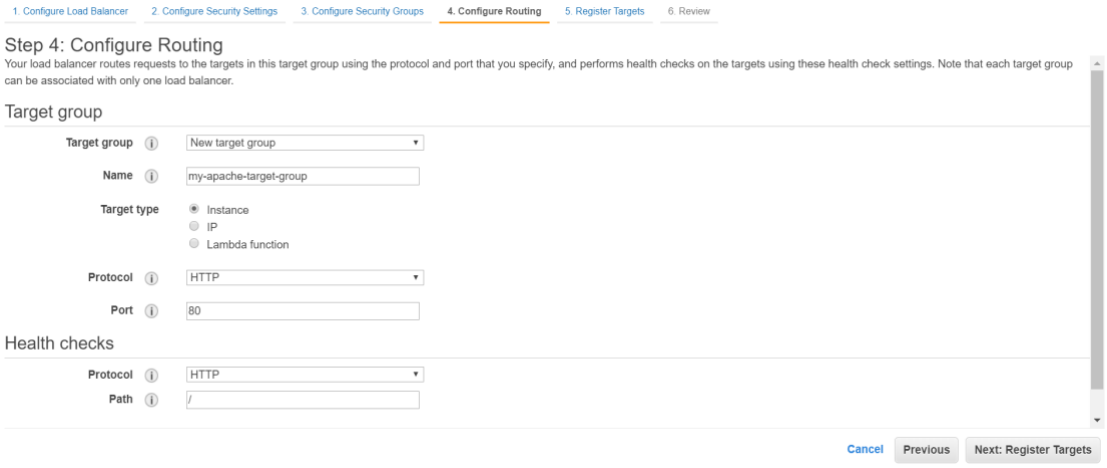
- It redirects to Configure Security Settings: Improve your load balancer's security. Your load balancer is not using any secure listener. If your traffic to the load balancer needs to be secure, use the HTTPS protocol for your front-end connection. You can go back to the first step to add/configure secure listeners under Basic Configuration section. You can also continue with current settings and Tap on Configure Security Groups as shown in below screen.



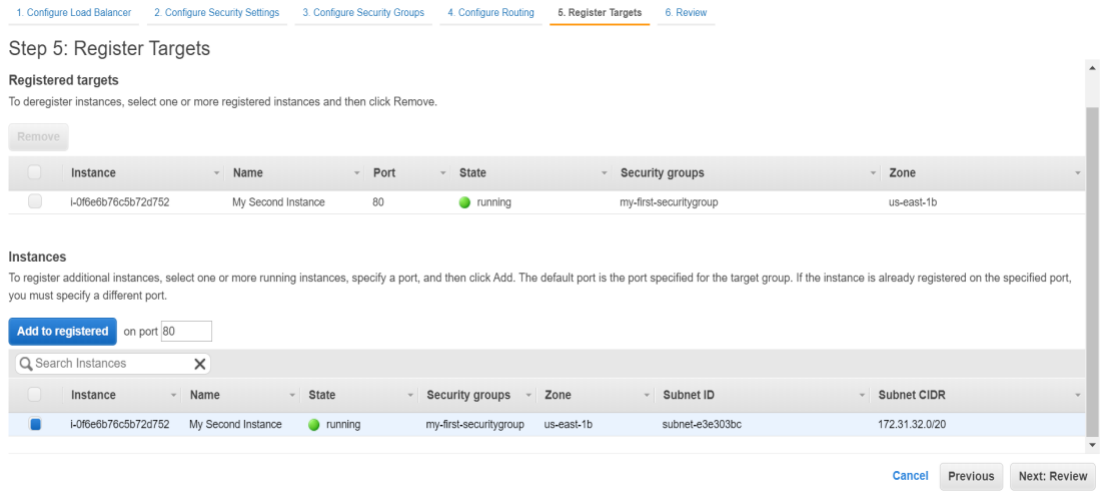
- We need to Configure Security Groups: A security group is a set of firewall rules that control the traffic to your load balancer. On this page, you can add rules to allow specific traffic to reach your load balancer. First, decide whether to create a new security group or select an existing one.



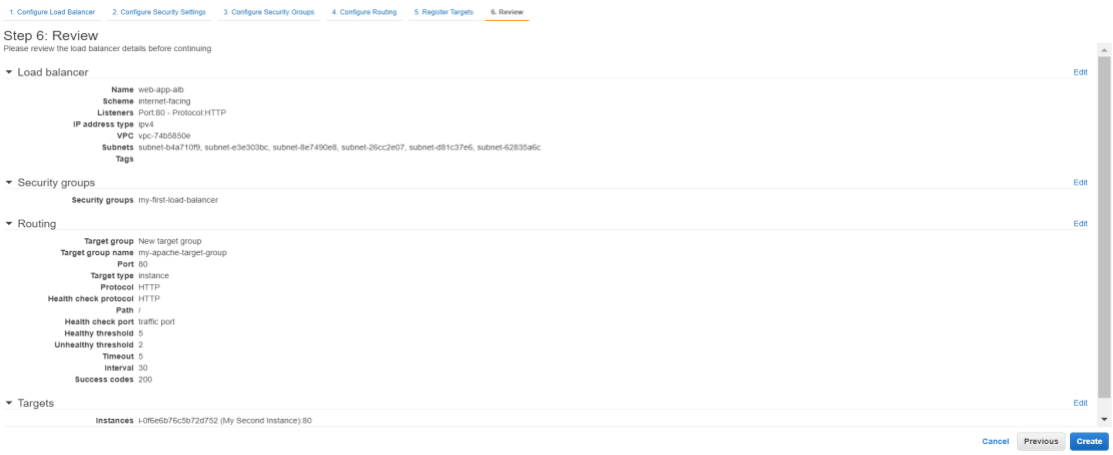
- Configure Routing: Your load balancer routes requests to the targets in this target group using the protocol and port that you specify, and performs health checks on the targets using these health check settings. Note that each target group can be associated with only one load balancer as shown in the below screen.



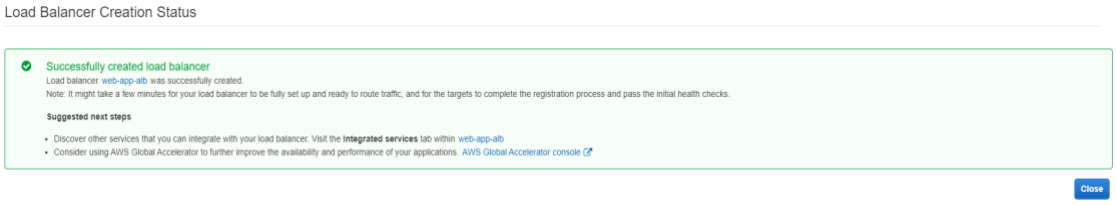
- Register Targets: To deregister instances, select one or more registered instances and then click Remove. Select the Instance and Tap on Add to registered and click on Review.



