Adv. Devops Experiment no. 1

Name: Rohan Lalchandani
Class: D15A Roll no.: 25

Aim:
Part A) To host static and dynamic website on AWS and EC2.
Part B) To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.

Theory:

AWS EC2 (Elastic Compute Cloud)

1. What is it? AWS EC2 provides scalable virtual servers (instances) in the cloud. You can use EC2 to run applications and services without managing physical hardware.

2. Key Features:

- **Instance Types:** Various types optimized for compute, memory, storage, or GPU capabilities.
- **Auto Scaling:** Automatically adjusts the number of instances based on traffic.
- **Elastic Load Balancing (ELB):** Distributes incoming traffic across multiple instances.
- **Security:** Integrates with AWS Identity and Access Management (IAM) and provides security groups and network ACLs.
- **Pricing:** Pay-as-you-go or reserved instances for long-term commitments.

3. Common Use Cases:

- Hosting websites and web applications.
- Running big data analytics.
- Building development and test environments.
- Running high-performance computing (HPC) applications.

AWS S3 (Simple Storage Service)

1. What is it? AWS S3 is an object storage service that provides highly scalable and durable storage for a wide variety of data types.

2. Key Features:

- **Storage Classes:** Multiple classes like Standard, Intelligent-Tiering, One Zone-IA, Glacier for different use cases and cost savings.
- **Data Durability:** Designed for 99.999999999% durability over a given year.
- **Scalability:** Automatically scales to accommodate data growth.
- **Security:** Data encryption (at rest and in transit), and access control policies.
- **Versioning:** Keeps multiple versions of an object to recover from accidental deletions.

3. Common Use Cases:

- Backup and restore.
- Data archiving.
- Application data storage.
- Content distribution and hosting.

AWS Cloud9

1. What is it? AWS Cloud9 is a cloud-based Integrated Development Environment (IDE) that allows you to write, run, and debug code from a web browser.

2. Key Features:

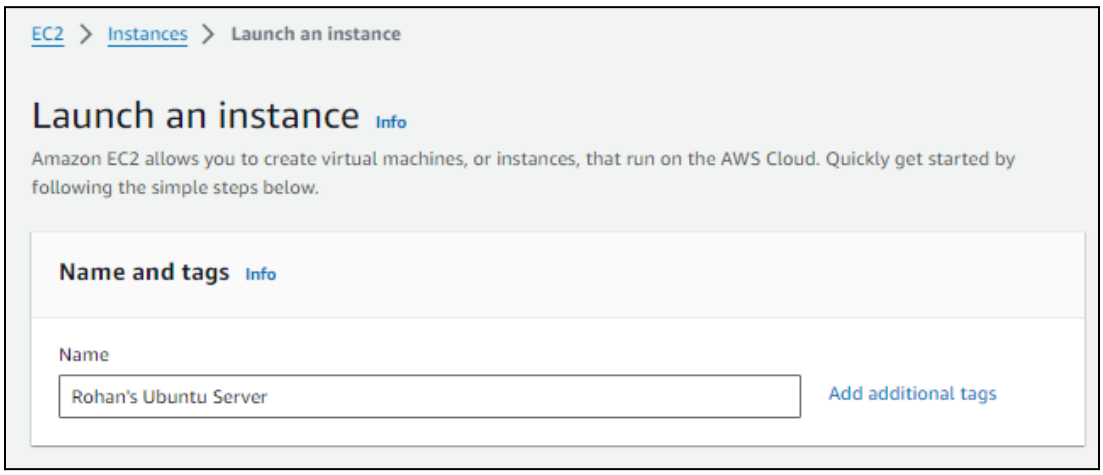
- **Environment:** Comes with pre-configured environments (Ubuntu-based) that include necessary tools and libraries.
- **Collaboration:** Multiple users can collaborate in real-time within the same IDE environment.
- **Integrated Tools:** Built-in terminal, debugger, and support for various programming languages.
- **AWS Integration:** Seamless integration with AWS services for deploying applications directly from the IDE.
- **Customizable:** Allows custom configurations and preferences for your development environment.

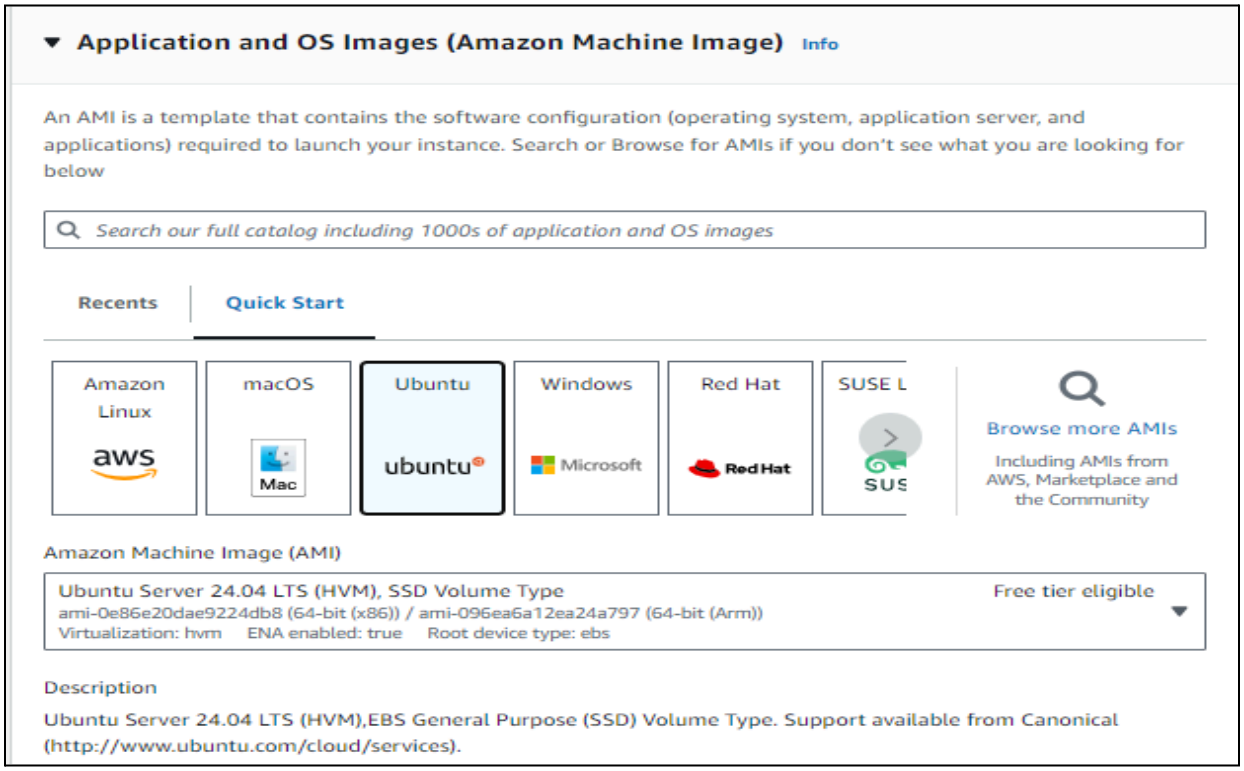
3. Common Use Cases:

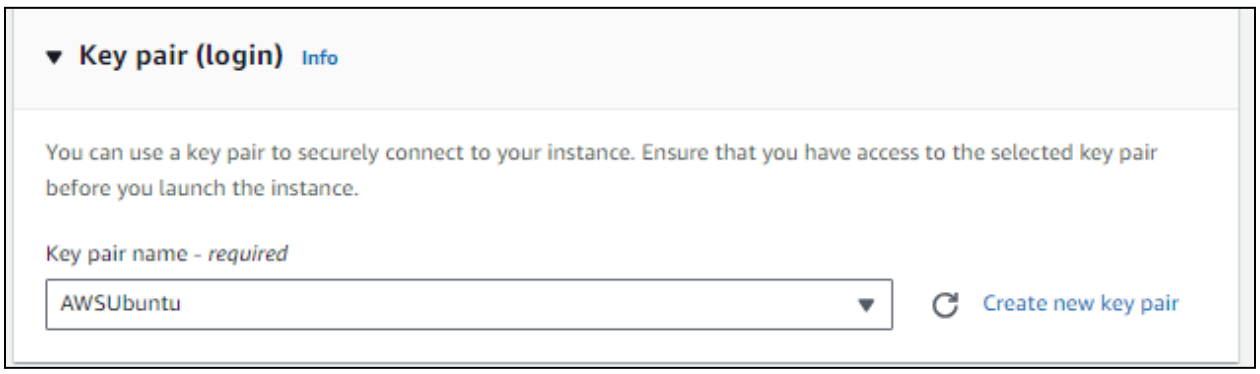
- Developing applications in a managed environment.
- Collaborative coding and debugging sessions.
- Learning and experimenting with code in a cloud-based IDE.

1. Static Website hosting using EC2(Ubuntu):

1) Instance creation and configuration:







Network settings Info

Edit

Network | Info

vpc-0f5e8abf0225b8a45

Subnet | Info

No preference (Default subnet in any availability zone)

Auto-assign public IP | Info

Enable

Additional charges apply when outside of free tier allowance

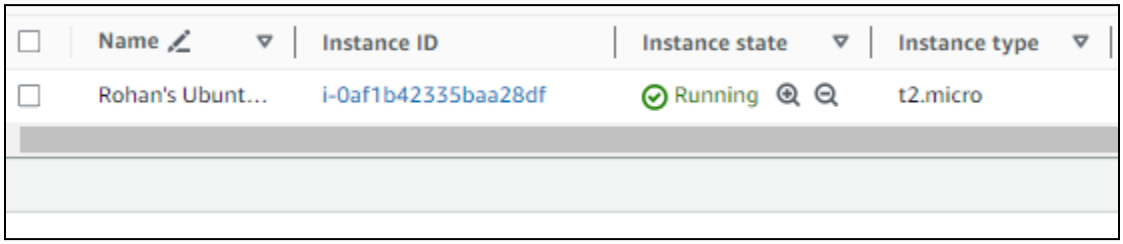
Firewall (security groups) | Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

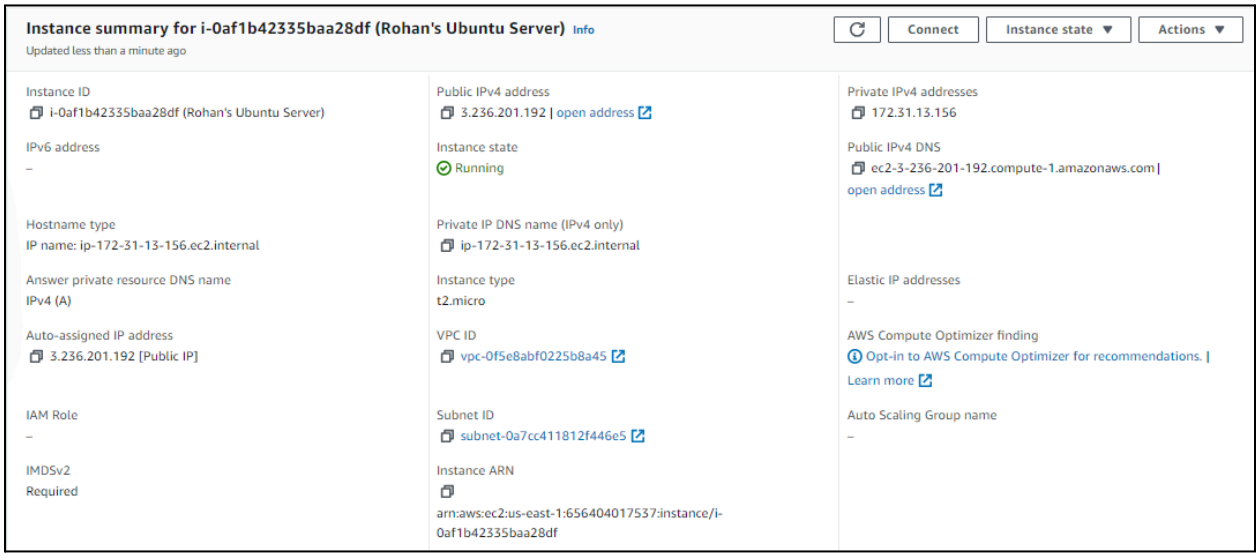
- (•) Create security group
- ( ) Select existing security group

We'll create a new security group called '**launch-wizard-3**' with the following rules:

- [x] Allow SSH traffic from
  Helps you connect to your instance
  Anywhere 0.0.0.0/0
- [ ] Allow HTTPS traffic from the internet
  To set up an endpoint, for example when creating a web server
- [ ] Allow HTTP traffic from the internet
  To set up an endpoint, for example when creating a web server



Name	Instance ID	Instance state	Instance type
Rohan's Ubunt...	i-0af1b42335baa28df	Running	t2.micro



Instance summary for i-0af1b42335baa28df (Rohan's Ubuntu Server) Info

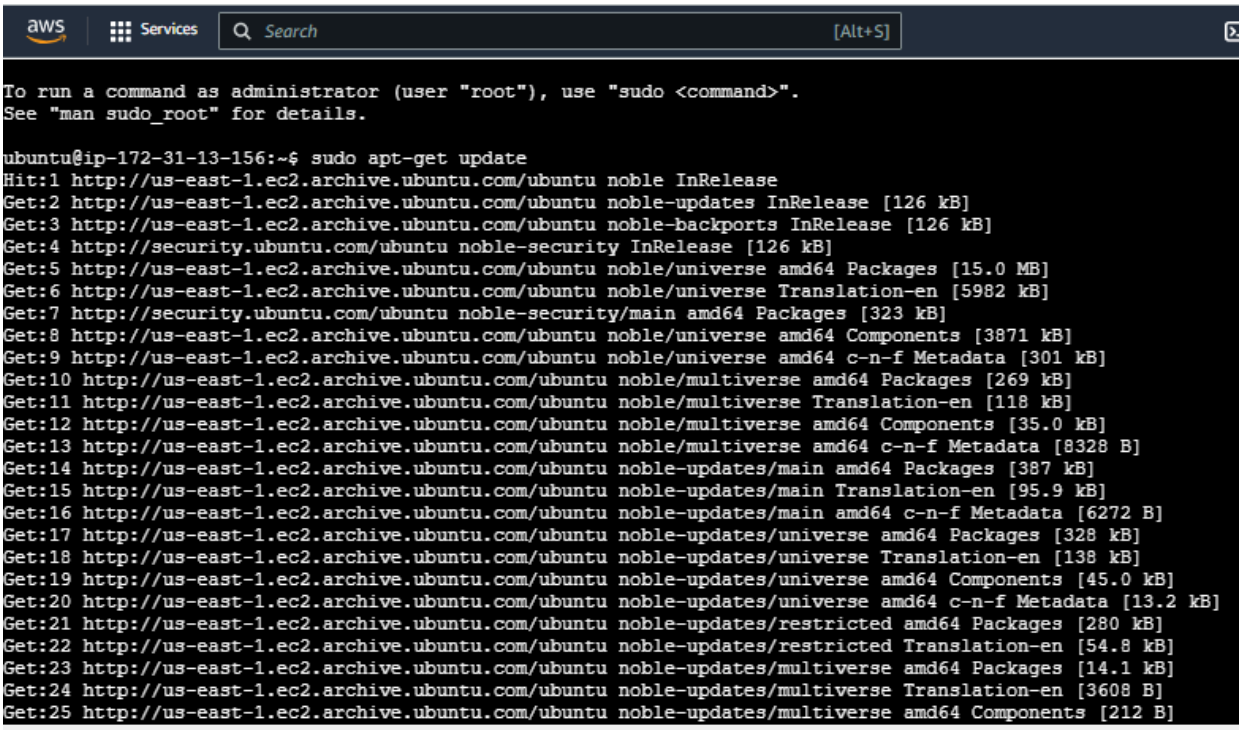
Updated less than a minute ago

Connect | Instance state | Actions

Instance ID i-0af1b42335baa28df (Rohan's Ubuntu Server)	Public IPv4 address 3.236.201.192 | open address	Private IPv4 addresses 172.31.13.156
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-3-236-201-192.compute-1.amazonaws.com | open address
Hostname type IP name: ip-172-31-13-156.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-13-156.ec2.internal	
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address 3.236.201.192 [Public IP]	VPC ID vpc-0f5e8abf0225b8a45	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role –	Subnet ID subnet-0a7cc411812f446e5	Auto Scaling Group name –
IMDSv2 Required	Instance ARN arn:aws:ec2:us-east-1:656404017537:instance/i-0af1b42335baa28df	

2) Connect to the instance and execute commands:

1. sudo apt-get update
2. sudo apt-get install apache2
3. systemctl status apache2

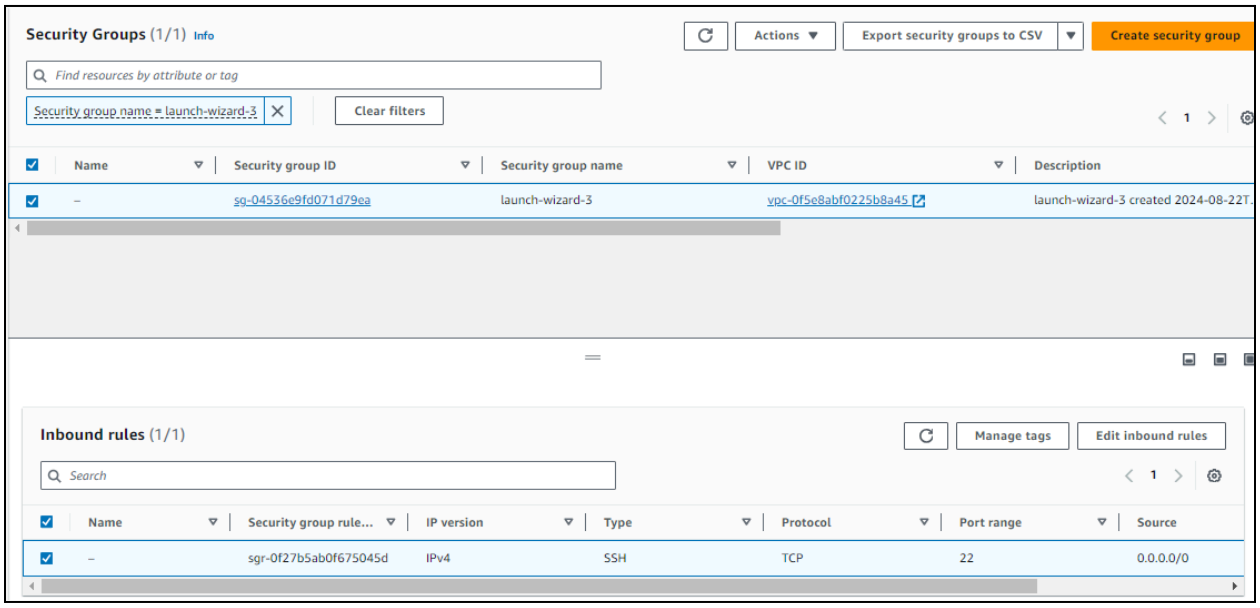


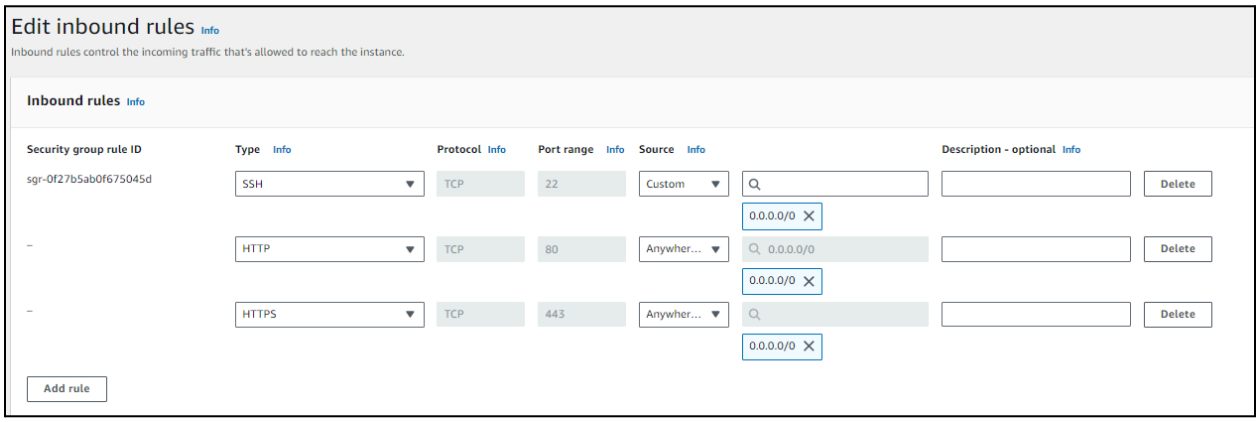

```
ubuntu@ip-172-31-13-156:~$ sudo apt-get install apache2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1-dbd-sqlite3 libaprutil1-ldap libaprutil1t64 liblua5.4-0 ssl-cert
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1-dbd-sqlite3 libaprutil1-ldap libaprutil1t64 liblua5.4-0 ssl-cert
0 upgraded, 10 newly installed, 0 to remove and 55 not upgraded.
Need to get 2083 kB of archives.
After this operation, 8094 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libapr1t64 amd64 1.7.2-3.1build2 [107 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1t64 amd64 1.6.3-1.1ubuntu7 [91.9 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1-dbd-sqlite3 amd64 1.6.3-1.1ubuntu7 [11.2 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1-ldap amd64 1.6.3-1.1ubuntu7 [9116 B]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 liblua5.4-0 amd64 5.4.6-3build2 [166 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-bin amd64 2.4.58-1ubuntu8.4 [1329 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-data all 2.4.58-1ubuntu8.4 [163 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-utils amd64 2.4.58-1ubuntu8.4 [97.1 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2 amd64 2.4.58-1ubuntu8.4 [90.2 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 ssl-cert all 1.1.2ubuntu1 [17.8 kB]
Fetched 2083 kB in 0s (25.3 MB/s)
```

```
Do you want to continue? [Y/n] y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libapr1t64 amd64 1.7.2-3.1build2 [107 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1t64 amd64 1.6.3-1.1ubuntu7 [91.9 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1-dbd-sqlite3 amd64 1.6.3-1.1ubuntu7 [11.2 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1-ldap amd64 1.6.3-1.1ubuntu7 [9116 B]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 liblua5.4-0 amd64 5.4.6-3build2 [166 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-bin amd64 2.4.58-1ubuntu8.4 [1329 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-data all 2.4.58-1ubuntu8.4 [163 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-utils amd64 2.4.58-1ubuntu8.4 [97.1 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2 amd64 2.4.58-1ubuntu8.4 [90.2 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 ssl-cert all 1.1.2ubuntu1 [17.8 kB]
Fetched 2083 kB in 0s (25.3 MB/s)
Preconfiguring packages ...
Selecting previously unselected package libapr1t64:amd64.
(Reading database ... 67741 files and directories currently installed.)
Preparing to unpack .../0-libapr1t64_1.7.2-3.1build2_amd64.deb ...
Unpacking libapr1t64:amd64 (1.7.2-3.1build2) ...
Selecting previously unselected package libaprutil1t64:amd64.
Preparing to unpack .../1-libaprutil1t64_1.6.3-1.1ubuntu7_amd64.deb ...
Unpacking libaprutil1t64:amd64 (1.6.3-1.1ubuntu7) ...
Selecting previously unselected package libaprutil1-dbd-sqlite3:amd64.
Preparing to unpack .../2-libaprutil1-dbd-sqlite3_1.6.3-1.1ubuntu7_amd64.deb ...
Unpacking libaprutil1-dbd-sqlite3:amd64 (1.6.3-1.1ubuntu7) ...
Selecting previously unselected package libaprutil1-ldap:amd64.
Preparing to unpack .../3-libaprutil1-ldap_1.6.3-1.1ubuntu7_amd64.deb ...
```

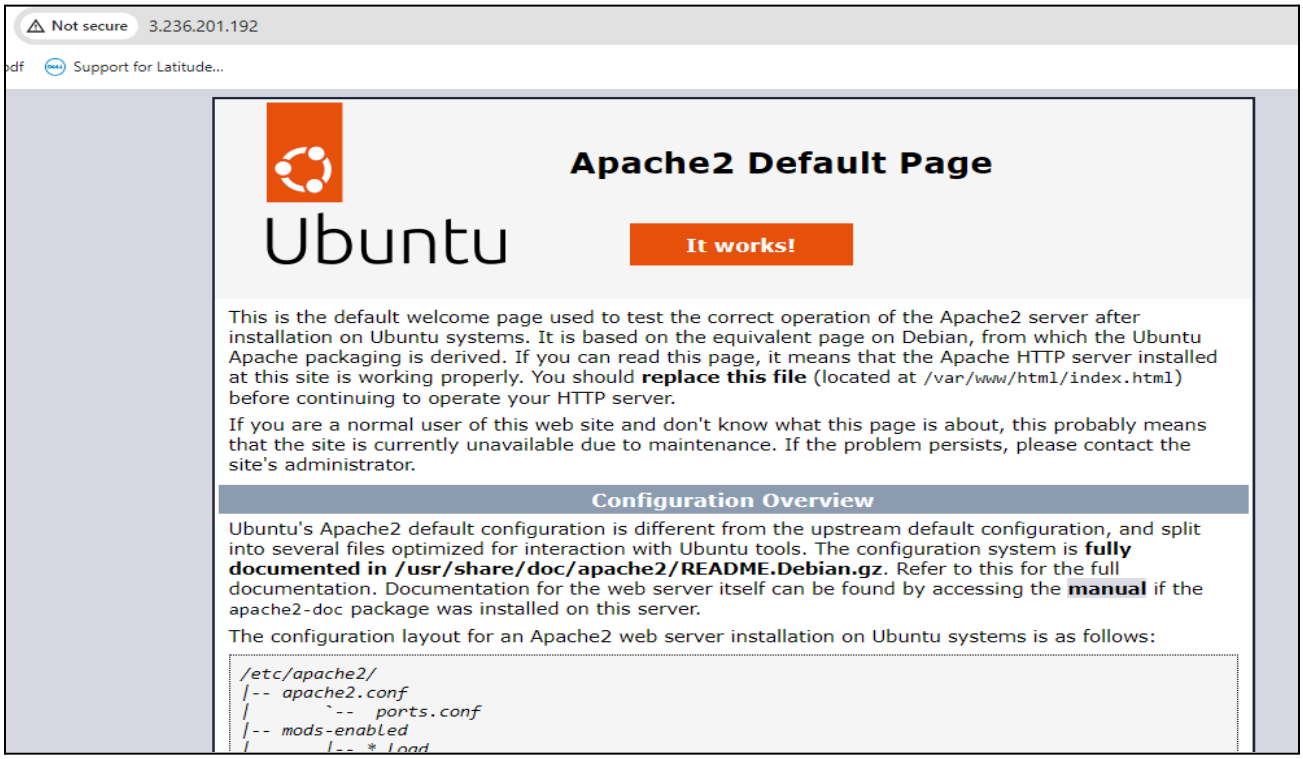
```
ubuntu@ip-172-31-13-156:~$ systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Thu 2024-08-22 17:23:12 UTC; 11min ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 2399 (apache2)
      Tasks: 55 (limit: 1130)
     Memory: 5.4M (peak: 5.6M)
        CPU: 70ms
     CGroup: /system.slice/apache2.service
             ├─2399 /usr/sbin/apache2 -k start
             ├─2402 /usr/sbin/apache2 -k start
             └─2403 /usr/sbin/apache2 -k start

Aug 22 17:23:12 ip-172-31-13-156 systemd[1]: Starting apache2.service - The Apache HTTP Server...
Aug 22 17:23:12 ip-172-31-13-156 systemd[1]: Started apache2.service - The Apache HTTP Server.
ubuntu@ip-172-31-13-156:~$ sudo su
root@ip-172-31-13-156:/home/ubuntu# cd /var/ww/html/
bash: cd: /var/ww/html/: No such file or directory
root@ip-172-31-13-156:/home/ubuntu# cd /var/www/html/
root@ip-172-31-13-156:/var/www/html#
```

3) Editing the inbound rules:



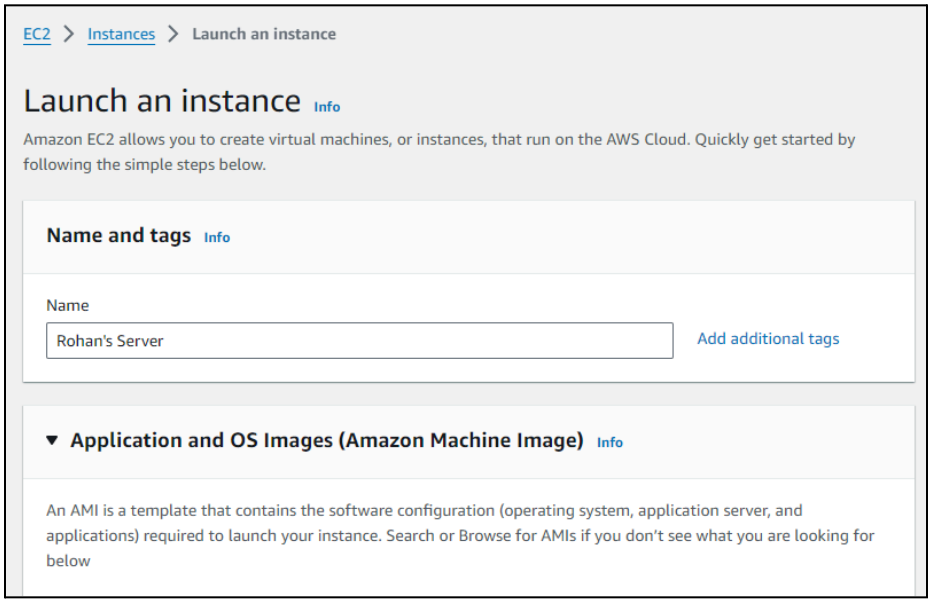


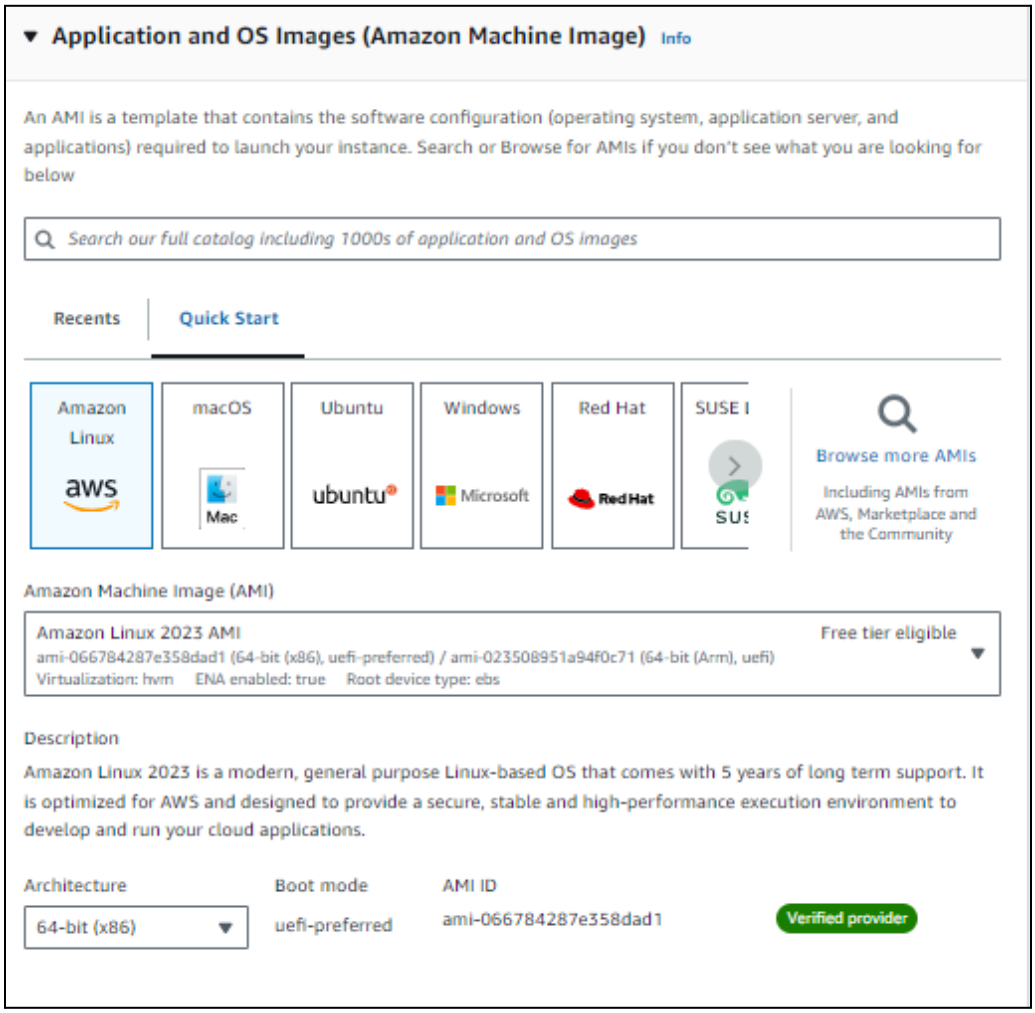
4) Go to the public IP link in new tab.

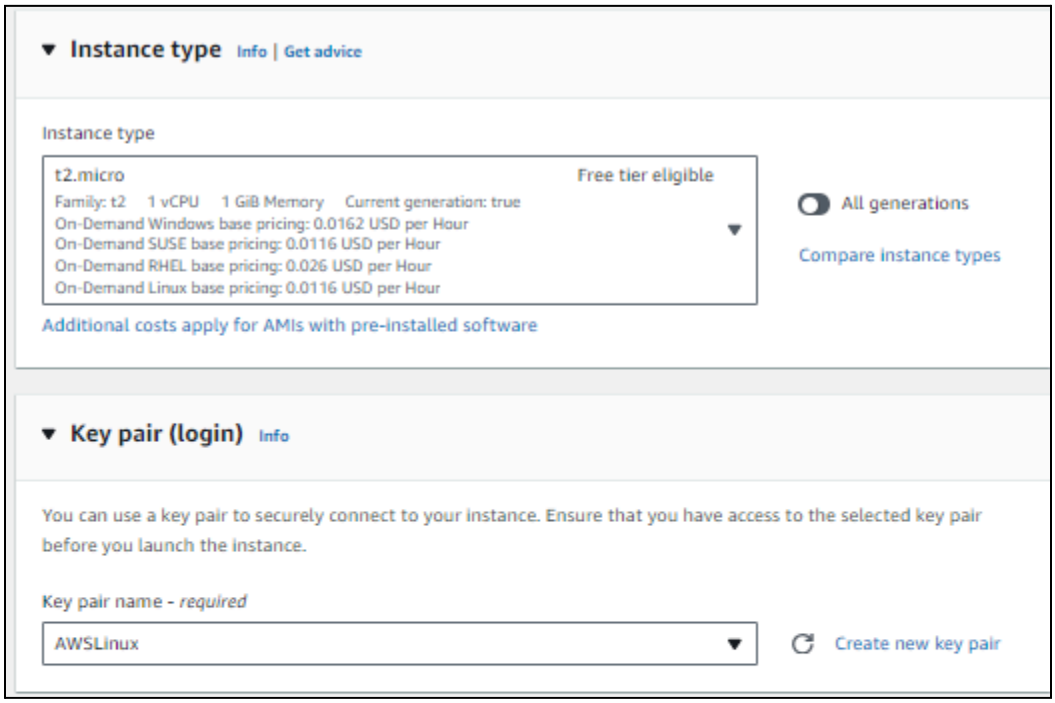


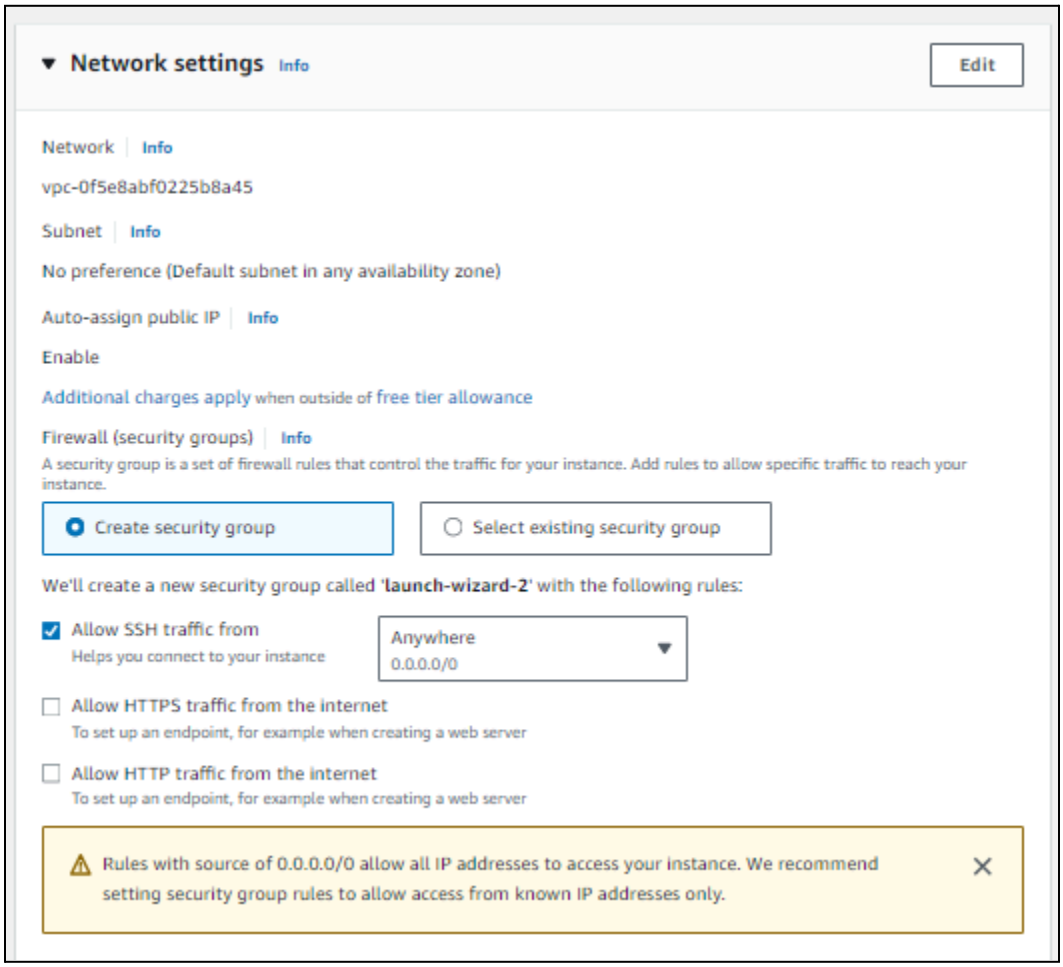
1. Dynamic hosting using EC2(Amazon Linux):

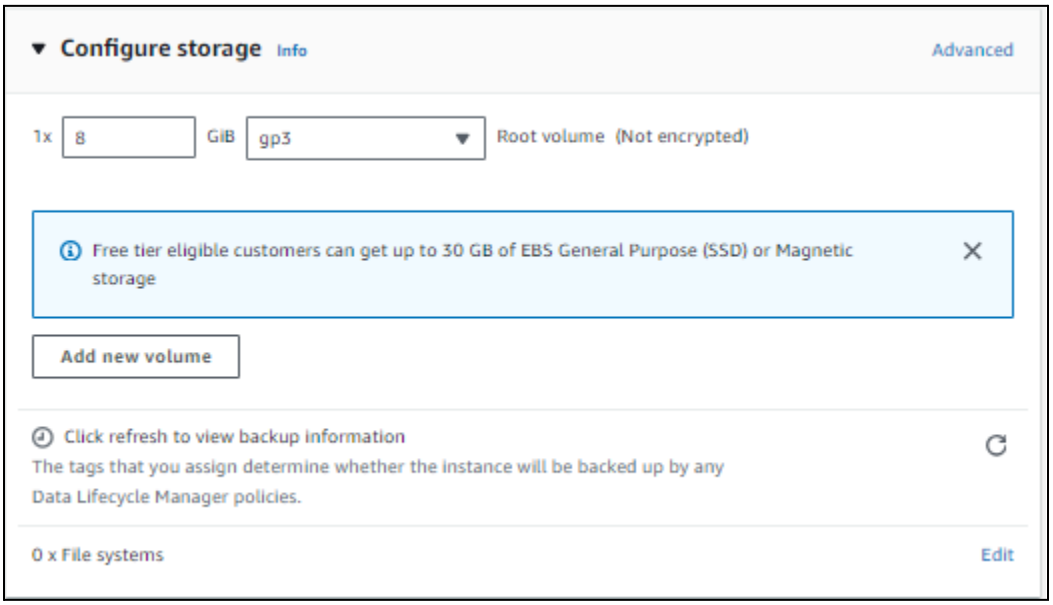
1) Instance creation:




Application and OS Images (Amazon Machine Image) Info
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below
Search our full catalog including 1000s of application and OS images
Recents
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE I
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)
Amazon Linux 2023 AMI
ami-066784287e358dad1 (64-bit (x86), uefi-preferred) / ami-023508951a94f0c71 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs
Free tier eligible
Description
Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.
Architecture
64-bit (x86)
Boot mode
uefi-preferred
AMI ID
ami-066784287e358dad1
Verified provider


Instance type Info | Get advice
Instance type
t2.micro
Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.026 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour
All generations
Compare instance types
Additional costs apply for AMIs with pre-installed software
Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.
Key pair name - required
AWSLinux
Create new key pair


Network settings Info
Edit
Network Info
vpc-0f5e8abf0225b8a45
Subnet Info
No preference (Default subnet in any availability zone)
Auto-assign public IP Info
Enable
Additional charges apply when outside of free tier allowance
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
We'll create a new security group called 'launch-wizard-2' with the following rules:
Allow SSH traffic from
Helps you connect to your instance
Anywhere
0.0.0.0/0
Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server
Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.