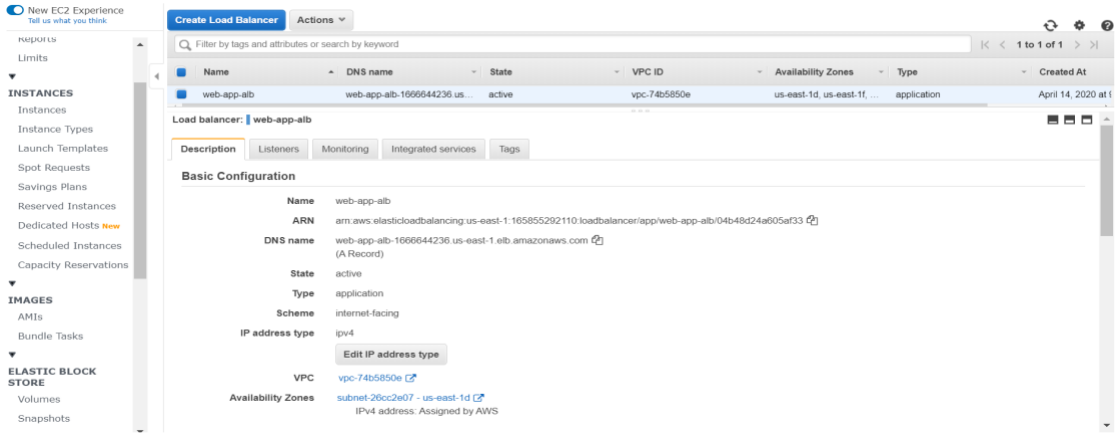
- Review: Please review the load balancer details before continuing, then tap on Create option.



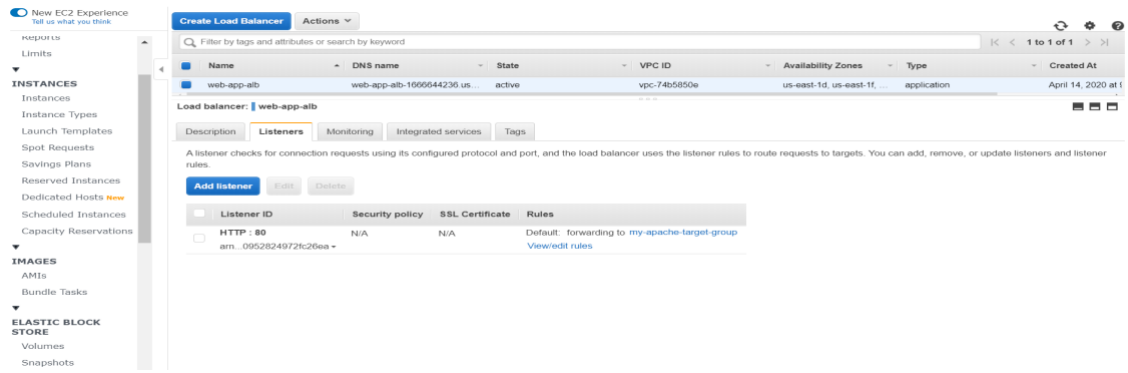
- Successfully Creation of Load Balancer is done.



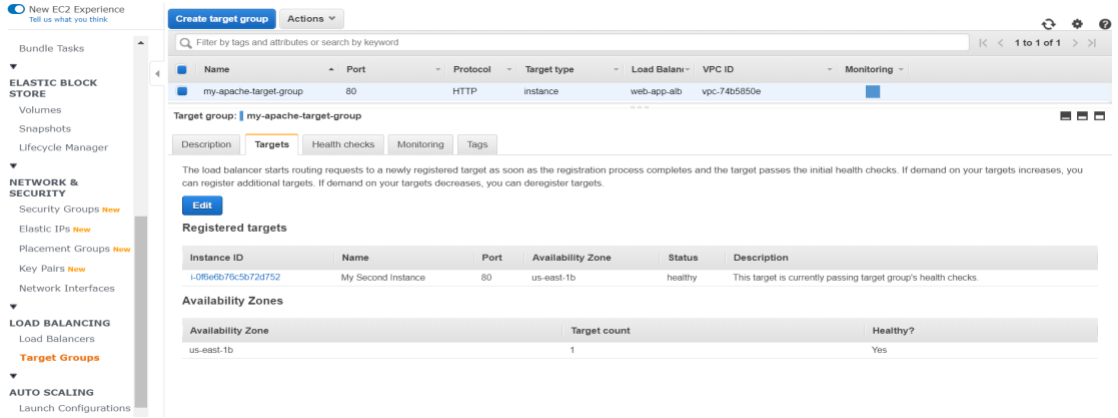
- After creating the Load Balancer, get the DNS name: web-app-alb-1666644236.us-east-1.elb.amazonaws.com and paste it on the browser. It gives us the results of the Instances requested for.



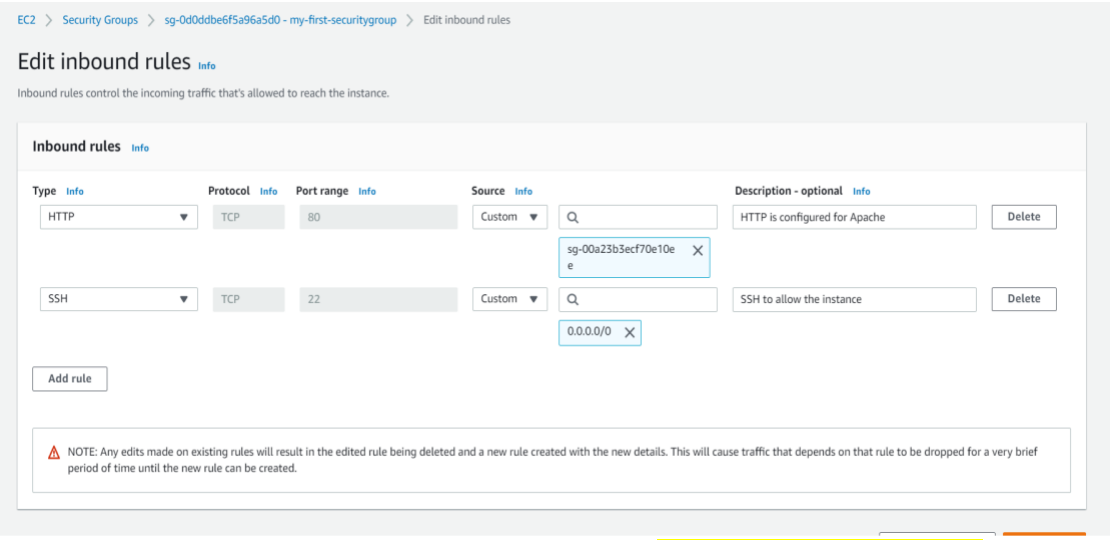
- If we check for the Listeners of Load Balancer, we can see that the requests target group is towards my-apache-target-group.



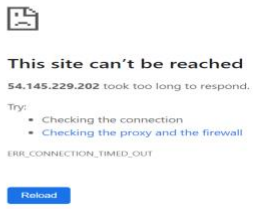
- We'll check for the Target Groups and EC2 Instances as shown in the below screen.

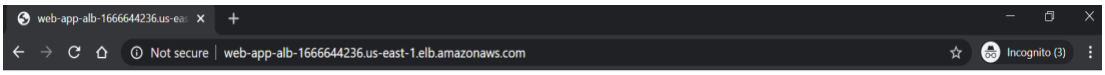


- In the Security Groups, Replace the Inbound traffic of EC2 Instance Security Groups Custom value from 0.0.0.0/0 to the Load Balancers value and Save it as shown in the below screen.



- Now check the Apache Server by running with the Public IP of EC2 Instance: 54.145.229.202 and DNS: web-app-alb-1666644236.us-east-1.elb.amazonaws.com of the Load Balancer connected with EC2 Instance as shown in the below screen.