Configure storage Info
Advanced
1x
8
GiB
gp3
Root volume (Not encrypted)
Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage
Add new volume
Click refresh to view backup information
The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.
0 x File systems
Edit

2) Connect to the instance:

Execute commands:

1. sudo su -
2. yum update -y
3. yum install -y httpd
4. systemctl status httpd
5. mkdir aws_assg3
6. cd aws_assg3
7. For this experiment we have created a website which we have uploaded on Github.com.
8. Copy the Download Link for the .zip file of the website.
9. using the wget command, download the zip file to the folder.
10. unzip the main.zip file and navigate into the “OnlinePrintingServices-main” folder using the cd command.
11. move all the contents from the folder to “/var/www/html/”

```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Wed Aug 21 20:17:27 2024 from 18.206.107.28
[ec2-user@ip-172-31-51-115 ~]$ sudo su -
[root@ip-172-31-51-115 ~]# yum update -y
Last metadata expiration check: 0:10:51 ago on Wed Aug 21 20:12:36 2024.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-51-115 ~]# yum install -y httpd
Last metadata expiration check: 0:11:57 ago on Wed Aug 21 20:12:36 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture  Version                 Repository     Size
================================================================================
Installing:
 httpd                   x86_64        2.4.62-1.amzn2023       amazonlinux    48 k
Installing dependencies:
 apr                     x86_64        1.7.2-2.amzn2023.0.2    amazonlinux   129 k
 apr-util                x86_64        1.6.3-1.amzn2023.0.1    amazonlinux    98 k
 generic-logos-httpd     noarch        18.0.0-12.amzn2023.0.3  amazonlinux    19 k
 httpd-core              x86_64        2.4.62-1.amzn2023       amazonlinux   1.4 M
 httpd-filesystem        noarch        2.4.62-1.amzn2023       amazonlinux    14 k
 httpd-tools             x86_64        2.4.62-1.amzn2023       amazonlinux    81 k
 libbrotli               x86_64        1.0.9-4.amzn2023.0.2    amazonlinux   315 k
 mailcap                 noarch        2.1.49-3.amzn2023.0.3   amazonlinux    33 k
Installing weak dependencies:
 apr-util-openssl        x86_64        1.6.3-1.amzn2023.0.1    amazonlinux    17 k
 mod_http2               x86_64        2.0.27-1.amzn2023.0.3   amazonlinux   166 k
 mod_lua                 x86_64        2.4.62-1.amzn2023       amazonlinux    61 k

Transaction Summary
================================================================================
Install  12 Packages

Total download size: 2.3 M
Installed size: 6.9 M
Downloading Packages:
(1/12): apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64.rpm   256 kB/s |  17 kB  00:00
(2/12): apr-1.7.2-2.amzn2023.0.2.x86_64.rpm                1.8 MB/s | 129 kB  00:00
(3/12): apr-util-1.6.3-1.amzn2023.0.1.x86_64.rpm           1.2 MB/s |  98 kB  00:00
(4/12): generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch.rpm 1.0 MB/s |  19 kB  00:00
(5/12): httpd-2.4.62-1.amzn2023.x86_64.rpm                 2.3 MB/s |  48 kB  00:00
(6/12): httpd-filesystem-2.4.62-1.amzn2023.noarch.rpm      657 kB/s |  14 kB  00:00
(7/12): httpd-core-2.4.62-1.amzn2023.x86_64.rpm             34 MB/s | 1.4 MB  00:00
(8/12): libbrotli-1.0.9-4.amzn2023.0.2.x86_64.rpm           14 MB/s | 315 kB  00:00
(9/12): mailcap-2.1.49-3.amzn2023.0.3.noarch.rpm           1.4 MB/s |  33 kB  00:00
(10/12): mod_http2-2.0.27-1.amzn2023.0.3.x86_64.rpm        5.9 MB/s | 166 kB  00:00
(11/12): mod_lua-2.4.62-1.amzn2023.x86_64.rpm              3.5 MB/s |  61 kB  00:00
(12/12): httpd-tools-2.4.62-1.amzn2023.x86_64.rpm          708 kB/s |  81 kB  00:00
--------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------------------------------------------------------------------------------------
Total                                                                                                                       8.7 MB/s | 2.3 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                                                               1/1
  Installing       : apr-1.7.2-2.amzn2023.0.2.x86_64                                                                                              1/12
  Installing       : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64                                                                                 2/12
  Installing       : apr-util-1.6.3-1.amzn2023.0.1.x86_64                                                                                         3/12
  Installing       : mailcap-2.1.49-3.amzn2023.0.3.noarch                                                                                         4/12
  Installing       : httpd-tools-2.4.62-1.amzn2023.x86_64                                                                                         5/12
  Installing       : libbrotli-1.0.9-4.amzn2023.0.2.x86_64                                                                                        6/12
  Running scriptlet: httpd-filesystem-2.4.62-1.amzn2023.noarch                                                                                    7/12
  Installing       : httpd-filesystem-2.4.62-1.amzn2023.noarch                                                                                    7/12
  Installing       : httpd-core-2.4.62-1.amzn2023.x86_64                                                                                          8/12
  Installing       : mod_http2-2.0.27-1.amzn2023.0.3.x86_64                                                                                       9/12
  Installing       : mod_lua-2.4.62-1.amzn2023.x86_64                                                                                            10/12
  Installing       : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch                                                                           11/12
  Installing       : httpd-2.4.62-1.amzn2023.x86_64                                                                                              12/12
  Running scriptlet: httpd-2.4.62-1.amzn2023.x86_64                                                                                              12/12
  Verifying        : apr-1.7.2-2.amzn2023.0.2.x86_64                                                                                              1/12
  Verifying        : apr-util-1.6.3-1.amzn2023.0.1.x86_64                                                                                         2/12
  Verifying        : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64                                                                                 3/12
  Verifying        : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch                                                                            4/12
  Verifying        : httpd-2.4.62-1.amzn2023.x86_64                                                                                               5/12
  Verifying        : httpd-core-2.4.62-1.amzn2023.x86_64                                                                                          6/12
  Verifying        : httpd-filesystem-2.4.62-1.amzn2023.noarch                                                                                    7/12
  Verifying        : httpd-tools-2.4.62-1.amzn2023.x86_64                                                                                         8/12
```

```
  Verifying        : httpd-tools-2.4.62-1.amzn2023.x86_64                                                                                         8/12
  Verifying        : libbrotli-1.0.9-4.amzn2023.0.2.x86_64                                                                                        9/12
  Verifying        : mailcap-2.1.49-3.amzn2023.0.3.noarch                                                                                        10/12
  Verifying        : mod_http2-2.0.27-1.amzn2023.0.3.x86_64                                                                                      11/12
  Verifying        : mod_lua-2.4.62-1.amzn2023.x86_64                                                                                            12/12

Installed:
  apr-1.7.2-2.amzn2023.0.2.x86_64          apr-util-1.6.3-1.amzn2023.0.1.x86_64    apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64    generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
  httpd-2.4.62-1.amzn2023.x86_64           httpd-core-2.4.62-1.amzn2023.x86_64     httpd-filesystem-2.4.62-1.amzn2023.noarch       httpd-tools-2.4.62-1.amzn2023.x86_64
  libbrotli-1.0.9-4.amzn2023.0.2.x86_64    mailcap-2.1.49-3.amzn2023.0.3.noarch    mod_http2-2.0.27-1.amzn2023.0.3.x86_64          mod_lua-2.4.62-1.amzn2023.x86_64

Complete!
[root@ip-172-31-51-115 ~]# systemctl status httpd
○ httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:httpd.service(8)
[root@ip-172-31-51-115 ~]# mkdir aws_assg3
[root@ip-172-31-51-115 ~]# cd aws_assg3
[root@ip-172-31-51-115 aws_assg3]# wget https://github.com/Rohan-Lalchandani08/OnlinePrintingServices.git
--2024-08-21 20:32:34--  https://github.com/Rohan-Lalchandani08/OnlinePrintingServices.git
Resolving github.com (github.com)... 140.82.113.4
Connecting to github.com (github.com)|140.82.113.4|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://github.com/Rohan-Lalchandani08/OnlinePrintingServices [following]
--2024-08-21 20:32:34--  https://github.com/Rohan-Lalchandani08/OnlinePrintingServices
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'OnlinePrintingServices.git'
```

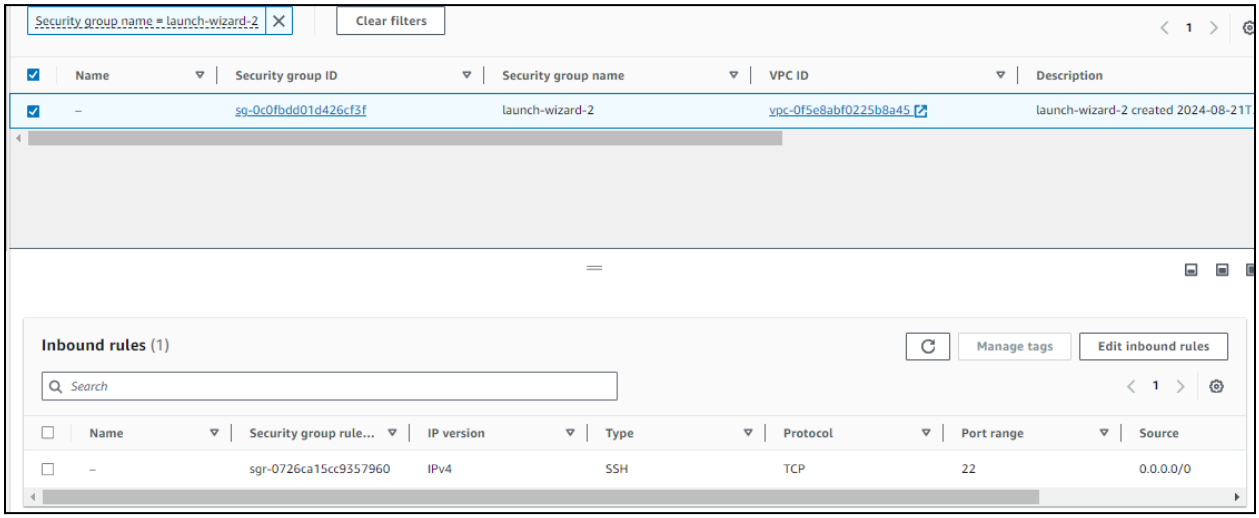
```
Saving to: 'OnlinePrintingServices.git'

OnlinePrintingServices.git                [ <=>                                                                ] 277.94K  --.-KB/s    in 0.007s

2024-08-21 20:32:35 (39.4 MB/s) - 'OnlinePrintingServices.git' saved [284608]

[root@ip-172-31-51-115 aws_assg3]# ls -lrt
total 280
-rw-r--r--. 1 root root 284608 Aug 21 20:32 OnlinePrintingServices.git
[root@ip-172-31-51-115 aws_assg3]# wget https://github.com/Rohan-Lalchandani08/OnlinePrintingServices/archive/refs/heads/main.zip
--2024-08-21 20:35:21--  https://github.com/Rohan-Lalchandani08/OnlinePrintingServices/archive/refs/heads/main.zip
Resolving github.com (github.com)... 140.82.113.3
Connecting to github.com (github.com)|140.82.113.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/Rohan-Lalchandani08/OnlinePrintingServices/zip/refs/heads/main [following]
--2024-08-21 20:35:21--  https://codeload.github.com/Rohan-Lalchandani08/OnlinePrintingServices/zip/refs/heads/main
Resolving codeload.github.com (codeload.github.com)... 140.82.113.9
Connecting to codeload.github.com (codeload.github.com)|140.82.113.9|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/zip]
Saving to: 'main.zip'

main.zip                                  [ <=>                                                                ] 450.42K  --.-KB/s    in 0.02s

2024-08-21 20:35:21 (28.4 MB/s) - 'main.zip' saved [461233]

[root@ip-172-31-51-115 aws_assg3]# ls -lrt
total 732
-rw-r--r--. 1 root root 284608 Aug 21 20:32 OnlinePrintingServices.git
-rw-r--r--. 1 root root 461233 Aug 21 20:35 main.zip
```

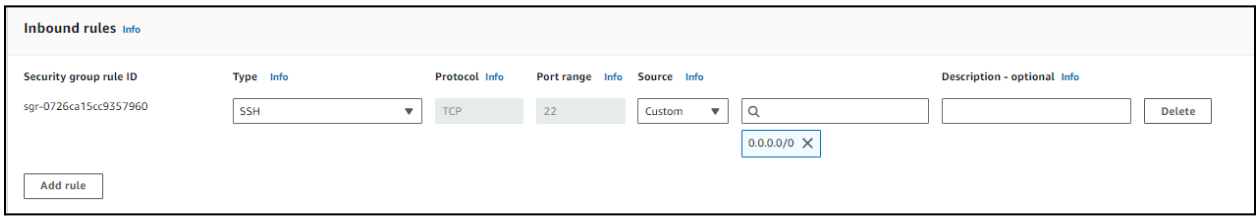
```
-rw-r--r--. 1 root root 284608 Aug 21 20:32 OnlinePrintingServices.git
-rw-r--r--. 1 root root 461233 Aug 21 20:35 main.zip
[root@ip-172-31-51-115 aws_assg3]# unzip main.zip
Archive:  main.zip
e89afae28997f8992f8d48da34c717b34da37c24
   creating: OnlinePrintingServices-main/
 extracting: OnlinePrintingServices-main/about.html
  inflating: OnlinePrintingServices-main/audio.mp3
 extracting: OnlinePrintingServices-main/banner.png
  inflating: OnlinePrintingServices-main/brochure.jpg
 extracting: OnlinePrintingServices-main/buisnesscard.png
  inflating: OnlinePrintingServices-main/contact.html
 extracting: OnlinePrintingServices-main/facebook.png
 extracting: OnlinePrintingServices-main/flyers.png
  inflating: OnlinePrintingServices-main/index.html
 extracting: OnlinePrintingServices-main/insta.png
 extracting: OnlinePrintingServices-main/poster.png
  inflating: OnlinePrintingServices-main/printer logo.png
  inflating: OnlinePrintingServices-main/x.png
[root@ip-172-31-51-115 aws_assg3]# ls -lrt
total 748
drwxr-xr-x. 2 root root  16384 Aug 20 04:47 OnlinePrintingServices-main
-rw-r--r--. 1 root root 284608 Aug 21 20:32 OnlinePrintingServices.git
-rw-r--r--. 1 root root 461233 Aug 21 20:35 main.zip
[root@ip-172-31-51-115 aws_assg3]# cd OnlinePrintingServices-main
[root@ip-172-31-51-115 OnlinePrintingServices-main]# ls -lrt
total 480
-rw-r--r--. 1 root root  14913 Aug 20 04:47  x.png
-rw-r--r--. 1 root root  17827 Aug 20 04:47 'printer logo.png'
-rw-r--r--. 1 root root  68827 Aug 20 04:47  poster.png
```

```
-rw-r--r--. 1 root root  17827 Aug 20 04:47 'printer logo.png'
-rw-r--r--. 1 root root  68827 Aug 20 04:47  poster.png
-rw-r--r--. 1 root root  28213 Aug 20 04:47  insta.png
-rw-r--r--. 1 root root   2821 Aug 20 04:47  index.html
-rw-r--r--. 1 root root 102141 Aug 20 04:47  flyers.png
-rw-r--r--. 1 root root  11110 Aug 20 04:47  facebook.png
-rw-r--r--. 1 root root    936 Aug 20 04:47  contact.html
-rw-r--r--. 1 root root  34395 Aug 20 04:47  buisnesscard.png
-rw-r--r--. 1 root root  86675 Aug 20 04:47  brochure.jpg
-rw-r--r--. 1 root root  69398 Aug 20 04:47  banner.png
-rw-r--r--. 1 root root  29997 Aug 20 04:47  audio.mp3
-rw-r--r--. 1 root root      2 Aug 20 04:47  about.html
[root@ip-172-31-51-115 OnlinePrintingServices-main]# mv * /var/www/html/
[root@ip-172-31-51-115 OnlinePrintingServices-main]# cd /var/www/html/
[root@ip-172-31-51-115 html]# ls -lrt
total 480
-rw-r--r--. 1 root root  14913 Aug 20 04:47  x.png
-rw-r--r--. 1 root root  17827 Aug 20 04:47 'printer logo.png'
-rw-r--r--. 1 root root  68827 Aug 20 04:47  poster.png
-rw-r--r--. 1 root root  28213 Aug 20 04:47  insta.png
-rw-r--r--. 1 root root   2821 Aug 20 04:47  index.html
-rw-r--r--. 1 root root 102141 Aug 20 04:47  flyers.png
-rw-r--r--. 1 root root  11110 Aug 20 04:47  facebook.png
-rw-r--r--. 1 root root    936 Aug 20 04:47  contact.html
-rw-r--r--. 1 root root  34395 Aug 20 04:47  buisnesscard.png
-rw-r--r--. 1 root root  86675 Aug 20 04:47  brochure.jpg
-rw-r--r--. 1 root root  69398 Aug 20 04:47  banner.png
-rw-r--r--. 1 root root  29997 Aug 20 04:47  audio.mp3
-rw-r--r--. 1 root root      2 Aug 20 04:47  about.html
[root@ip-172-31-51-115 html]#
```

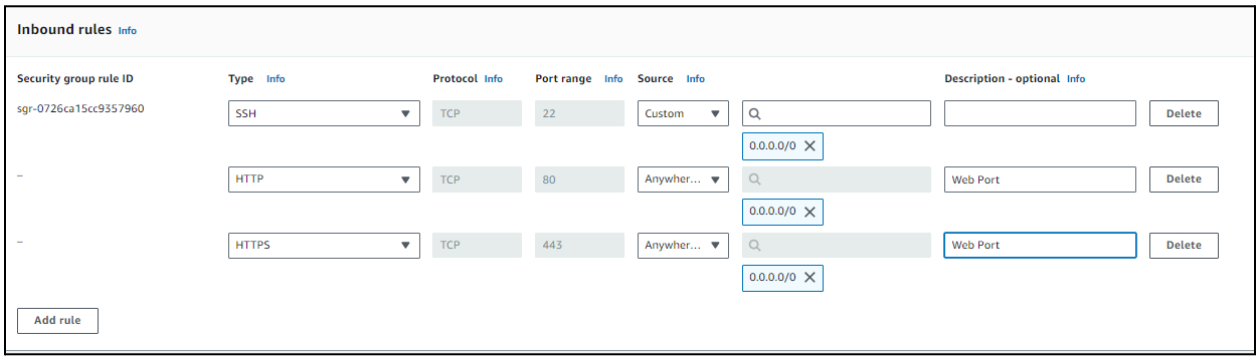
3) Editing the Inbound Rules:

Click on “Edit Inbound Rules” Button:





Now, Add new Rules as follows:



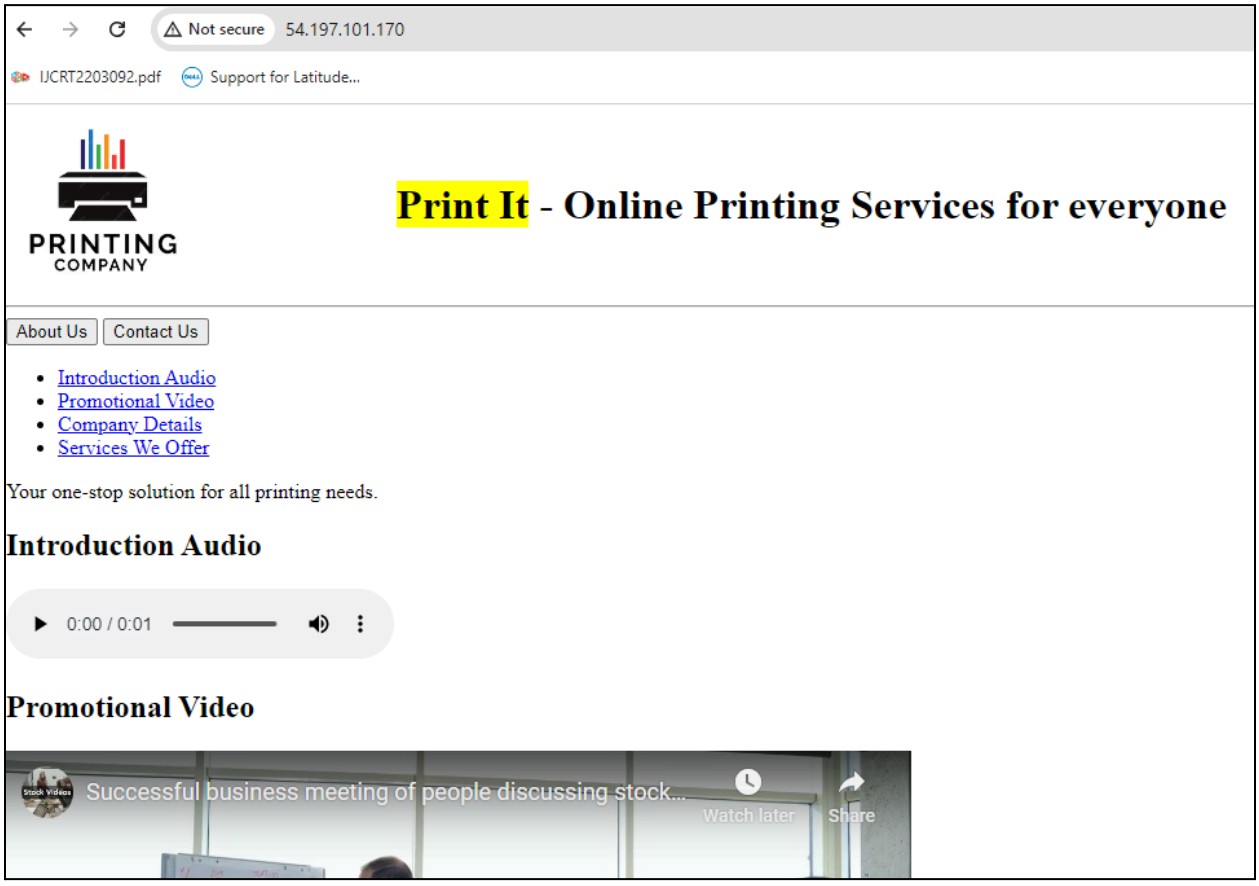
4) Check the status of httpd and then enable & start httpd using the following commands:

```
systemctl status httpd
systemctl enable httpd
systemctl start httpd
```

Now open the public ipv4 address allocated to the EC2 instance we created in new tab. We will be able to see the Website.

```
[root@ip-172-31-51-115 html]# systemctl status httpd
o httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:httpd.service(8)
[root@ip-172-31-51-115 html]# systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[root@ip-172-31-51-115 html]# start httpd
-bash: start: command not found
[root@ip-172-31-51-115 html]# systemctl start httpd
[root@ip-172-31-51-115 html]#
```

```
[root@ip-172-31-51-115 html]# systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
     Active: active (running) since Wed 2024-08-21 20:59:52 UTC; 3s ago
       Docs: man:httpd.service(8)
   Main PID: 28176 (httpd)
     Status: "Started, listening on: port 80"
      Tasks: 177 (limit: 1112)
     Memory: 12.9M
        CPU: 63ms
     CGroup: /system.slice/httpd.service
             ├─28176 /usr/sbin/httpd -DFOREGROUND
             ├─28177 /usr/sbin/httpd -DFOREGROUND
             ├─28178 /usr/sbin/httpd -DFOREGROUND
             ├─28179 /usr/sbin/httpd -DFOREGROUND
             └─28180 /usr/sbin/httpd -DFOREGROUND

Aug 21 20:59:52 ip-172-31-51-115.ec2.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Aug 21 20:59:52 ip-172-31-51-115.ec2.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Aug 21 20:59:52 ip-172-31-51-115.ec2.internal httpd[28176]: Server configured, listening on: port 80
```



Not secure 54.197.101.170

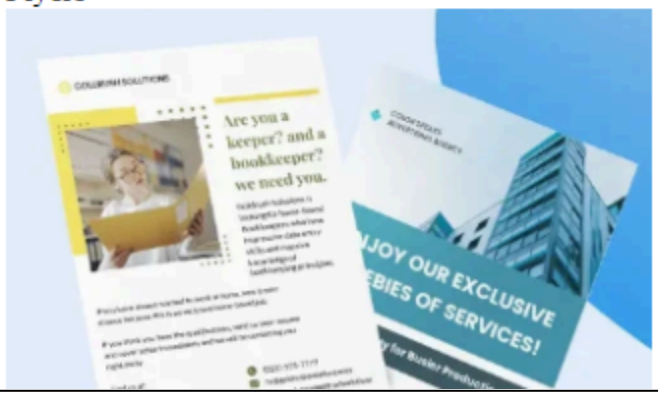
IJCRT2203092.pdf Support for Latitude...

Company Details

Services We Offer

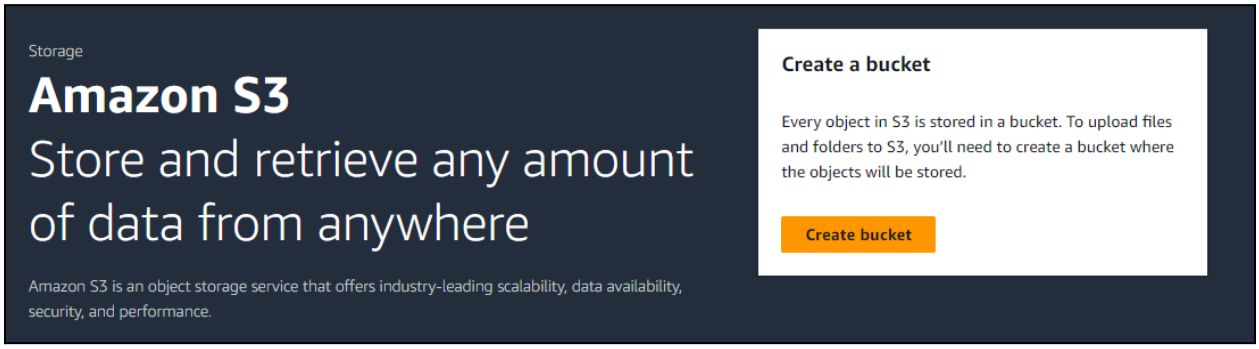
- Business Cards

- Flyers

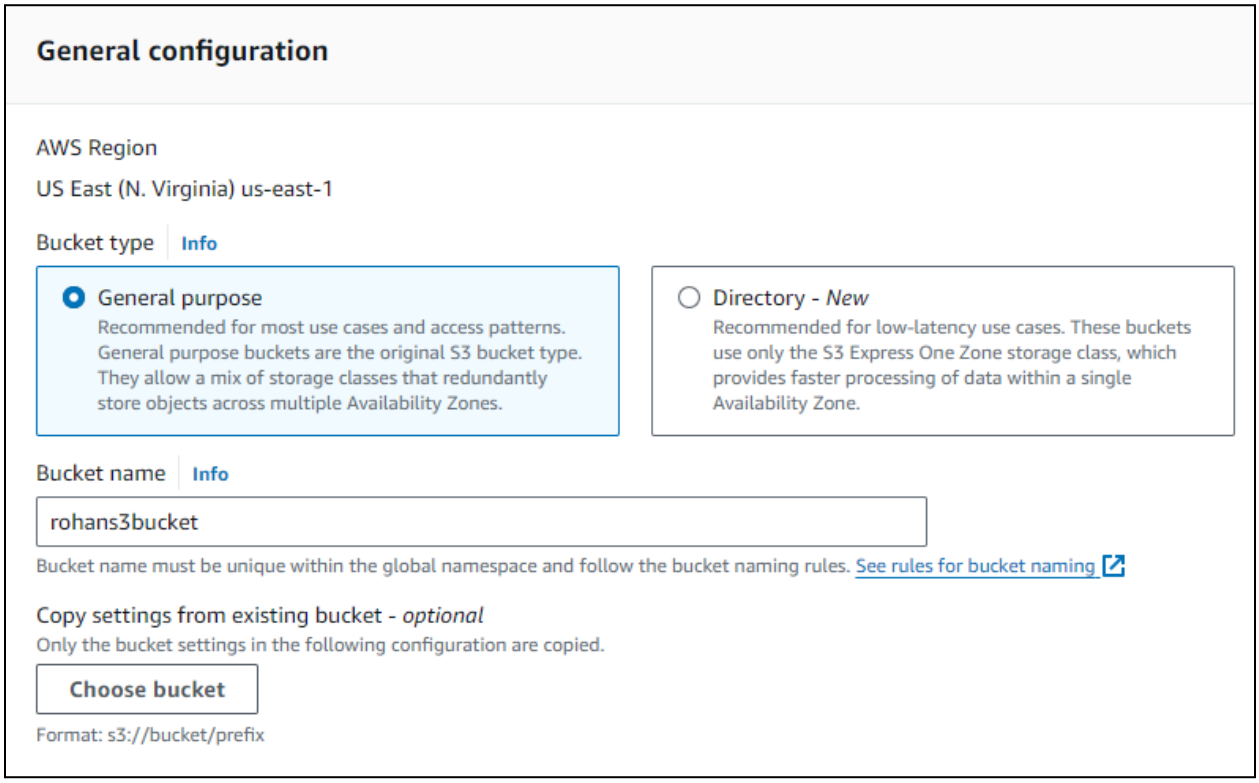


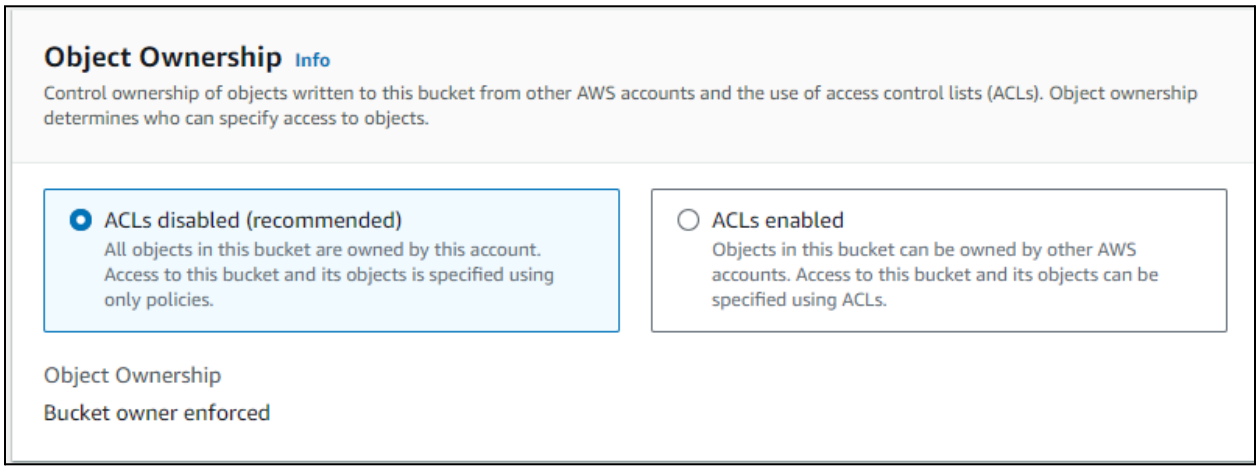
3) Using S3 Bucket:

1) Search for s3 bucket and go to the link and click on create bucket.



2) Configure the settings:





Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning

- ○ Disable
- ◉ Enable

Default encryption Info

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type Info

- ◉ Server-side encryption with Amazon S3 managed keys (SSE-S3)
- ○ Server-side encryption with AWS Key Management Service keys (SSE-KMS)
- ○ Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)

  Secure your objects with two separate layers of encryption. For details on pricing, see **DSSE-KMS pricing** on the **Storage** tab of the Amazon S3 pricing page.

Bucket Key

Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. Learn more

- ○ Disable
- ◉ Enable

General purpose buckets (1) Info All AWS Regions

Buckets are containers for data stored in S3.

Copy ARN | Empty | Delete | Create bucket

Find buckets by name

1

Name	AWS Region	IAM Access Analyzer	Creation date
rohans3bucket1	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 23, 2024, 21:32:27 (UTC+05:30)

3) Upload the files.

Files and folders (13 Total, 456.9 KB)

Remove | Add files | Add folder

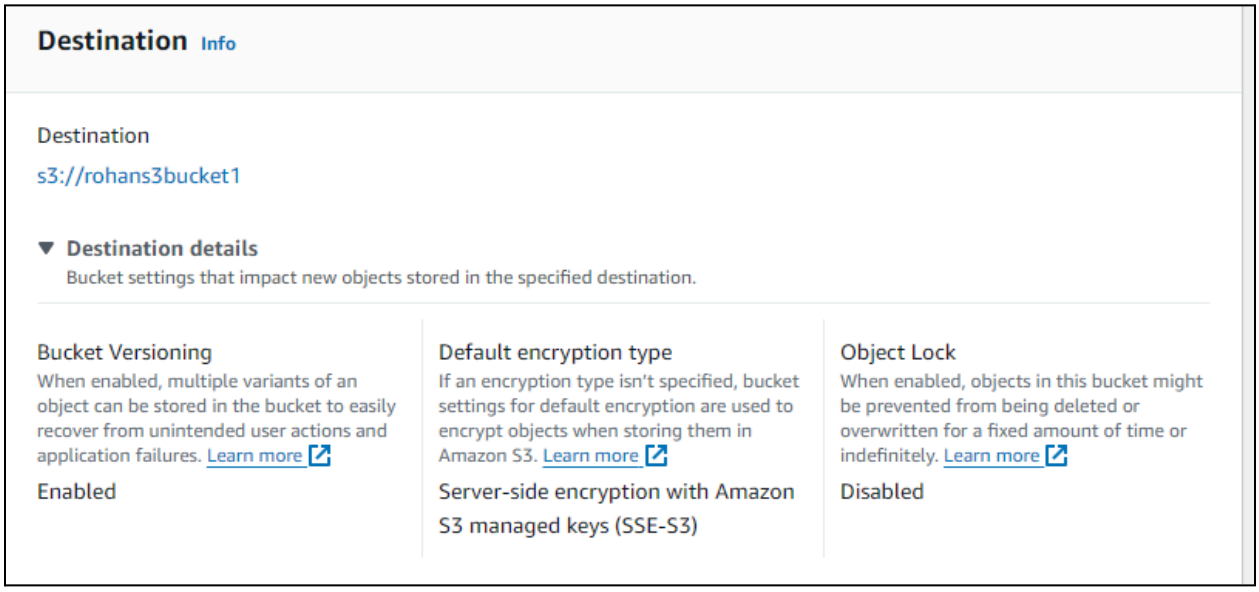
All files and folders in this table will be uploaded.

Find by name

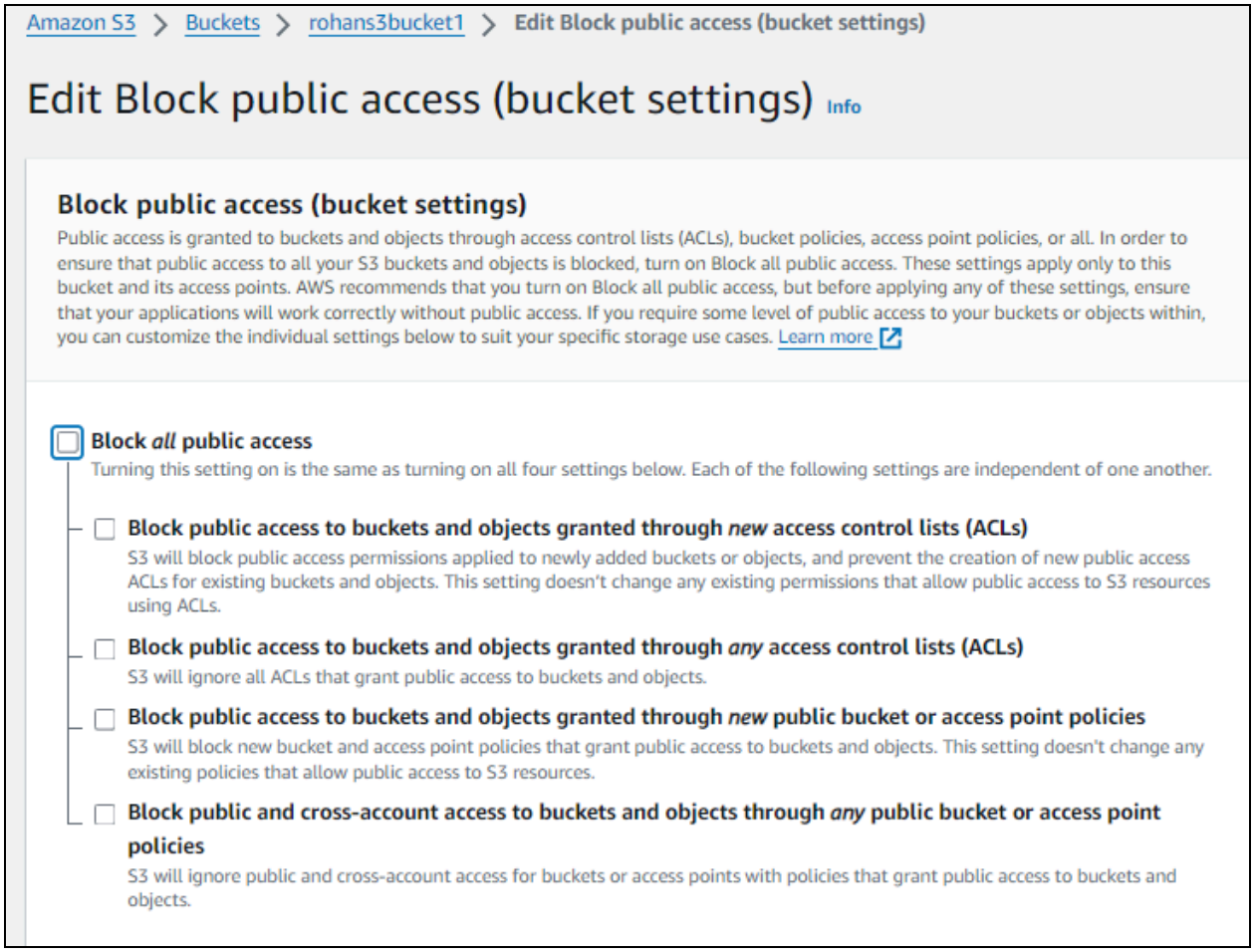
1 2

Name	Folder	Type
x.png	-	image/png
about.html	-	text/html
audio.mp3	-	audio/mpeg
banner.png	-	image/png
brochure.jpg	-	image/jpeg
buisnesscard.png	-	image/png
contact.html	-	text/html
facebook.png	-	image/png
flyers.png	-	image/png
index.html	-	text/html

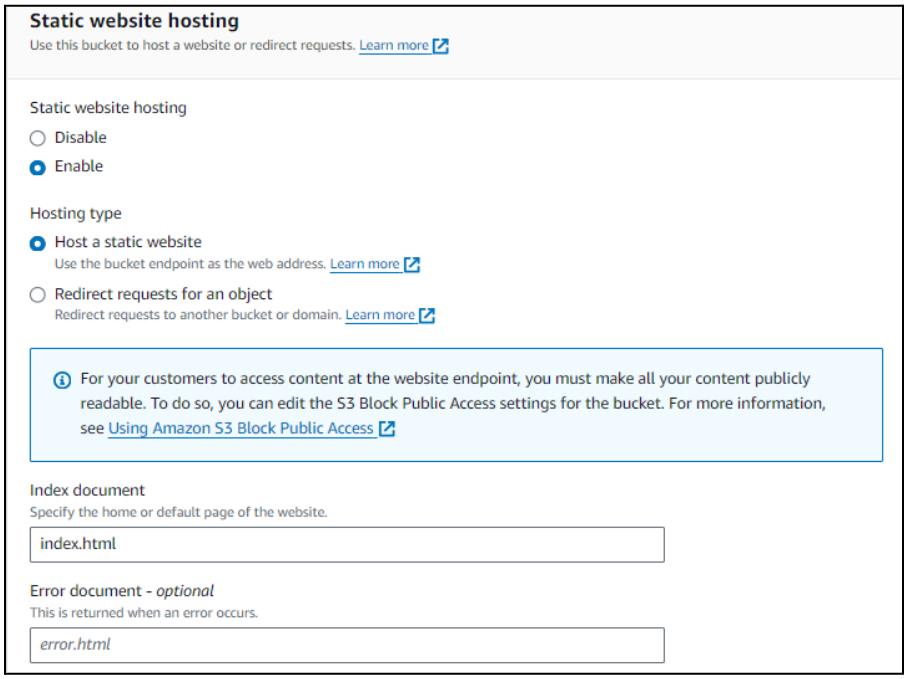
4) Go to your bucket destination link.



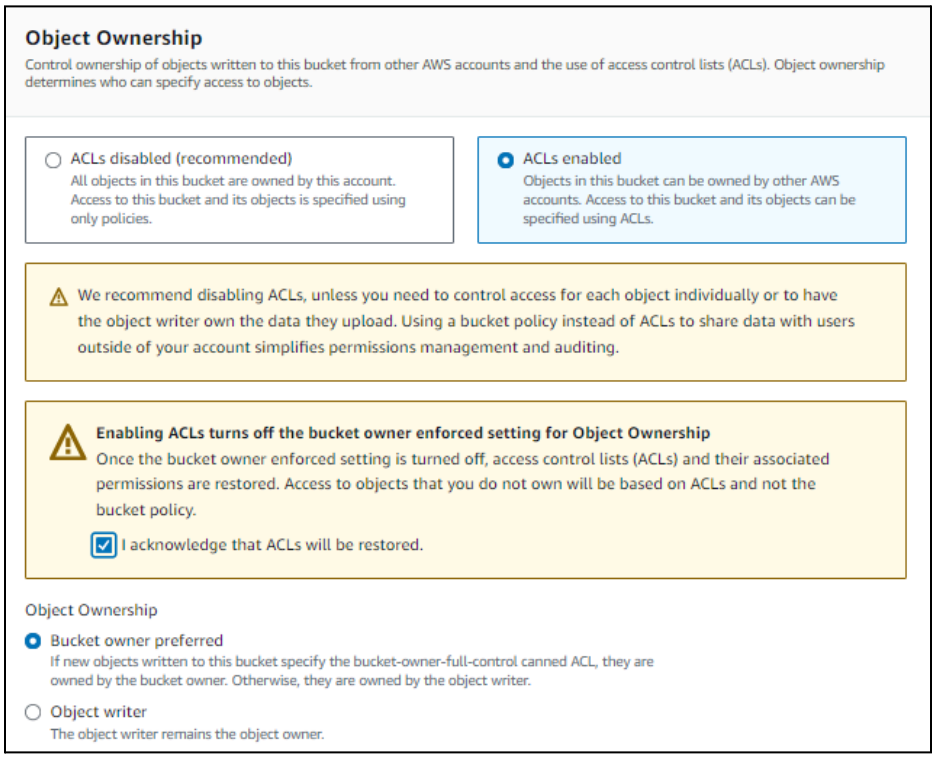
5) Uncheck the block public access.



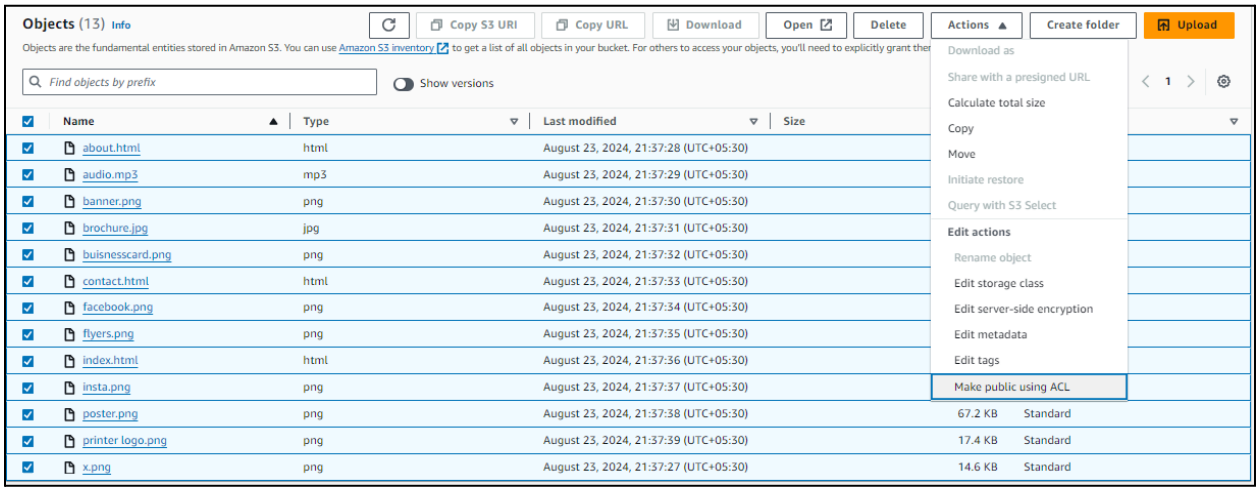
6) Go to the properties section and scroll down to static website hosting and enable it.



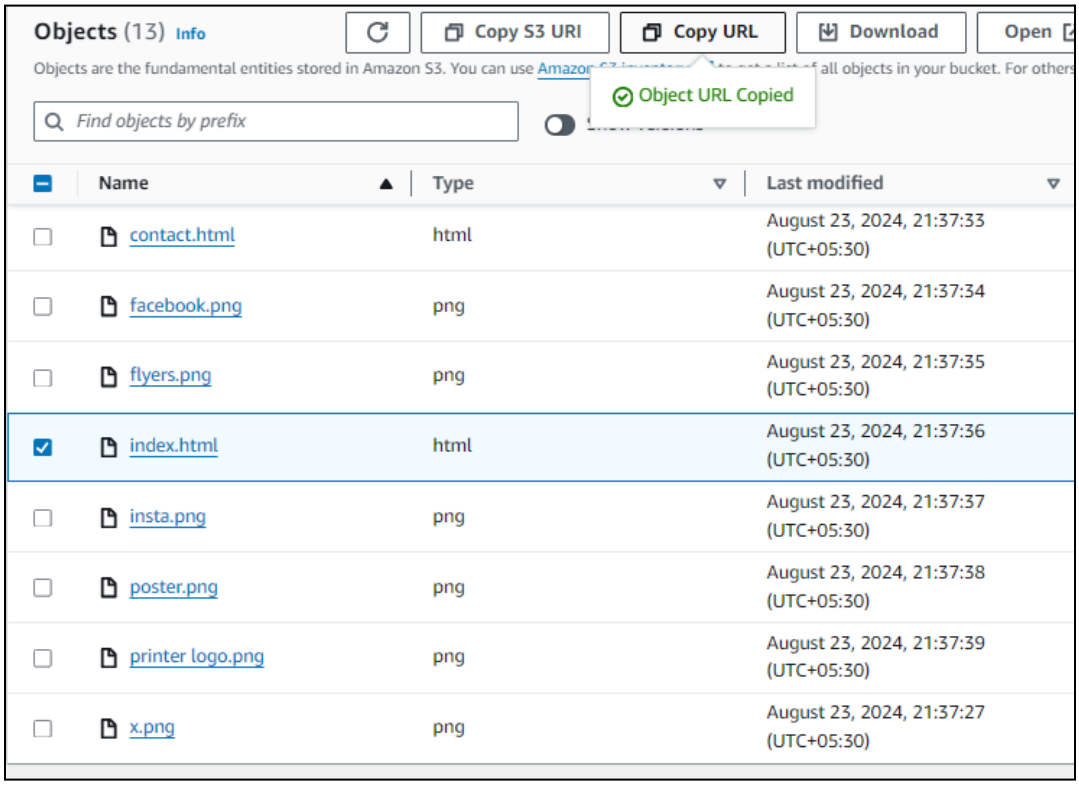
7) Go to the permissions section and see for ACL's option and enable it.



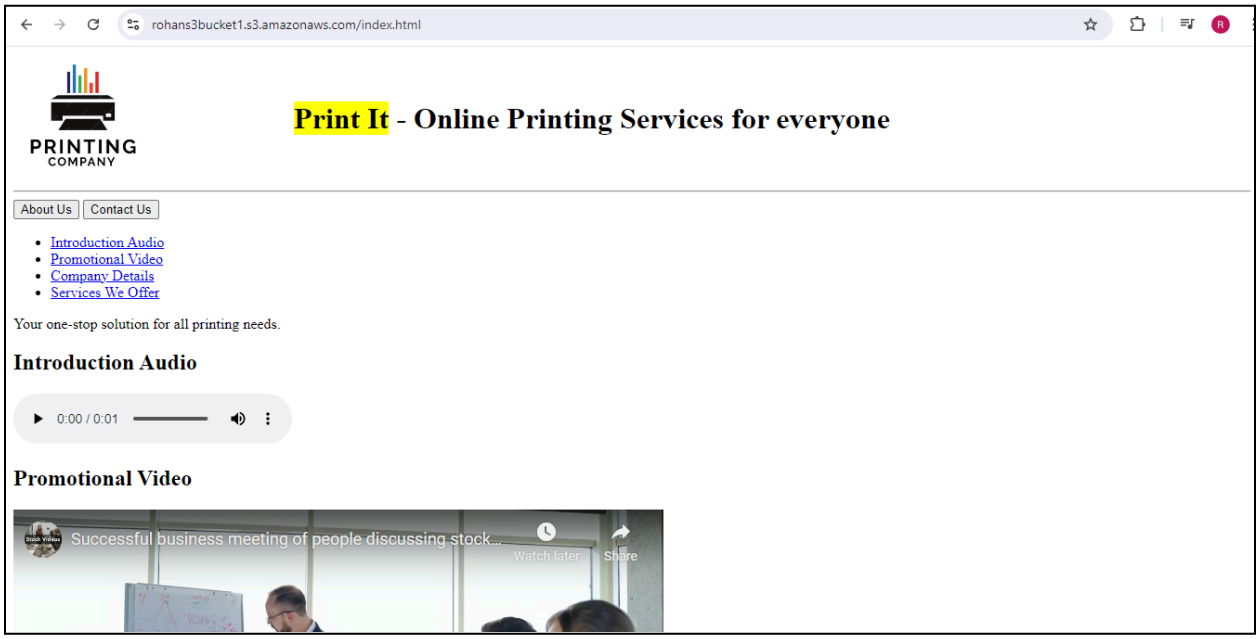
8) Now, go to the objects section and select all objects and click on action button and select “Make public using ACL”



9) Copy the URL of the index.html file and go to the copied link in the new tab of the browser.

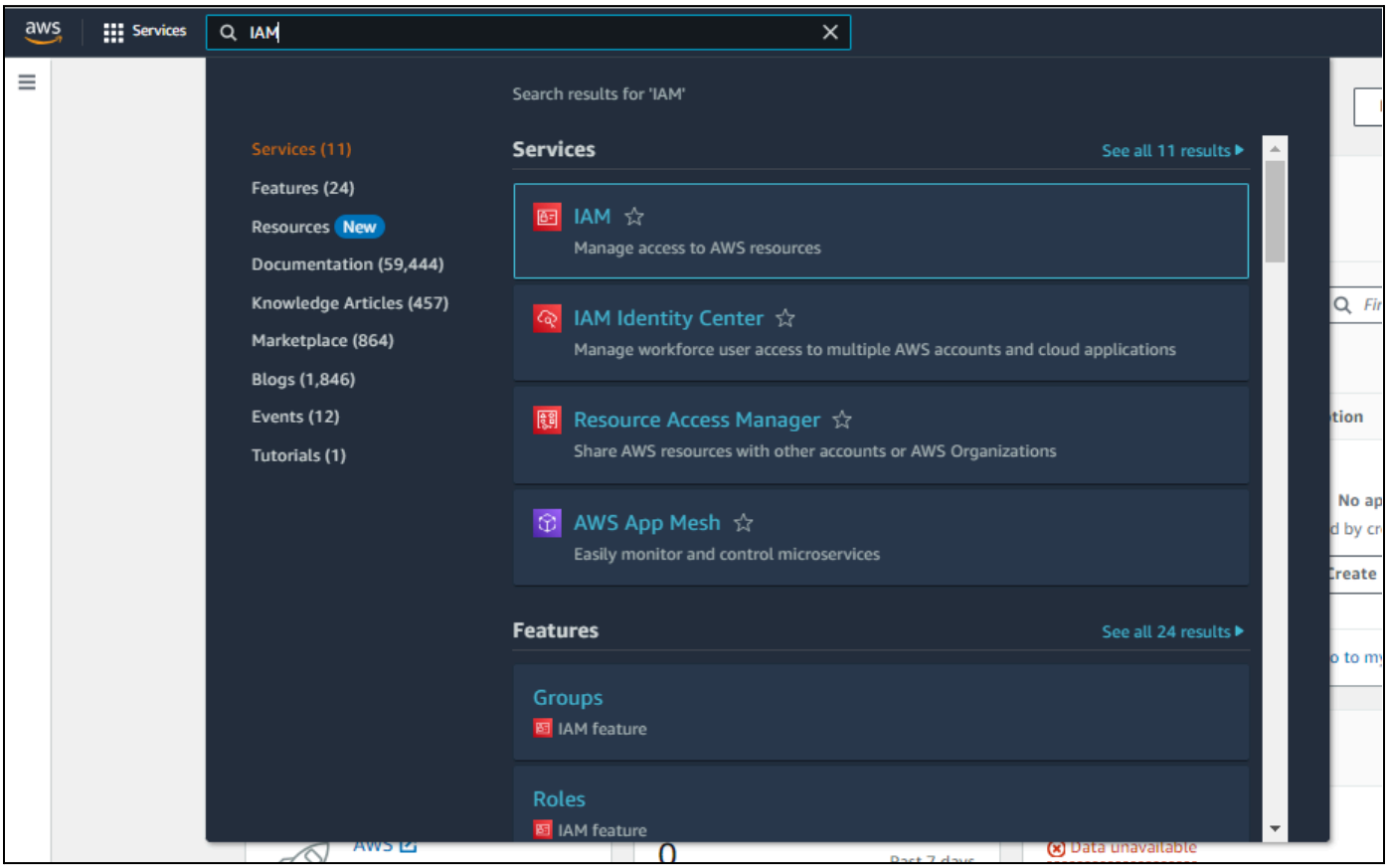


10) Website is successfully deployed.

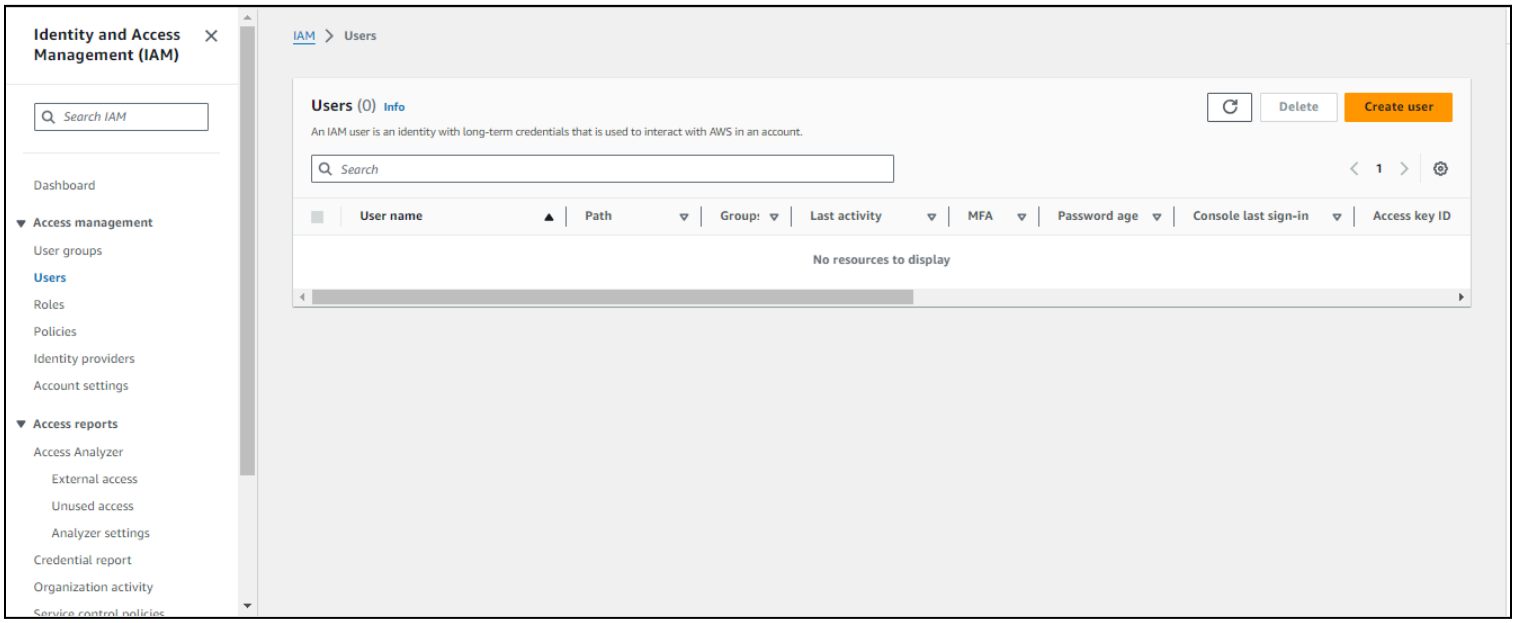


Part B) AWS Cloud9:

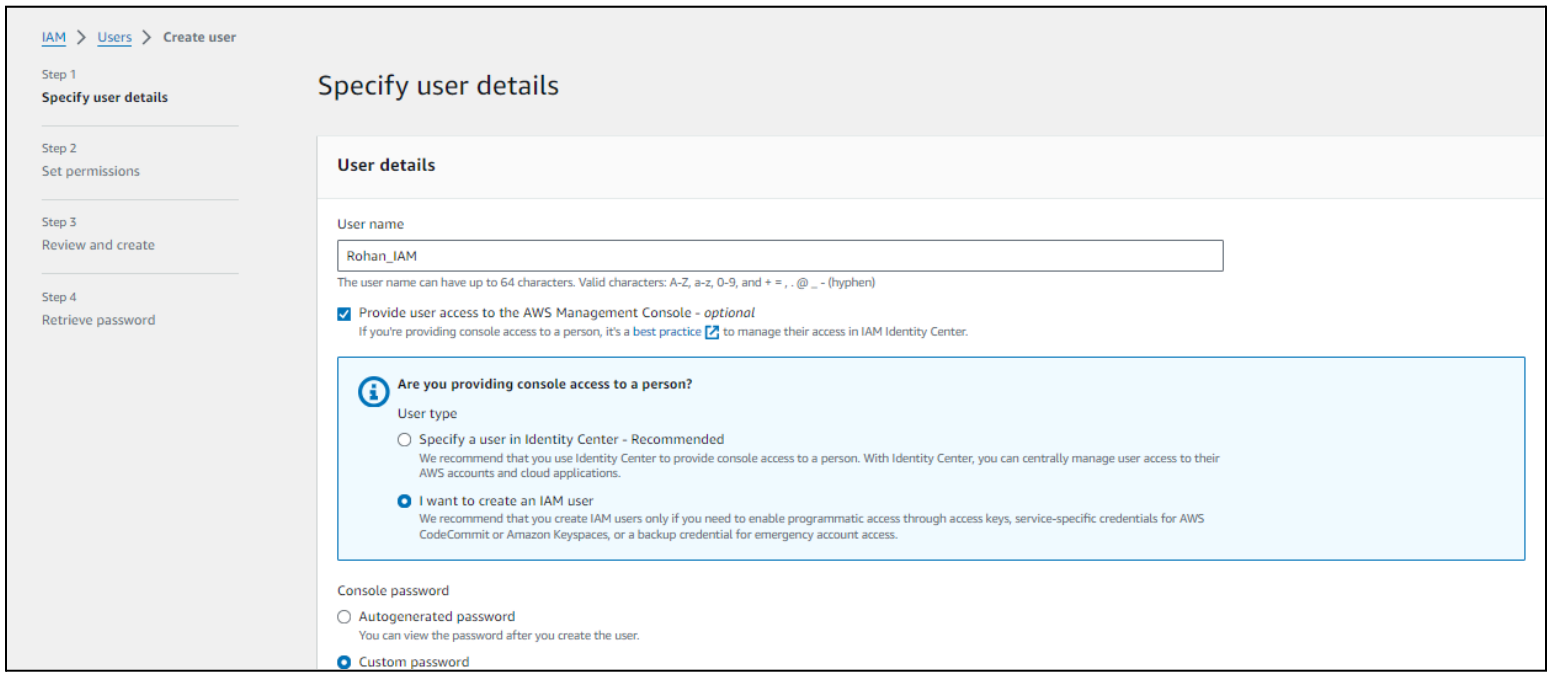
1) Search for IAM in the search box.

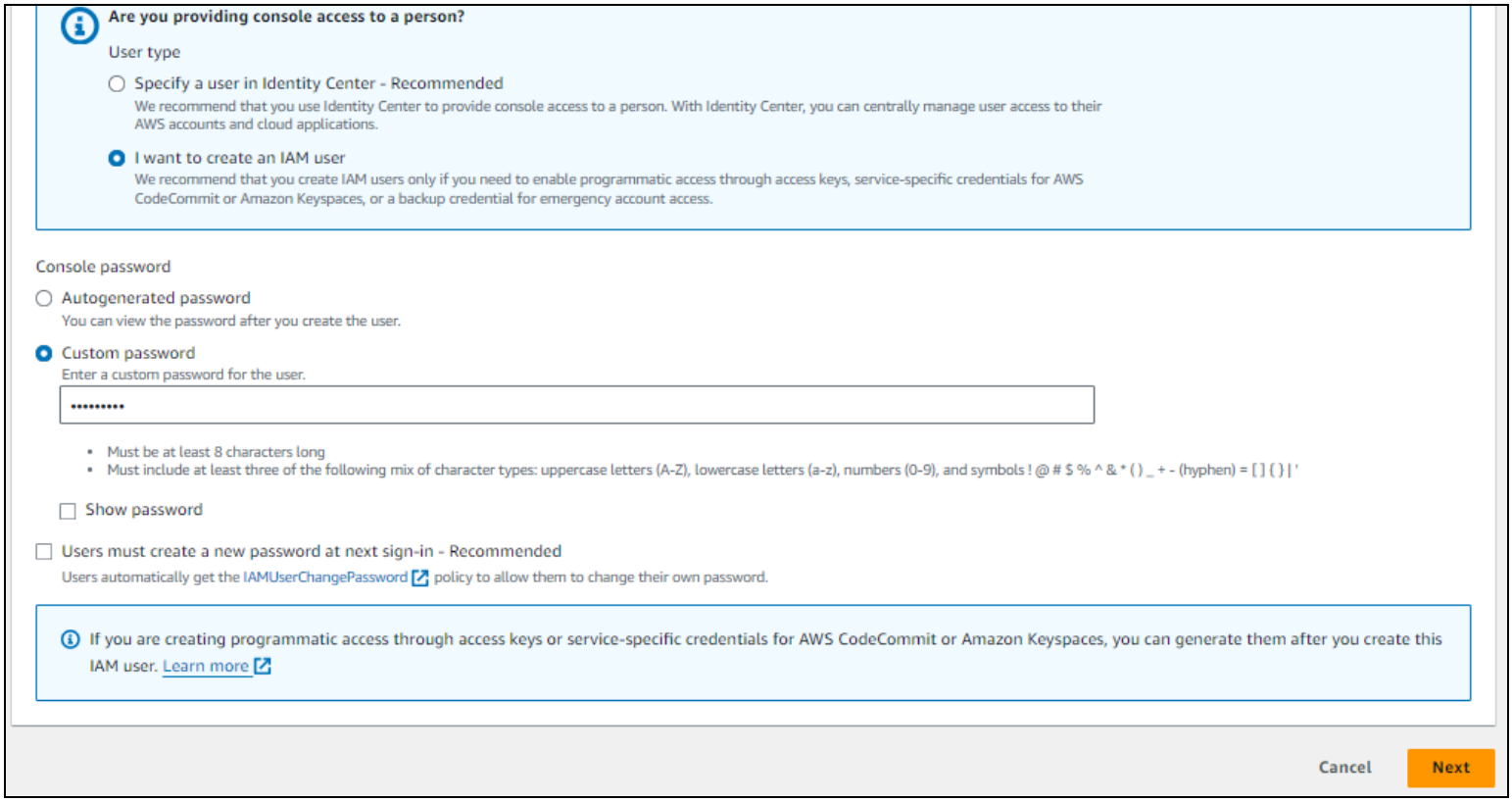


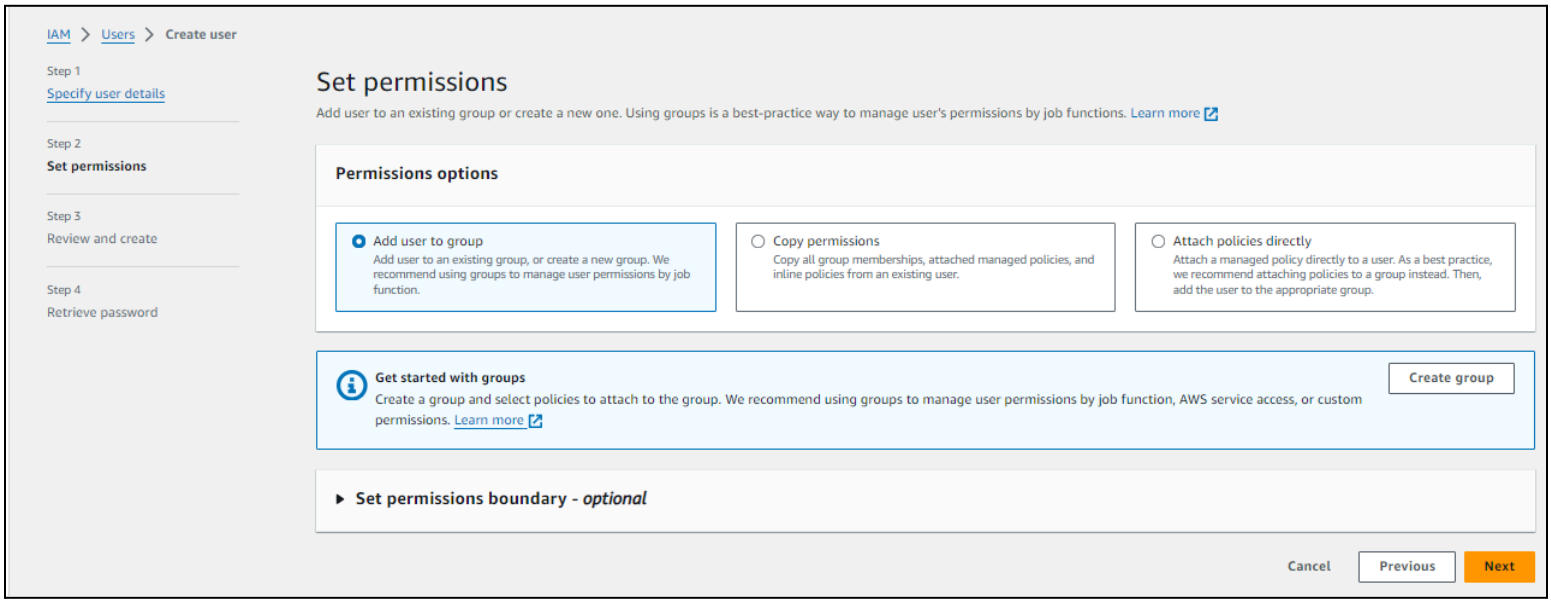
2) Click on create user.

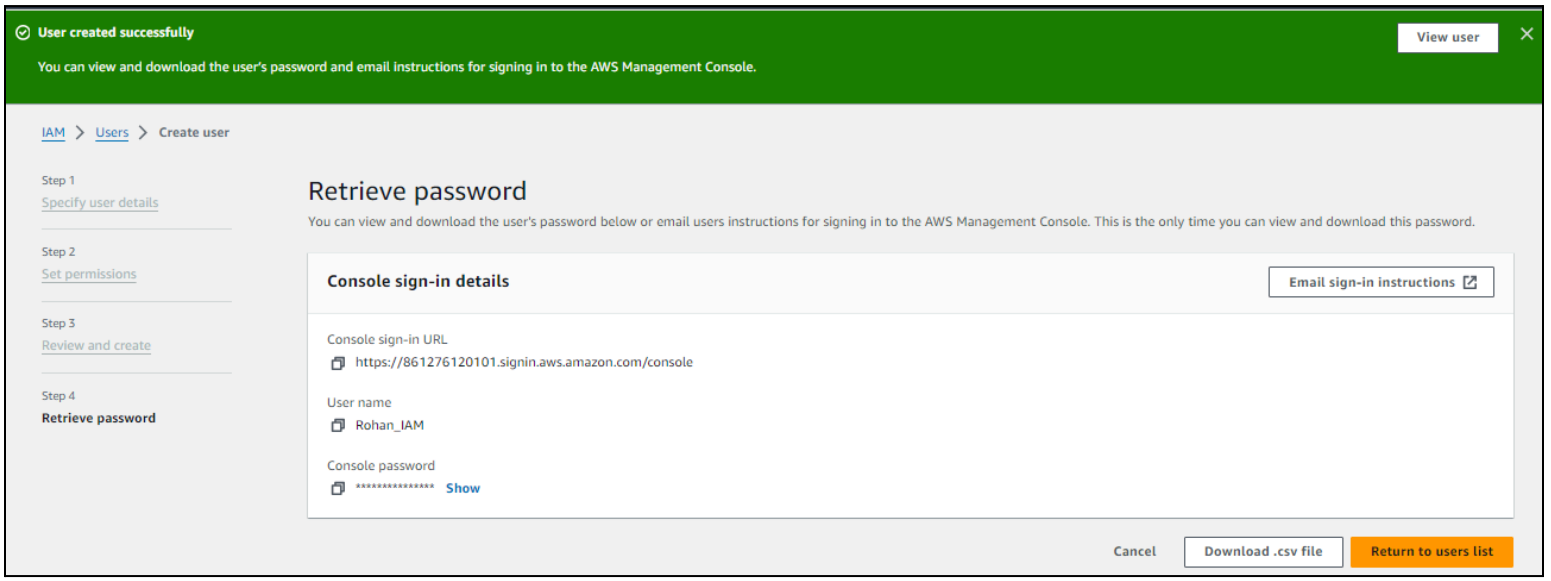


3) Configuring the IAM role.

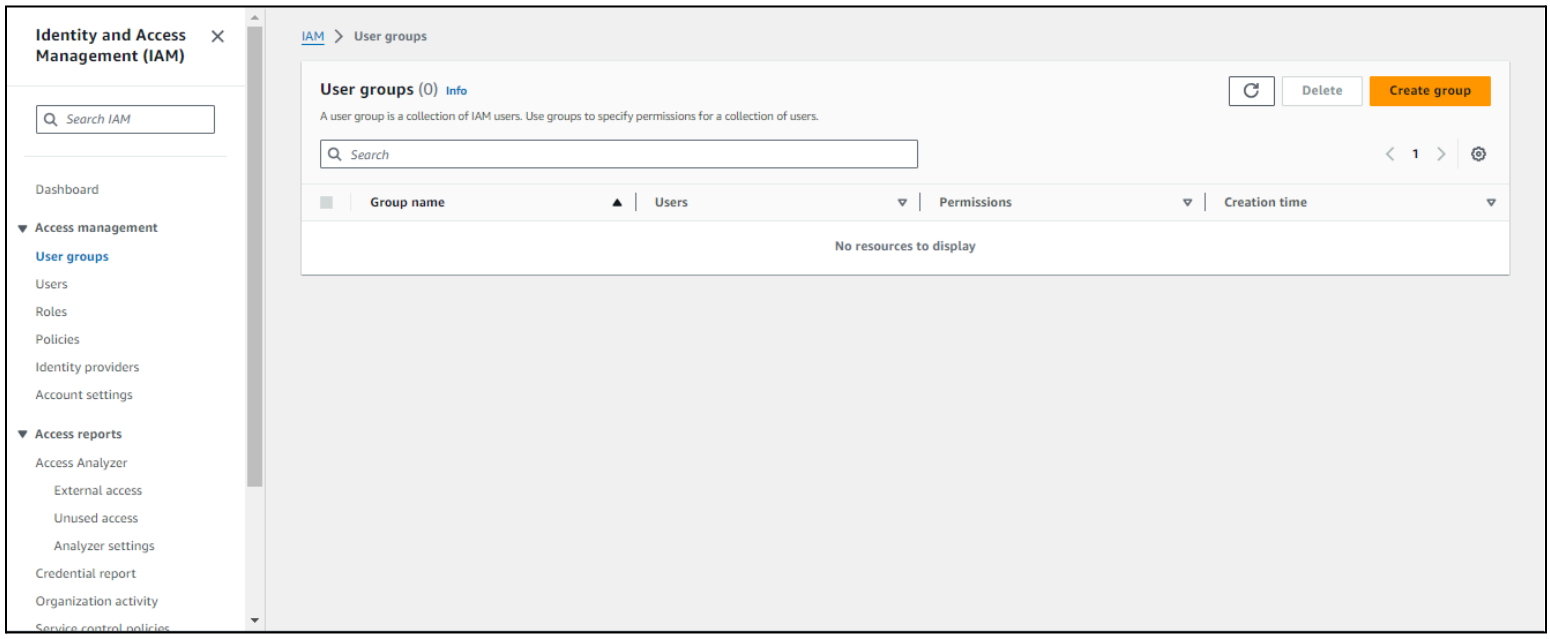




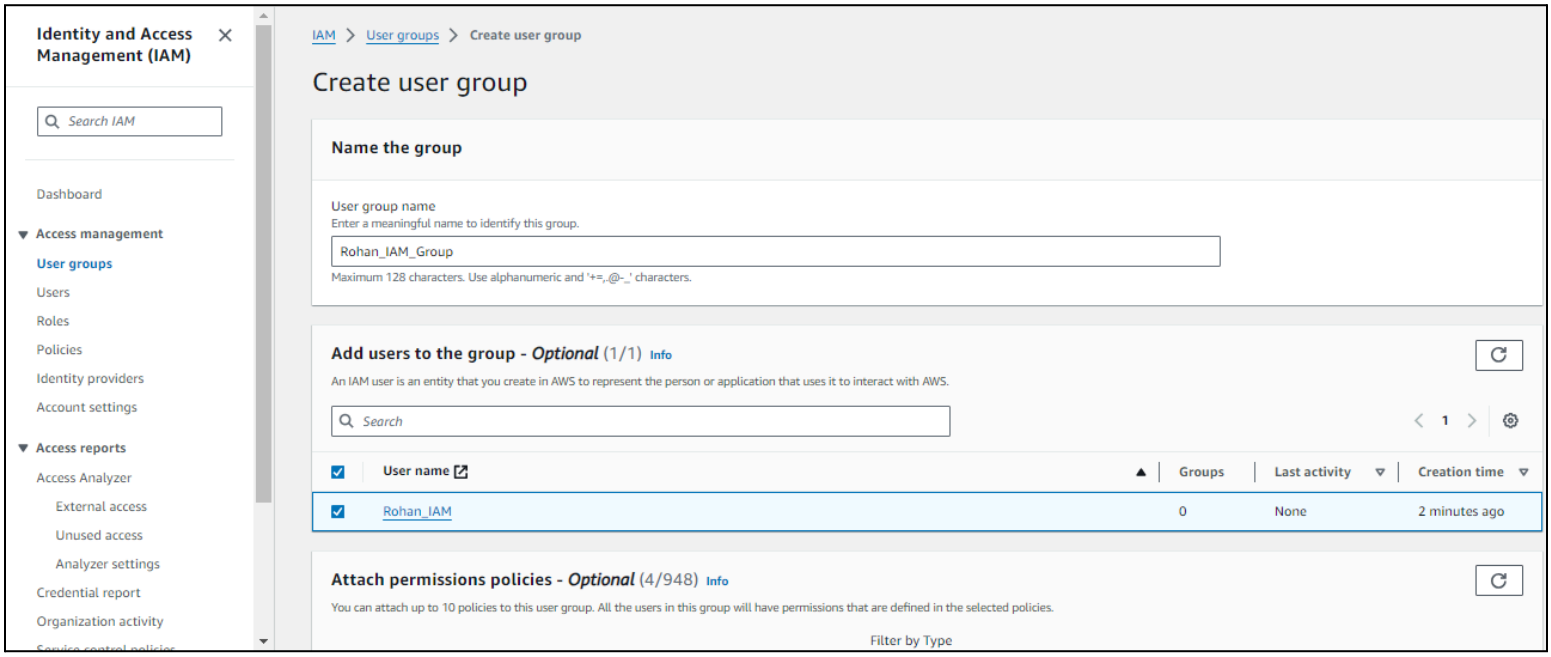

IAM > Users > Create user
Step 1
Specify user details
Step 2
Set permissions
Step 3
Review and create
Step 4
Retrieve password
Set permissions
Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more
Permissions options
Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.
Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.
Get started with groups
Create a group and select policies to attach to the group. We recommend using groups to manage user permissions by job function, AWS service access, or custom permissions. Learn more
Create group
Set permissions boundary - optional
Cancel
Previous
Next

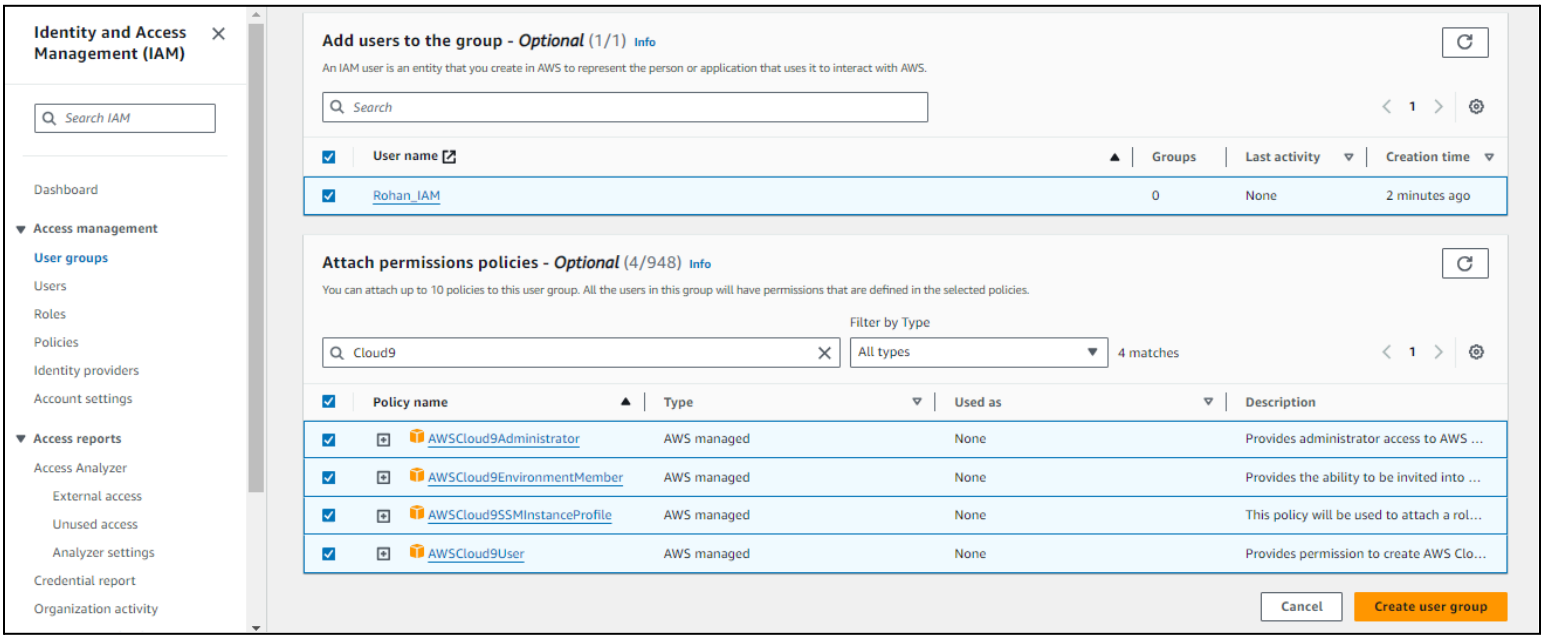

User created successfully
You can view and download the user's password and email instructions for signing in to the AWS Management Console.
View user
IAM > Users > Create user
Step 1
Specify user details
Step 2
Set permissions
Step 3
Review and create
Step 4
Retrieve password
Retrieve password
You can view and download the user's password below or email users instructions for signing in to the AWS Management Console. This is the only time you can view and download this password.
Console sign-in details
Email sign-in instructions
Console sign-in URL
https://861276120101.signin.aws.amazon.com/console
User name
Rohan_IAM
Console password
********** Show
Cancel
Download .csv file
Return to users list

4) After creating IAM role, go to User groups, and click on create group.

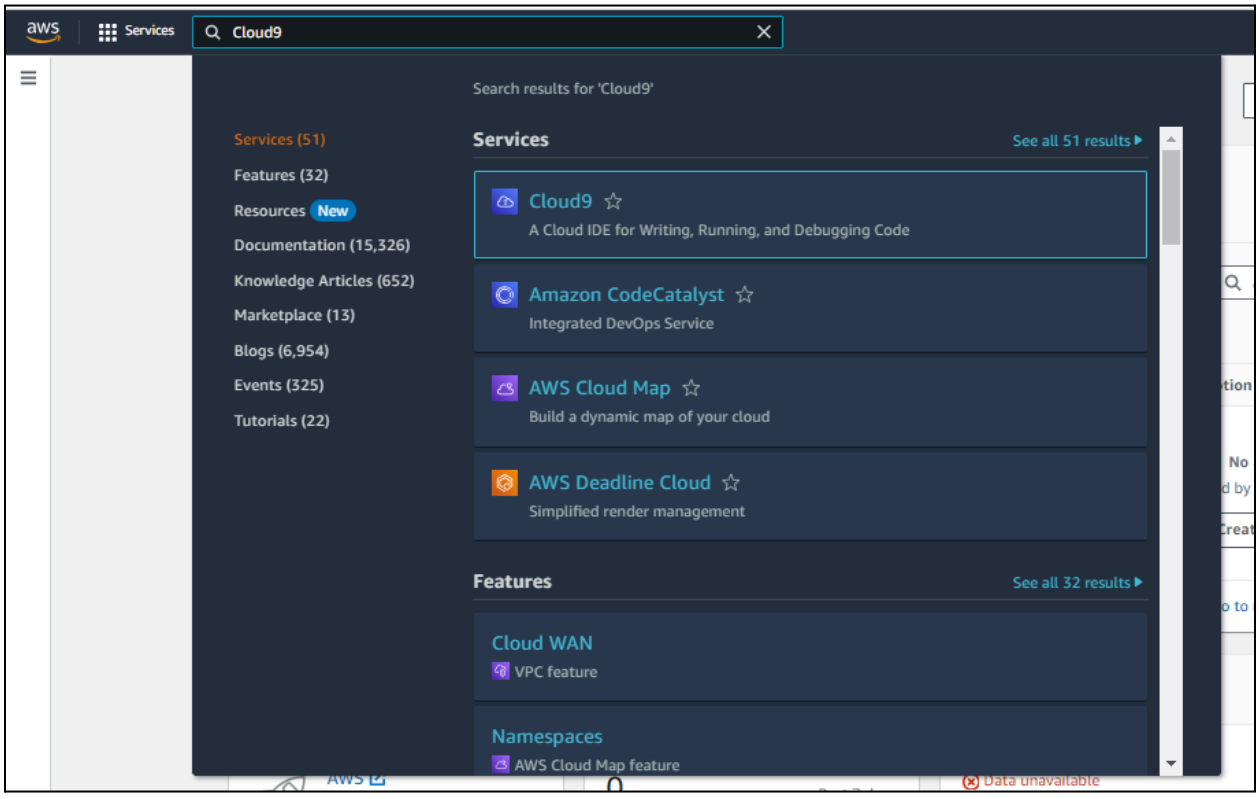


5) Configuring the user group.

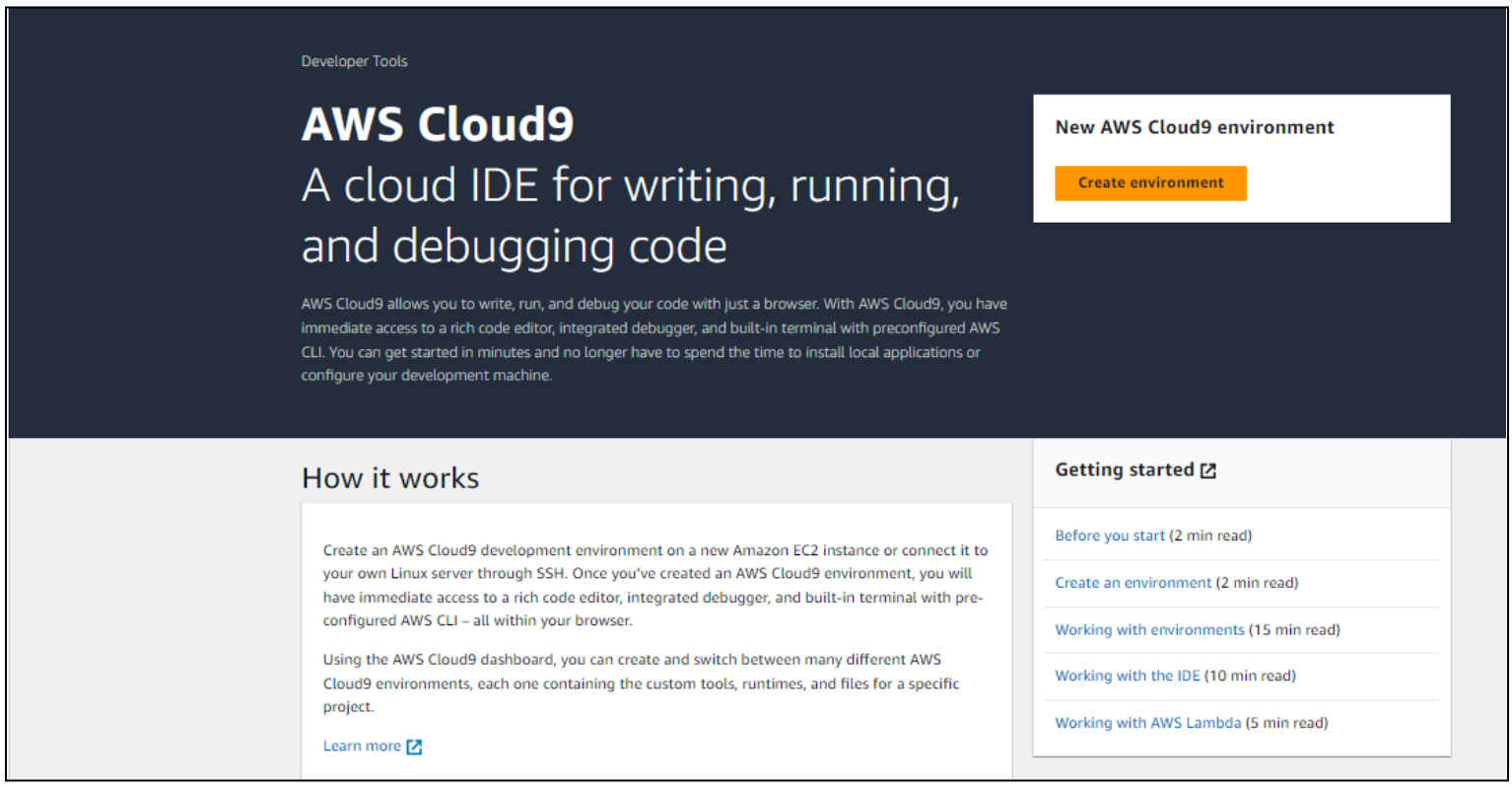




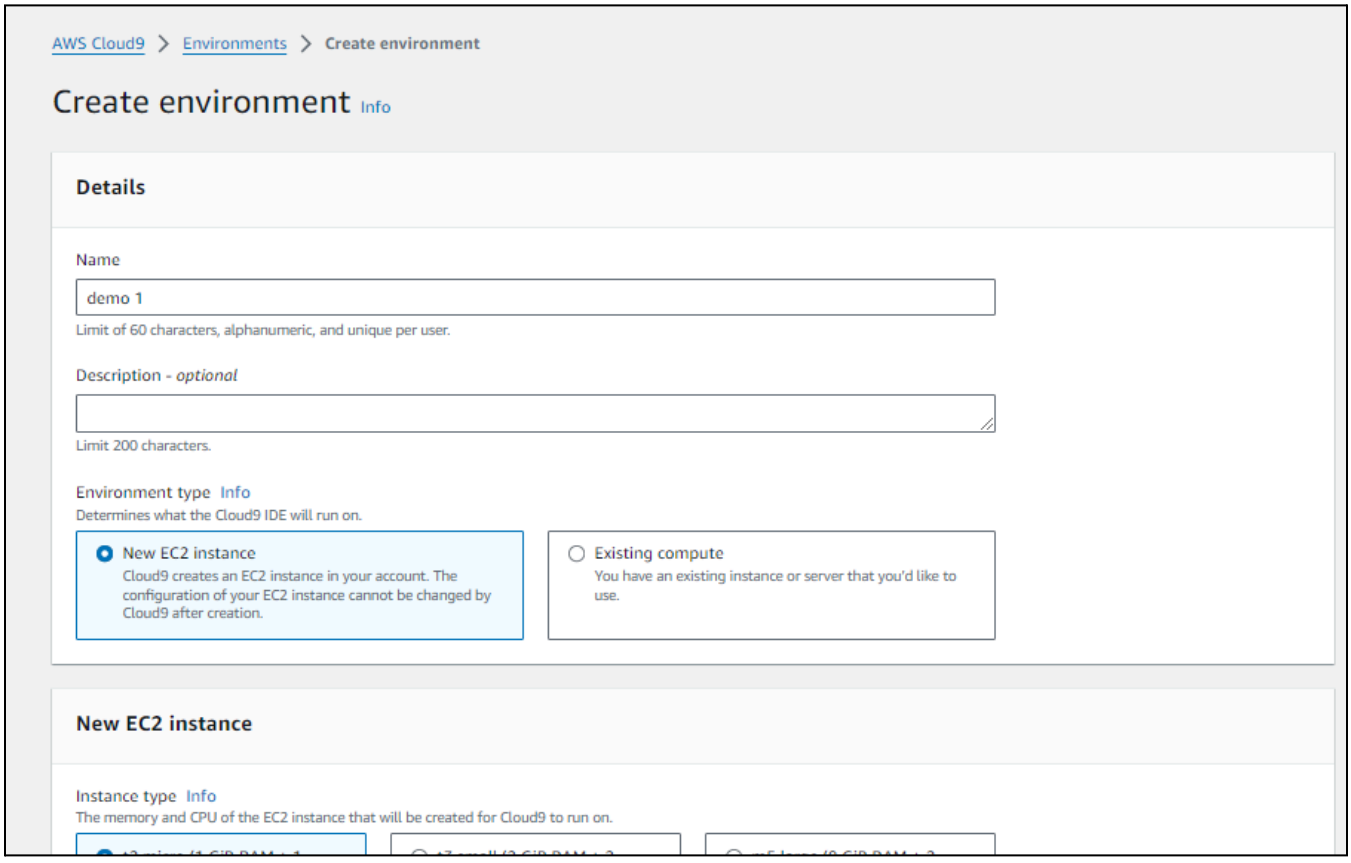
6) After creating a user group, search for Cloud9 in the search box.

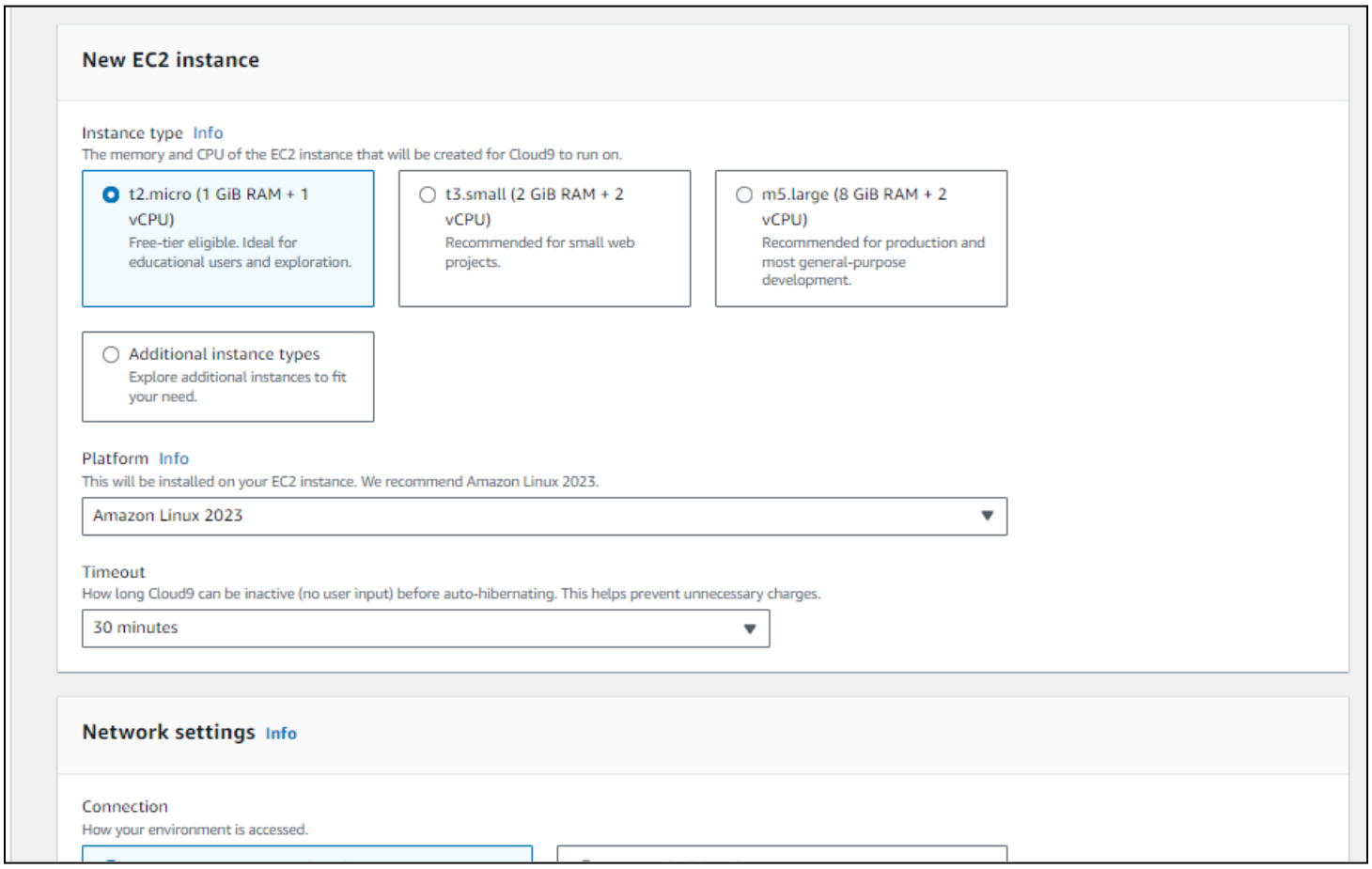


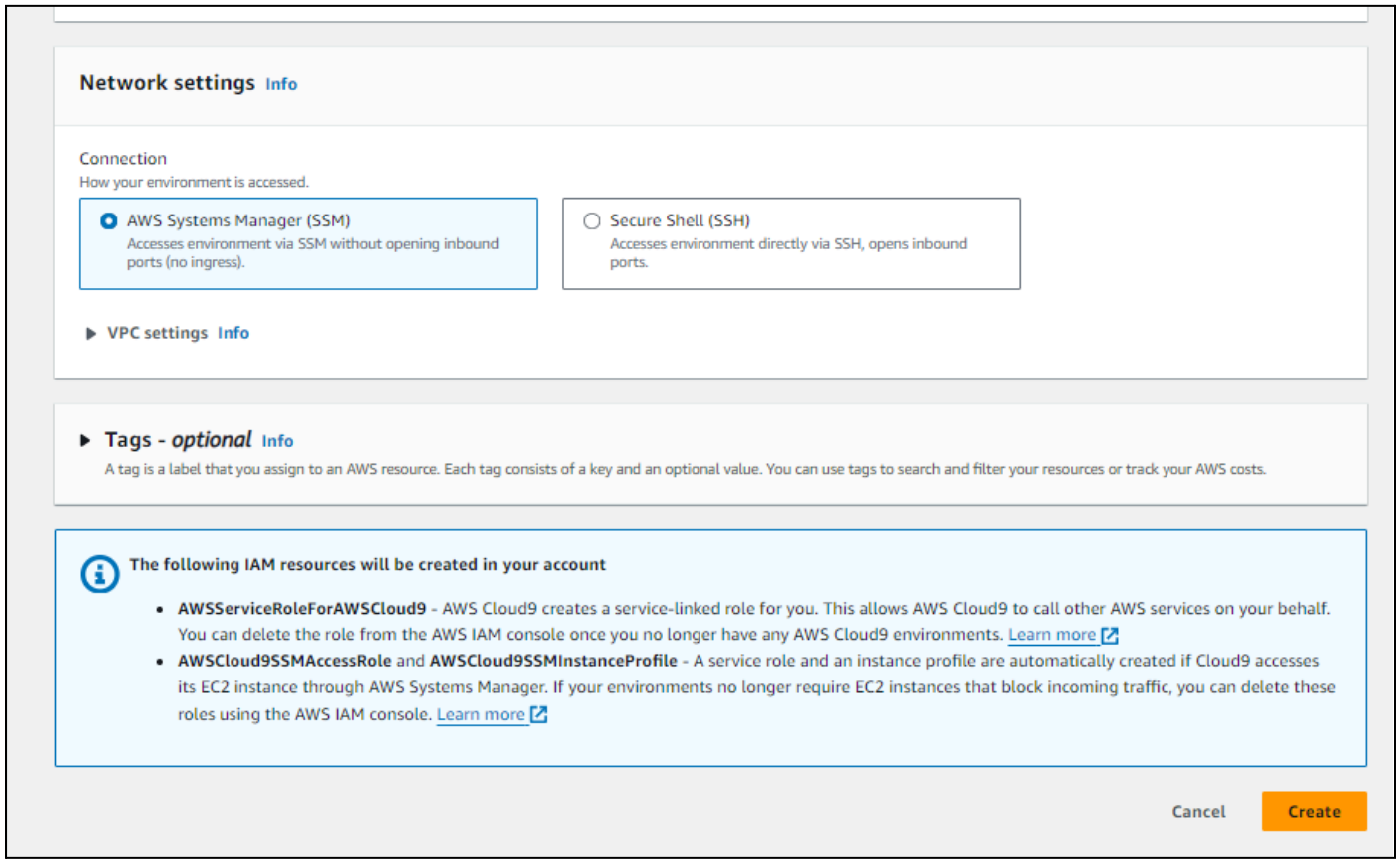
7) Click on create environment.