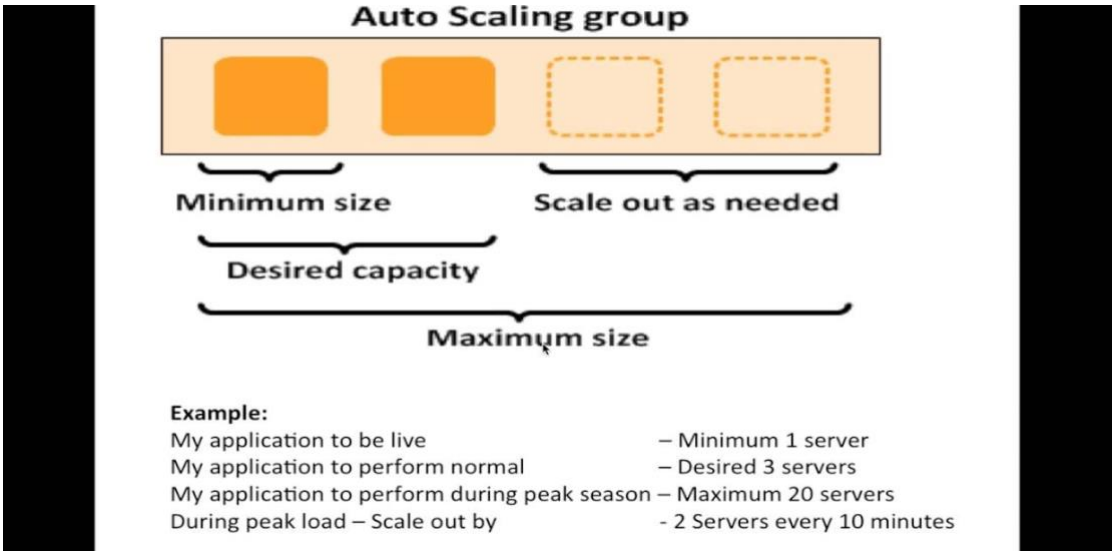
Hello Jay from ip-172-31-37-236.ec2.internal

7. Auto Scaling Group

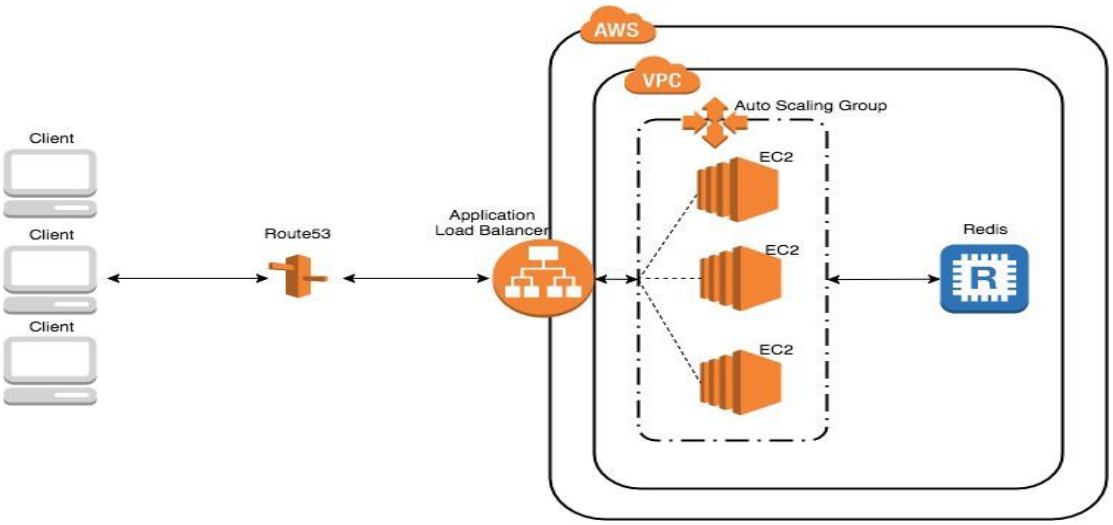
What's an Auto Scaling Group?

- In real-life, the load on your websites and application can change
- In the cloud, you can create and get rid of servers very quickly
- The goal of an Auto Scaling Group (ASG) is to:
  - Scale out (add EC2 Instances) to match an increased load
  - Scale in (remove EC2 Instances) to match a decreased load
  - Ensure we have a minimum and a maximum number of machines running
  - Automatically Register new instances to a load balancer

Auto Scaling Group in AWS



Auto Scaling Group in AWS with Load Balancer



ASGs have the following Attributes

- A launch configuration
  - AMI + Instance Type
  - EC2 User Data
  - EBS Volumes
  - Security Groups
  - SSH Key Pair
- Min Size / Max Size / Initial Capacity
- Network + Subnets Information
- Load Balancer Information
- Scaling Policies

Auto Scaling Alarms

- It is possible to scale an ASG based on CloudWatch alarms
- An Alarm monitors a metric (such as Average CPU)
- Metrics are computed for the overall ASG instances
- Based on the alarm:
    - We can scale-out policies (increase the number of instances)
    - We can create scale-in policies (decrease the number of instances)

Auto Scaling New Rules

- It is now possible to define "better" auto scaling rules that are directly managed by EC2
    - Target Average CPU Usage
    - Number of requests on the ELB per instance
    - Average Network In
    - Average Network Out
- These rules are easier to set up and can make more sense

Auto Scaling Custom Metric

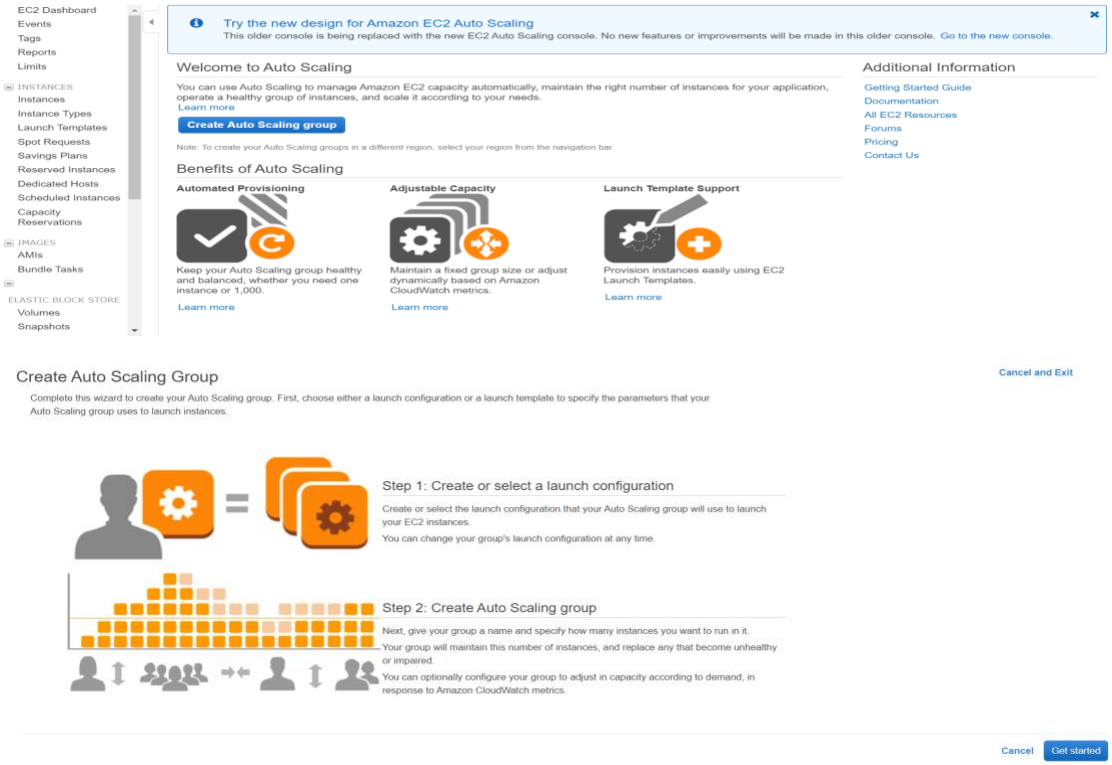
- We can auto scale based on a custom metric (ex: number of connected users)
- 1. Send custom metric from application on EC2 to CloudWatch (PutMatric API)
- 2. Create CloudWatch alarm to react to low / high values
- 3. Use the CloudWatch alarms as the scaling policy for ASG

ASG Brain dump

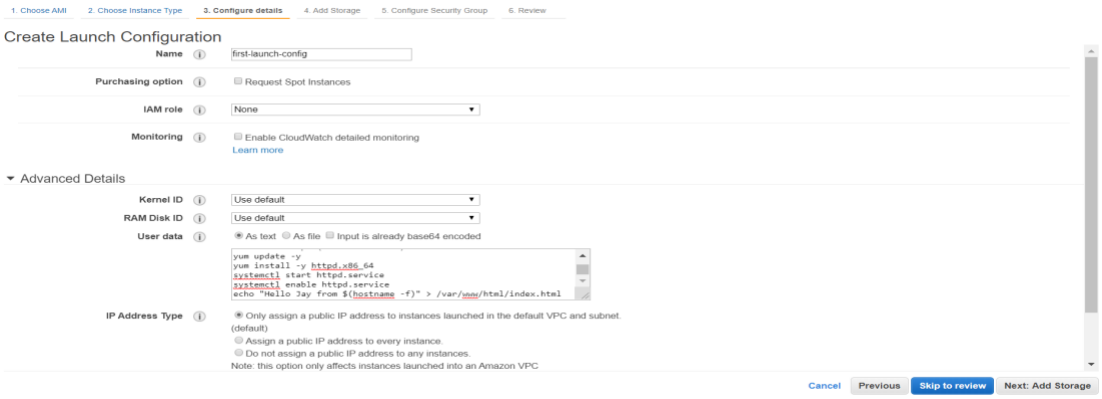
- Scaling policies can be on CPU, Network… and can even be on custom metrics or based on a schedule (if you know your visitor patterns)
- ASGs use Launch configurations and you update an ASG by providing a new launch configuration
- IAM roles attached to an ASG will get assigned to EC2 instances
- ASG are free. You pay for the underlying resources being launched
- Having instances under an ASG means that if they get terminated for whatever reason, the ASG will restart them. Extra safety!
- ASG can terminate instances marked as unhealthy by an LB (and hence replace them)

Auto Scaling Groups Hands-On

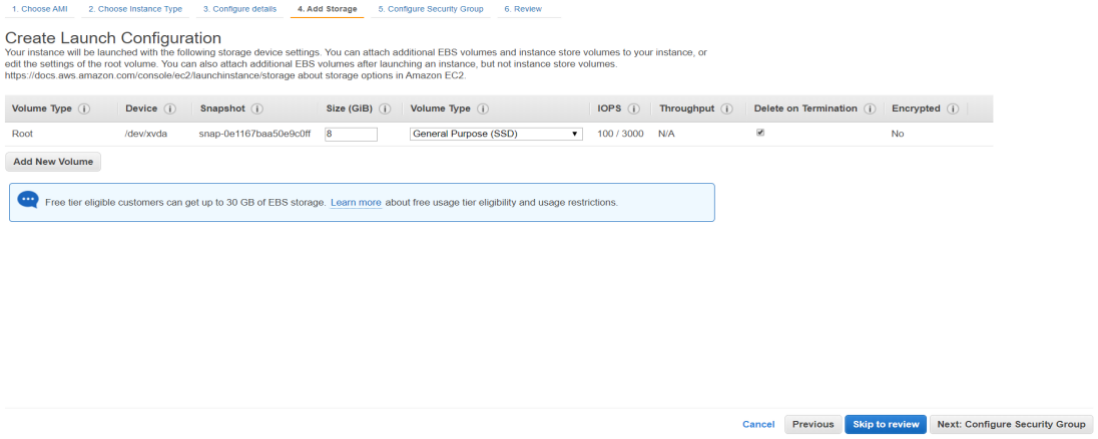
- Select Auto Scaling Groups which is present below the Load Balancer on the left hand side and it redirects to Create Auto Scaling Group, you can use Auto Scaling to manage Amazon EC2 capacity automatically, maintain the right number of instances for your application, operate a healthy group of instances, and scale it according to your needs. Tap on Get Started.



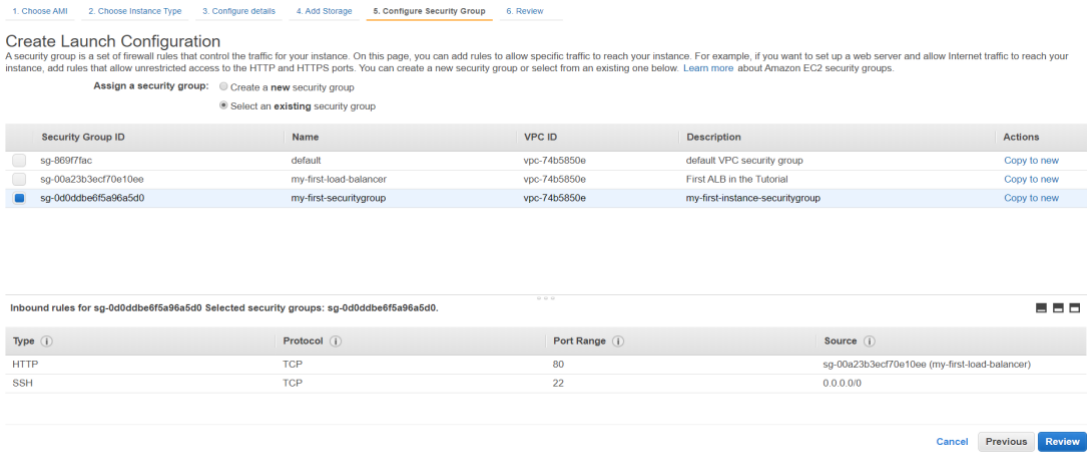
- It redirects to Create Launch Configuration page. An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs. Select Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0323c3dd2da7fb37d, same as we've did in the EC2 Instance Creation.
- Select t2.micro free tier eligible in the Create Launch Configuration and tap on Configure details.
- In this page, we need to specify the name (first-launch-config) of the launch configuration and in the advanced settings write the below steps in the user data and Tap on Add Storage.
    - #!/bin/bash
    - # install httpd (Linux 2 version)
    - yum update -y
    - yum install -y httpd.x86_64
    - systemctl start httpd.service
    - systemctl enable httpd.service
    - echo "Hello Jay from $(hostname -f)" > /var/www/html/index.html