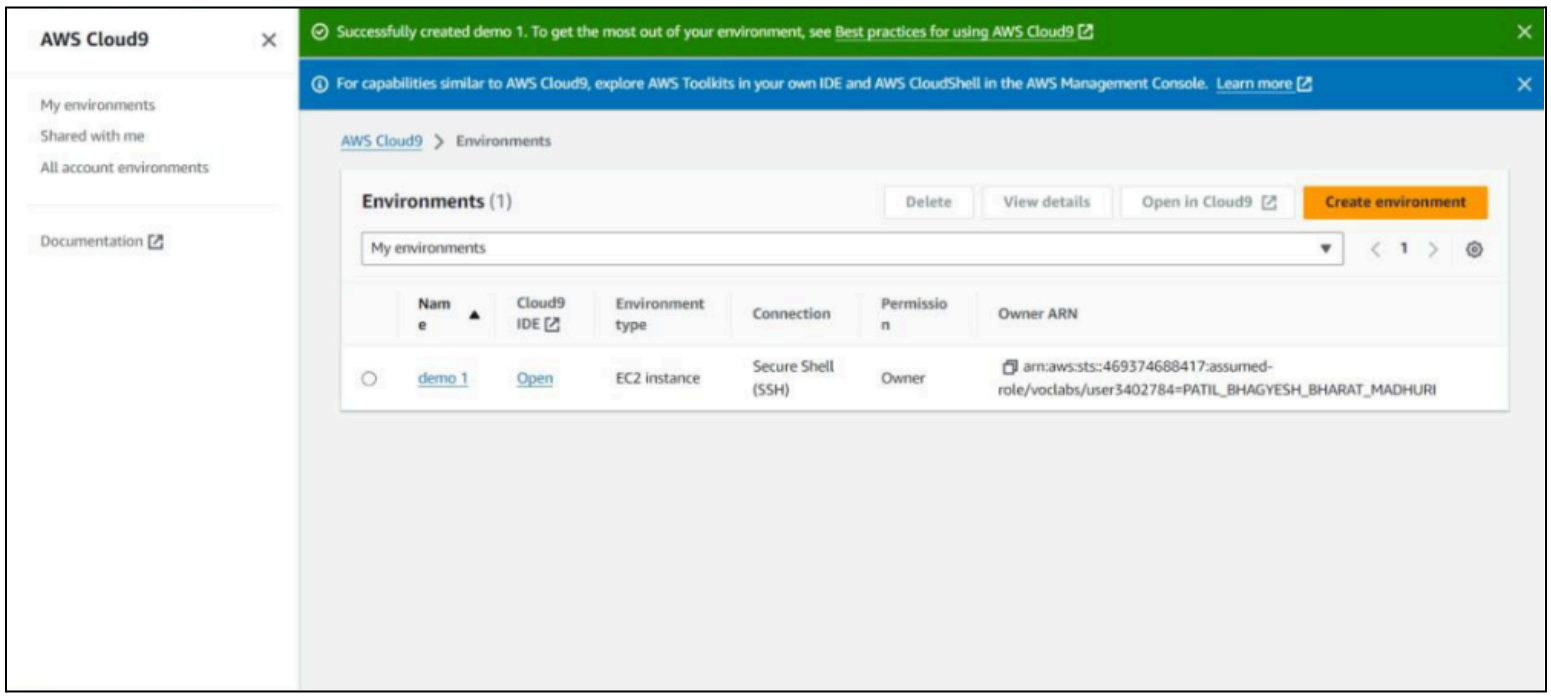
8) Configuring the environment.

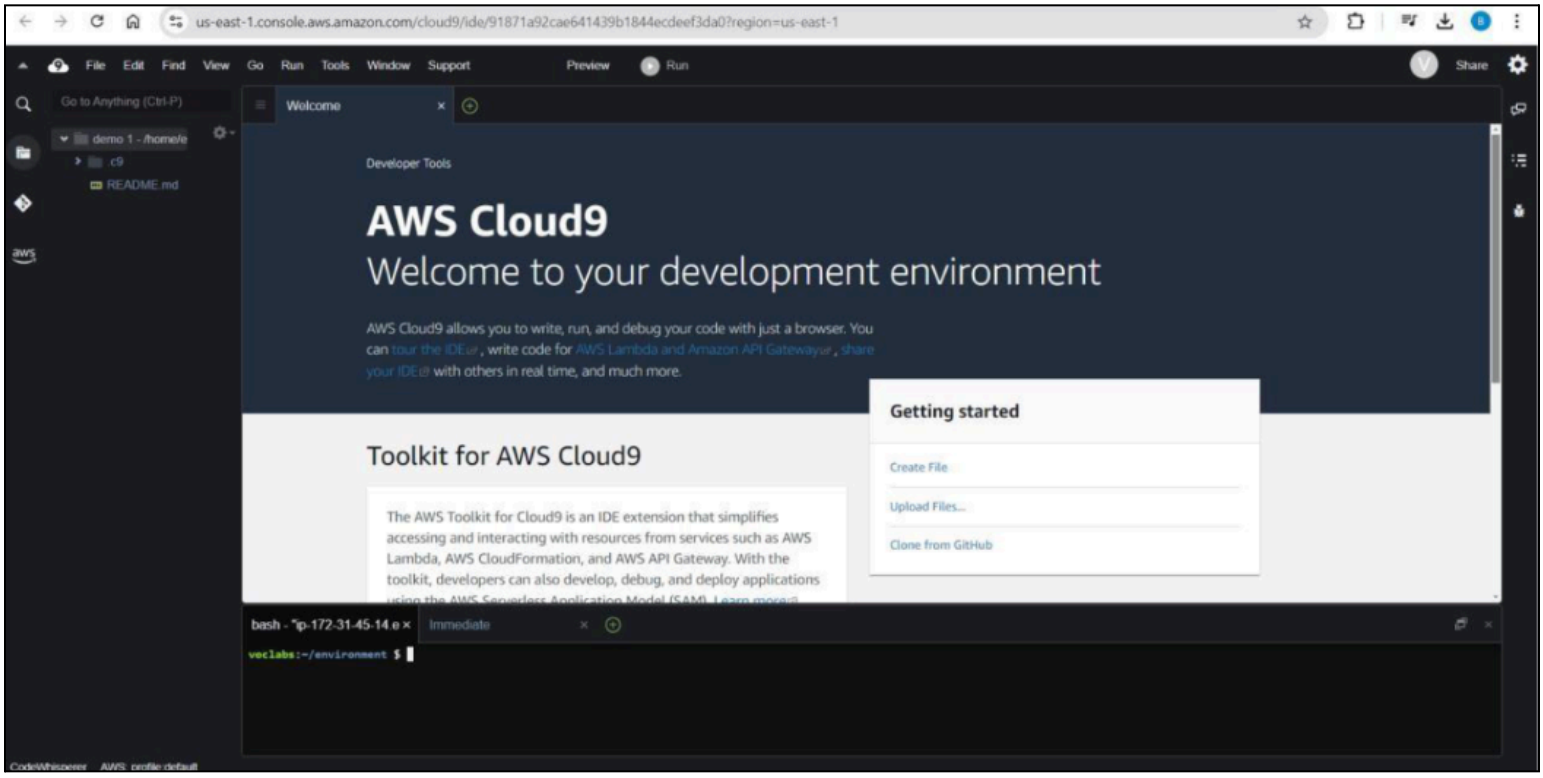


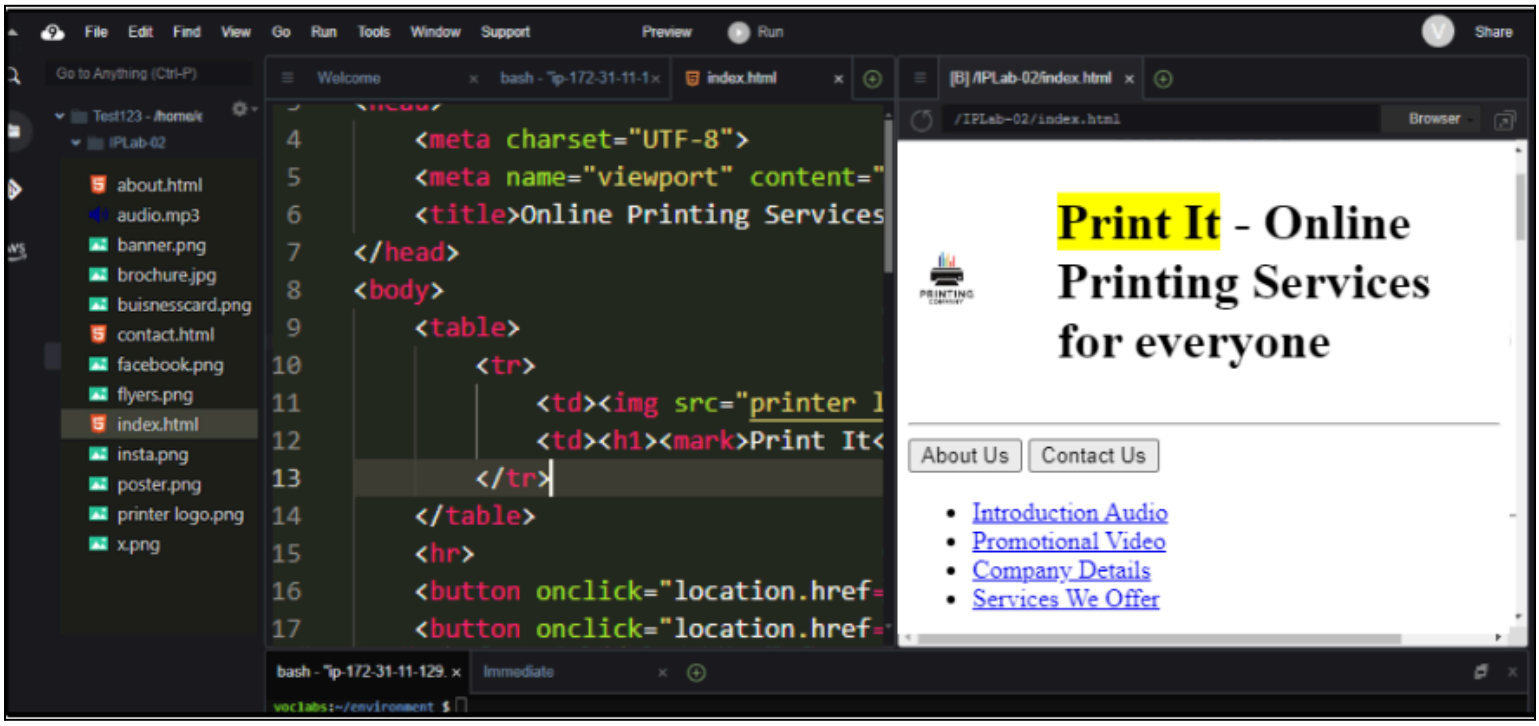

New EC2 instance
Instance type Info
The memory and CPU of the EC2 instance that will be created for Cloud9 to run on.
t2.micro (1 GiB RAM + 1 vCPU)
Free-tier eligible. Ideal for educational users and exploration.
t3.small (2 GiB RAM + 2 vCPU)
Recommended for small web projects.
m5.large (8 GiB RAM + 2 vCPU)
Recommended for production and most general-purpose development.
Additional instance types
Explore additional instances to fit your need.
Platform Info
This will be installed on your EC2 instance. We recommend Amazon Linux 2023.
Amazon Linux 2023
Timeout
How long Cloud9 can be inactive (no user input) before auto-hibernating. This helps prevent unnecessary charges.
30 minutes
Network settings Info
Connection
How your environment is accessed.


Network settings Info
Connection
How your environment is accessed.
AWS Systems Manager (SSM)
Accesses environment via SSM without opening inbound ports (no ingress).
Secure Shell (SSH)
Accesses environment directly via SSH, opens inbound ports.
VPC settings Info
Tags - optional Info
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.
The following IAM resources will be created in your account
AWSServiceRoleForAWSCloud9 - AWS Cloud9 creates a service-linked role for you. This allows AWS Cloud9 to call other AWS services on your behalf. You can delete the role from the AWS IAM console once you no longer have any AWS Cloud9 environments. Learn more
AWSCloud9SSMAccessRole and AWSCloud9SSMInstanceProfile - A service role and an instance profile are automatically created if Cloud9 accesses its EC2 instance through AWS Systems Manager. If your environments no longer require EC2 instances that block incoming traffic, you can delete these roles using the AWS IAM console. Learn more
Cancel
Create

9) Environment is successfully created.






File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
Test123 - /home/e
IPLab-02
about.html
audio.mp3
banner.png
brochure.jpg
buisnesscard.png
contact.html
facebook.png
flyers.png
index.html
insta.png
poster.png
printer logo.png
x.png
Welcome
bash - "ip-172-31-11-1
index.html
<meta charset="UTF-8">
<meta name="viewport" content="
<title>Online Printing Services
</head>
<body>
<table>
<tr>
<td><img src="printer l
<td><h1><mark>Print It<
</tr>
</table>
<hr>
<button onclick="location.href=
<button onclick="location.href=
bash - "ip-172-31-11-129.
Immediate
voclabs:~/environment $
[B] /IPLab-02/index.html
/IPLab-02/index.html
Browser
Print It - Online Printing Services for everyone
About Us
Contact Us
Introduction Audio
Promotional Video
Company Details
Services We Offer

Adv. Devops Experiment no. 2

Name: Rohan Lalchandani
Class: D15A Roll no.: 25

Aim: To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

Theory:

Amazon Elastic Beanstalk is a Platform-as-a-Service (PaaS) offered by AWS that simplifies the deployment, management, and scaling of applications. It abstracts the underlying infrastructure, allowing developers to focus on writing code rather than managing servers and infrastructure.

Key Features

1. Easy Deployment:

- **Deployment Options:** Supports multiple deployment methods, including the AWS Management Console, CLI, and SDKs. You can deploy applications using ZIP files, Docker images, or from a source control repository.
- **Managed Platform Updates:** Automatically handles platform updates and patches for the underlying infrastructure.

2. Application Management:

- **Environment Management:** Provides pre-configured environments for popular application platforms like Node.js, Python, Java, .NET, PHP, Ruby, and Docker.
- **Monitoring and Logging:** Integrated with Amazon CloudWatch and AWS X-Ray for monitoring performance and logging. Offers health monitoring and application logs through the Elastic Beanstalk console.

3. Auto Scaling and Load Balancing:

- **Auto Scaling:** Automatically adjusts the number of instances based on traffic and resource utilization.

- **Load Balancing:** Distributes incoming application traffic across multiple instances to ensure high availability.

4. Customization and Flexibility:

- **Configuration Options:** Allows customization through configuration files (e.g., `.ebextensions`) and environment variables. You can configure instance types, scaling policies, and load balancer settings.
- **Integration with AWS Services:** Easily integrates with other AWS services like Amazon RDS (Relational Database Service), S3 (Simple Storage Service), and IAM (Identity and Access Management).

5. Environment Types:

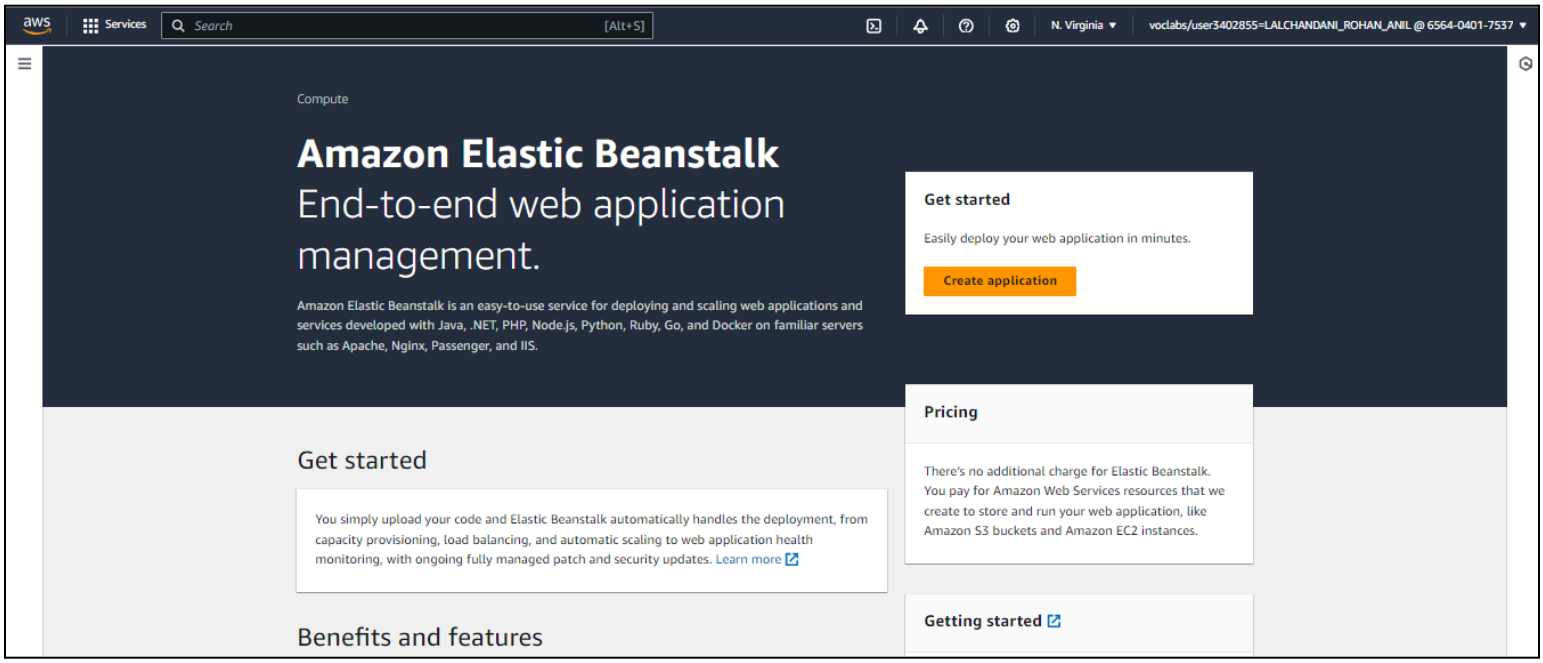
- **Single Instance Environment:** Suitable for development and testing where high availability and fault tolerance are not required.
- **Load Balanced Environment:** Designed for production with auto-scaling, load balancing, and high availability.

Common Use Cases

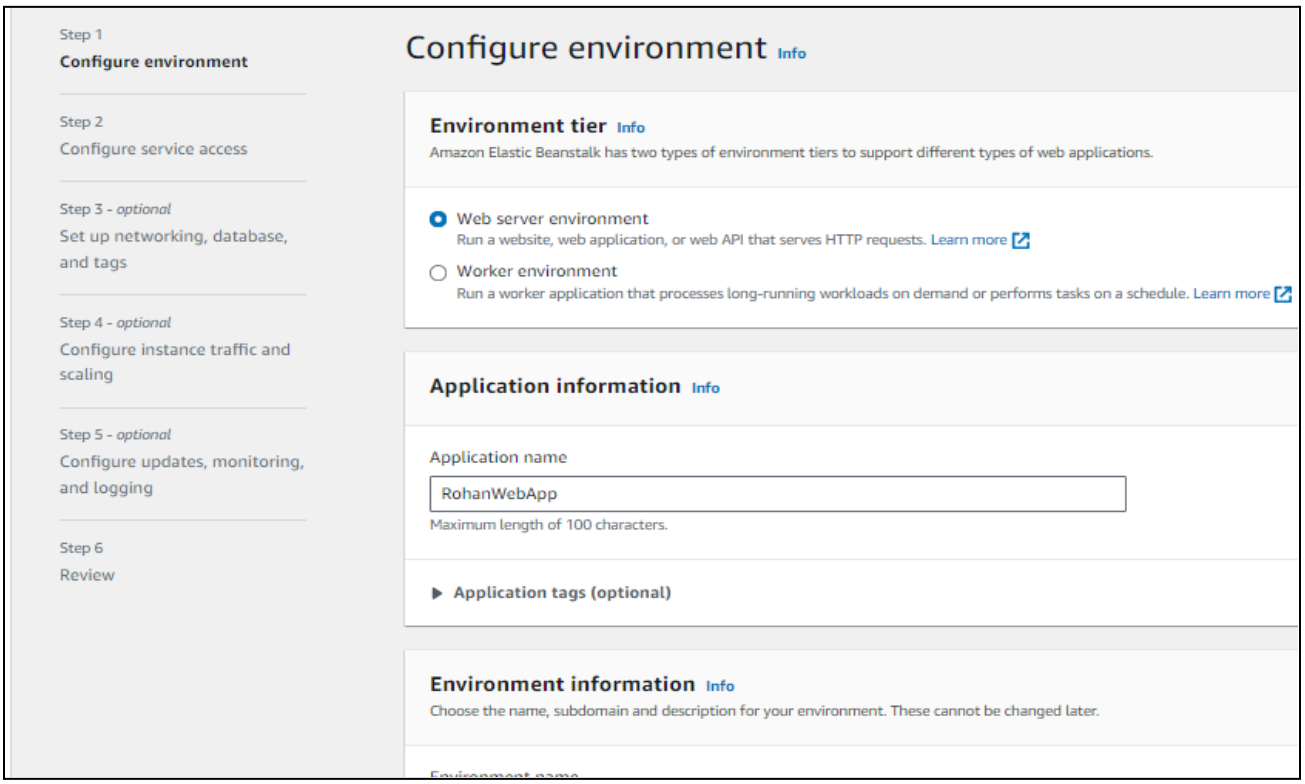
- **Web Applications:** Deploy and manage web applications with a scalable backend and frontend.
- **APIs:** Host RESTful APIs or microservices with auto-scaling and high availability.
- **Development and Testing:** Quickly spin up environments for development and testing purposes.

Implementation:

1) Search for Elastic Beanstalk in the search box and click on create application.



2) Configuring the environment.



Platform Info

Platform type

- (•) Managed platform
  Platforms published and maintained by Amazon Elastic Beanstalk. Learn more
- ( ) Custom platform
  Platforms created and owned by you. This option is unavailable if you have no platforms.

Platform

Python

Platform branch

Python 3.11 running on 64bit Amazon Linux 2023

Platform version

4.1.3 (Recommended)

Application code Info

- (•) Sample application
- ( ) Existing version
  Application versions that you have uploaded.
- ( ) Upload your code
  Upload a source bundle from your computer or copy one from Amazon S3.

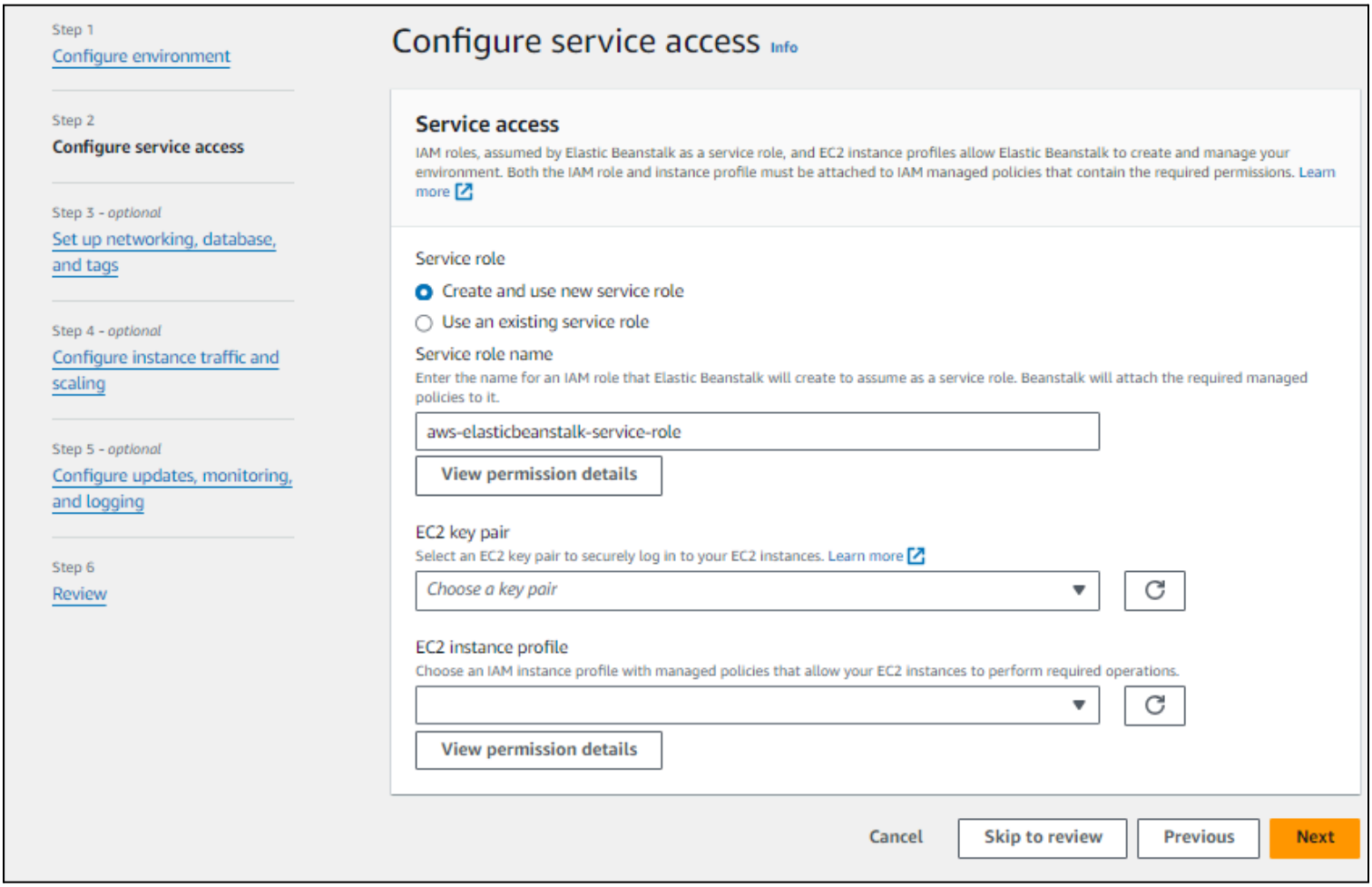
Presets Info

Start from a preset that matches your use case or choose custom configuration to unset recommended values and use the service's default values.

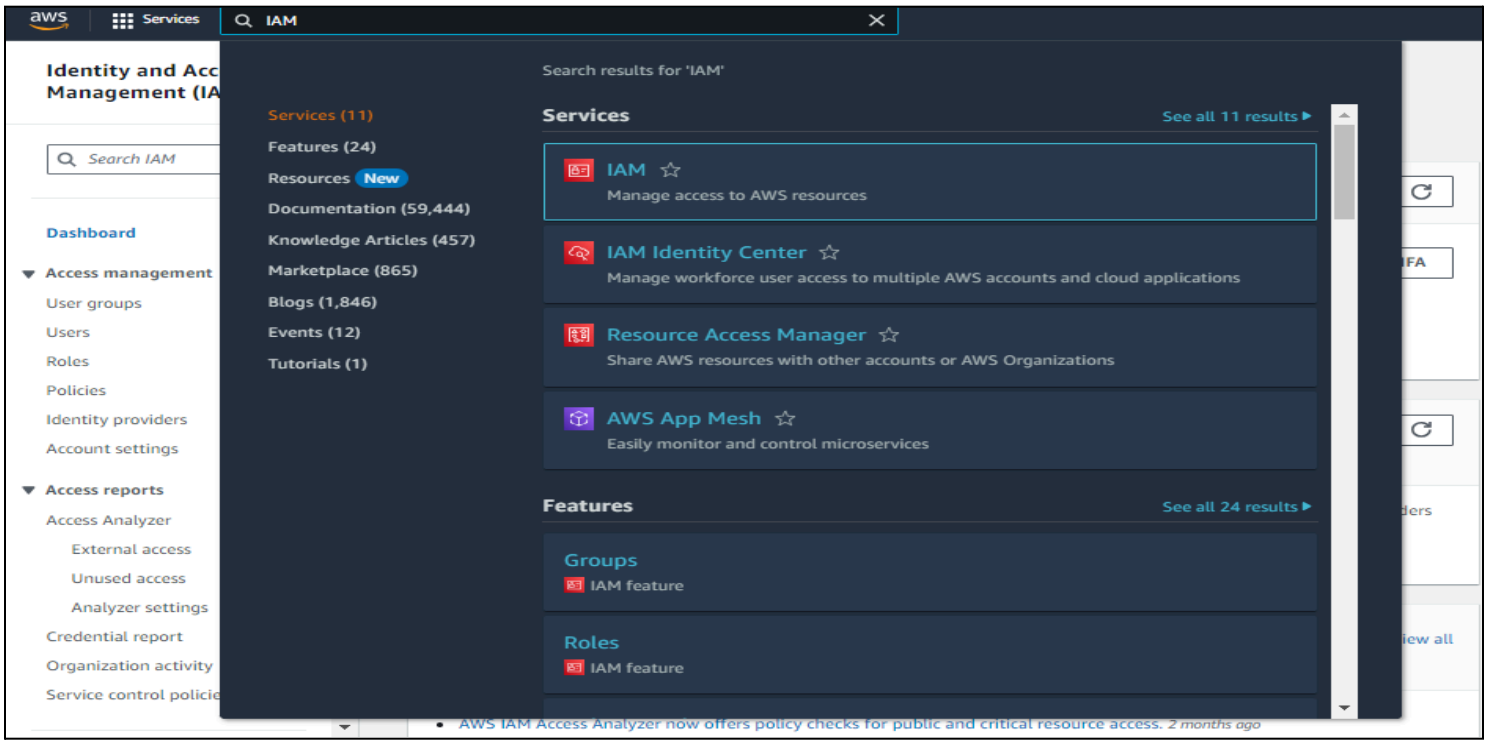
Configuration presets

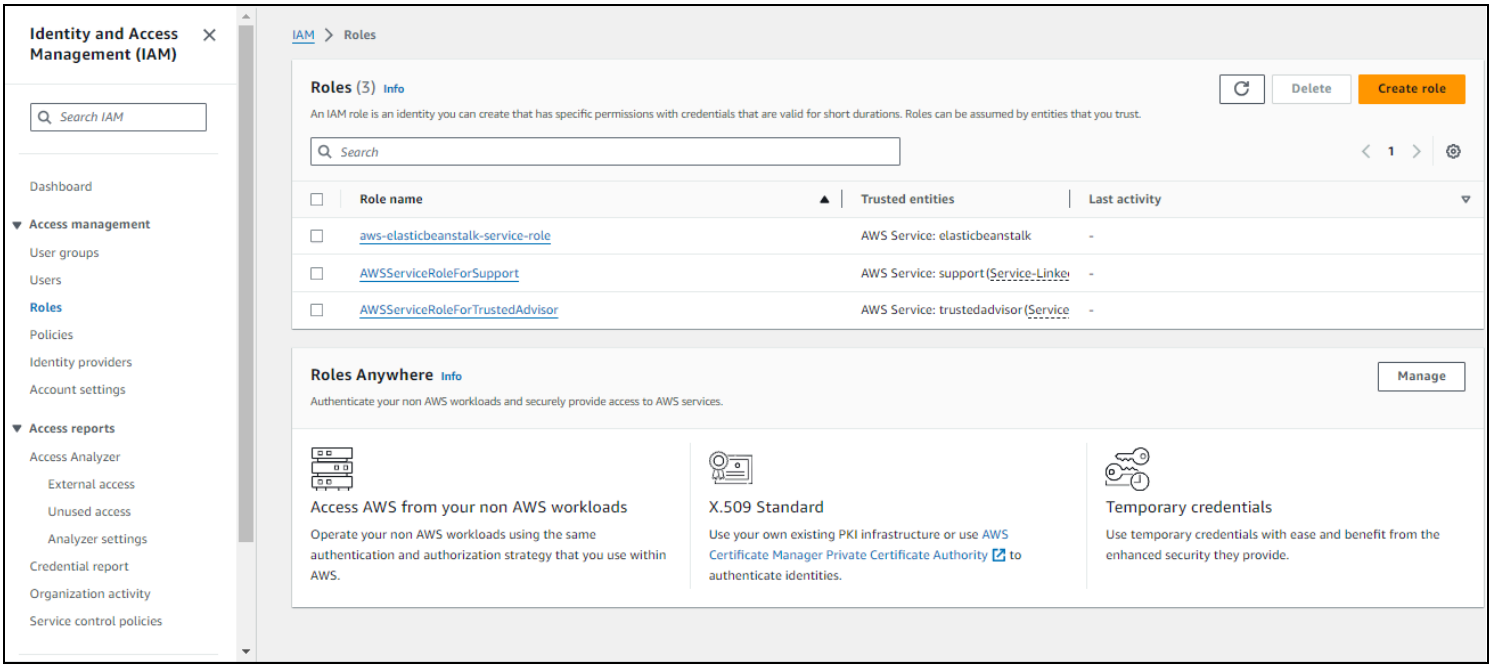
- (•) Single instance (free tier eligible)
- ( ) Single instance (using spot instance)
- ( ) High availability
- ( ) High availability (using spot and on-demand instances)
- ( ) Custom configuration

3) Click on create service role and for selecting EC2 instance profile we need to create an IAM policy for it.

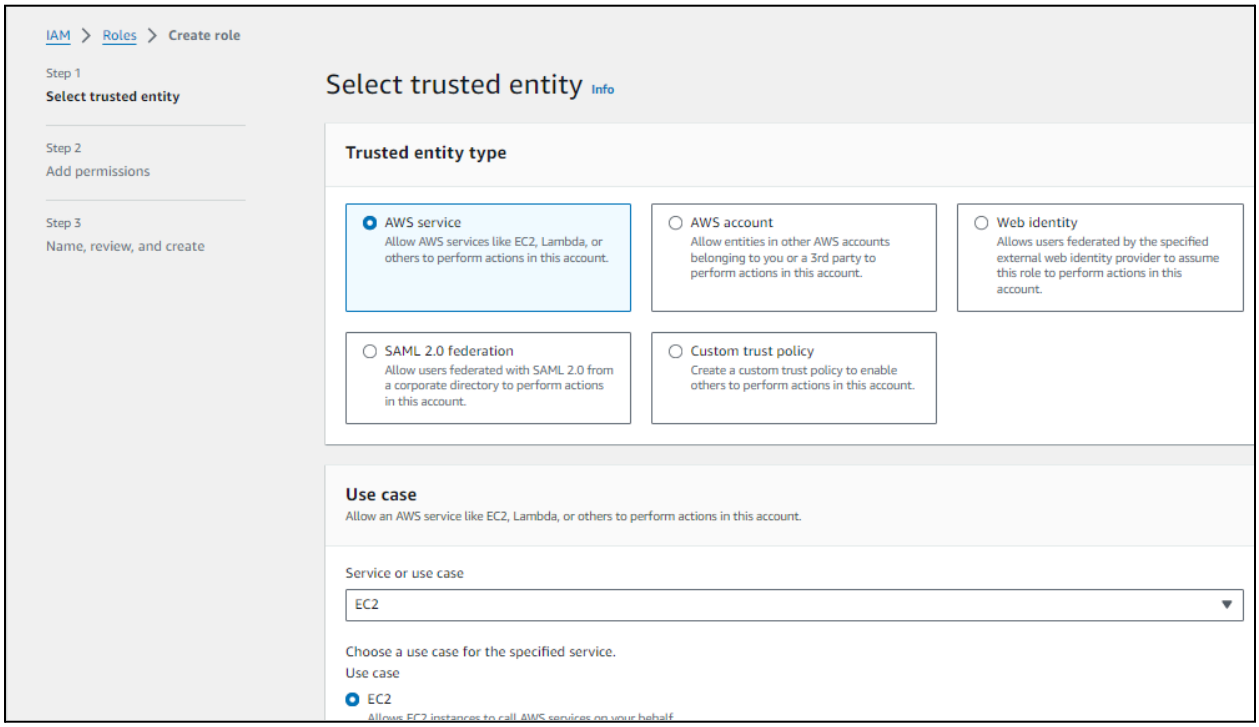


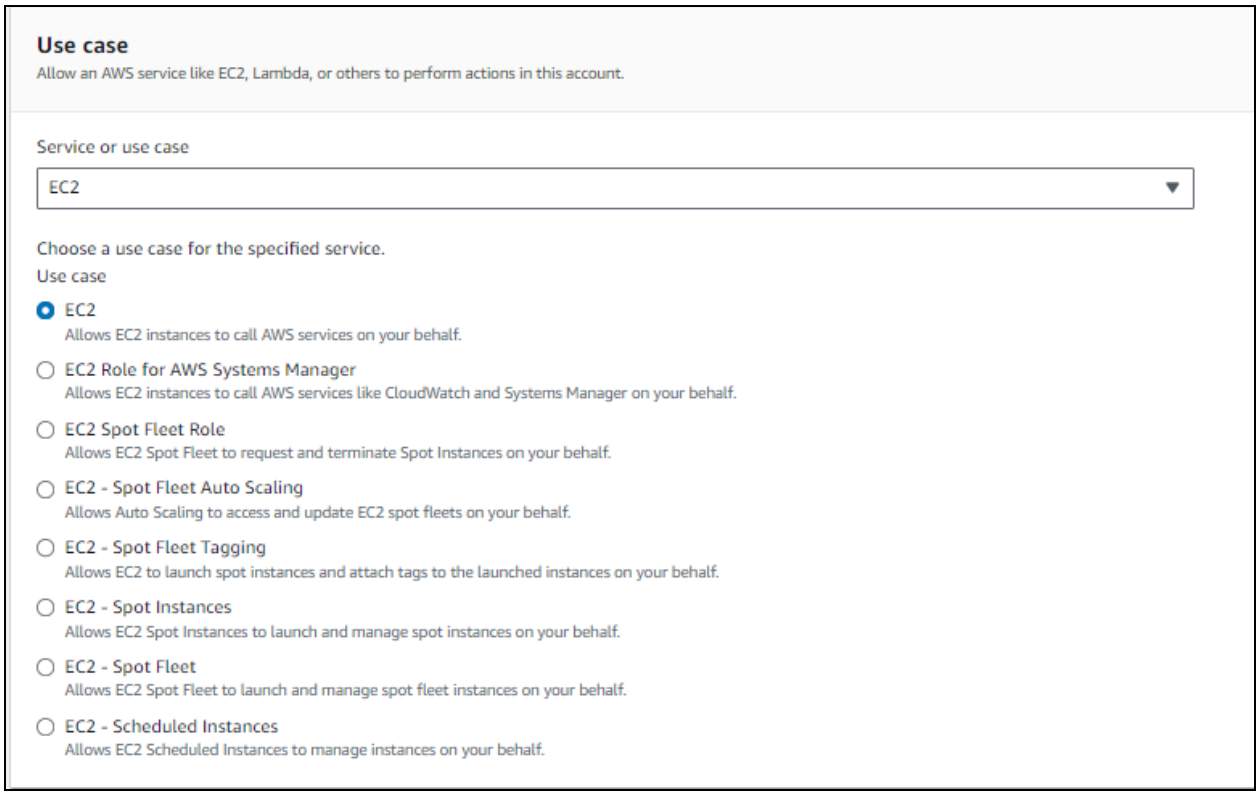
4) Search for IAM in the search box and click on “roles” on the left hand side bar and click on create role.



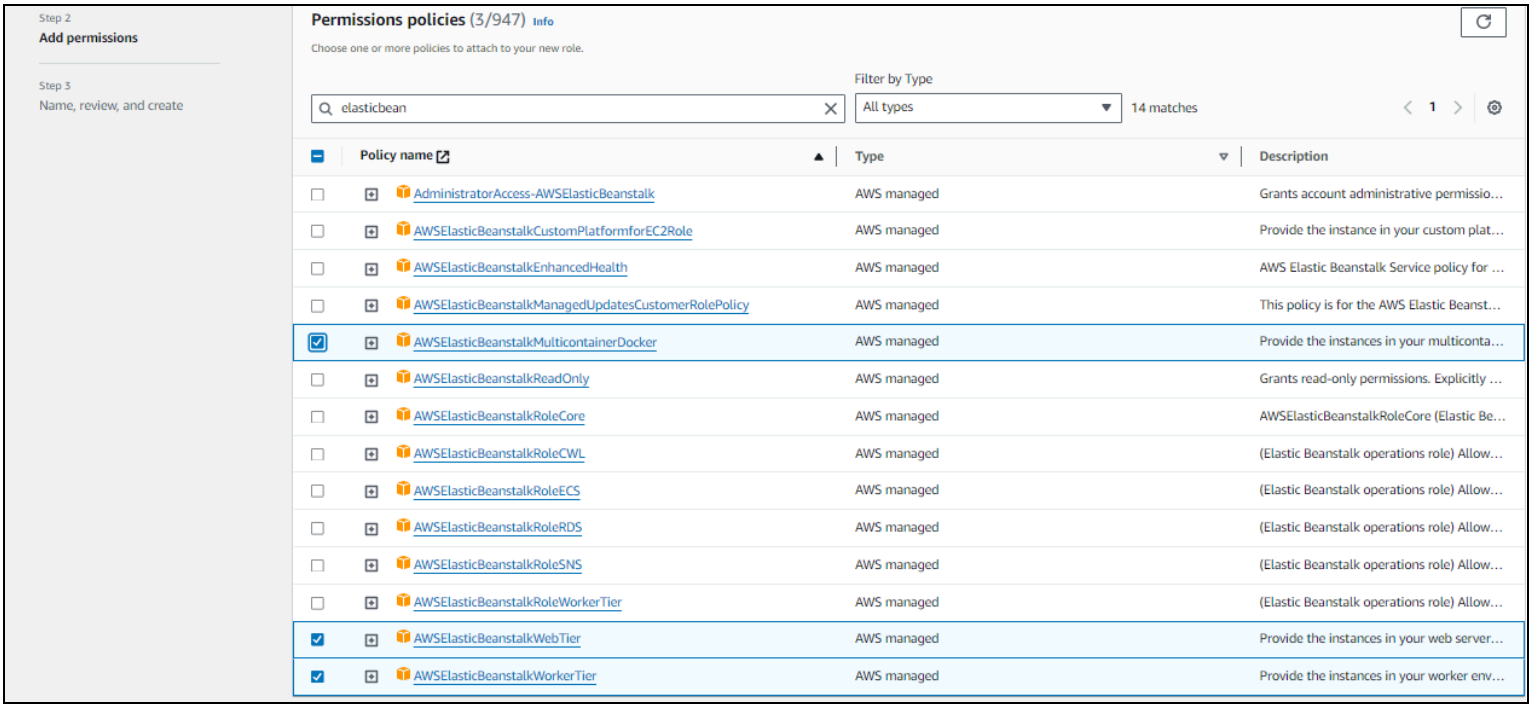


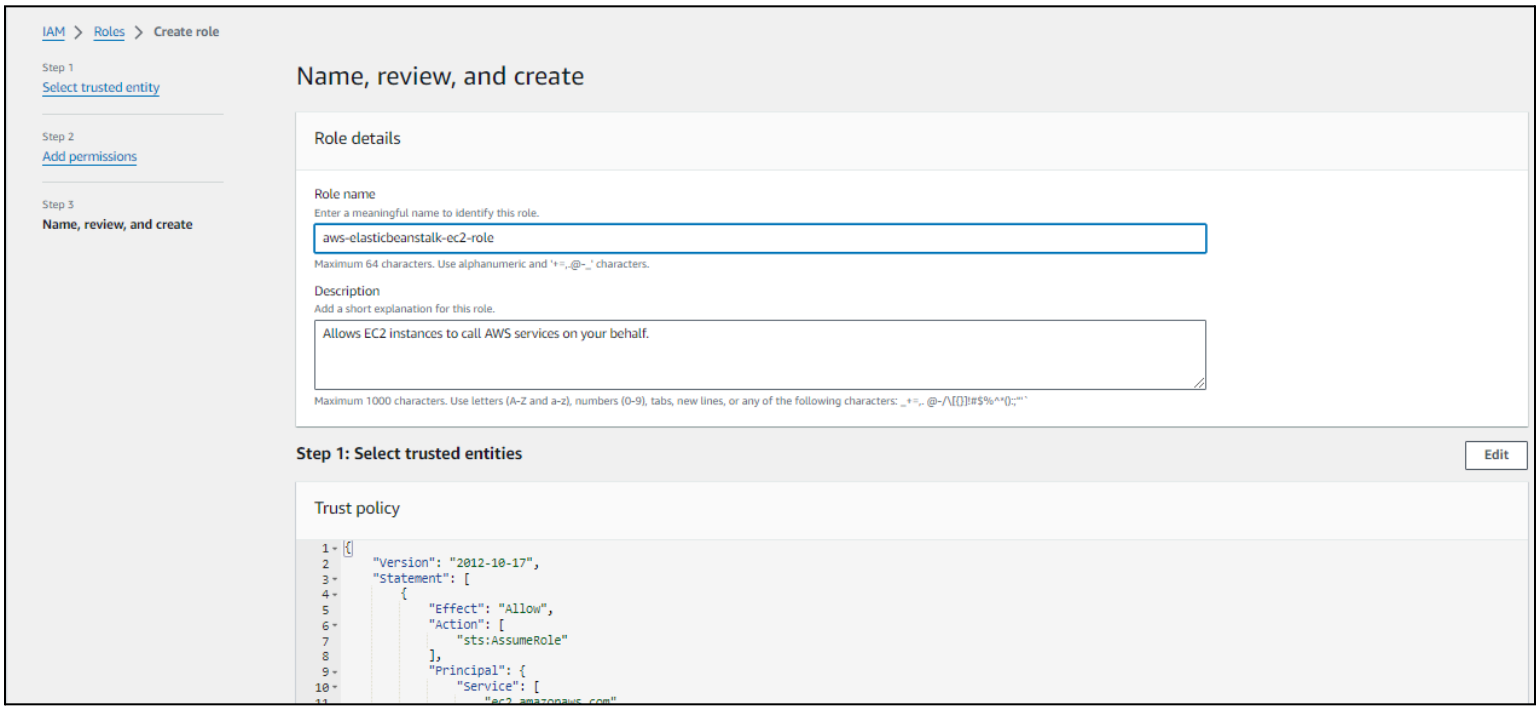
5) Configuring the IAM role.