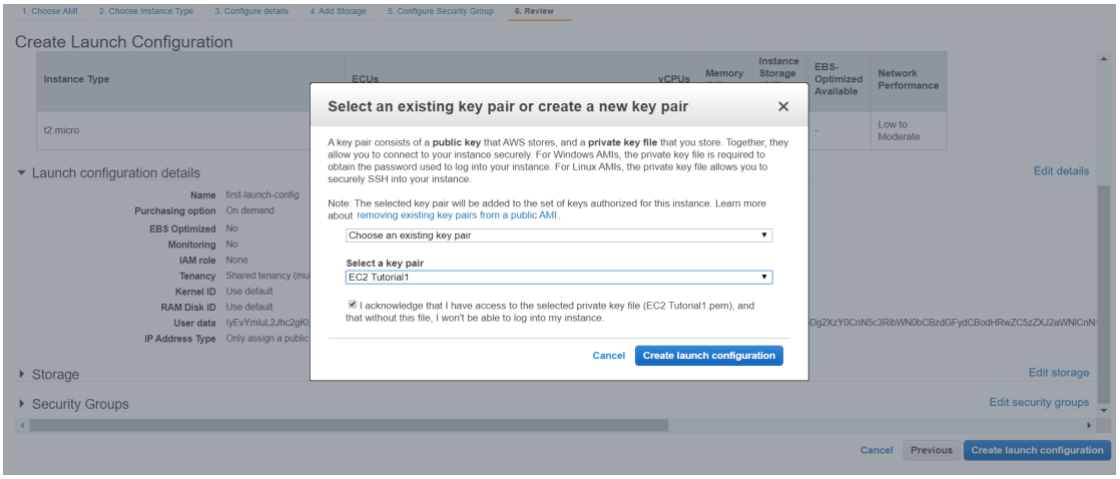
- it redirects to Add Storage screen, your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. https://docs.aws.amazon.com/console/ec2/launchinstance/storage about storage options in Amazon EC2 and tap on Configure Security Group.



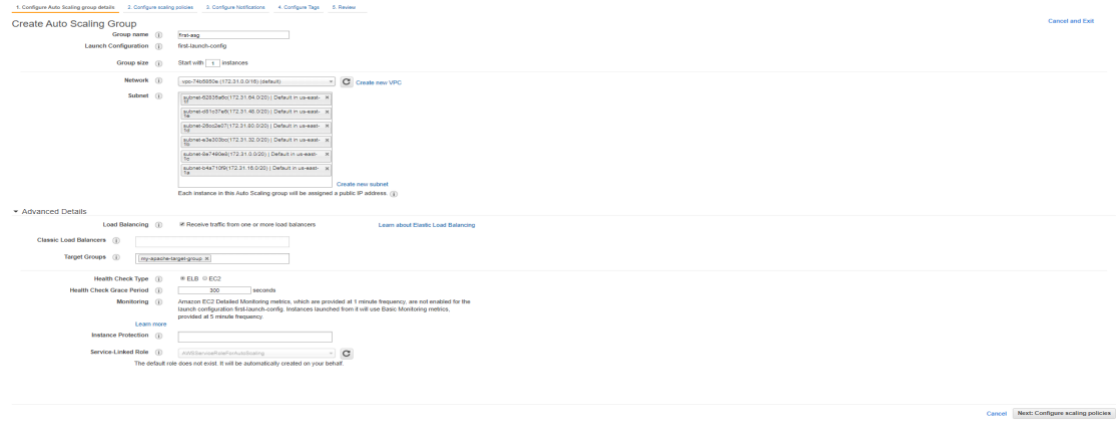
- In the Security Group, select an existing security group, tap on security group created for the EC2 Instances for Auto Scaling Group as well, as it contains SSH and HTTP protocol types and Load Balancer is configured to HTTP with port 80 as shown in the below screen. Tap on Review and we'll verify the Launch Configuration details.



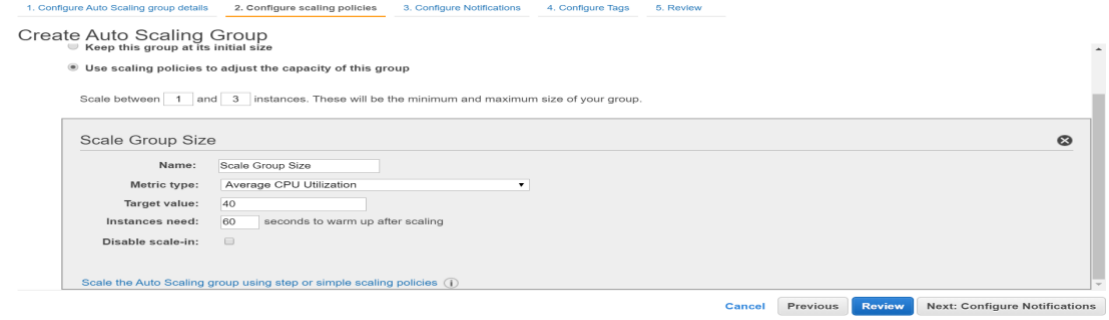
- A pop-up is generated for configuring key pair, we'll select existing key pair and Create launch configuration as shown below screen.



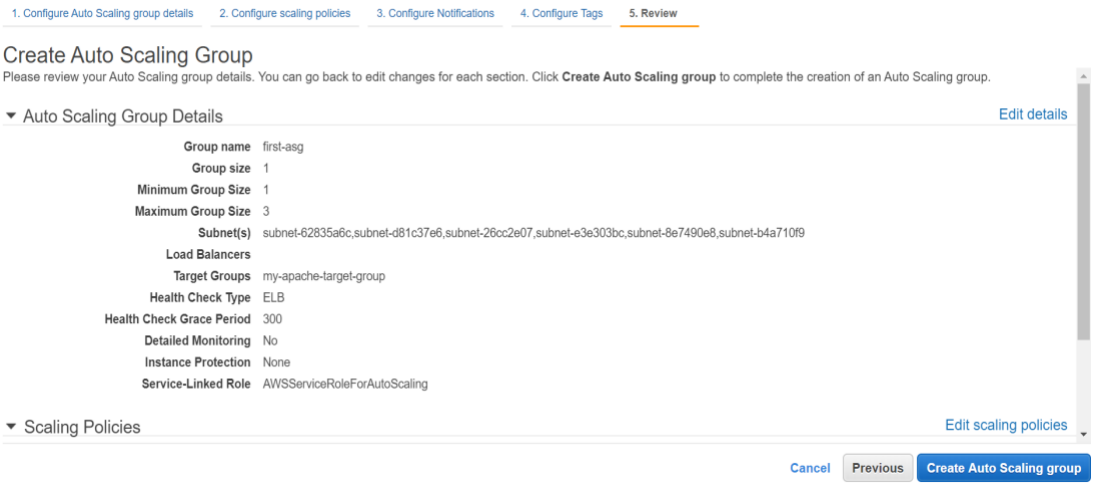
- It redirects to Create Auto Scaling Group, Name the ASG (first-asg), Group Size (1), Select the Network and Subnets and in the advanced details select the Target Groups, tick mark the LB traffic receiver and ELB for Health Check and Tap on Configure Scaling Policies.



- You can optionally add scaling policies if you want to adjust the size (number of instances) of your group automatically. A scaling policy is a set of instructions for making such adjustments in response to an Amazon CloudWatch alarm that you assign to it. In each policy, you can choose to add or remove a specific number of instances or a percentage of the existing group size, or you can set the group to an exact size. When the alarm triggers, it will execute the policy and adjust the size of your group accordingly. Select the use scaling policies to adjust the capacity of this group ash shown in the below screen.