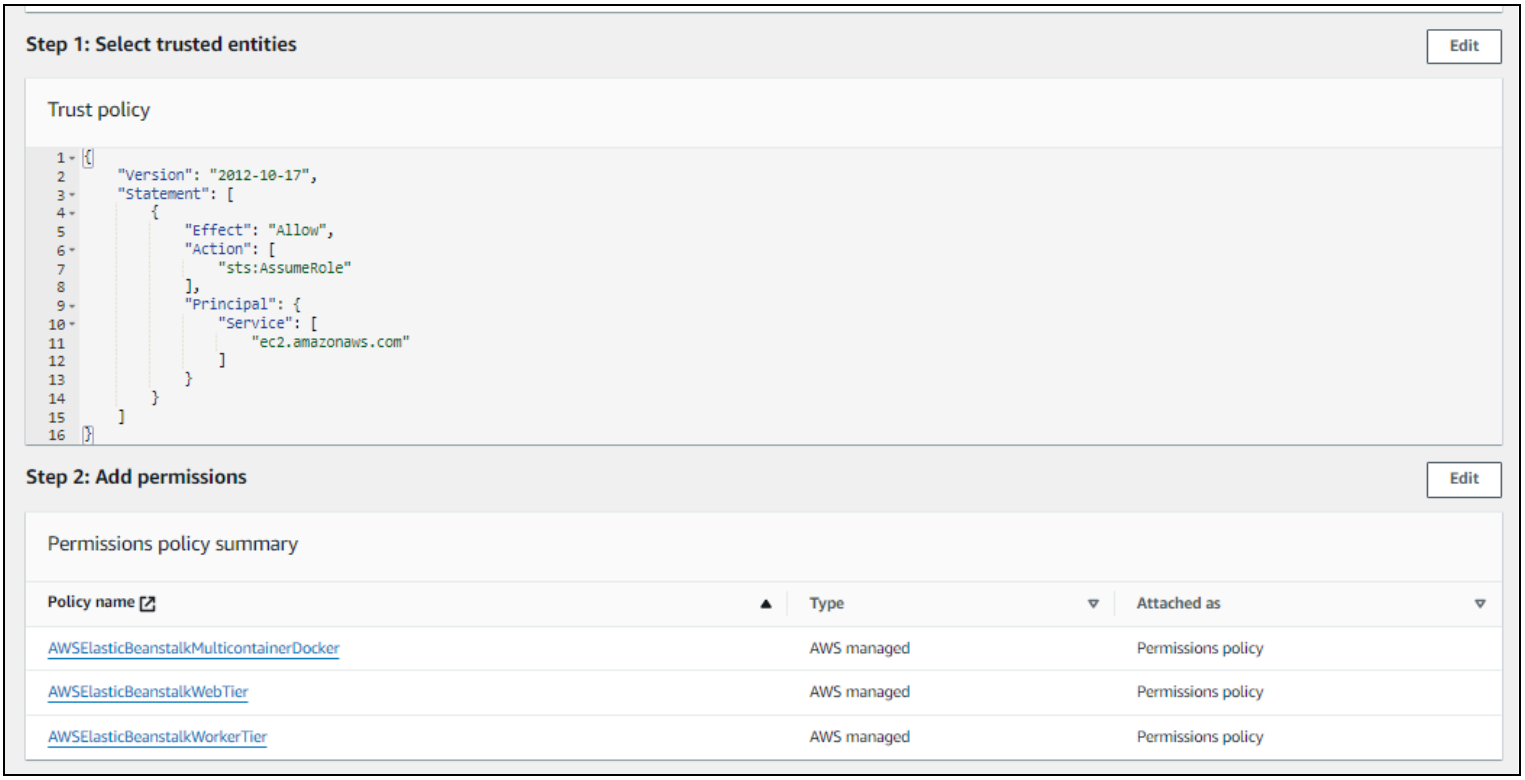




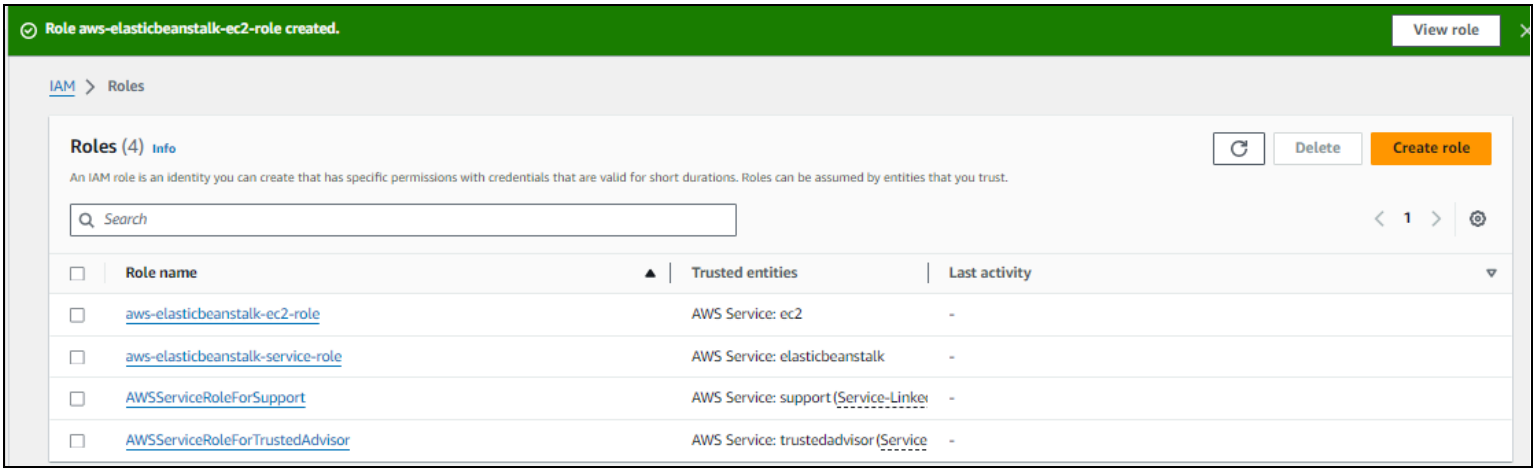
6) We need to select three permission policies as shown below.


Step 2
Add permissions
Step 3
Name, review, and create
Permissions policies (3/947) Info
Choose one or more policies to attach to your new role.
Filter by Type
elasticbean
All types
14 matches
1
Policy name
Type
Description
AdministratorAccess-AWSElasticBeanstalk
AWS managed
Grants account administrative permissio...
AWSElasticBeanstalkCustomPlatformforEC2Role
AWS managed
Provide the instance in your custom plat...
AWSElasticBeanstalkEnhancedHealth
AWS managed
AWS Elastic Beanstalk Service policy for ...
AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy
AWS managed
This policy is for the AWS Elastic Beanst...
AWSElasticBeanstalkMulticontainerDocker
AWS managed
Provide the instances in your multiconta...
AWSElasticBeanstalkReadOnly
AWS managed
Grants read-only permissions. Explicitly ...
AWSElasticBeanstalkRoleCore
AWS managed
AWSElasticBeanstalkRoleCore (Elastic Be...
AWSElasticBeanstalkRoleCWL
AWS managed
(Elastic Beanstalk operations role) Allow...
AWSElasticBeanstalkRoleECS
AWS managed
(Elastic Beanstalk operations role) Allow...
AWSElasticBeanstalkRoleRDS
AWS managed
(Elastic Beanstalk operations role) Allow...
AWSElasticBeanstalkRoleSNS
AWS managed
(Elastic Beanstalk operations role) Allow...
AWSElasticBeanstalkRoleWorkerTier
AWS managed
(Elastic Beanstalk operations role) Allow...
AWSElasticBeanstalkWebTier
AWS managed
Provide the instances in your web server...
AWSElasticBeanstalkWorkerTier
AWS managed
Provide the instances in your worker env...

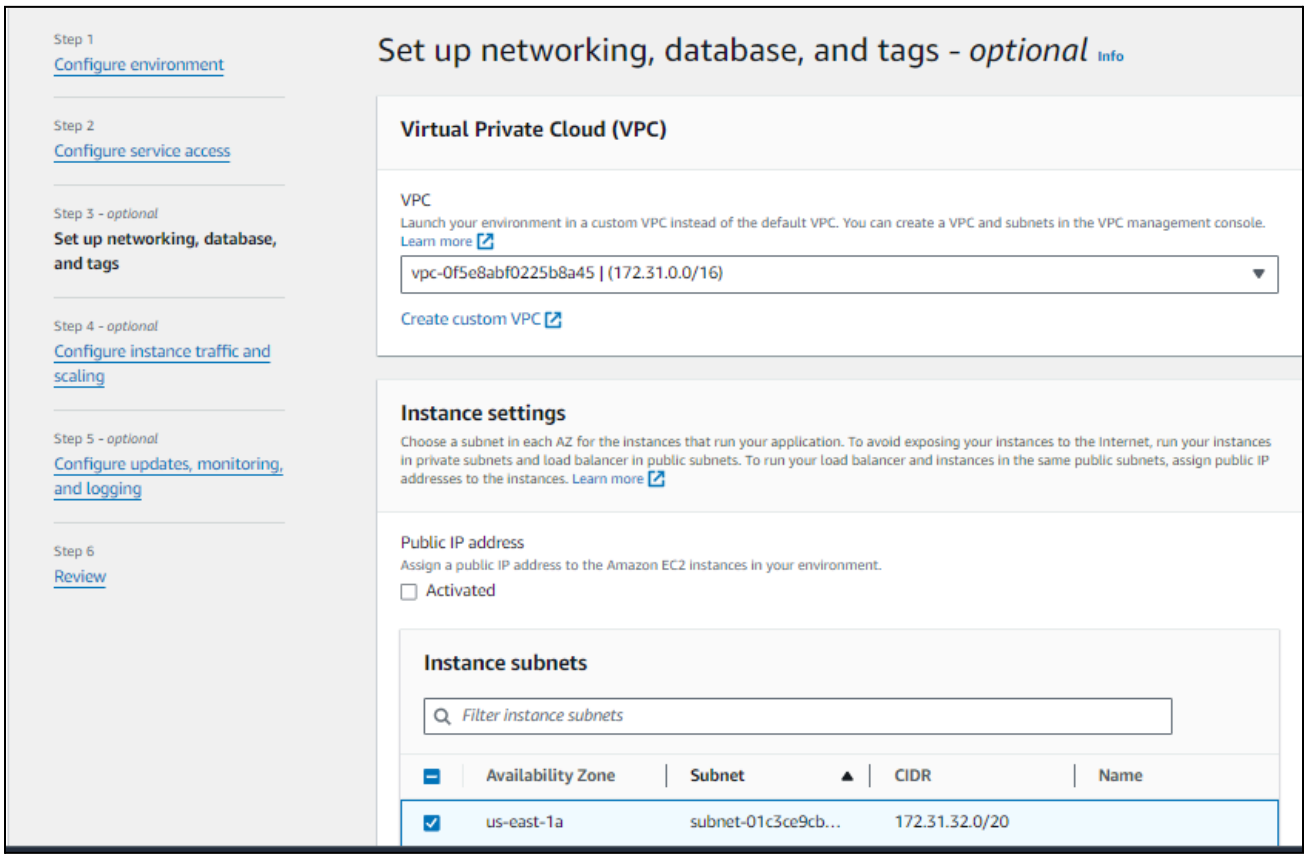

IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Name, review, and create
Role details
Role name
Enter a meaningful name to identify this role.
aws-elasticbeanstalk-ec2-role
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.
Description
Add a short explanation for this role.
Allows EC2 instances to call AWS services on your behalf.
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,.@-/\[{}]!#$%^*();:"'`
Step 1: Select trusted entities
Edit
Trust policy
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"sts:AssumeRole"
],
"Principal": {
"Service": [

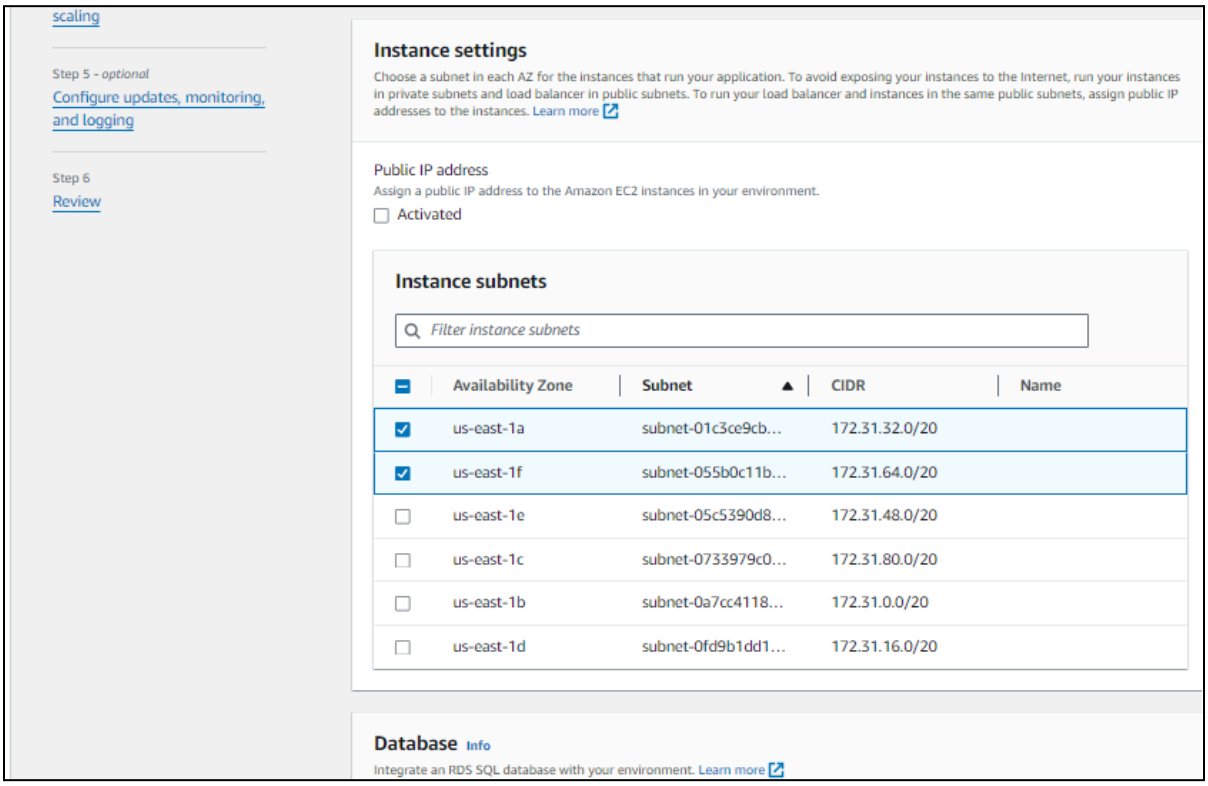


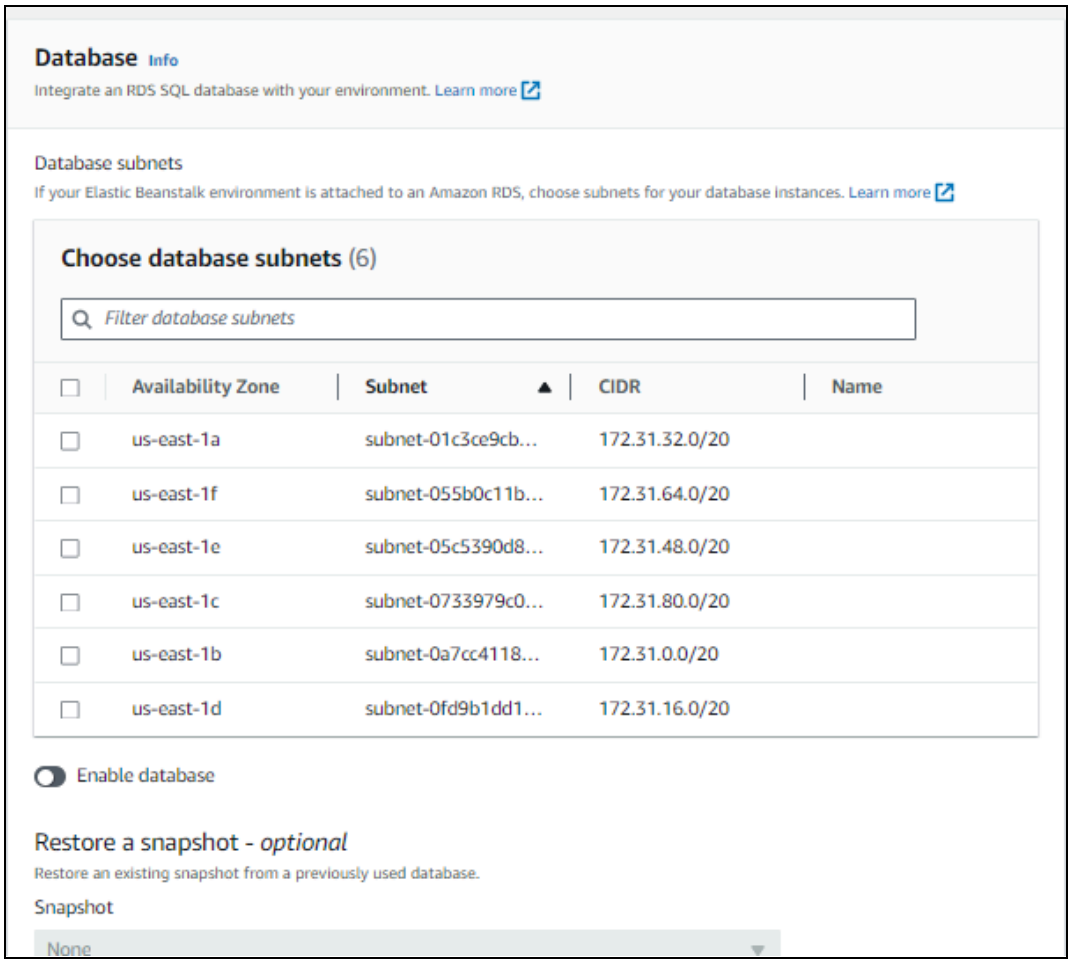
7) IAM Role for EC2 instance is successfully created, go back to “Configuring Service access” for elastic beanstalk and select the EC2 instance profile.

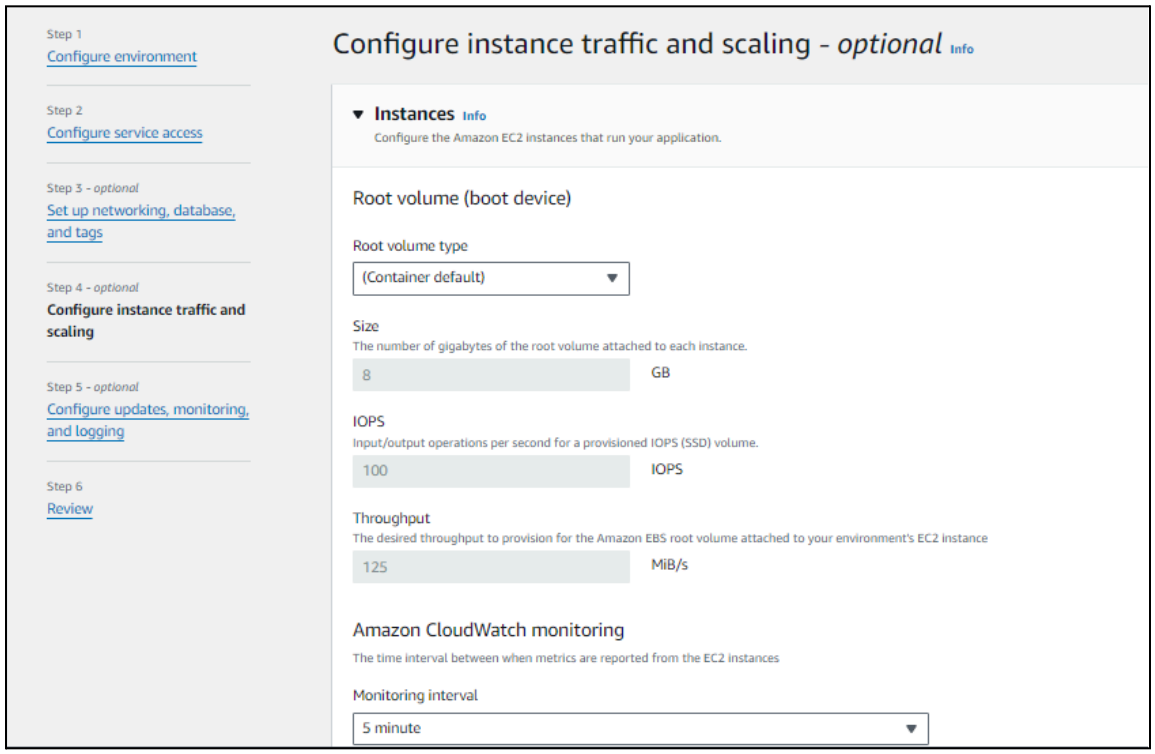


8) Select default settings as shown below in the next section.






Database Info
Integrate an RDS SQL database with your environment. Learn more
Database subnets
If your Elastic Beanstalk environment is attached to an Amazon RDS, choose subnets for your database instances. Learn more
Choose database subnets (6)
Filter database subnets
Availability Zone
Subnet
CIDR
Name
us-east-1a
subnet-01c3ce9cb...
172.31.32.0/20
us-east-1f
subnet-055b0c11b...
172.31.64.0/20
us-east-1e
subnet-05c5390d8...
172.31.48.0/20
us-east-1c
subnet-0733979c0...
172.31.80.0/20
us-east-1b
subnet-0a7cc4118...
172.31.0.0/20
us-east-1d
subnet-0fd9b1dd1...
172.31.16.0/20
Enable database
Restore a snapshot - optional
Restore an existing snapshot from a previously used database.
Snapshot
None


Step 1
Configure environment
Step 2
Configure service access
Step 3 - optional
Set up networking, database, and tags
Step 4 - optional
Configure instance traffic and scaling
Step 5 - optional
Configure updates, monitoring, and logging
Step 6
Review
Configure instance traffic and scaling - optional Info
Instances Info
Configure the Amazon EC2 instances that run your application.
Root volume (boot device)
Root volume type
(Container default)
Size
The number of gigabytes of the root volume attached to each instance.
8
GB
IOPS
Input/output operations per second for a provisioned IOPS (SSD) volume.
100
IOPS
Throughput
The desired throughput to provision for the Amazon EBS root volume attached to your environment's EC2 instance
125
MiB/s
Amazon CloudWatch monitoring
The time interval between when metrics are reported from the EC2 instances
Monitoring interval
5 minute

Amazon CloudWatch monitoring

The time interval between when metrics are reported from the EC2 instances

Monitoring interval

5 minute

Instance metadata service (IMDS)

Your environment's platform supports both IMDSv1 and IMDSv2. To enforce IMDSv2, deactivate IMDSv1. Learn more

IMDSv1

With the current setting, the environment enables only IMDSv2.

☑ Deactivated

EC2 security groups

Select security groups to control traffic.

EC2 security groups (4)

Filter security groups

	Group name	Group ID	Name
☑	default	sg-009f9ea9079c28fd9	
☐	launch-wizard-1	sg-094cfd5e04d89a25f	
☐	launch-wizard-2	sg-0c0fbdd01d426cf3f	
☐	launch-wizard-3	sg-04536e9fd071d79ea	

Capacity Info

Configure the compute capacity of your environment and auto scaling settings to optimize the number of instances used.

Auto scaling group

Environment type

Select a single-instance or load-balanced environment. You can develop and test an application in a single-instance environment to save costs and then upgrade to a load-balanced environment when the application is ready for production. Learn more

Single instance

Instances

1 Min

1 Max

Fleet composition

Spot instances are launched at the lowest available price. Learn more

◉ On-Demand instance

○ Spot instance

Maximum spot price

The maximum price per instance-hour, in USD, that you're willing to pay for a Spot Instance. Setting a custom price limits your chances to fulfill your target capacity using Spot instances.

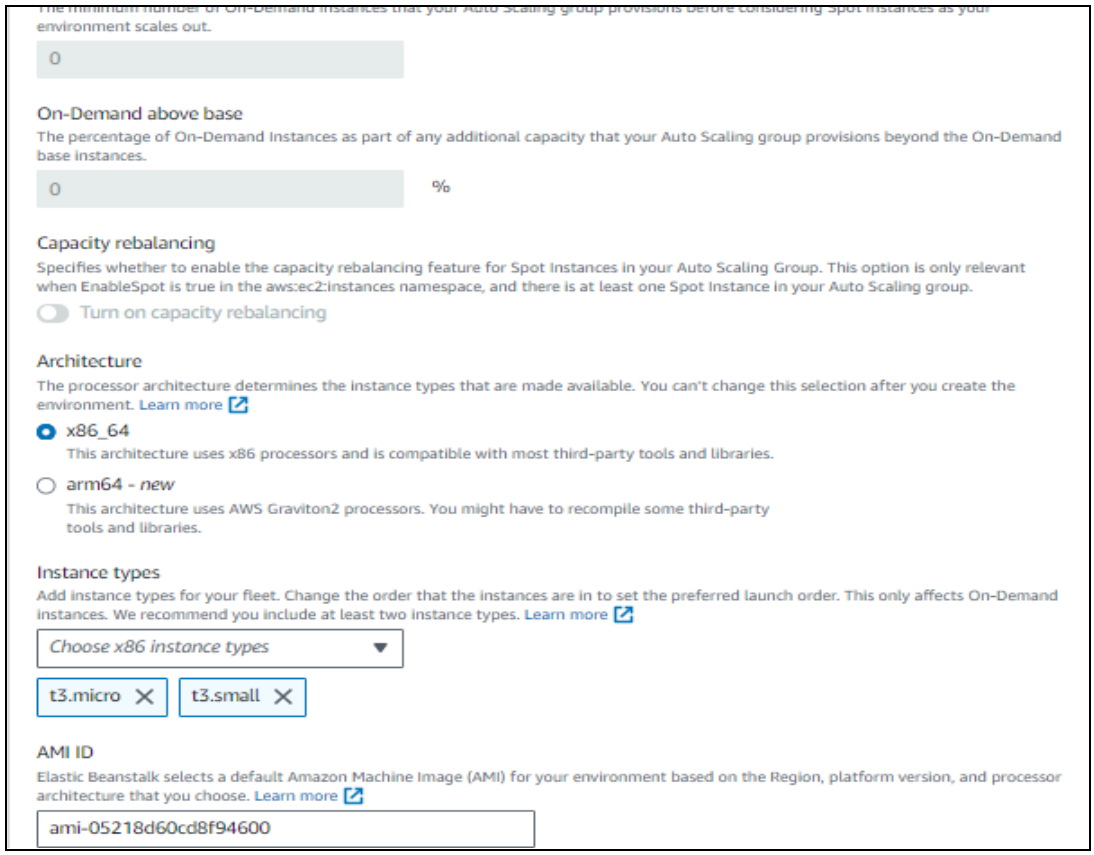
◉ Default

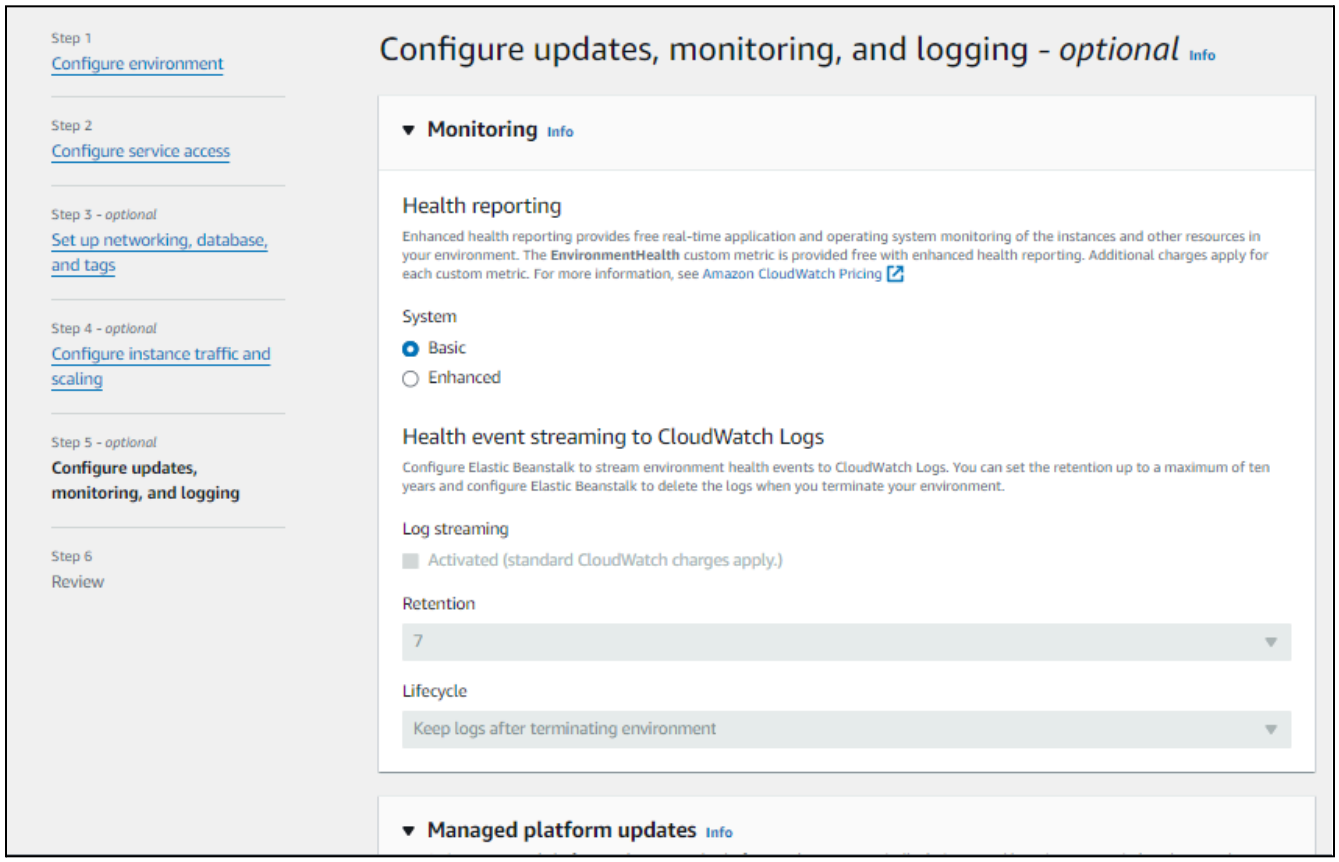
○ Set your maximum price

On-Demand base

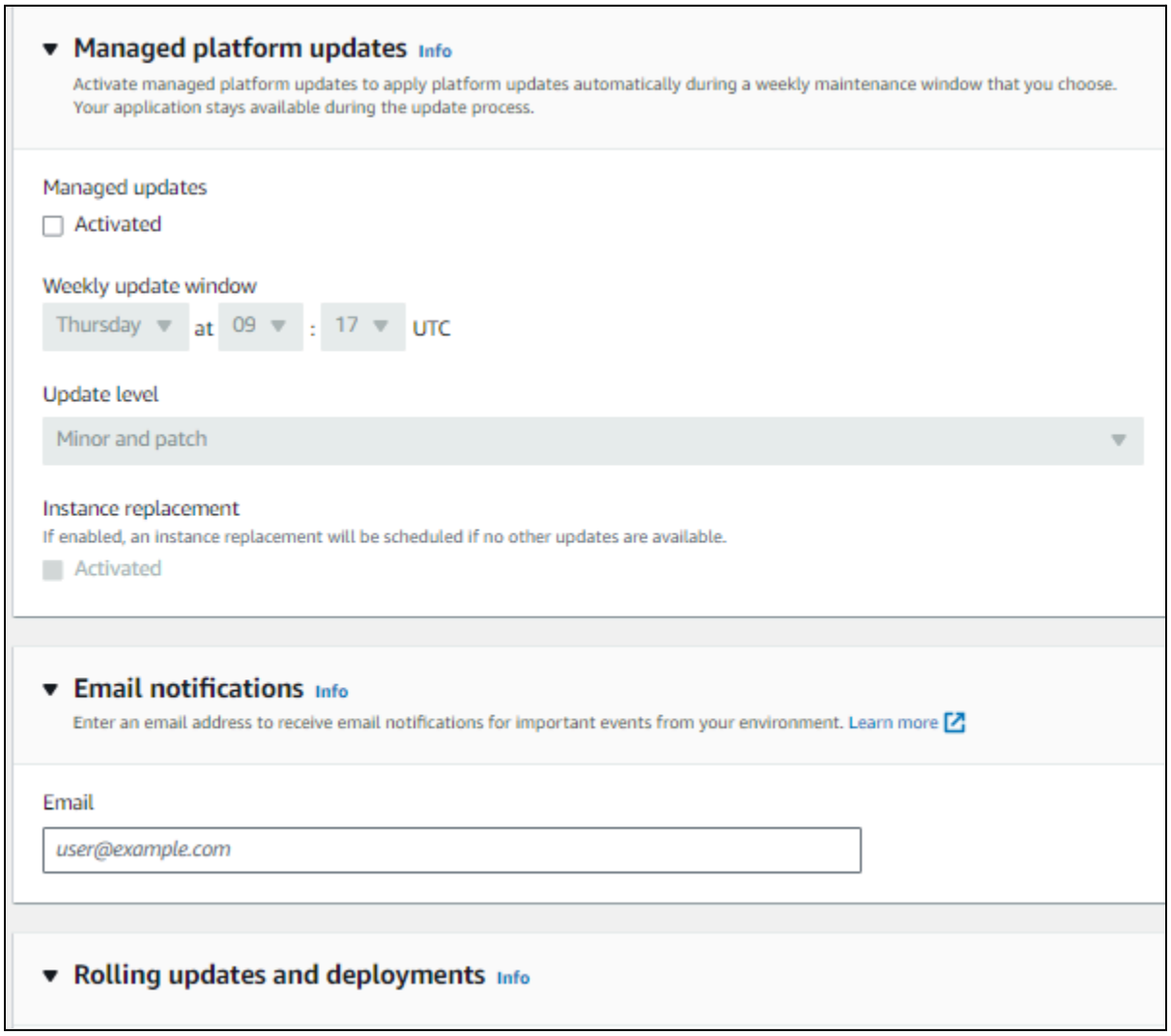
The minimum number of On-Demand Instances that your Auto Scaling group provisions before considering Spot Instances as your environment scales out.

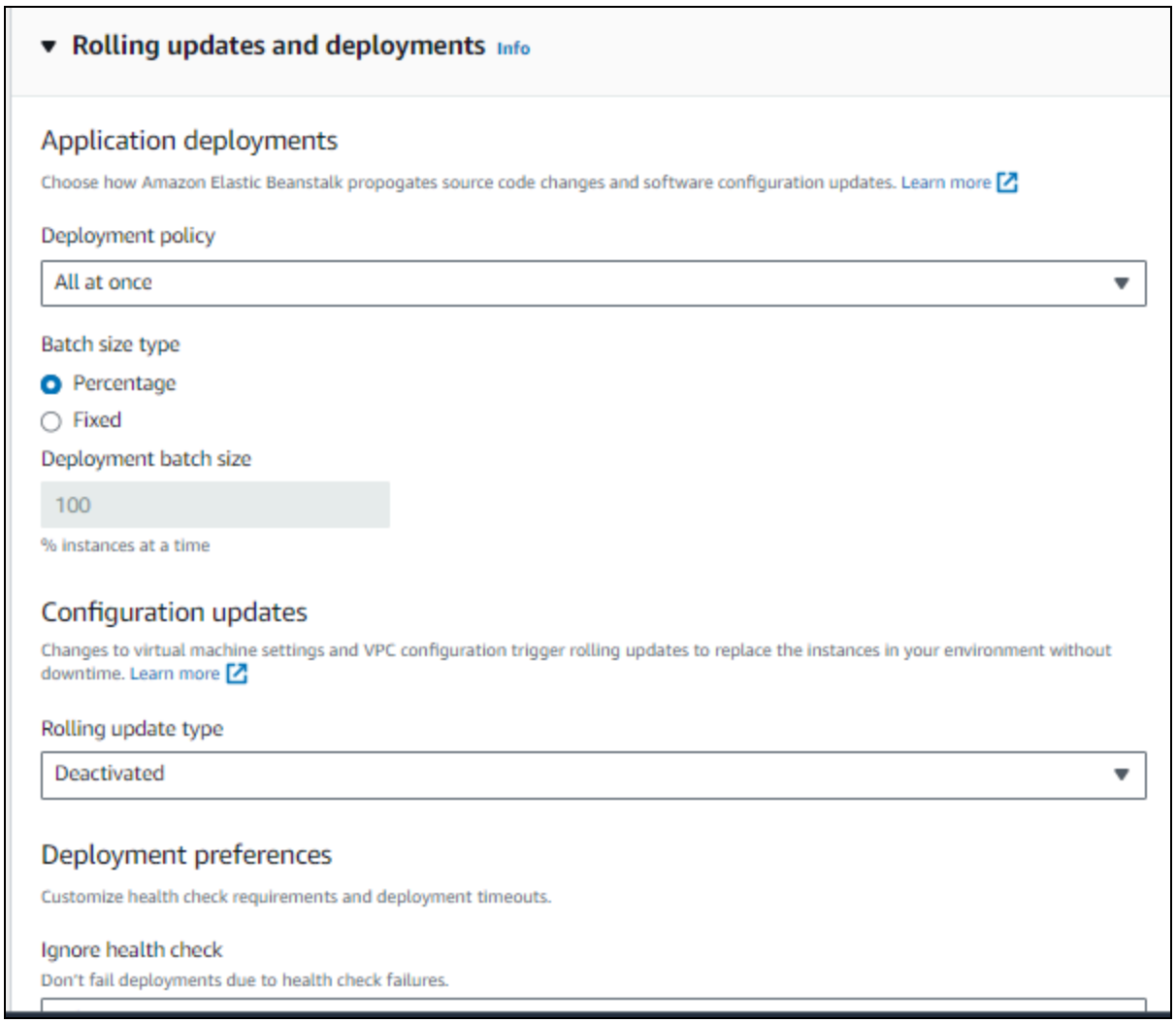
0


environment scales out.
0
On-Demand above base
The percentage of On-Demand Instances as part of any additional capacity that your Auto Scaling group provisions beyond the On-Demand base instances.
0
%
Capacity rebalancing
Specifies whether to enable the capacity rebalancing feature for Spot Instances in your Auto Scaling Group. This option is only relevant when EnableSpot is true in the aws:ec2:instances namespace, and there is at least one Spot Instance in your Auto Scaling group.
Turn on capacity rebalancing
Architecture
The processor architecture determines the instance types that are made available. You can't change this selection after you create the environment. Learn more
x86_64
This architecture uses x86 processors and is compatible with most third-party tools and libraries.
arm64 - new
This architecture uses AWS Graviton2 processors. You might have to recompile some third-party tools and libraries.
Instance types
Add instance types for your fleet. Change the order that the instances are in to set the preferred launch order. This only affects On-Demand instances. We recommend you include at least two instance types. Learn more
Choose x86 instance types
t3.micro
t3.small
AMI ID
Elastic Beanstalk selects a default Amazon Machine Image (AMI) for your environment based on the Region, platform version, and processor architecture that you choose. Learn more
ami-05218d60cd8f94600


Step 1
Configure environment
Step 2
Configure service access
Step 3 - optional
Set up networking, database, and tags
Step 4 - optional
Configure instance traffic and scaling
Step 5 - optional
Configure updates, monitoring, and logging
Step 6
Review
Configure updates, monitoring, and logging - optional Info
Monitoring Info
Health reporting
Enhanced health reporting provides free real-time application and operating system monitoring of the instances and other resources in your environment. The EnvironmentHealth custom metric is provided free with enhanced health reporting. Additional charges apply for each custom metric. For more information, see Amazon CloudWatch Pricing
System
Basic
Enhanced
Health event streaming to CloudWatch Logs
Configure Elastic Beanstalk to stream environment health events to CloudWatch Logs. You can set the retention up to a maximum of ten years and configure Elastic Beanstalk to delete the logs when you terminate your environment.
Log streaming
Activated (standard CloudWatch charges apply.)
Retention
7
Lifecycle
Keep logs after terminating environment
Managed platform updates Info

9) Uncheck Managed Updates, Activated.




Rolling updates and deployments Info
Application deployments
Choose how Amazon Elastic Beanstalk propogates source code changes and software configuration updates. Learn more
Deployment policy
All at once
Batch size type
Percentage
Fixed
Deployment batch size
100
% instances at a time
Configuration updates
Changes to virtual machine settings and VPC configuration trigger rolling updates to replace the instances in your environment without downtime. Learn more
Rolling update type
Deactivated
Deployment preferences
Customize health check requirements and deployment timeouts.
Ignore health check
Don't fail deployments due to health check failures.

Platform software Info

Configure the options available to your specific platform. These include the proxy server and OS environment properties. Learn more

Container options

Proxy server

Nginx

Amazon X-Ray

Amazon X-Ray is a service that collects data about the requests and responses that your application serves and receives. You can use the tools that X-Ray offers to view and filter the data that it provides to identify potential issues and optimization opportunities.

X-Ray daemon

(service charges may apply.)

Activated

S3 log storage

Configure the instances in your environment to upload rotated logs to Amazon S3. Learn more

Rotate logs

(standard S3 charges apply.)

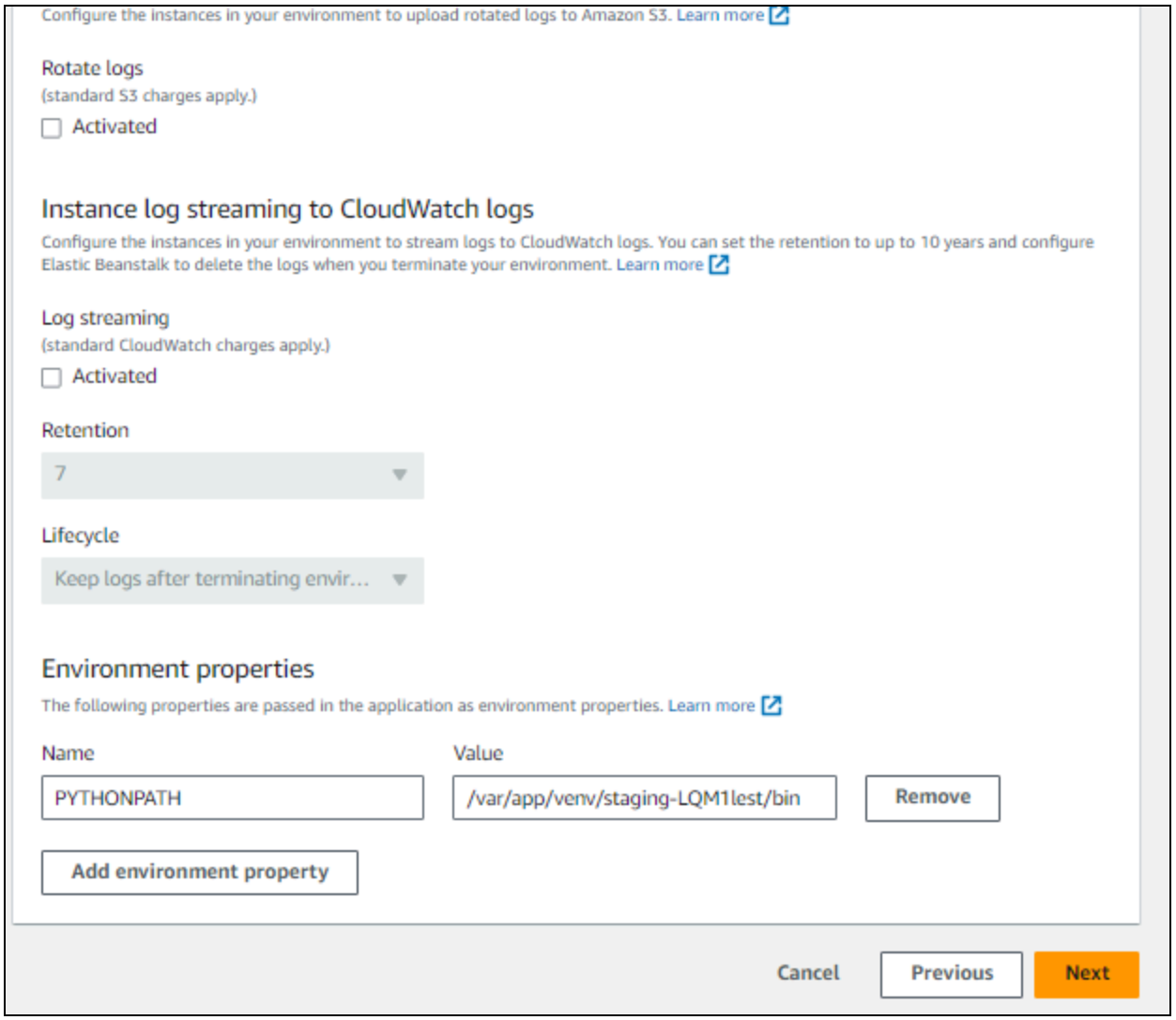
Activated

Instance log streaming to CloudWatch logs

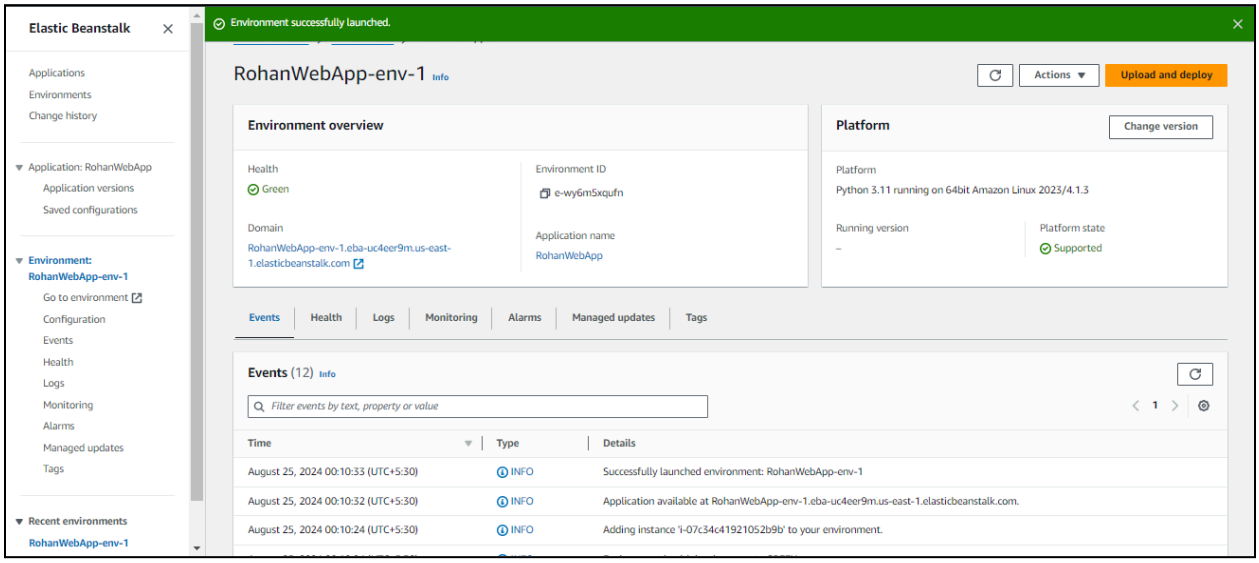
Configure the instances in your environment to stream logs to CloudWatch logs. You can set the retention to up to 10 years and configure Elastic Beanstalk to delete the logs when you terminate your environment. Learn more

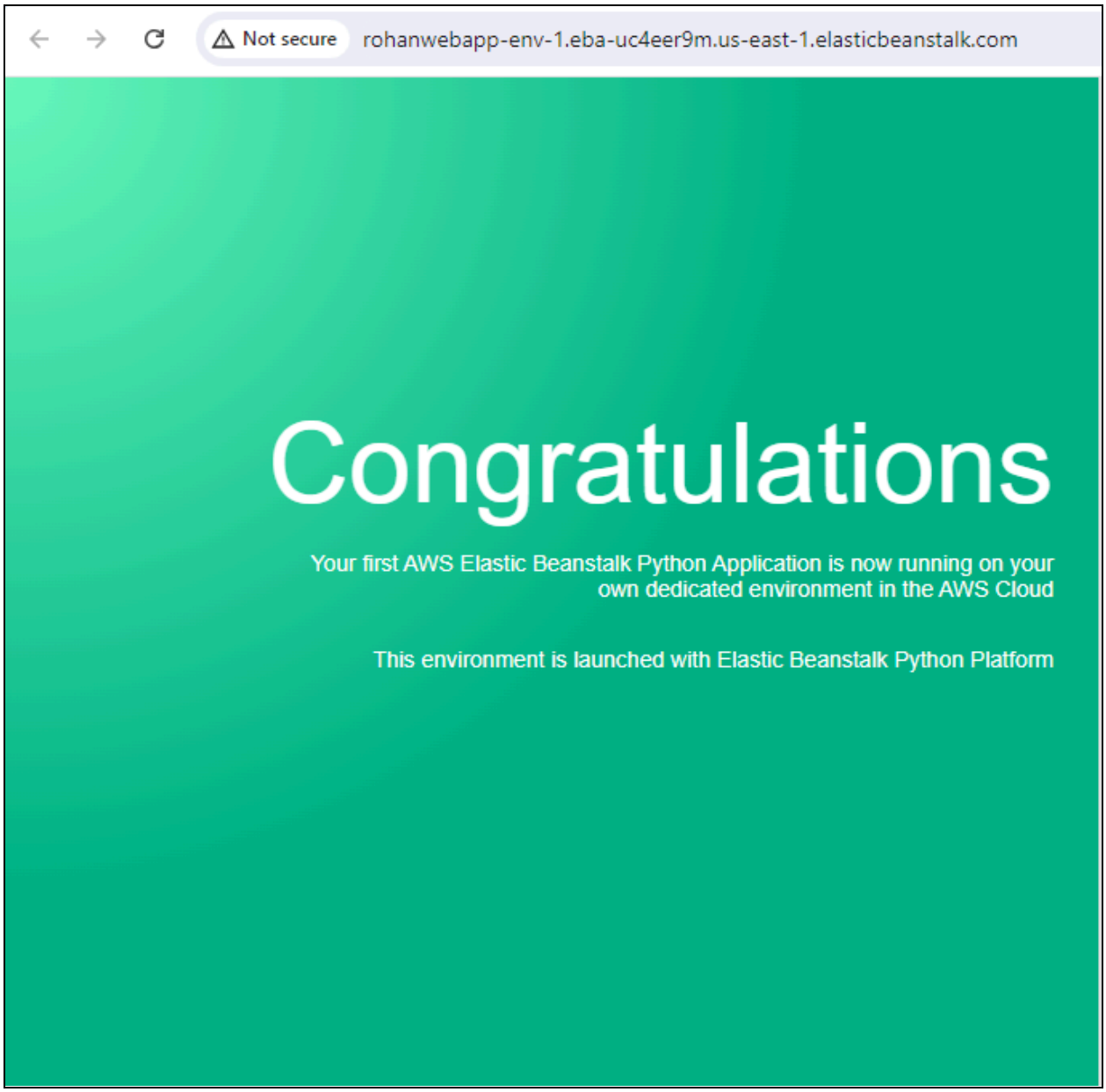
Log streaming

(standard CloudWatch charges apply.)