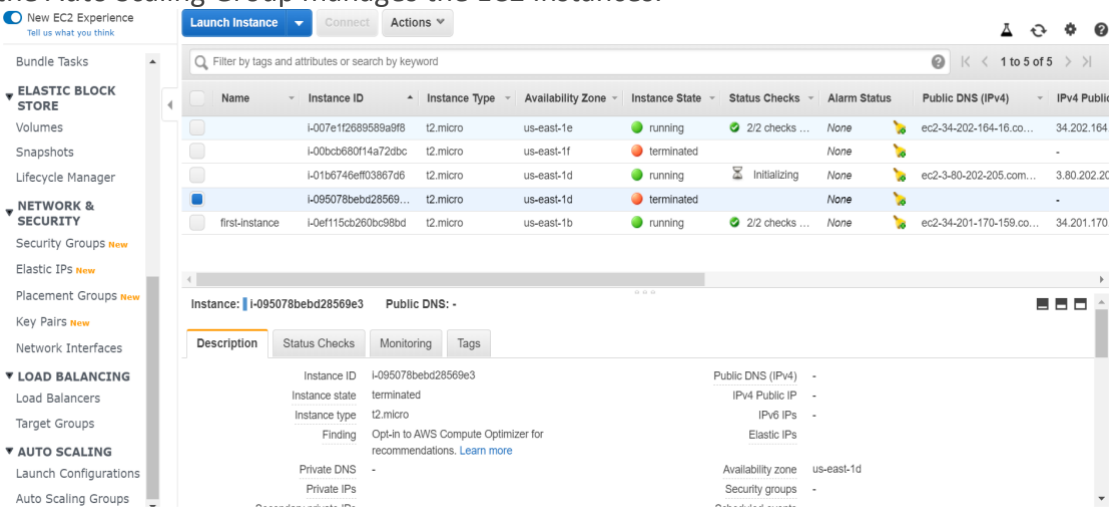


- Configure Notifications: Configure your Auto Scaling group to send notifications to a specified endpoint, such as an email address, whenever a specified event takes place, including: successful launch of an instance, failed instance launch, instance termination, and failed instance termination. If you created a new topic, check your email for a confirmation message and click the included link to confirm your subscription. Notifications can only be sent to confirmed addresses. Currently we won't need any Notifications.
- Configure Tags: A tag consists of a case sensitive key-value pair that you can use to identify your group. For example, you could define a tag with Key = Environment and Value = Production. You can optionally choose to apply these tags to instances in the group when they launch. Currently we won't need any Tags and Tap on Review.
- Please review your Auto Scaling group details. You can go back to edit changes for each section. Click **Create Auto Scaling group** to complete the creation of an Auto Scaling group.



Note: if you get an error here, just retry and it should fix itself

- Auto Scaling Group is created by following the above steps, it automatically creates EC2 Instances based on the traffic on the Instance.
- Copy the DNS Public name of the Load Balancer and paste on the browser, we'll understand that response is getting from the two different instances. As the Auto Scaling Group manages the EC2 Instances.



8. EBS Volumes Overview

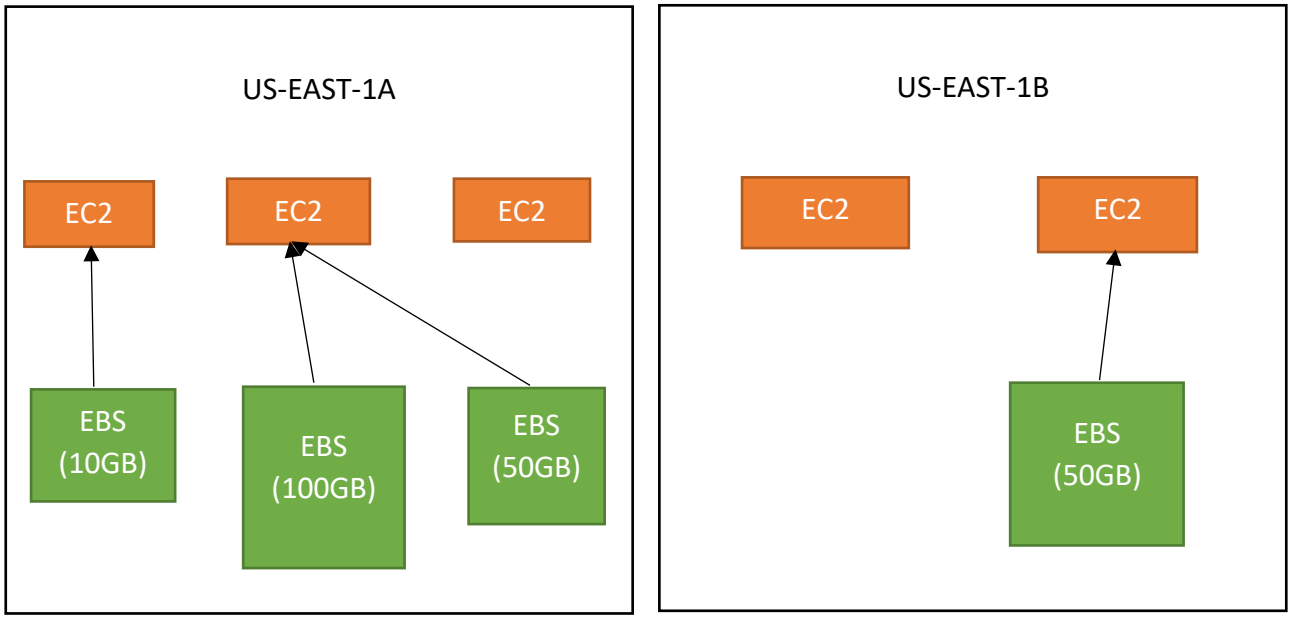
What's an EBS Volume?

- An EC2 machine loses its root volume (main drive) when it is manually terminated.
- Unexpected terminations might happen from time to time (AWS would email you)
- Sometimes, you need a way to store your instance data somewhere
- An EBS (Elastic Block Store) Volume is a network drive you can attach to your instances while they run
- It allows your instances to persist data

EBS Volume

- It's a network drive (i.e., not a physical drive)
    - It uses the network to communicate the instance, which means there might be a bit of latency
    - It can be detached from an EC2 instance and attached to another one quickly
- It's locked to an Availability Zone (AZ)
    - An EBS Volume in the us-east-1a can't be attached to us-east-1b
    - To move a volume across, you first need to snapshot it
- Have a provisioned capacity (size in GBs, and IOPS)
    - You get billed for all the provisioned capacity
    - You can increase the capacity of the drive over time

EBS Volume Example



EBS Volume Types

- EBS Volumes come in 4 types

    - **GP2 (SSD):** General Purpose SSD volume that balances price and performance for a wide variety of workloads
    - **IO1 (SSD):** Highest-performance SSD volume for mission-critical low-latency or high-throughput workloads
    - **ST1 (HDD):** Low cost HDD volume designed for frequently accessed, throughput-intensive workloads
    - **SC1 (HDD):** Lowest cost HDD volume designed for less frequently accessed workloads
- EBS Volumes are characterized in Size | Throughput | IOPS
- When in doubt always consult the AWS documentation – it's good!
- We've been only using GP2 volumes so far

EBS Volume Resizing

- Feb 2017: You can resize the EBS volumes
- You can only increase the EBS volumes:
    - Size (any volume type)
    - IOPS (only in IO1)
- After resizing an EBS volume, you need to repartition your drive

EBS Snapshots

- EBS Volumes can be backed up using "snapshots"
- Snapshots only take the actual space of the blocks on the volume
- If you snapshot a 100GB drive that only has 5GB of data, then your EBS snapshot will only be 5GB
- Snapshots are used for:
    - Backups: ensuring you can save your data in case of catastrophe
    - Volume migration:
        - Resizing a volume down
        - Changing the volume type
        - Encrypt a volume

EBS Encryption

- When you create an encrypted EBS volume, you get the following:
    - Data at rest is encrypted inside the volume
    - All the data in flight moving between the instance and the volume is encrypted
    - All snapshots are encrypted
    - All volumes created from the snapshot
- Encryption and decryption are handled transparently (you have nothing to do)
- Encryption has a minimal impact on latency
- EBS Encryption leverages keys from KMS (AES-256)
- Copying an unencrypted snapshot allows encryption

EBS Vs Instance Store

- Some instance does not come with Root EBS volumes
- Instead, they come with “Instance Store”.
- Instance store is physically attached to the machine
- Pros:
  - Better I/O performance
- Cons:
  - On termination, the instance store is lost
  - You can’t resize the instance store
  - Backups must be operated by the user
- Overall, EBS-backed instances should fit most applications workloads

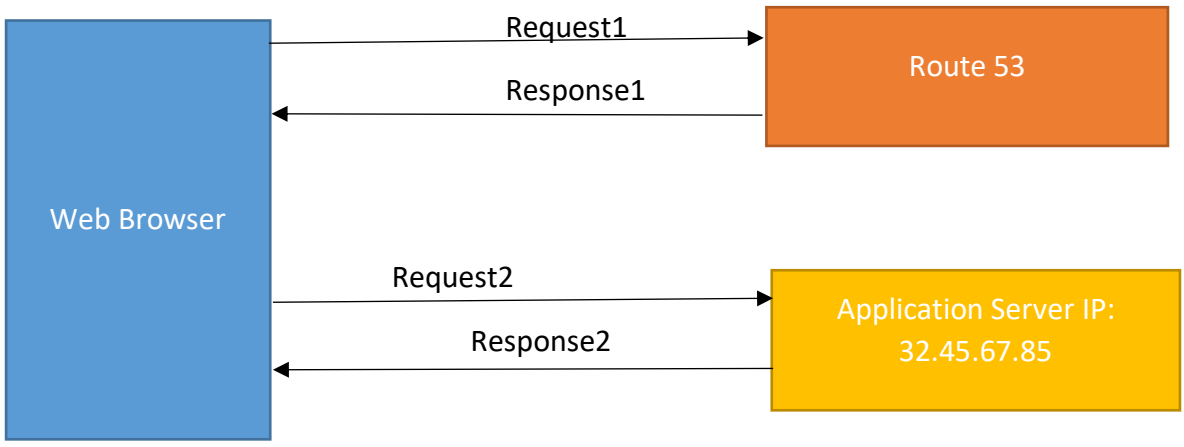
EBS Brain Dump

- EBS can be attached to only one instance at a time
- EBS are locked at the AZ level
- Migrating an EBS volume across AZ means first backing it up (snapshot), then recreating it in the other AZ
- EBS backups use IO and you shouldn’t run them while your application is handling a lot of traffic
- Root EBS Volumes of instances get terminated by default if the EC2 instance gets terminated. (you can disable that)

9. AWS Route 53

- Route53 is a Managed DNS (Domain Name System)
- DNS is a collection of rules and records which helps clients understand how to reach a server through URLs.
- In AWS, the most common records are:
  - A: URL to IPv4
  - AAAA: URL to IPv6
  - CNAME: URL to URL
  - Alias: URL to AWS resource.

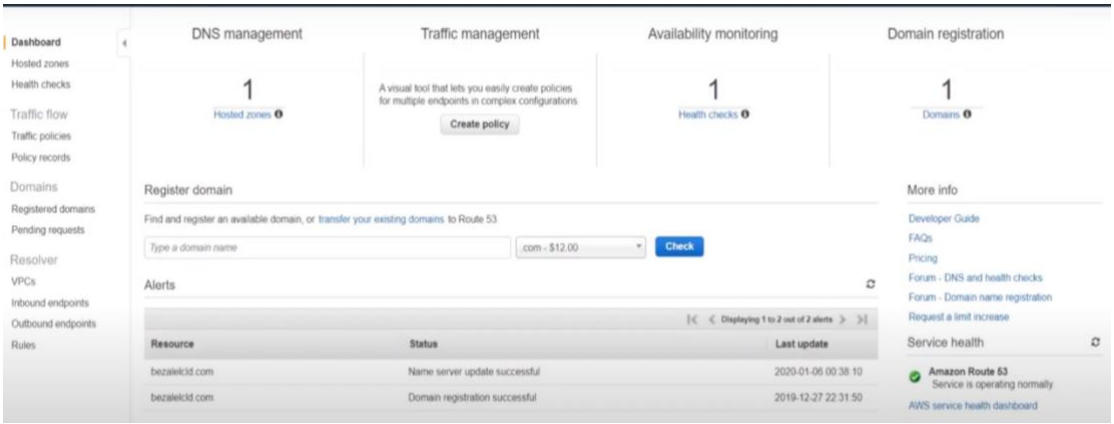
Route 53 – Diagram for a Record



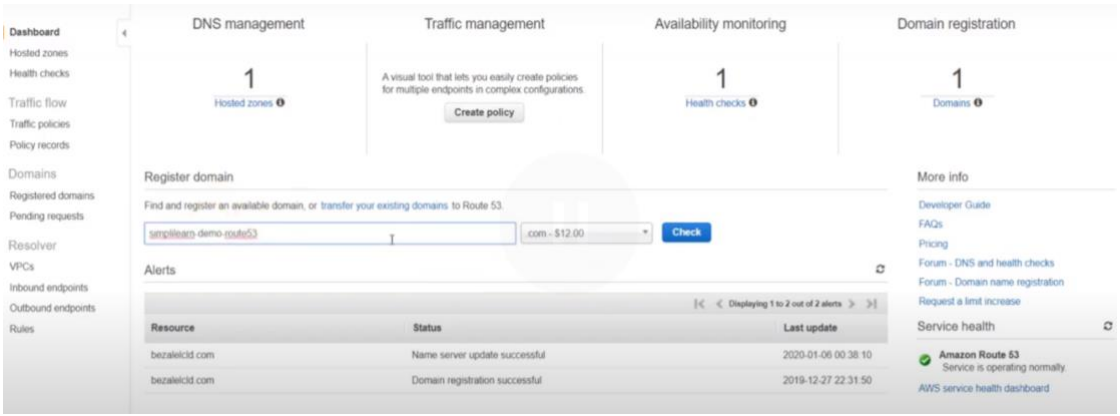
- Request1: http://myapp.mydomain.com
- Response1: Send back IP: 32.45.67.85 (A record: URL to IP)
- Request2: HTTP Request http://myapp.mydomain.com
- Response2: HTTP Response
- Route53 can use:
    - Public domain names you own (or buy) application1.mypublicdomain.com
    - Private domain names that can be resolved by your instances in your VPCs. application1.company.internal
- Route53 has advanced features such as:
    - Load balancing (through DNS – also called client load balancing)
    - Health checks (although limited. . .)
    - Routing policy: simple, failover, geolocation, geo-proximity, latency, weighted
- Prefer Alias over CNAME for AWS resources (for performance reasons)
- Overall Route53 is not much used in the AWS Certified Developer Exam.