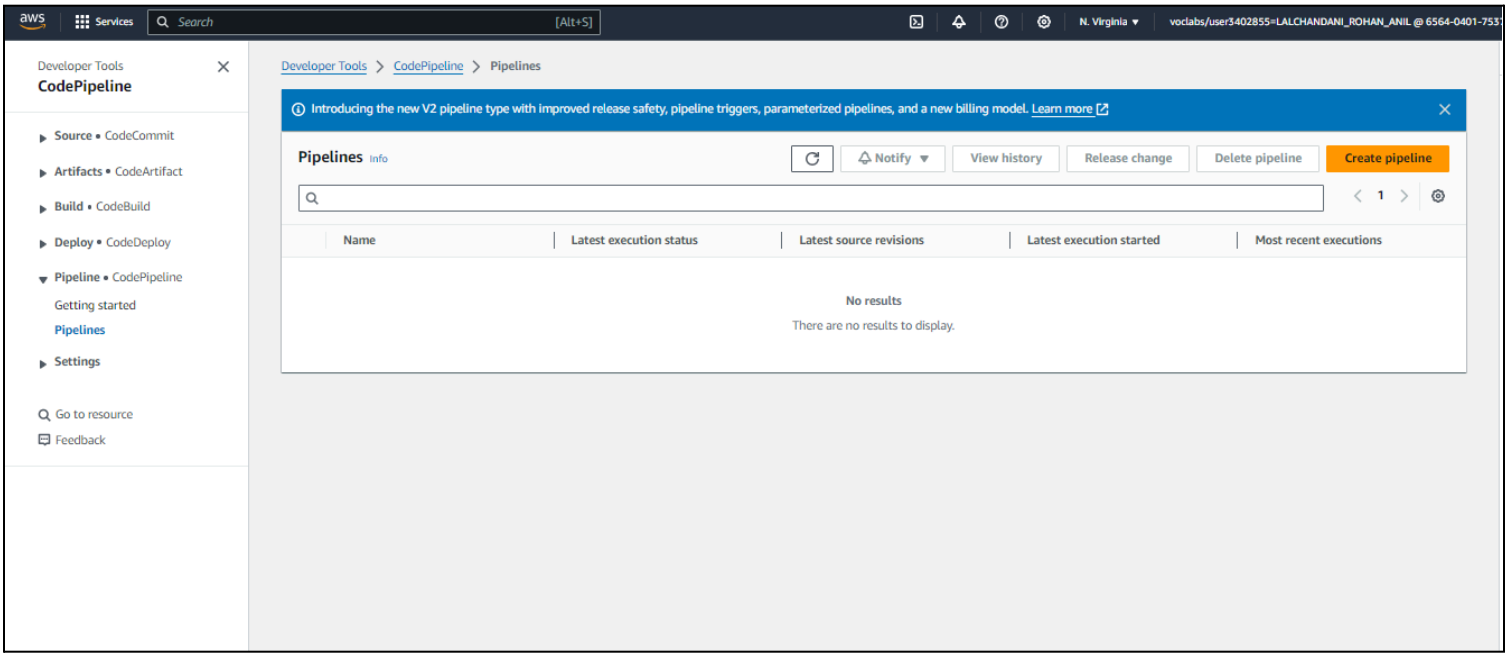
10) Environment is successfully launched.



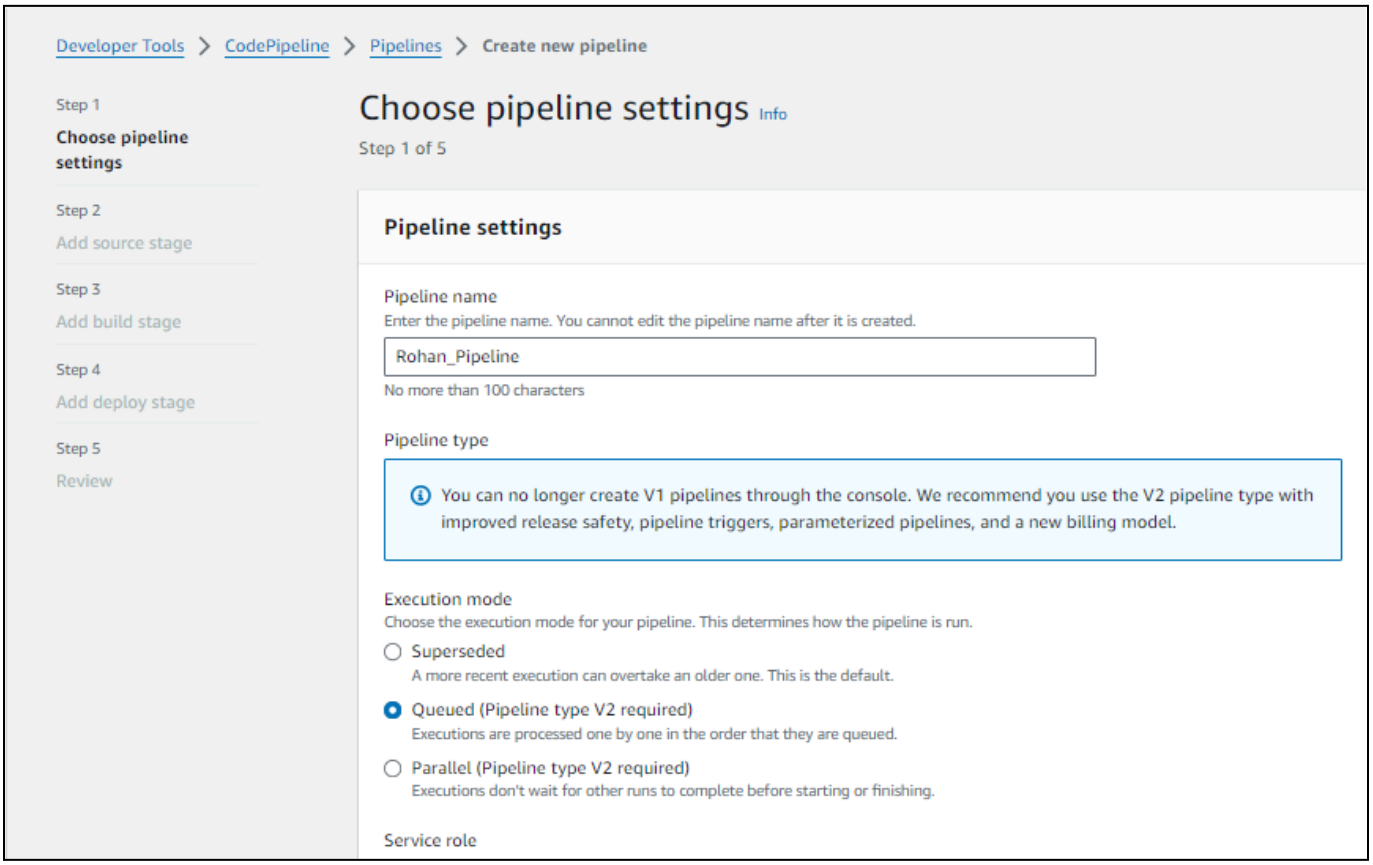

Not secure
rohanwebapp-env-1.eba-uc4eer9m.us-east-1.elasticbeanstalk.com
Congratulations
Your first AWS Elastic Beanstalk Python Application is now running on your own dedicated environment in the AWS Cloud
This environment is launched with Elastic Beanstalk Python Platform

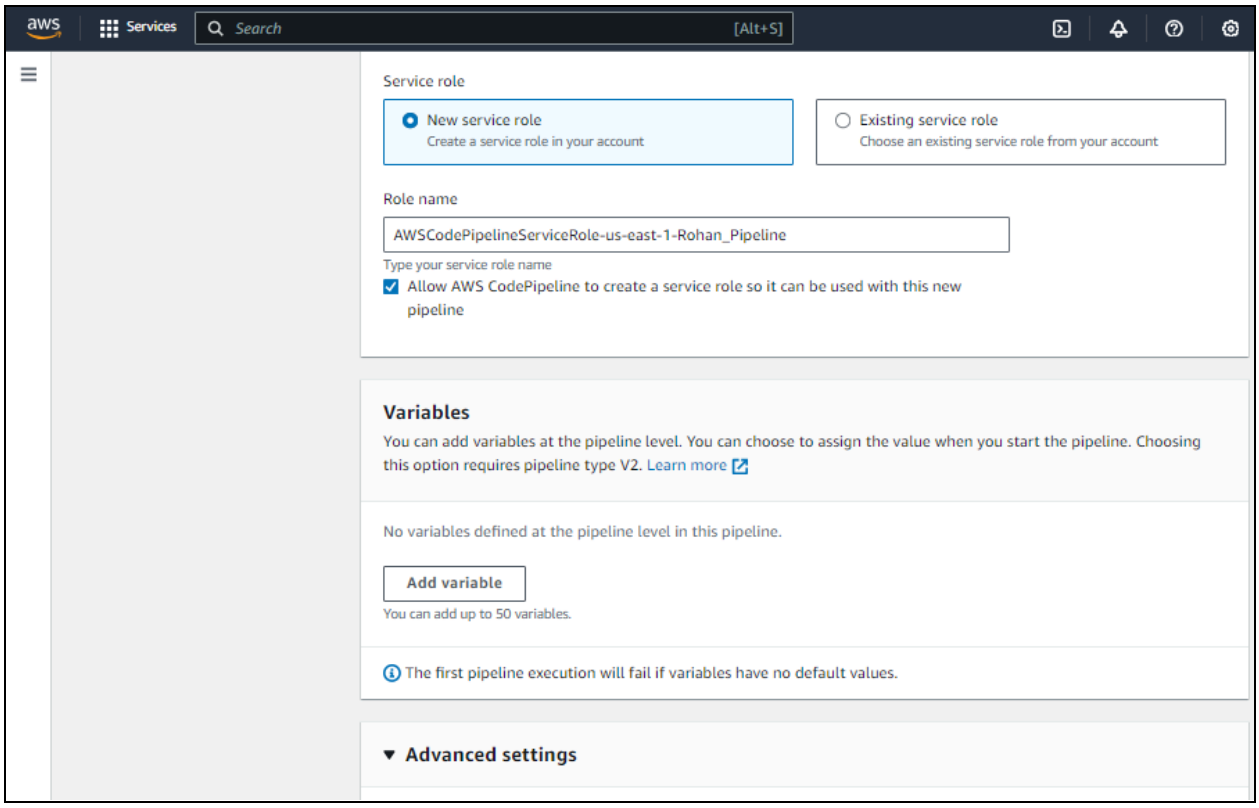
Code Deploy and Code Pipeline:

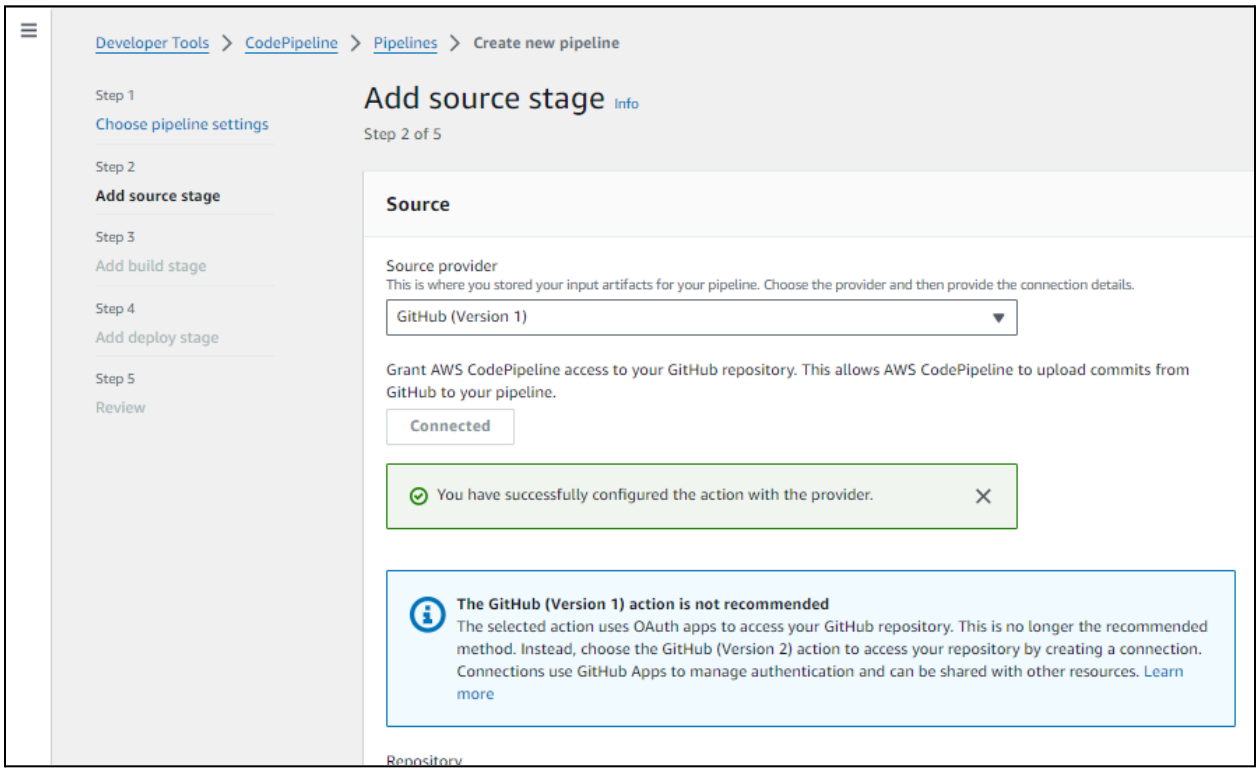
1) Search for Codepipeline in the search box and click on create pipeline.

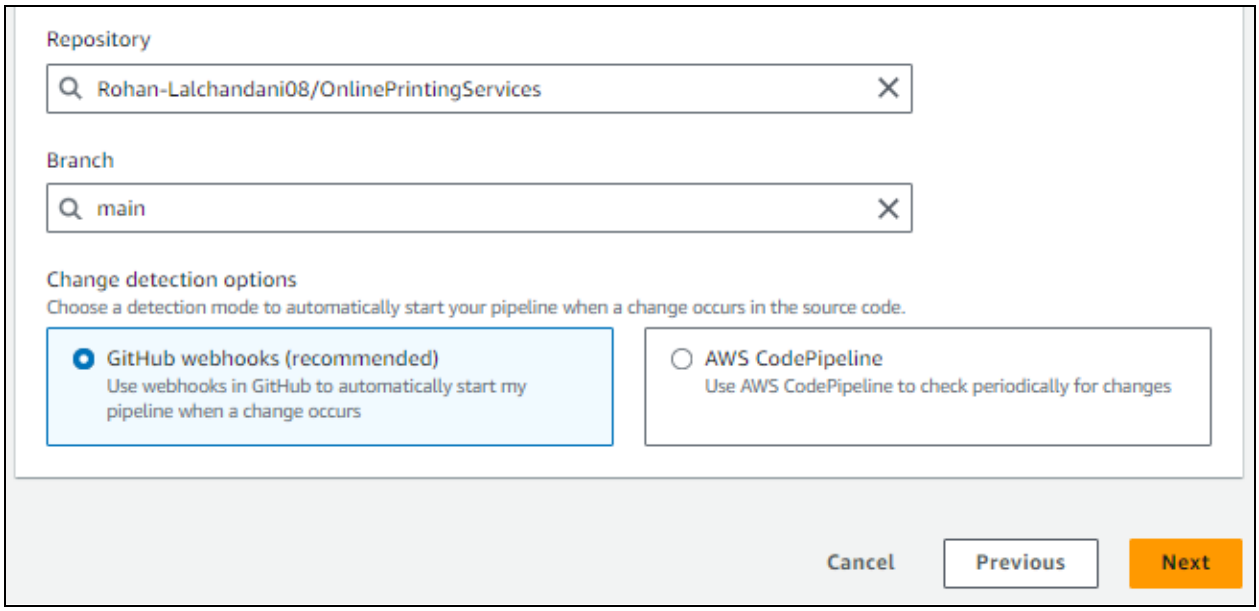


2) Configuring the pipeline.

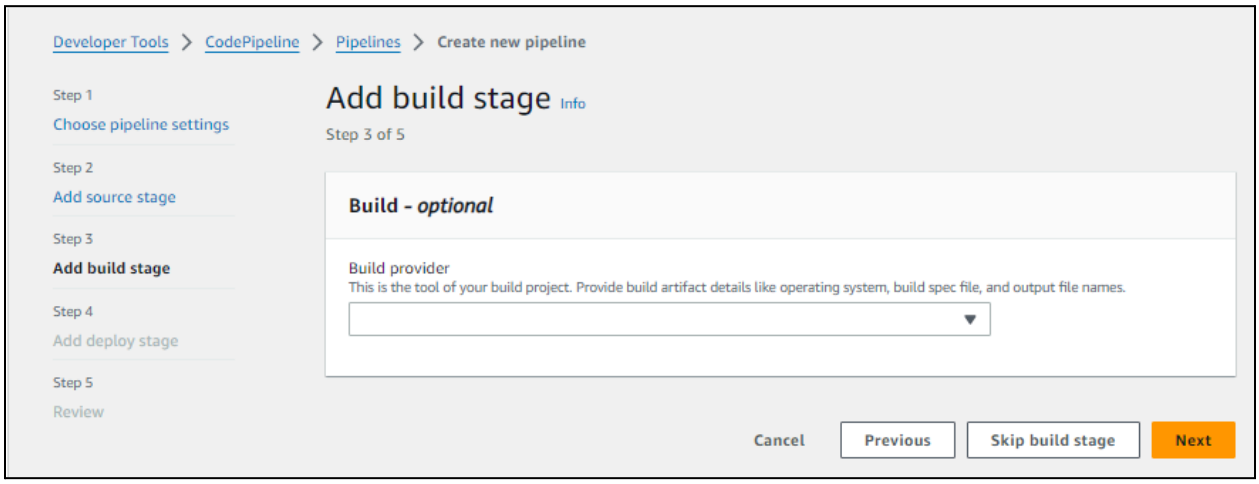



aws
Services
Search
[Alt+S]
Service role
New service role
Create a service role in your account
Existing service role
Choose an existing service role from your account
Role name
AWSCodePipelineServiceRole-us-east-1-Rohan_Pipeline
Type your service role name
Allow AWS CodePipeline to create a service role so it can be used with this new pipeline
Variables
You can add variables at the pipeline level. You can choose to assign the value when you start the pipeline. Choosing this option requires pipeline type V2. Learn more
No variables defined at the pipeline level in this pipeline.
Add variable
You can add up to 50 variables.
The first pipeline execution will fail if variables have no default values.
Advanced settings

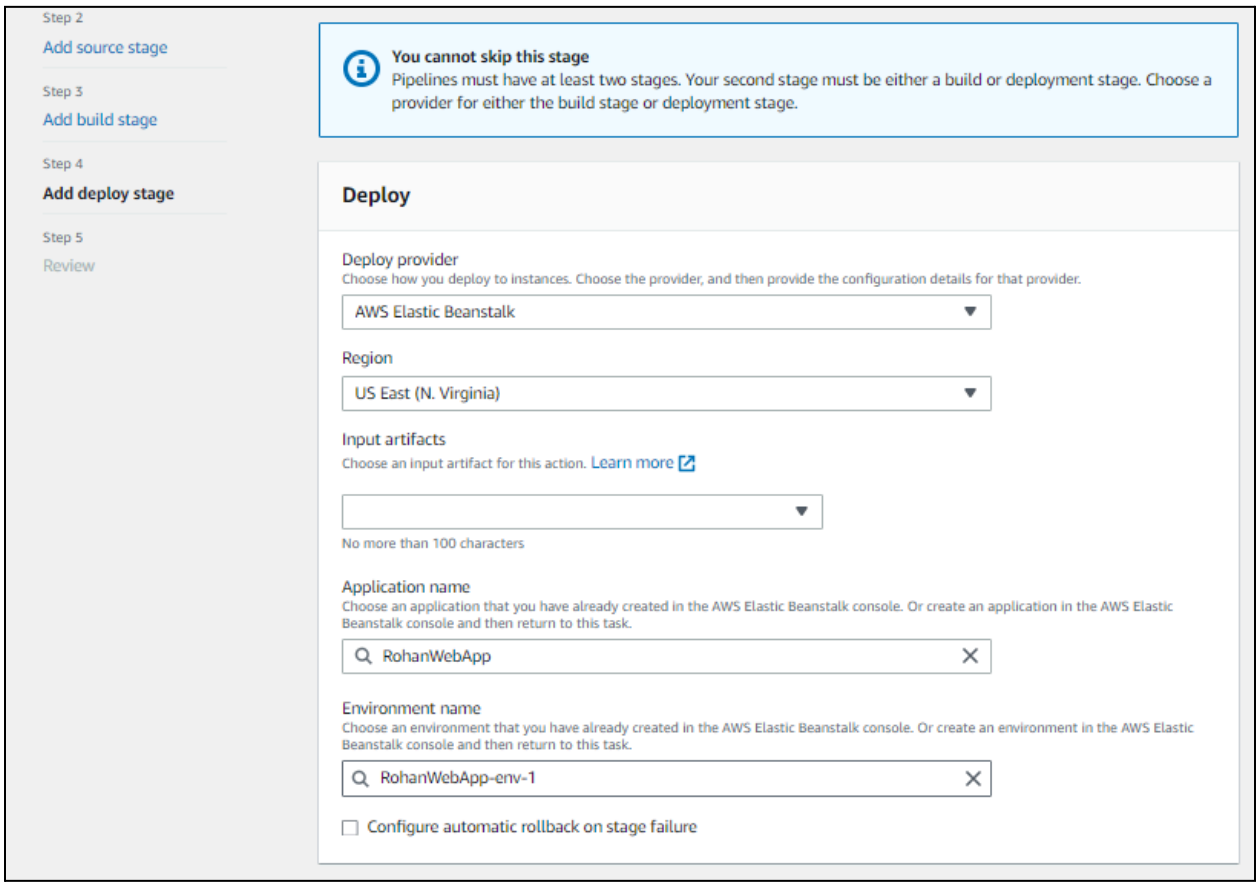

Developer Tools
CodePipeline
Pipelines
Create new pipeline
Step 1
Choose pipeline settings
Step 2
Add source stage
Step 3
Add build stage
Step 4
Add deploy stage
Step 5
Review
Add source stage
Info
Step 2 of 5
Source
Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.
GitHub (Version 1)
Grant AWS CodePipeline access to your GitHub repository. This allows AWS CodePipeline to upload commits from GitHub to your pipeline.
Connected
You have successfully configured the action with the provider.
The GitHub (Version 1) action is not recommended
The selected action uses OAuth apps to access your GitHub repository. This is no longer the recommended method. Instead, choose the GitHub (Version 2) action to access your repository by creating a connection. Connections use GitHub Apps to manage authentication and can be shared with other resources. Learn more
Repository

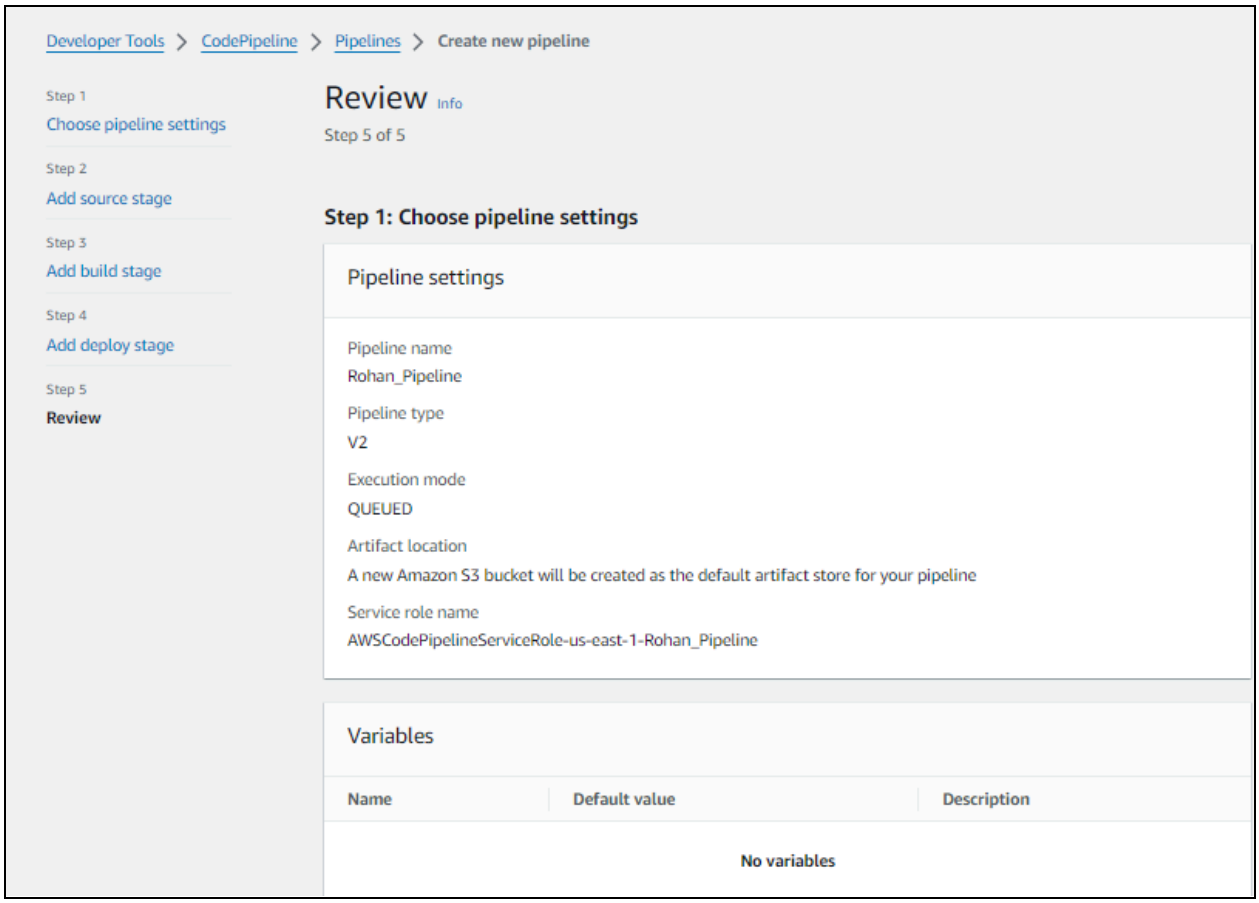


3) Skip the build stage.



4) Choose Elastic Beanstalk as the Deploy Provider.




Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose pipeline settings
Step 2
Add source stage
Step 3
Add build stage
Step 4
Add deploy stage
Step 5
Review
Review Info
Step 5 of 5
Step 1: Choose pipeline settings
Pipeline settings
Pipeline name
Rohan_Pipeline
Pipeline type
V2
Execution mode
QUEUED
Artifact location
A new Amazon S3 bucket will be created as the default artifact store for your pipeline
Service role name
AWSCodePipelineServiceRole-us-east-1-Rohan_Pipeline
Variables
Name
Default value
Description
No variables

Step 2: Add source stage

Source action provider

Source action provider
GitHub (Version 1)

PollForSourceChanges
false

Repo
OnlinePrintingServices

Owner
Rohan-Lalchandani08

Branch
main

Step 3: Add build stage

Build action provider

Build stage
No build

Step 4: Add deploy stage

Deploy action provider

Deploy action provider
AWS Elastic Beanstalk

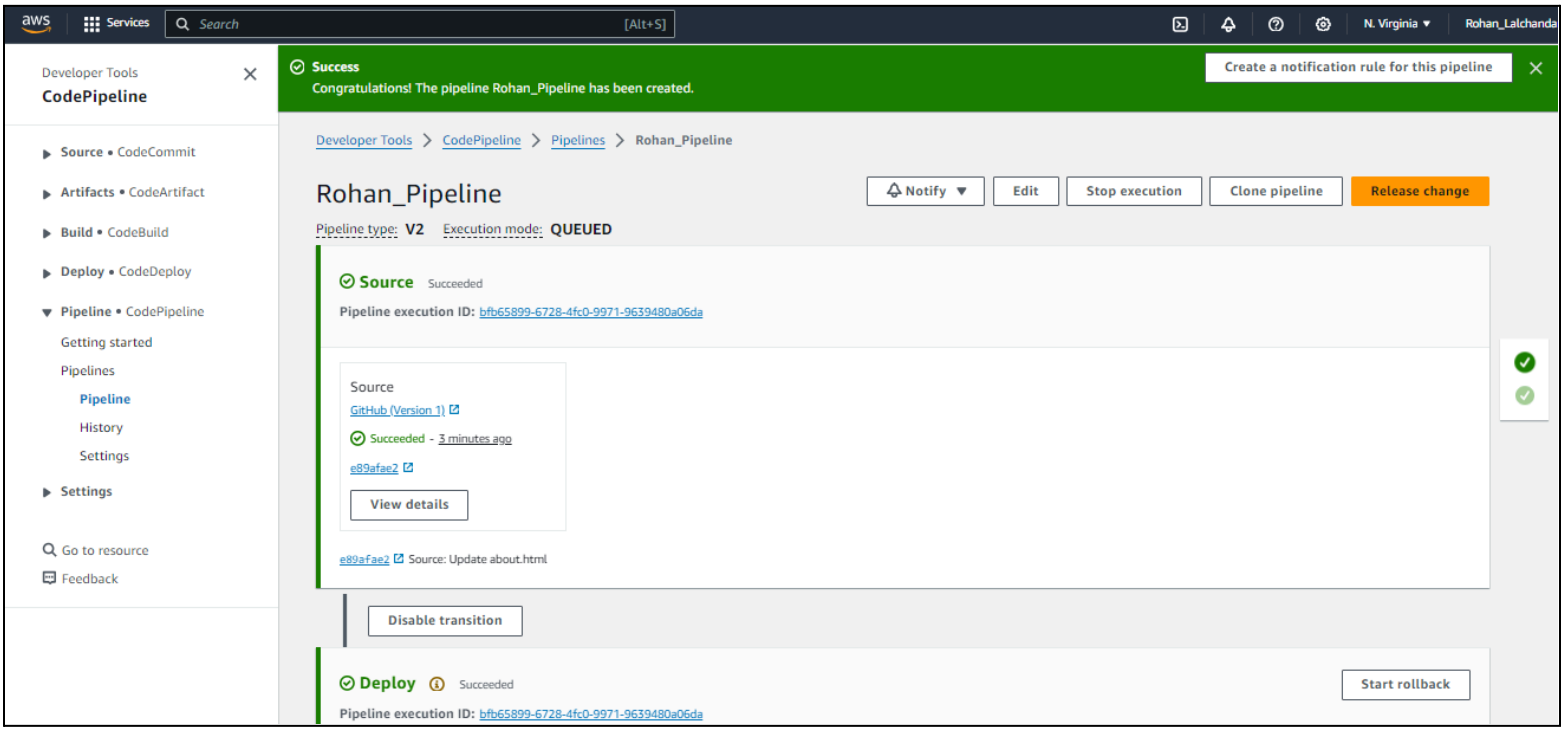
ApplicationName
RohanWebApp

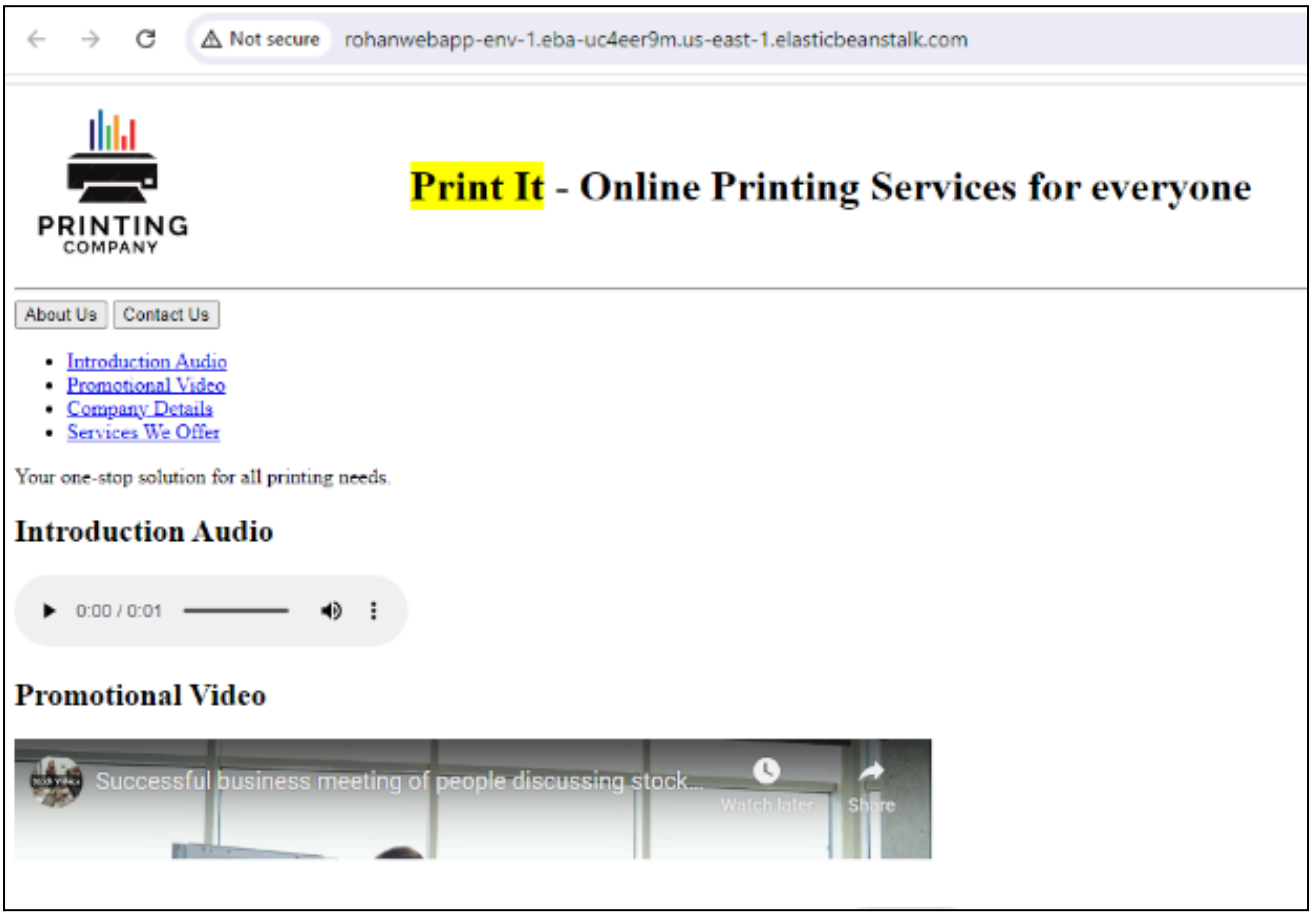
EnvironmentName
RohanWebApp-env-1

Configure automatic rollback on stage failure
Disabled

Cancel | Previous | Create pipeline

5) Pipeline has been created.





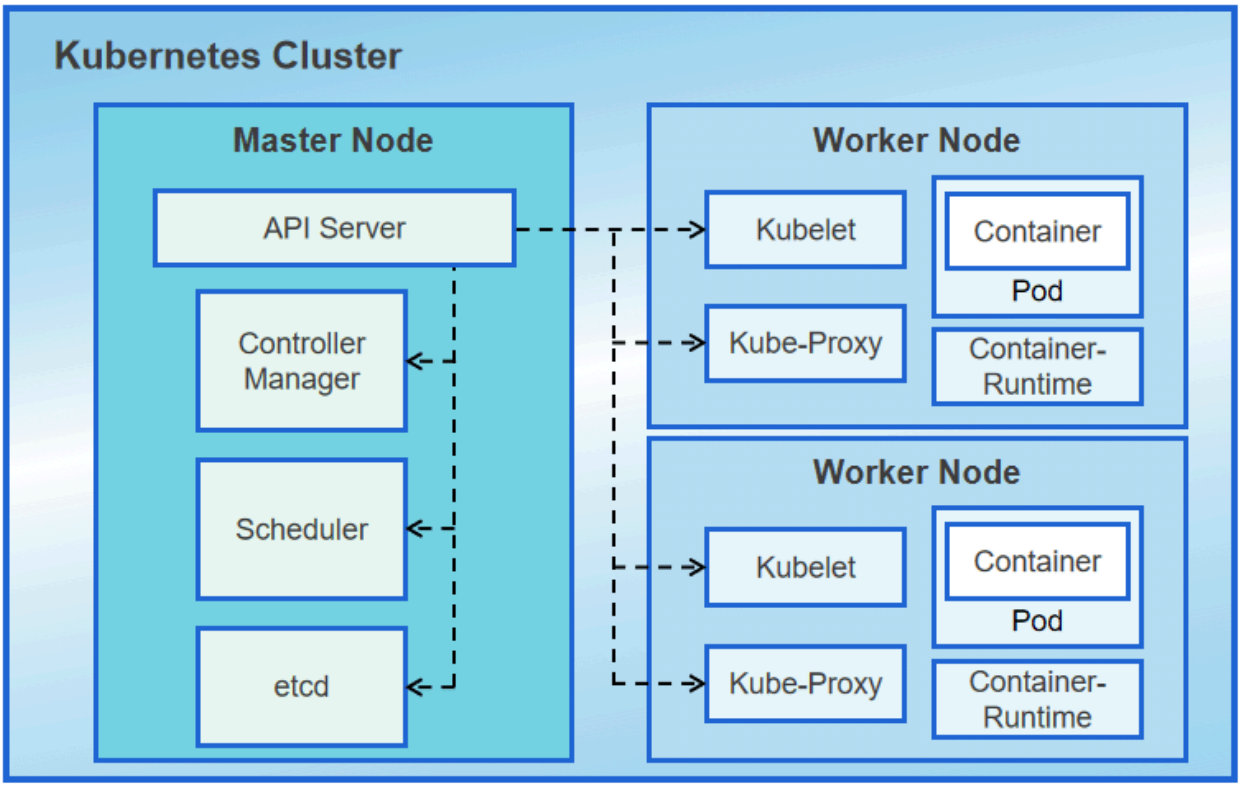
Adv. Devops Experiment no. 3

Name: Rohan Lalchandani
Class: D15A Roll no.: 25

Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Theory:

Kubernetes cluster architecture is designed to manage containerized applications efficiently. It consists of two main components: the **Control Plane** and the **Worker Nodes**.



1. Control Plane

The control plane is responsible for managing the overall state of the cluster. It makes global decisions about the cluster, such as scheduling applications, and detects/responds to cluster events (like a container failure).

- **API Server**: The entry point for all administrative tasks in the cluster. It exposes the Kubernetes API, acting as the front end of the control plane.
- **etcd**: A distributed key-value store that stores all the cluster's data. It is a highly reliable store for all cluster state and configuration.
- **Controller Manager**: Monitors the cluster state and performs routine tasks like handling node failures, maintaining the correct number of replicas for pods, and balancing the load across the cluster.
- **Scheduler**: Assigns newly created pods to nodes based on resource availability, affinity rules, and other policies.

2. Worker Nodes

Worker nodes run the containerized applications (pods). Each node has a set of components that communicate with the control plane to receive and execute tasks.

- **Kubelet**: The agent that runs on every node in the cluster. It ensures that containers are running in pods and communicates with the API server to get the desired state of the node.
- **Container Runtime**: The software responsible for running containers, such as Docker, containerd, or CRI-O. It manages the lifecycle of containers on the node.
- **Kube Proxy**: Ensures networking rules are applied to allow communication between the different components in the cluster, handling services and load balancing.

3. Cluster Networking

Kubernetes provides an abstracted networking layer for communication between pods. Key features include:

- **Pod-to-Pod Communication**: Every pod in a cluster can communicate with every other pod without using NAT (Network Address Translation).

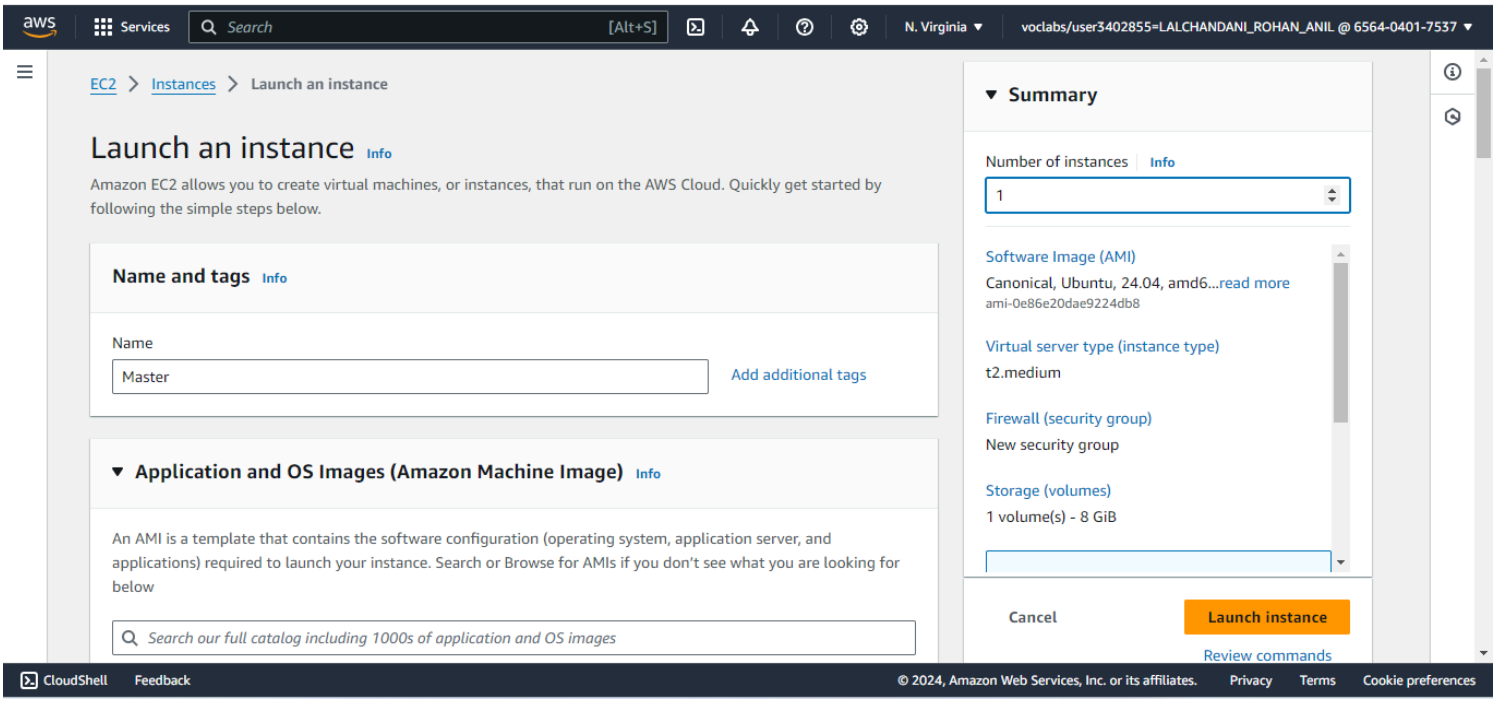
- **Service Abstraction**: Services provide a stable endpoint to access a group of pods. This decouples the frontend from backend pods, offering load balancing and failure recovery.

4. Storage

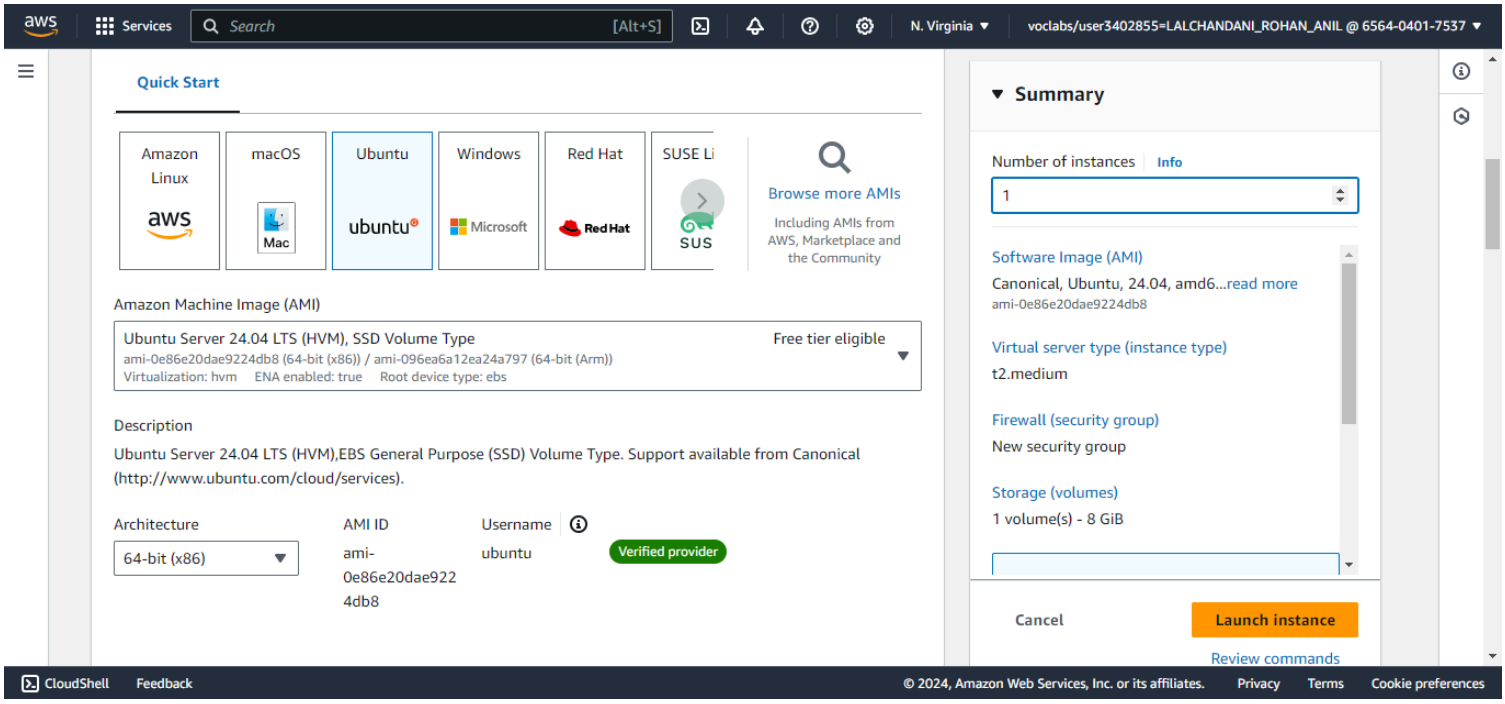
Kubernetes uses persistent storage systems like persistent volumes (PVs) and persistent volume claims (PVCs) to manage data storage needs.

Implementation:

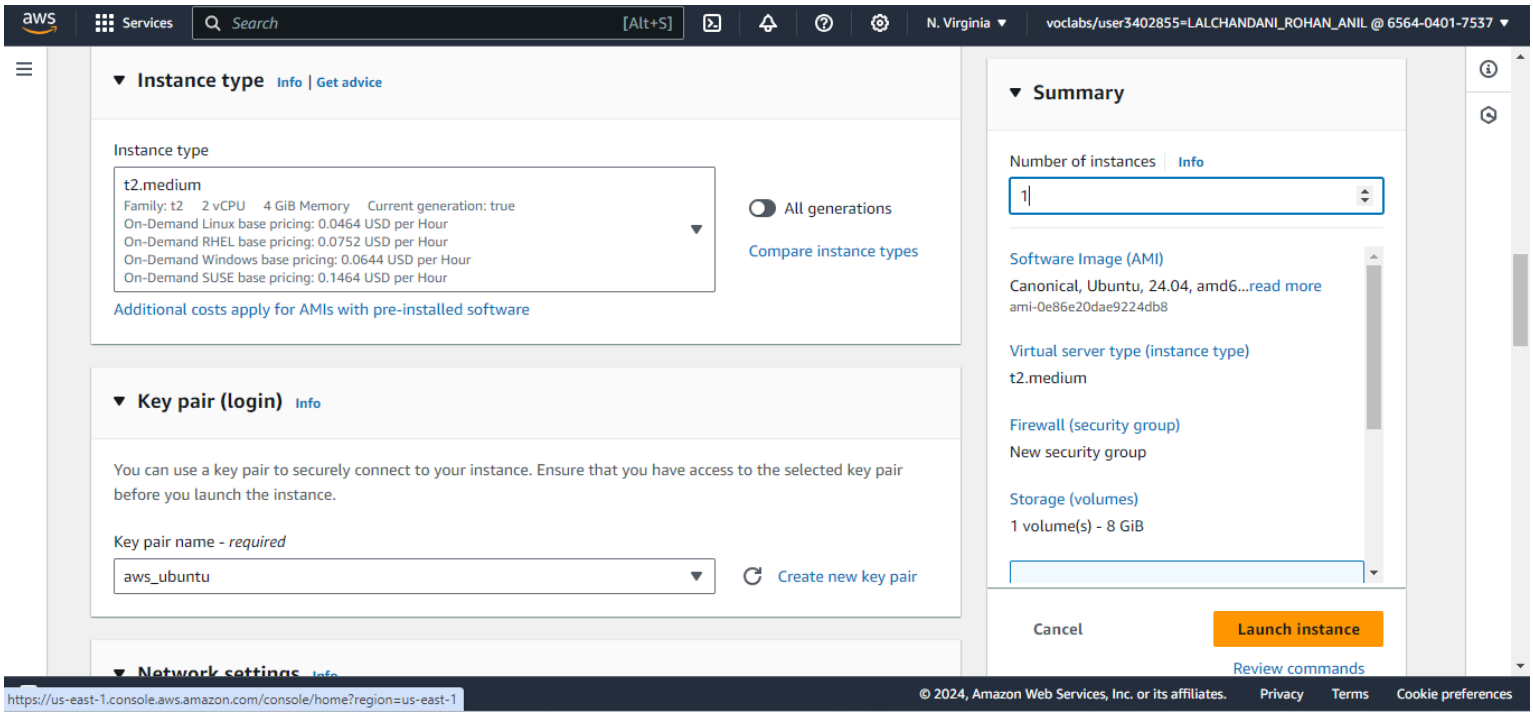
1. Create three EC2 Ubuntu Instances - Master, Worker 1 and Worker 2.



2. Select the following AMI image.

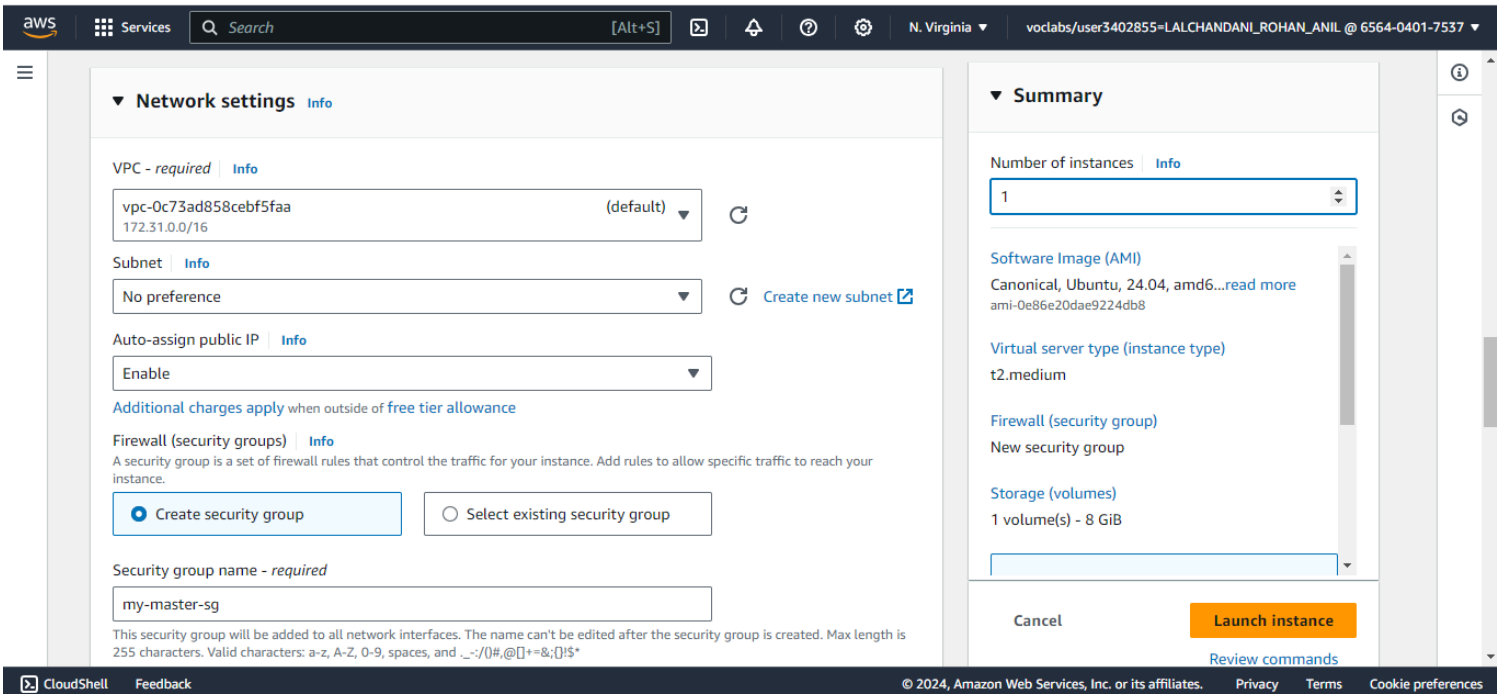


3. Select t2.medium in instance type.
Why?
t2.medium has 2 vCPUs and 4 GB of RAM, which allows it to run more pods and handle larger or more resource-intensive containers compared to **t2.micro** (1 vCPU, 1 GB of RAM).

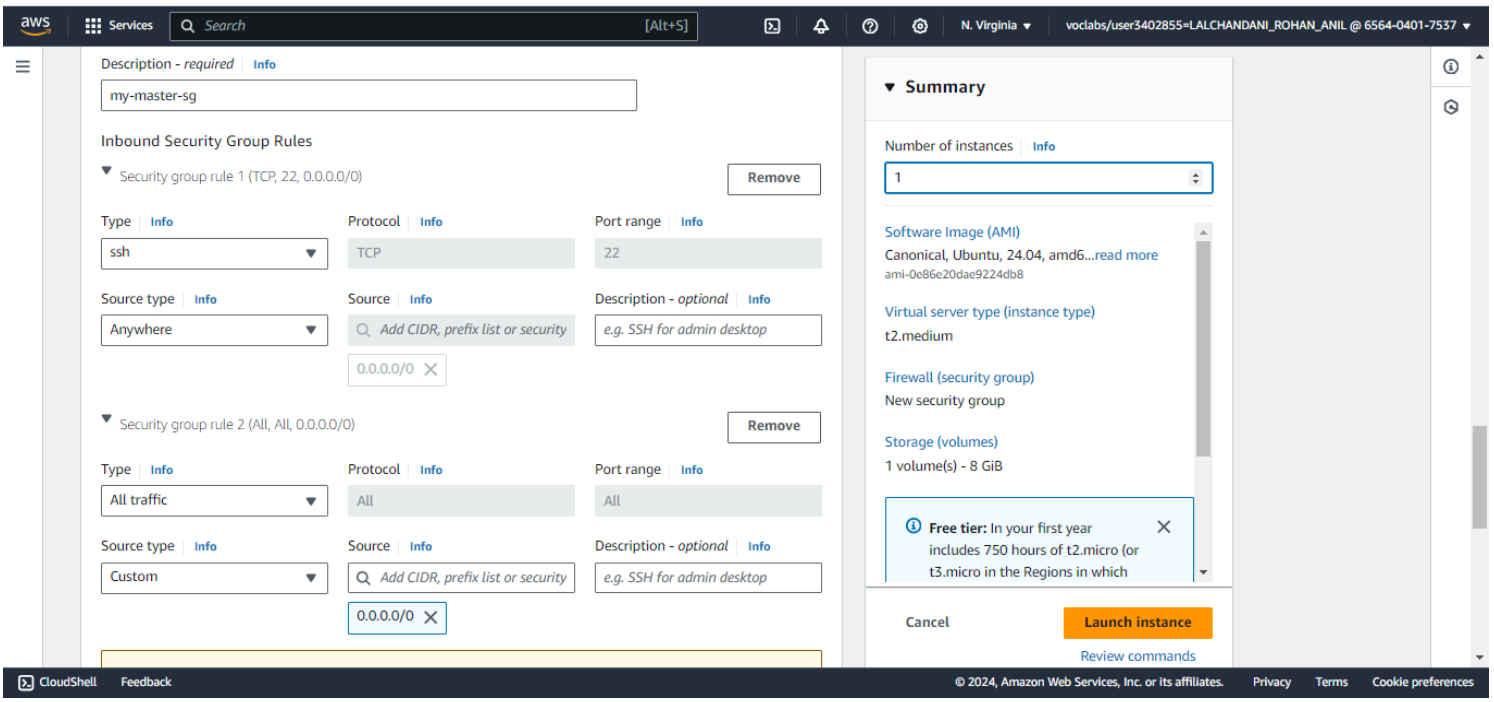


4. Create security group

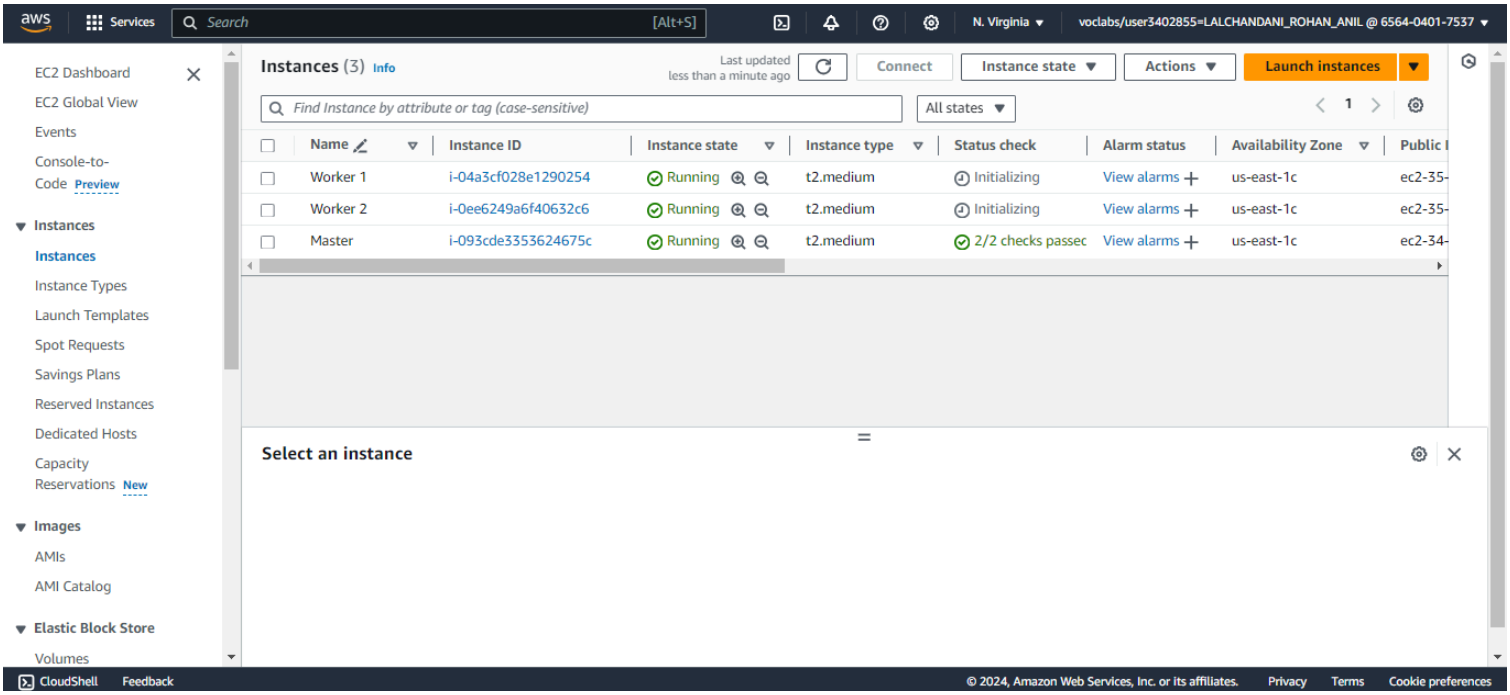
In it type the group name and description as my-master-sg



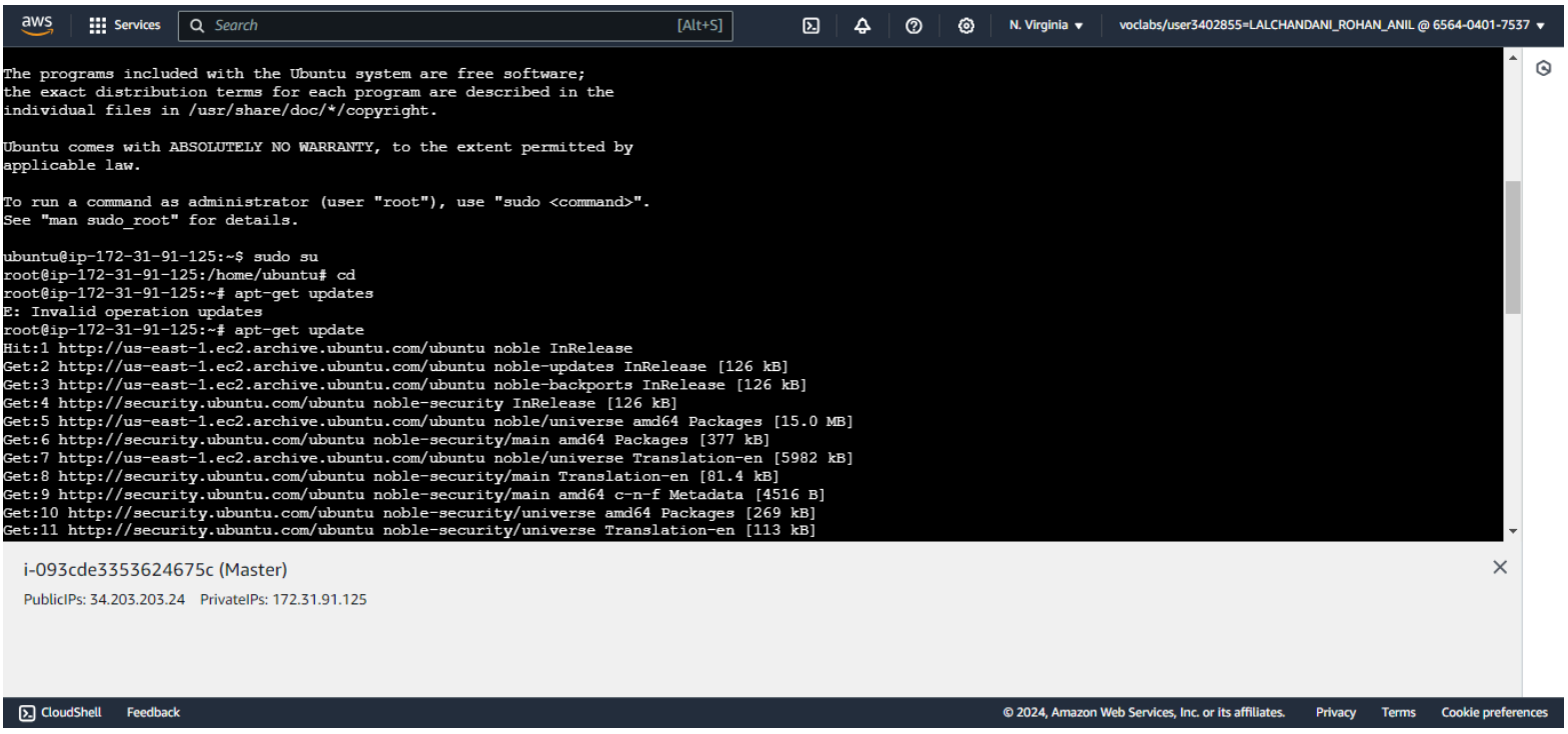
5. Edit the inbound rules and add a new rule to accept “All traffic” as shown below.



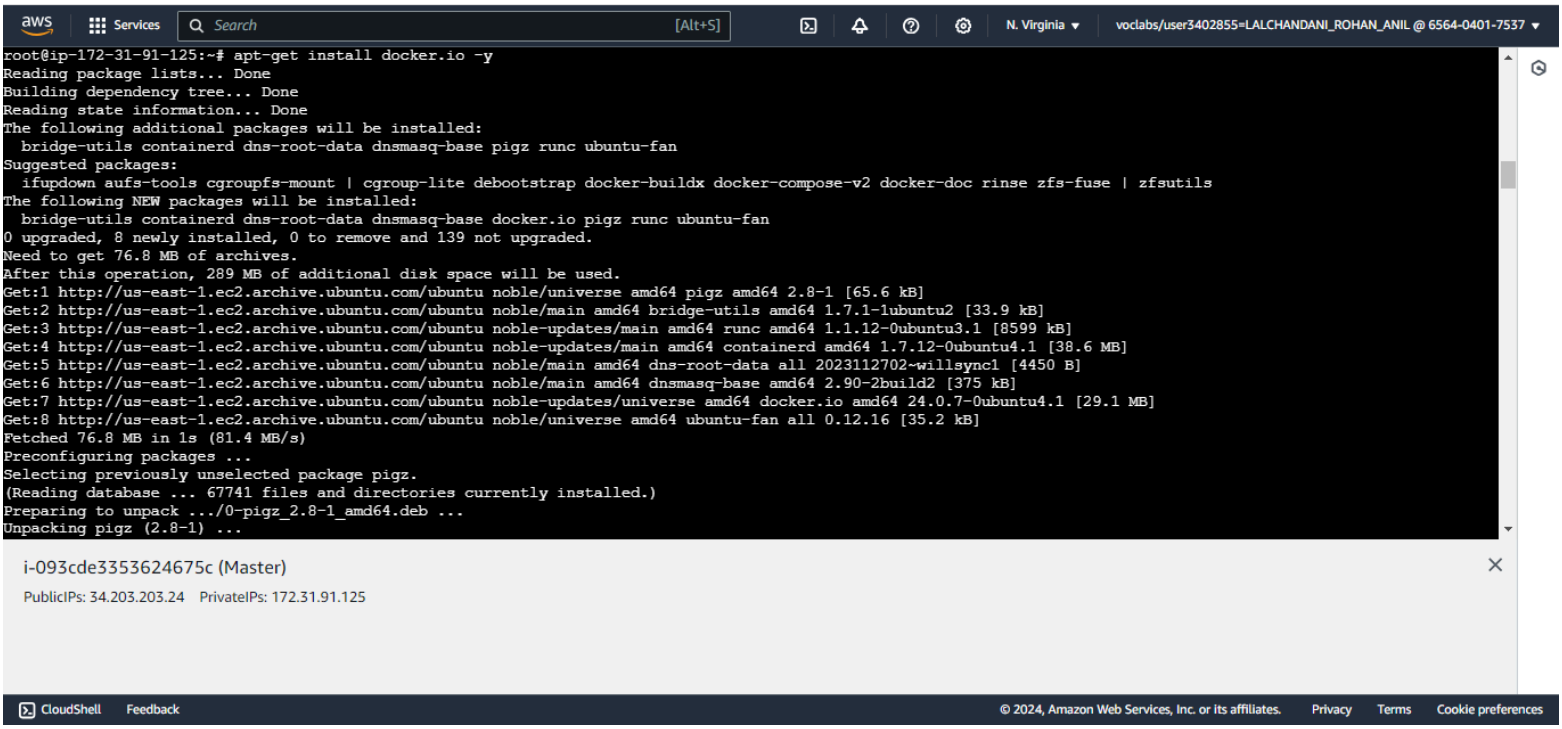
6. Create Worker 1 and Worker 2 also with same settings and select the existing security group of “my-master-sg” in them. Connect all the instances.

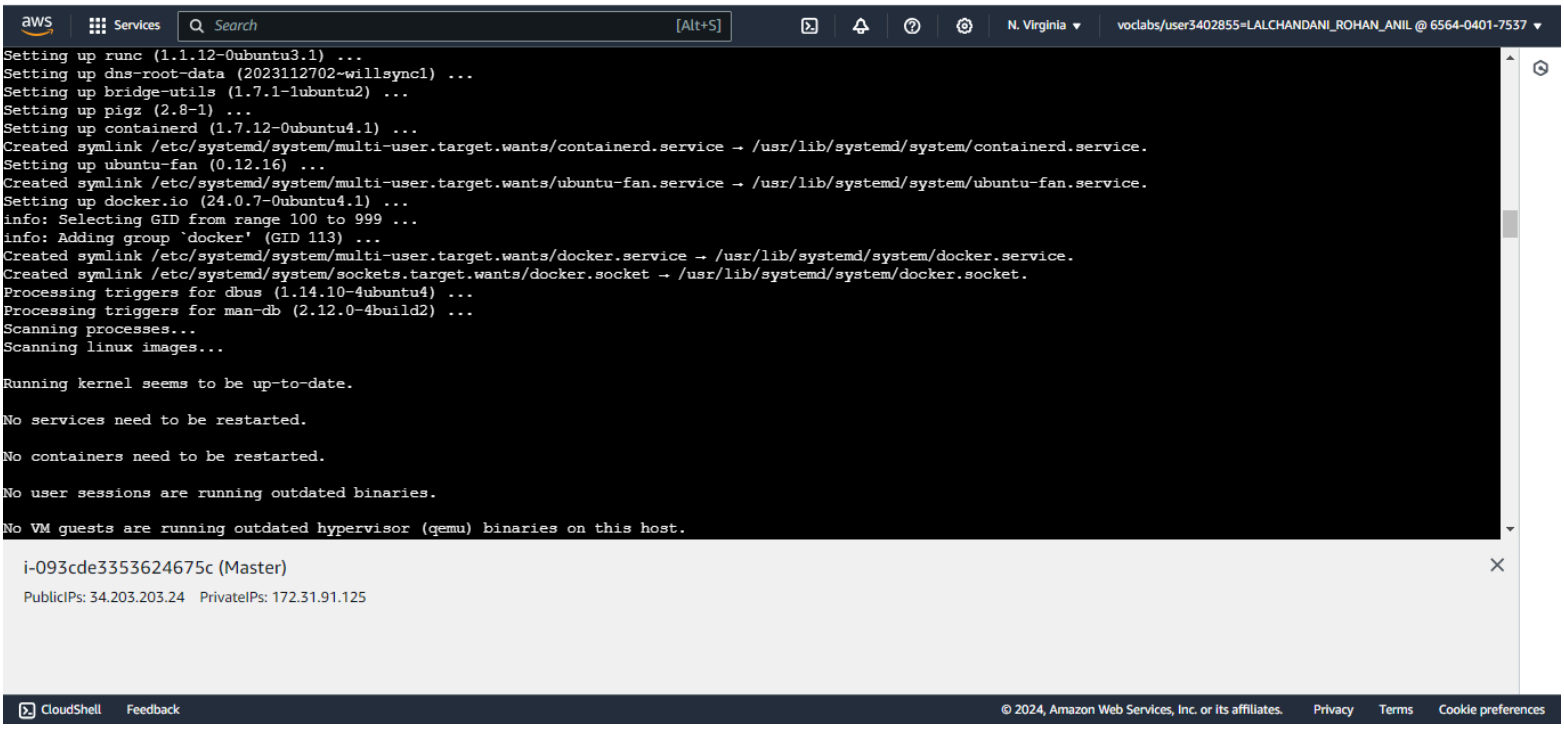


7. Now run “sudo su” to execute commands as root user.
“cd” to change directory
then “apt-get update”

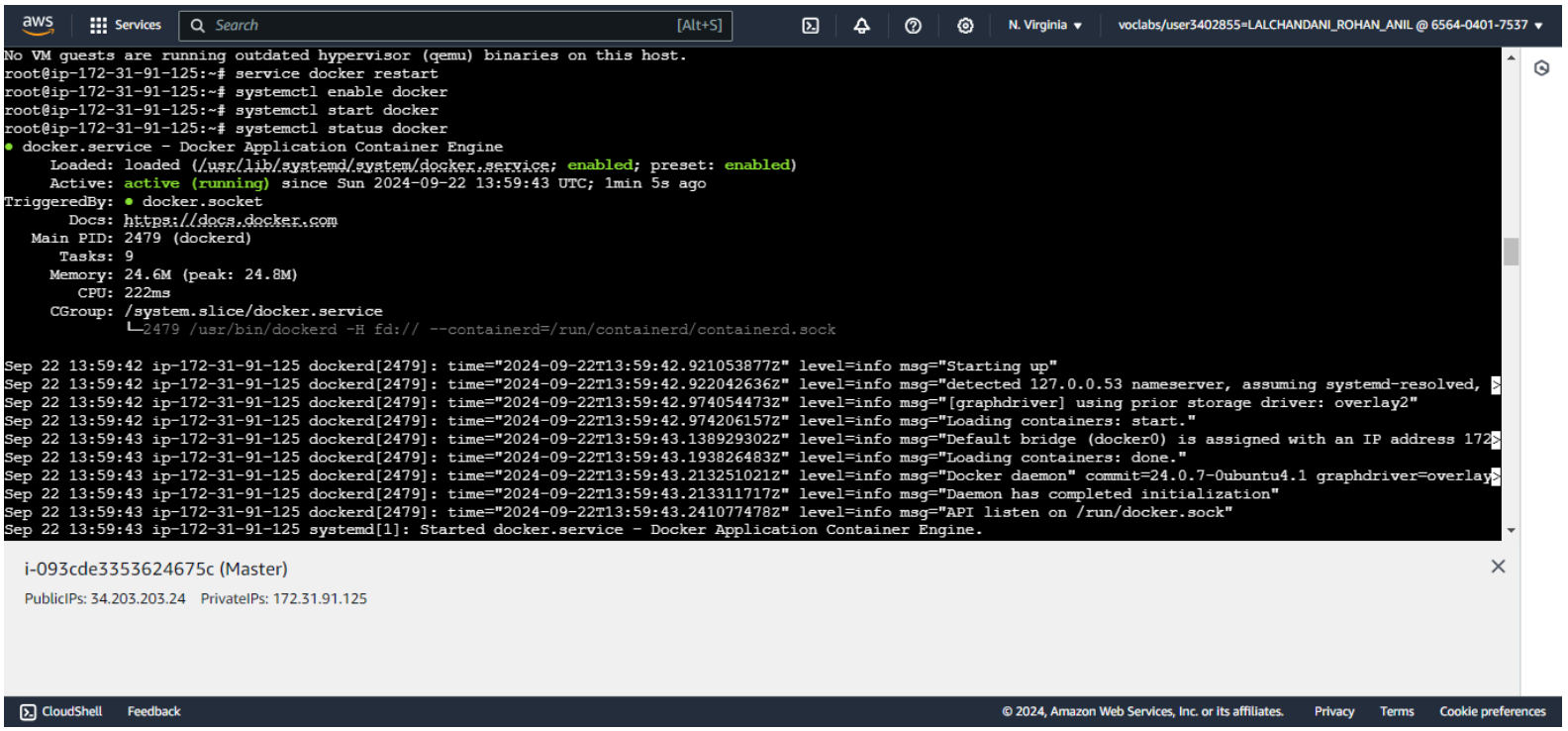


8. Install docker by “apt-get install docker.io -y”



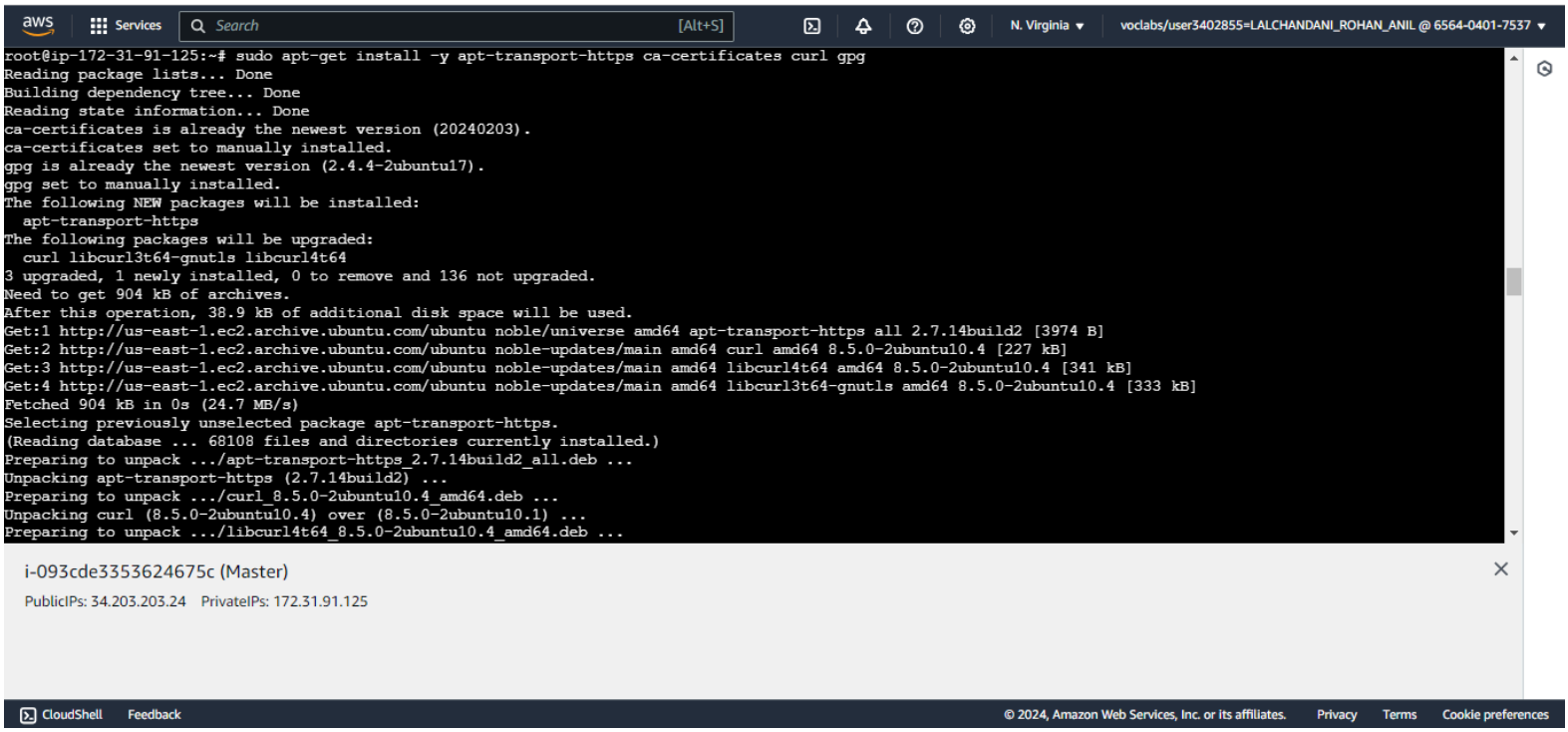


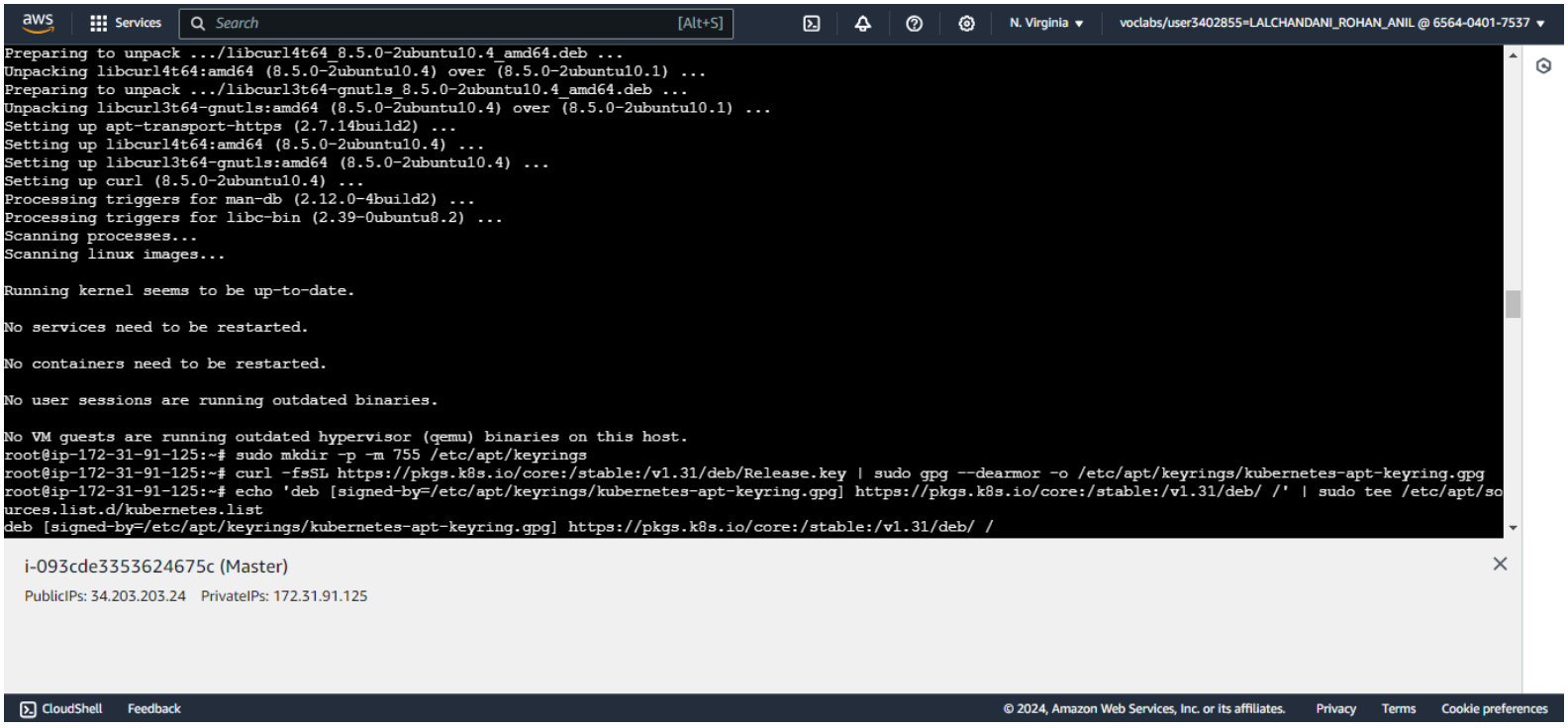
9. enable, start and check status of docker if it’s active.