AWS Route53 Hands-On

- To change the visibility of the URL on the browser, we use Route53 for changing the URL as per we need.
- Select Route53 from the Services of the AWS, go to DNS Management and Tap on Create Hosted Zone as shown in the below screen.



- Check for the domain, if exists select the particular domain or register with the new domain because it costs us $12.00 and tap on check option.



- If the user exists, then add it to the cart and Tap on Continue. It redirects to Contact details for your 1 domain to be filled and after finishing this step. Our Domain is successfully registered and tap on Go to Domain option.
- Then it creates Hosted Zone, if we tap on that hosted zone we'll find 2 types of hosted zones (NS, SOA).
- Now go back to EC2 Instance and Open Load Balancer, select DNS name of the Load Balancer.
- After selecting the DNS Name, go to Hosted Zone and Create a record Set and choose the type A-IPv4 address, choose the Alias as Yes, Alias Target as our Load Balancer and Tap on Create.
- We can see the Hosted Zone for the Load Balancers DNS Name, verify it by giving the name of the hosted zone on the browser. So that we can see our response from the Load Balancer with different URL.

10. AWS RDS Overview

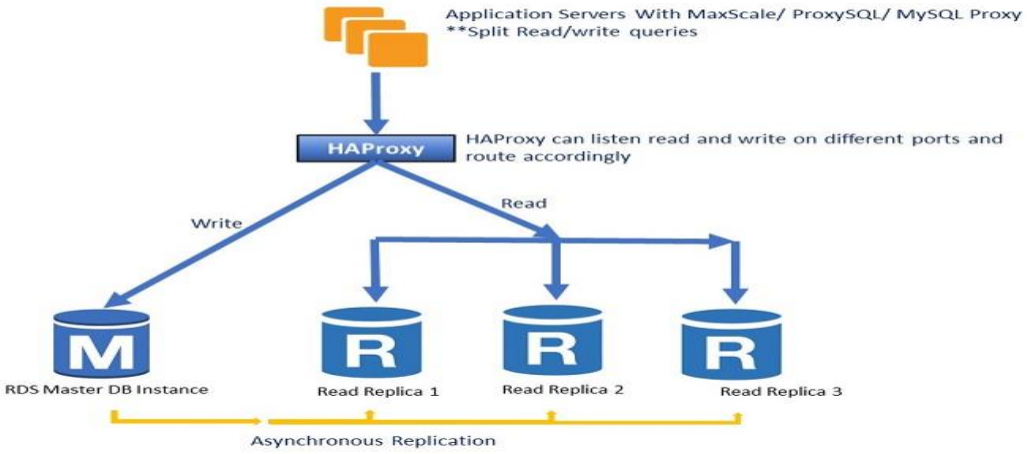
- RDS stands for Relational Database Service
- It's managed DB service for DB use SQL as a query language.
- It allows you to create databases in the cloud that are managed by AWS
    - PostgreSQL
    - Oracle
    - MySQL
    - Maria DB
    - Microsoft SQL Server
    - Aurora (AWS Proprietary database)

Advantage over using RDS versus deploying DB on EC2

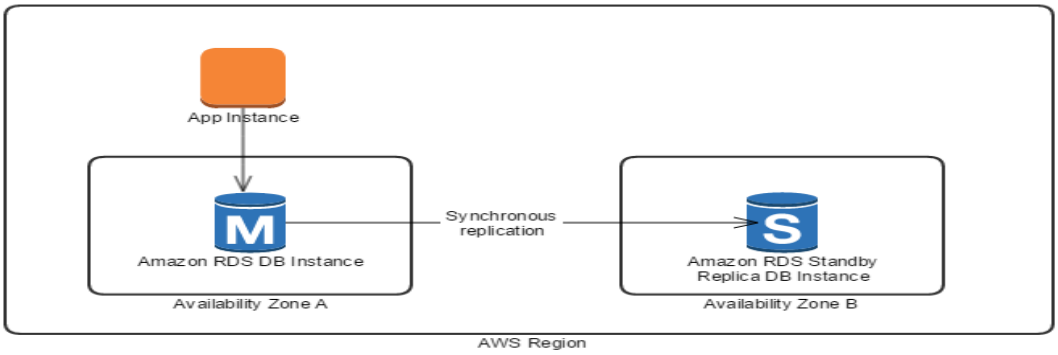
- Managed service:
- OS patching level
- Continuous backups and restore to specific timestamp (Point in Time Restore)!
- Monitoring dashboards
- Read replicas for improved read performance
- Multi AZ setup for DR (Disaster Recovery)
- Maintenance windows for upgrades
- Scaling capability (vertical and horizontal)
- But you can't SSH into your instances

RDS Read Replicas for read scalability

- Up to 5 Read Replicas
- Within AZ, Cross AZ or Cross Region
- Replication is ASYNC, so reads are eventually consistent
- Replicas can be promoted to their own DB
- Applications must update the connection string to leverage read replicas



RDS Multi AZ (Disaster Recovery)



- SYNC replication
- One DNS name – automatic app failure to standby
- Increase availability
- Failure in case of loss of AZ, loss of network, instance or strong failure
- No manual intervention in apps
- Not used for scaling

RDS Backups

- Backups are automatically enabled in RDS
- Automated backups:
  - Daily full snapshot of the database
  - Capture transaction logs in real time
  - => ability to restore to any point in time
  - 7 days' retention (can be increased to 35 days)
- DB Snapshots:
  - Manually triggered by the user
  - Retention of backup for as long as you want

RDS Encryption

- Encryption at rest capability with AWS KMS – AES-256 encryption
- SSL certificates to encrypt data to RDS in flight

- To Enforce SSL:
    - PostgreSQL: rds.force_ssl=1 in the AWS RDS Console (Parameter Groups)
    - MySQL: Within the DB:
      GRANT USAGE ON ** TO 'mysqluser'@'%' Require SSL;
- To connect using SSL:
    - Provide the SSL Trust certificate (can be download from AWS)
    - Provide SSL options when connecting to database

RDS Security

- RDS databases are usually deployed within a private subnet, not in a public one
- RDS Security works by leveraging security groups (the same concept as for EC2 instances) – it controls who can communicate with RDS
- IAM policies help control who can manage AWS RDS
- Traditional Username and Password can be used to login to the database
- IAM users can now be used too (for MySQL / Aurora – NEW!)

RDS vs Aurora

- Aurora is a proprietary, technology from AWS (not open sourced)
- PostgreSQL and MySQL are both supported as Aurora DB (that means your drivers will work as if Aurora was a PostgreSQL or MySQL database)
- Aurora is "AWS cloud optimized" and claims 5x performance improvement over MySQL on RDS, over 3x the performance of PostgreSQL on RDS
- Aurora storage automatically grows in increments of 10GB, up to 64TB.
- Aurora can have 15 replicas while MySQL has 5, and the replication process is faster (sub 10ms replica lag)
- Failover in Aurora is instantaneous. It's HA native
- Aurora costs more than RDS (20% more) – but id more efficient

AWS RDS Hands On

-