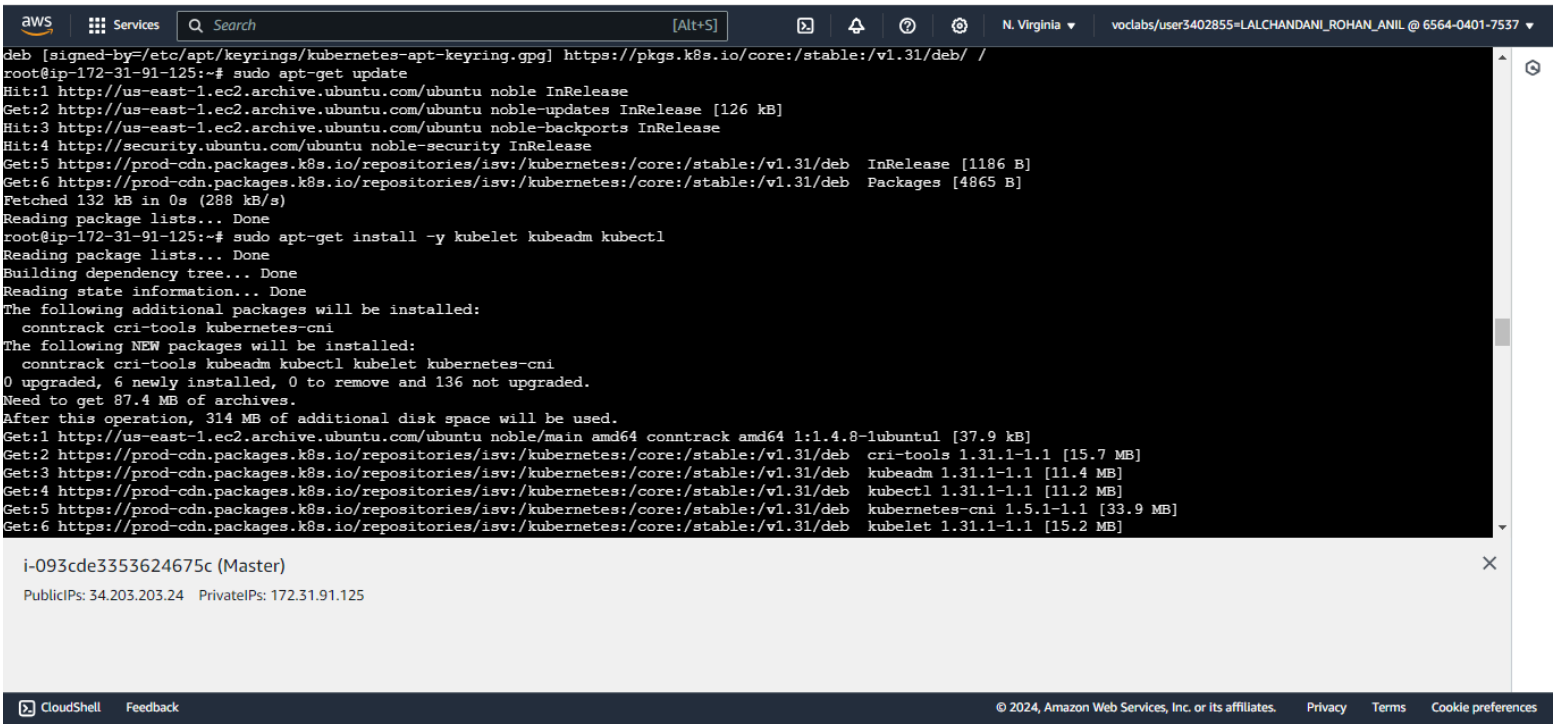


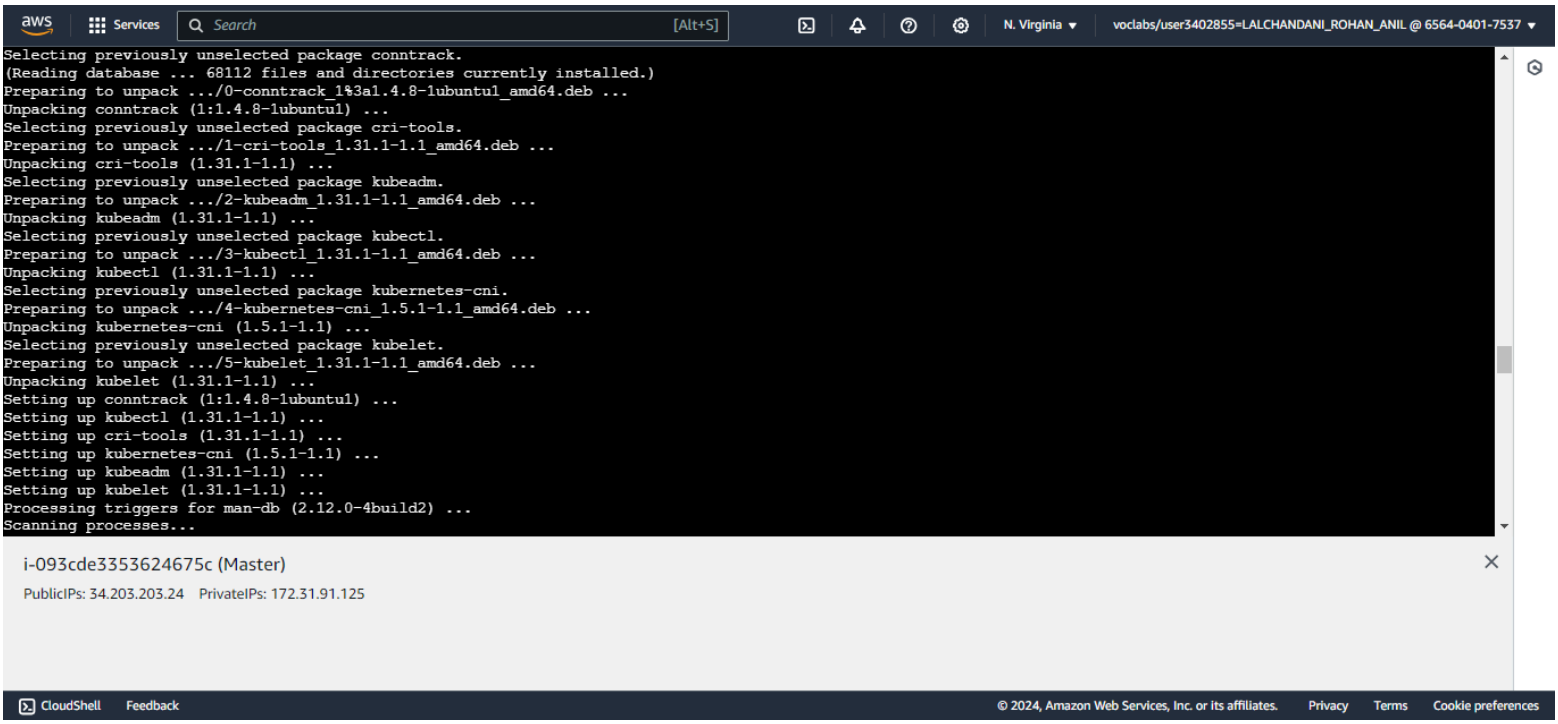
10. Now we have to install kubernetes, for that we will refer the documentation at https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/
Got to the “Debian-based distributions” and follow the instructions below to install kubeadm

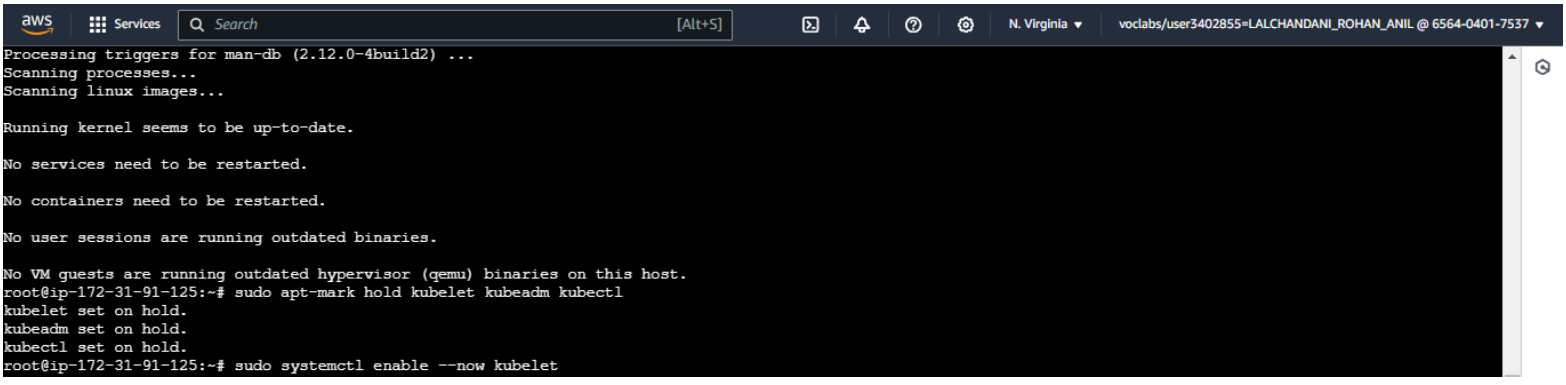
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
sudo mkdir -p -m 755 /etc/apt/keyrings
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
sudo systemctl enable --now kubelet
systemctl restart kubelet


Services
Search
[Alt+S]
N. Virginia
voclabs/user3402855=LALCHANDANI_ROHAN_ANIL @ 6564-0401-7537
root@ip-172-31-91-125:~# sudo apt-get install -y apt-transport-https ca-certificates curl gpg
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
gpg is already the newest version (2.4.4-2ubuntu17).
gpg set to manually installed.
The following NEW packages will be installed:
apt-transport-https
The following packages will be upgraded:
curl libcurl3t64-gnutls libcurl4t64
3 upgraded, 1 newly installed, 0 to remove and 136 not upgraded.
Need to get 904 kB of archives.
After this operation, 38.9 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 apt-transport-https all 2.7.14build2 [3974 B]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 curl amd64 8.5.0-2ubuntu10.4 [227 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libcurl4t64 amd64 8.5.0-2ubuntu10.4 [341 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libcurl3t64-gnutls amd64 8.5.0-2ubuntu10.4 [333 kB]
Fetched 904 kB in 0s (24.7 MB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 68108 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.7.14build2_all.deb ...
Unpacking apt-transport-https (2.7.14build2) ...
Preparing to unpack .../curl_8.5.0-2ubuntu10.4_amd64.deb ...
Unpacking curl (8.5.0-2ubuntu10.4) over (8.5.0-2ubuntu10.1) ...
Preparing to unpack .../libcurl4t64_8.5.0-2ubuntu10.4_amd64.deb ...
i-093cde3353624675c (Master)
PublicIPs: 34.203.203.24 PrivateIPs: 172.31.91.125
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

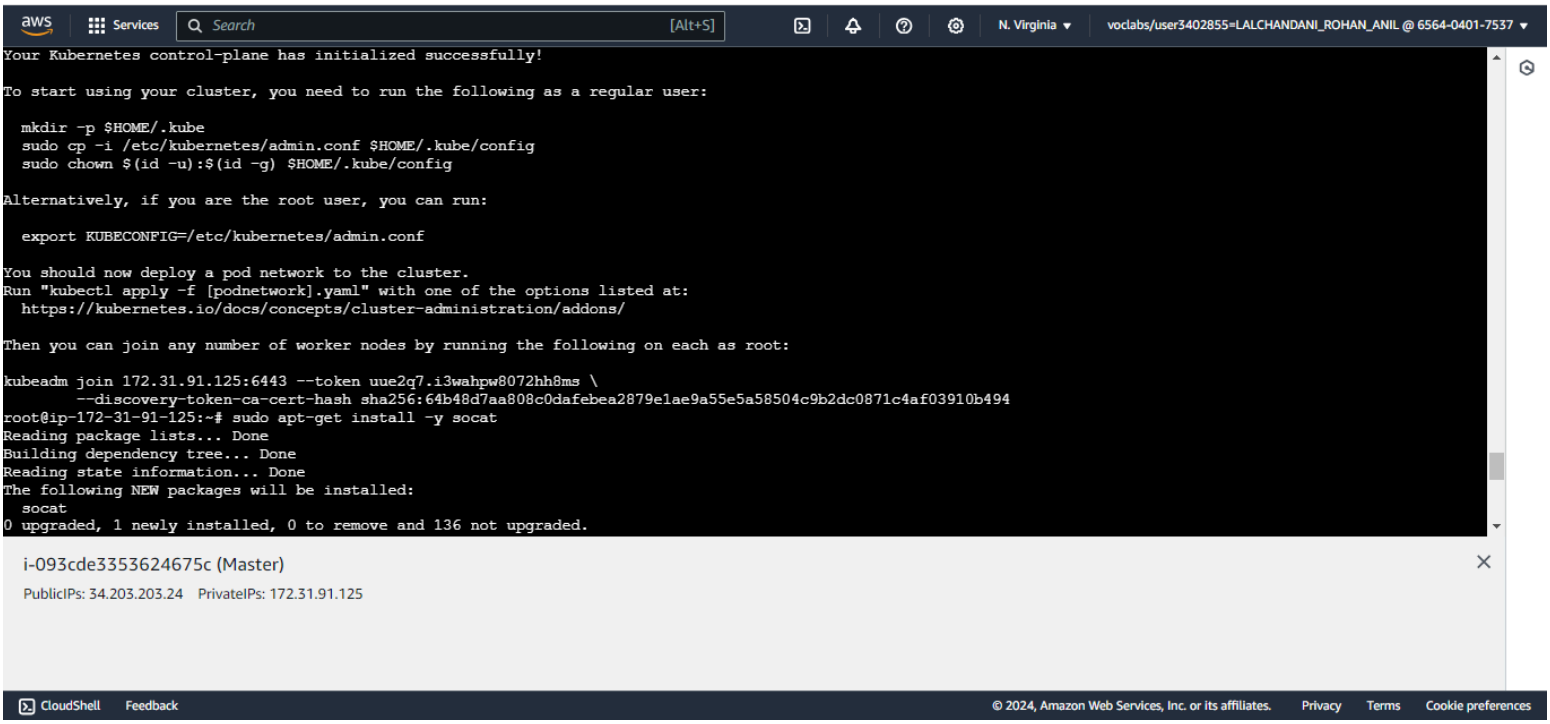

Services
Search
[Alt+S]
N. Virginia
voclabs/user3402855=LALCHANDANI_ROHAN_ANIL @ 6564-0401-7537
Preparing to unpack .../libcurl4t64_8.5.0-2ubuntu10.4_amd64.deb ...
Unpacking libcurl4t64:amd64 (8.5.0-2ubuntu10.4) over (8.5.0-2ubuntu10.1) ...
Preparing to unpack .../libcurl3t64-gnutls_8.5.0-2ubuntu10.4_amd64.deb ...
Unpacking libcurl3t64-gnutls:amd64 (8.5.0-2ubuntu10.4) over (8.5.0-2ubuntu10.1) ...
Setting up apt-transport-https (2.7.14build2) ...
Setting up libcurl4t64:amd64 (8.5.0-2ubuntu10.4) ...
Setting up libcurl3t64-gnutls:amd64 (8.5.0-2ubuntu10.4) ...
Setting up curl (8.5.0-2ubuntu10.4) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
Scanning processes...
Scanning linux images...
Running kernel seems to be up-to-date.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.
No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-91-125:~# sudo mkdir -p -m 755 /etc/apt/keyrings
root@ip-172-31-91-125:~# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
root@ip-172-31-91-125:~# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/so
urces.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
i-093cde3353624675c (Master)
PublicIPs: 34.203.203.24 PrivateIPs: 172.31.91.125
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3402855=LALCHANDANI_ROHAN_ANIL @ 6564-0401-7537
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
root@ip-172-31-91-125:~# sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb InRelease [1186 B]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb Packages [4865 B]
Fetched 132 kB in 0s (288 kB/s)
Reading package lists... Done
root@ip-172-31-91-125:~# sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 136 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb cri-tools 1.31.1-1.1 [15.7 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb kubeadm 1.31.1-1.1 [11.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb kubectl 1.31.1-1.1 [11.2 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb kubernetes-cni 1.5.1-1.1 [33.9 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb kubelet 1.31.1-1.1 [15.2 MB]
i-093cde3353624675c (Master)
PublicIPs: 34.203.203.24 PrivateIPs: 172.31.91.125
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

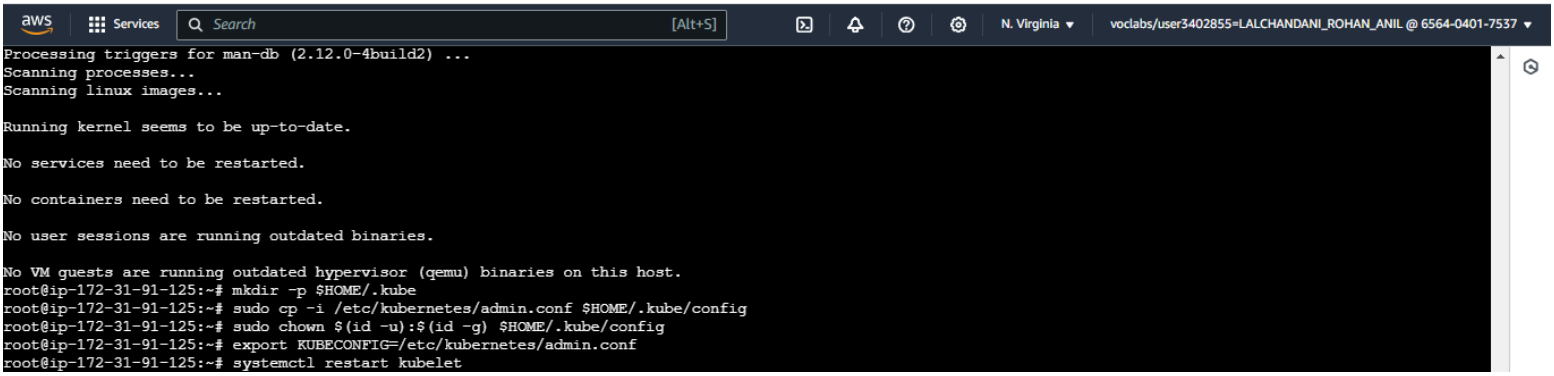

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3402855=LALCHANDANI_ROHAN_ANIL @ 6564-0401-7537
Selecting previously unselected package conntrack.
(Reading database ... 68112 files and directories currently installed.)
Preparing to unpack .../0-conntrack_1%3a1.4.8-1ubuntu1_amd64.deb ...
Unpacking conntrack (1:1.4.8-1ubuntu1) ...
Selecting previously unselected package cri-tools.
Preparing to unpack .../1-cri-tools_1.31.1-1.1_amd64.deb ...
Unpacking cri-tools (1.31.1-1.1) ...
Selecting previously unselected package kubeadm.
Preparing to unpack .../2-kubeadm_1.31.1-1.1_amd64.deb ...
Unpacking kubeadm (1.31.1-1.1) ...
Selecting previously unselected package kubectl.
Preparing to unpack .../3-kubectl_1.31.1-1.1_amd64.deb ...
Unpacking kubectl (1.31.1-1.1) ...
Selecting previously unselected package kubernetes-cni.
Preparing to unpack .../4-kubernetes-cni_1.5.1-1.1_amd64.deb ...
Unpacking kubernetes-cni (1.5.1-1.1) ...
Selecting previously unselected package kubelet.
Preparing to unpack .../5-kubelet_1.31.1-1.1_amd64.deb ...
Unpacking kubelet (1.31.1-1.1) ...
Setting up conntrack (1:1.4.8-1ubuntu1) ...
Setting up kubectl (1.31.1-1.1) ...
Setting up cri-tools (1.31.1-1.1) ...
Setting up kubernetes-cni (1.5.1-1.1) ...
Setting up kubeadm (1.31.1-1.1) ...
Setting up kubelet (1.31.1-1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
i-093cde3353624675c (Master)
PublicIPs: 34.203.203.24 PrivateIPs: 172.31.91.125
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3402855=LALCHANDANI_ROHAN_ANIL @ 6564-0401-7537
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...
Running kernel seems to be up-to-date.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.
No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-91-125:~# sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@ip-172-31-91-125:~# sudo systemctl enable --now kubelet

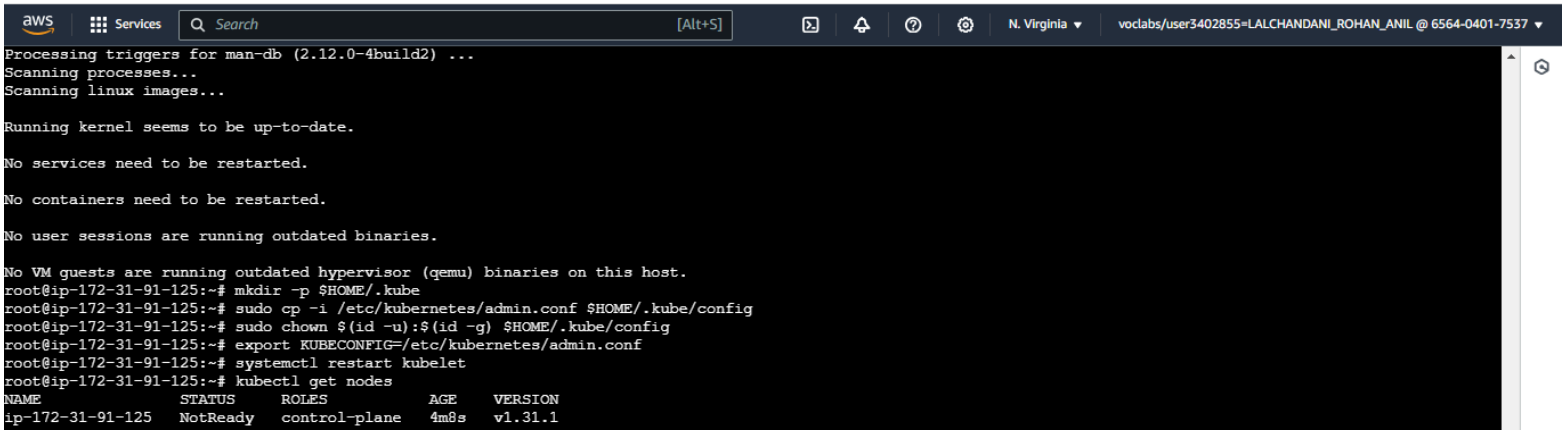
Now only in **Master Node**, execute the “kubeadm init” to initialize the cluster.



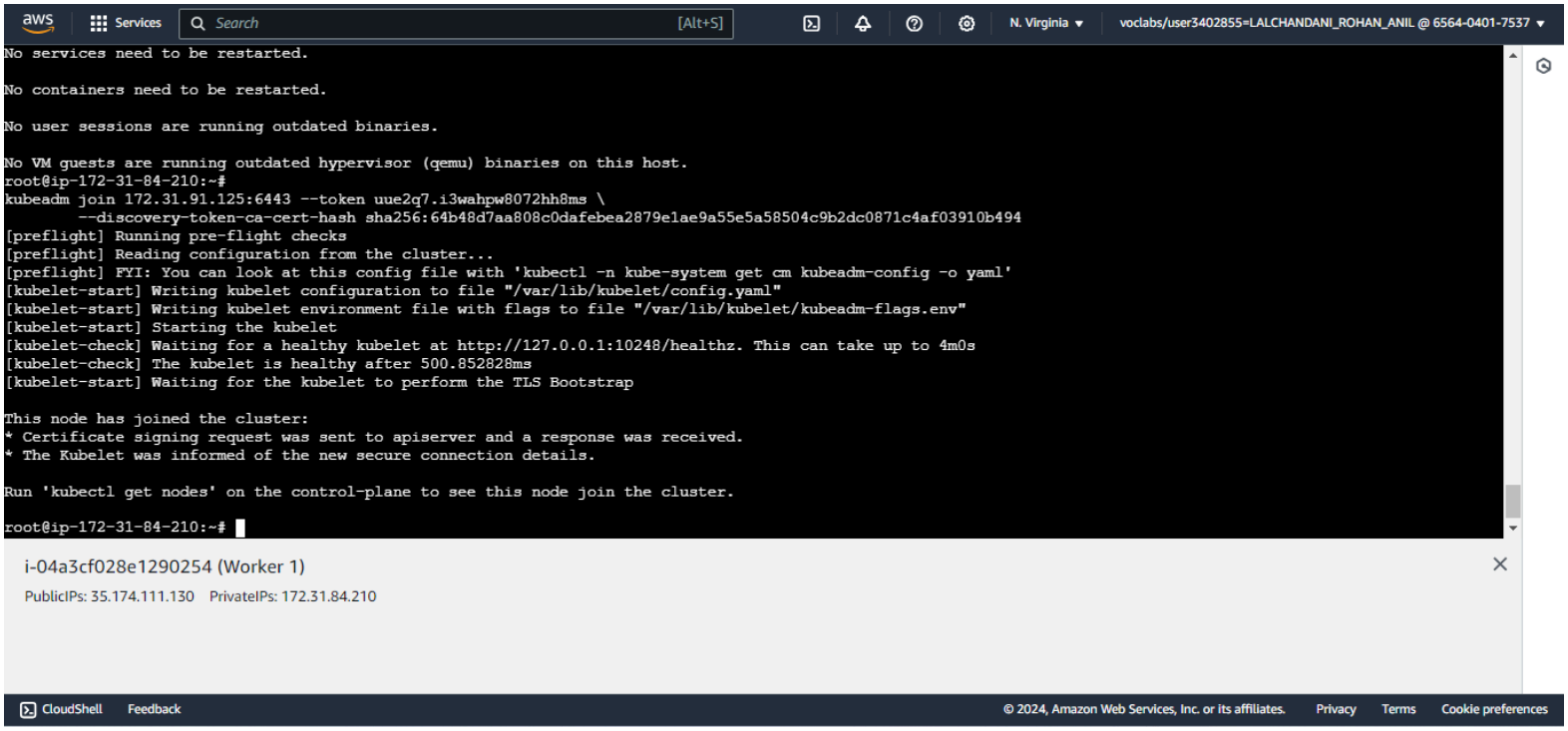
11. Now copy the 3 commands from the regular user and 1 command from root user and one by one execute them.
Execute the same set of steps from 1 to 11 in Worker 1 and Worker 2 except the kubeadm init command.

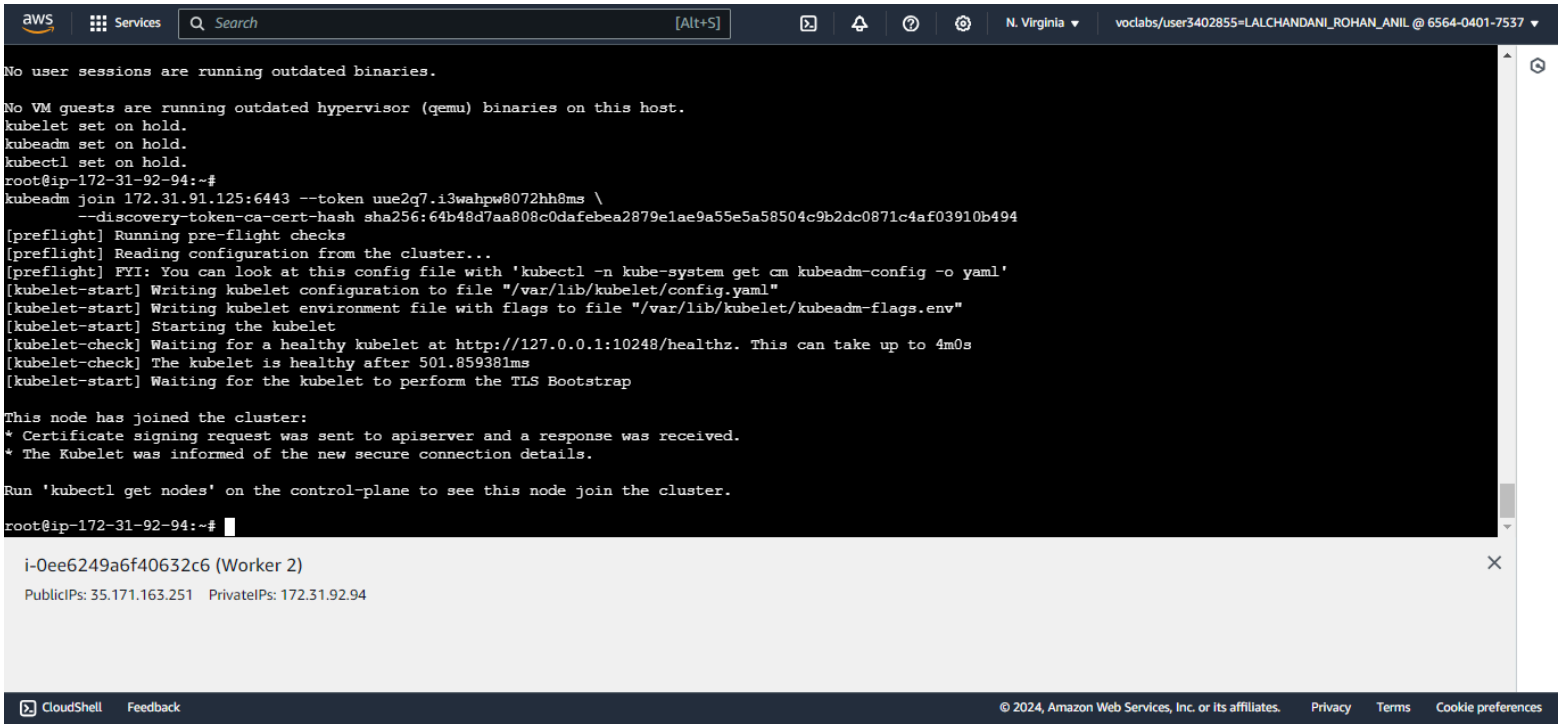


12. Execute “kubectl get nodes” in master to get a list of nodes present. Initially there will be no nodes present.

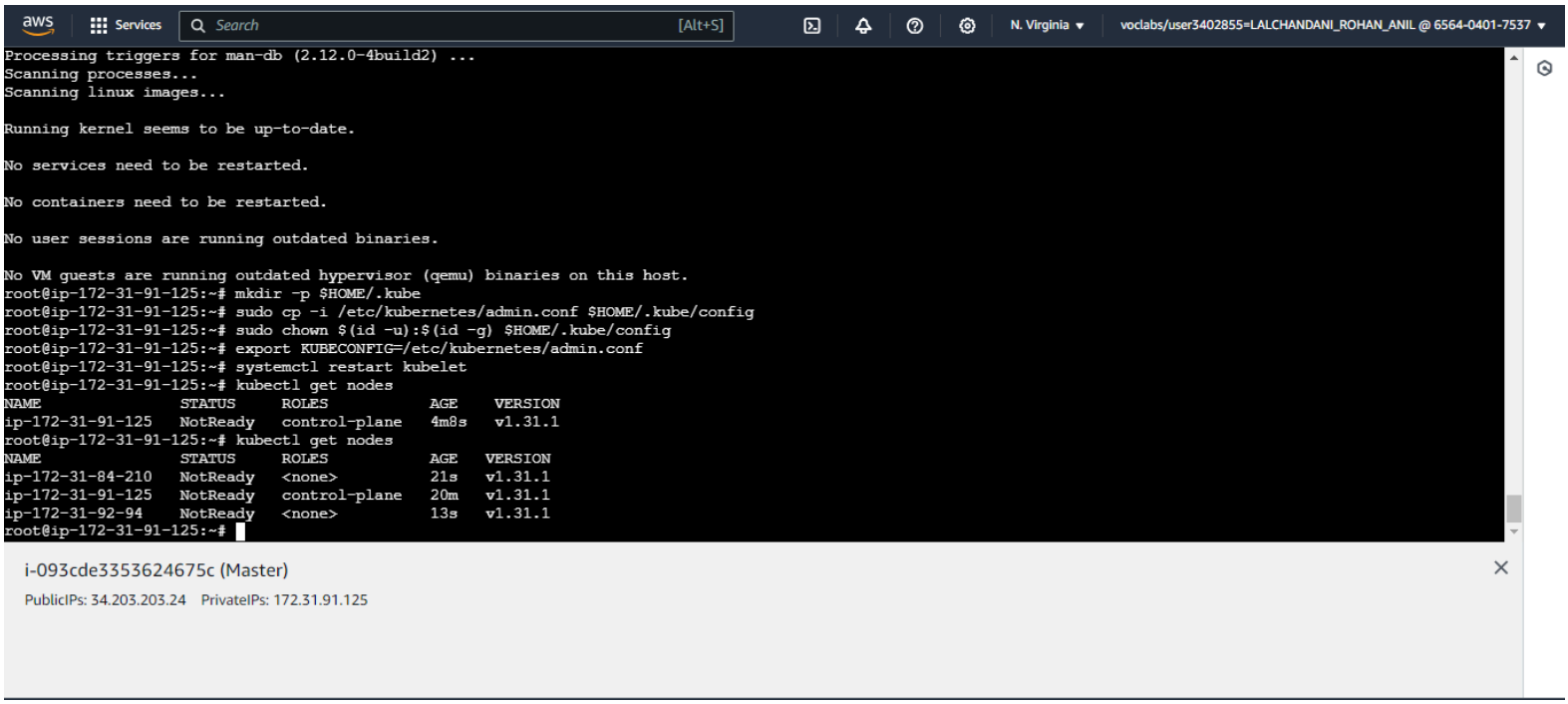


Copy the join command from Master and execute it in Worker 1 and Worker 2 to join them to the cluster.



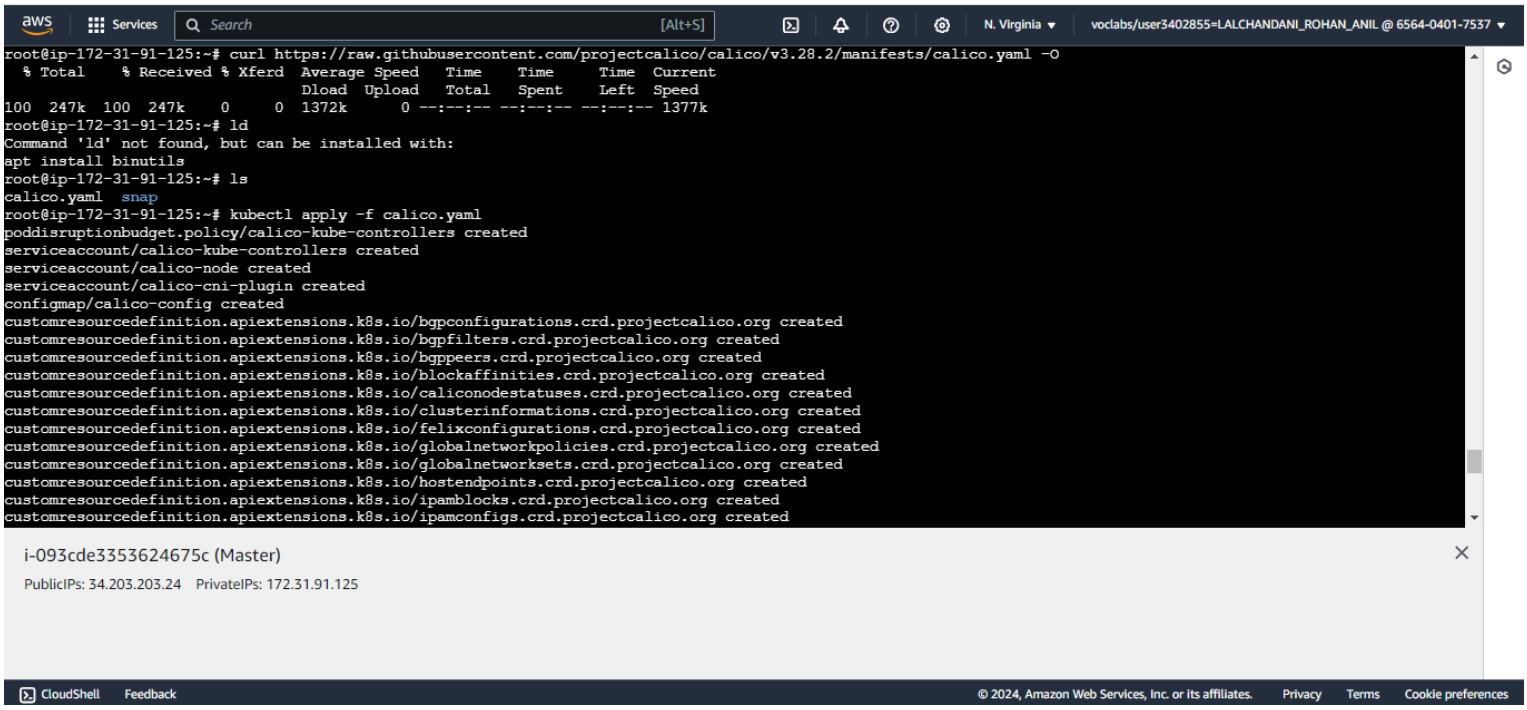


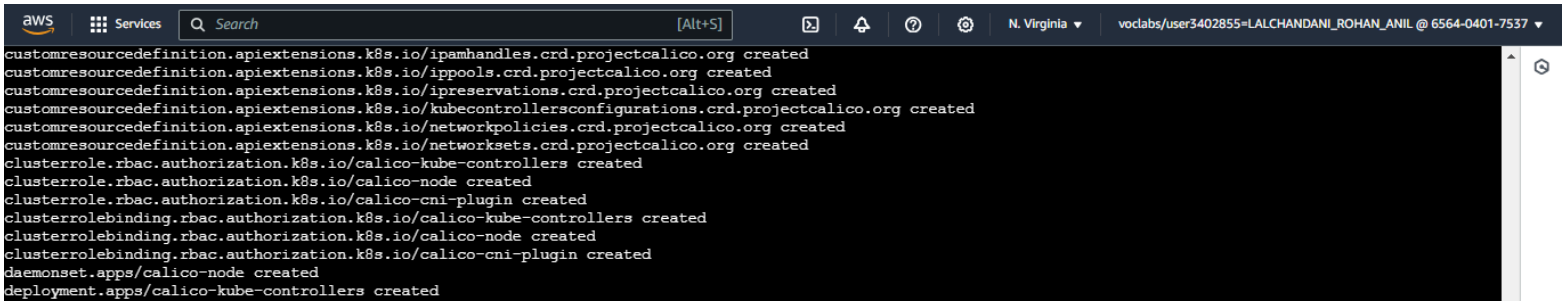
Again execute kubectl get nodes to see if they have joined. You can see the status of nodes to be Not Ready, we have to make them Ready.



Now, we have to manifest “calico.yaml” file to make the status ready for that execute:

curl https://raw.githubusercontent.com/projectcalico/calico/v3.28.2/manifests/calico.yaml -O
ls
kubectl apply -f calico.yaml

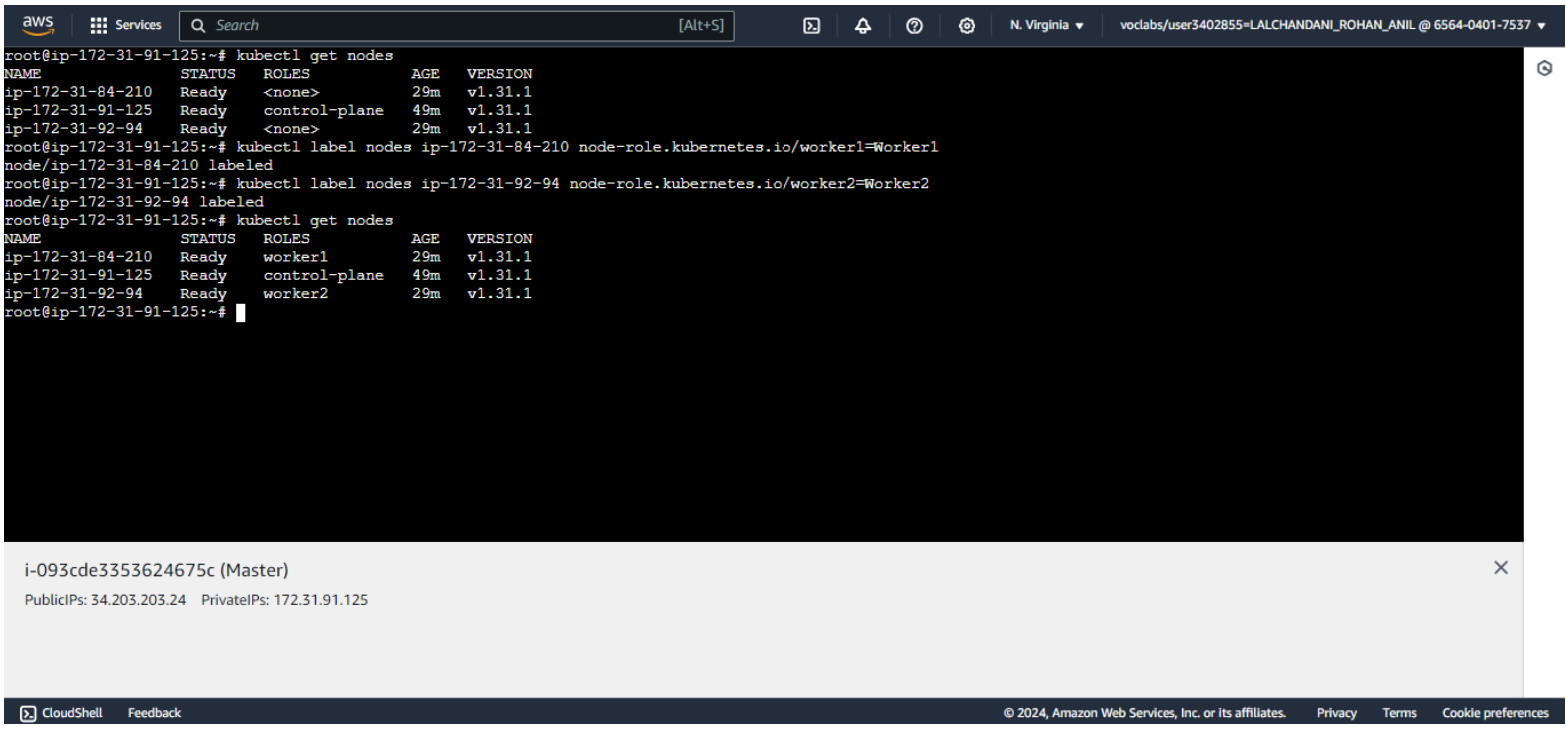




Now the status becomes ready.
EXTRA: Renaming the role names:

kubectl label nodes <node ip address> node-role.kubernetes.io/<nodename>=<nodename>

Replace node ip address with your worker 1 or 2 ip address.
Type the name of the node in nodename.



Adv. Devops Experiment no. 4

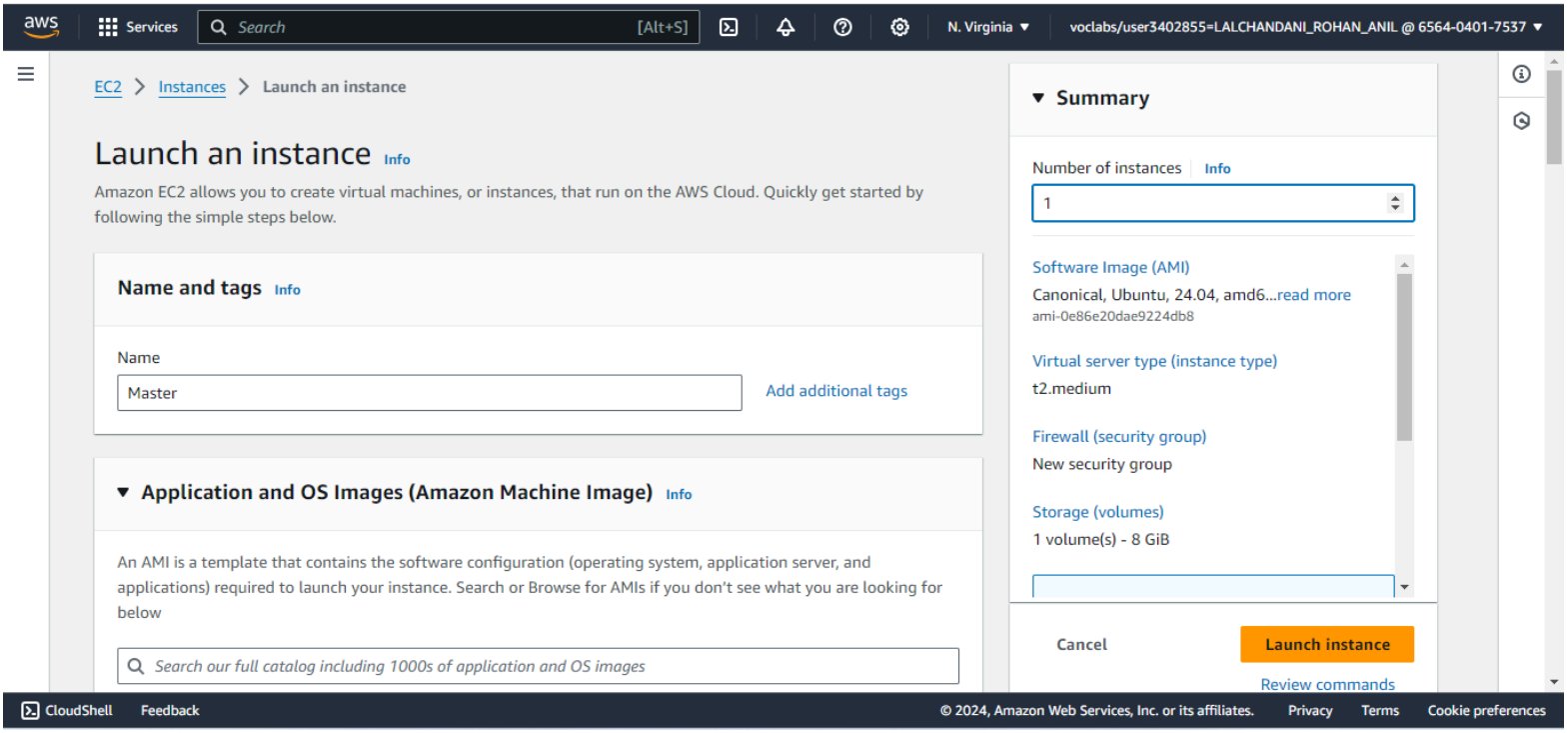
Name: Rohan Lalchandani
Class: D15A Roll no.: 25

Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

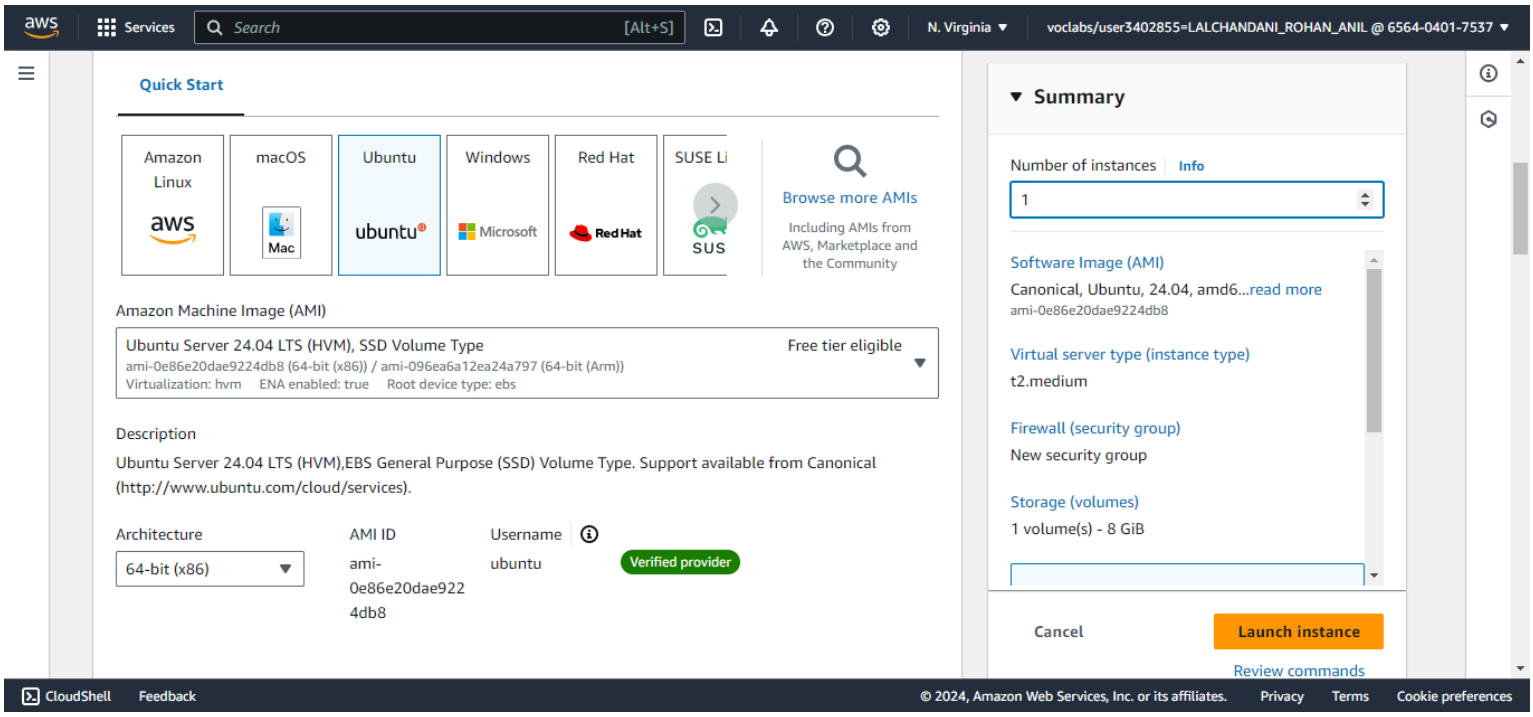
Theory:

Implementation:

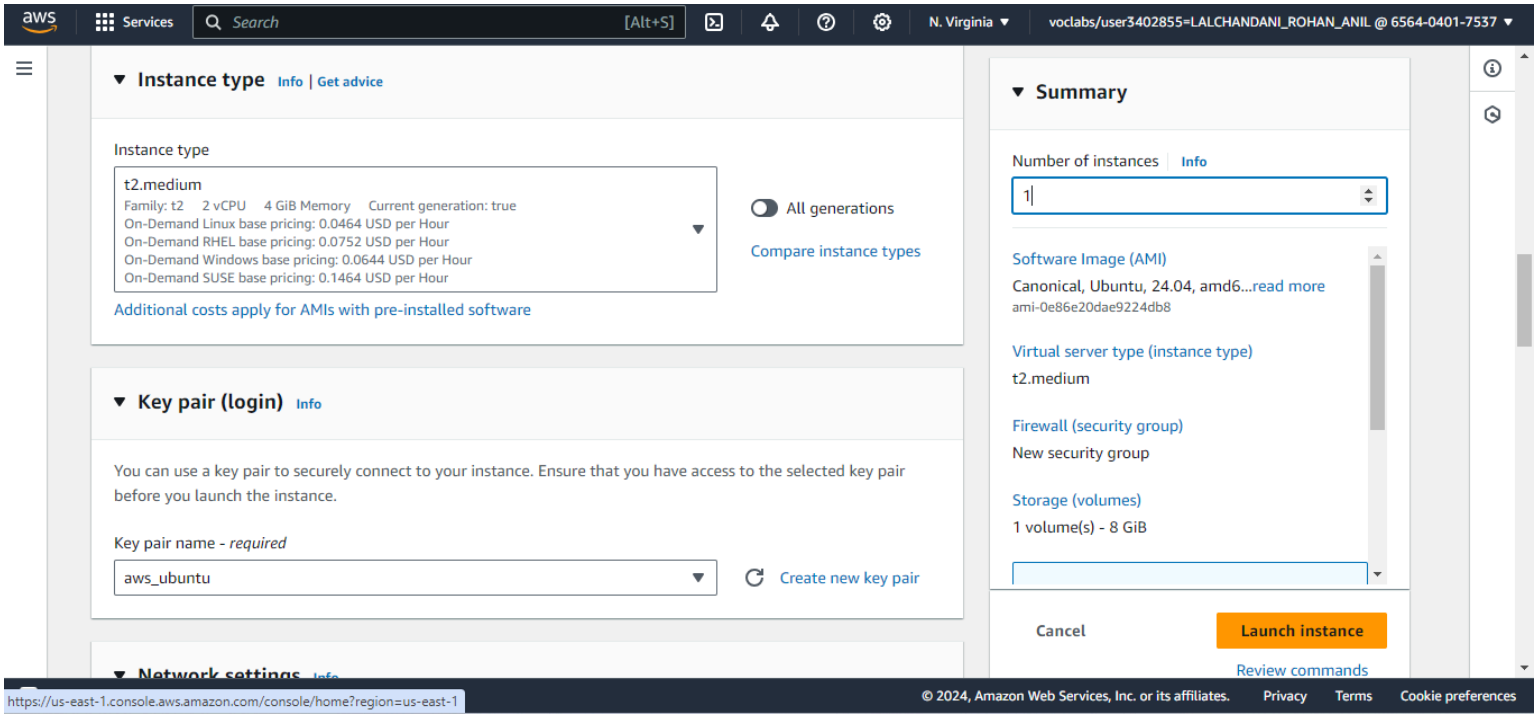
1. Create an EC2 Ubuntu Instance - Master.



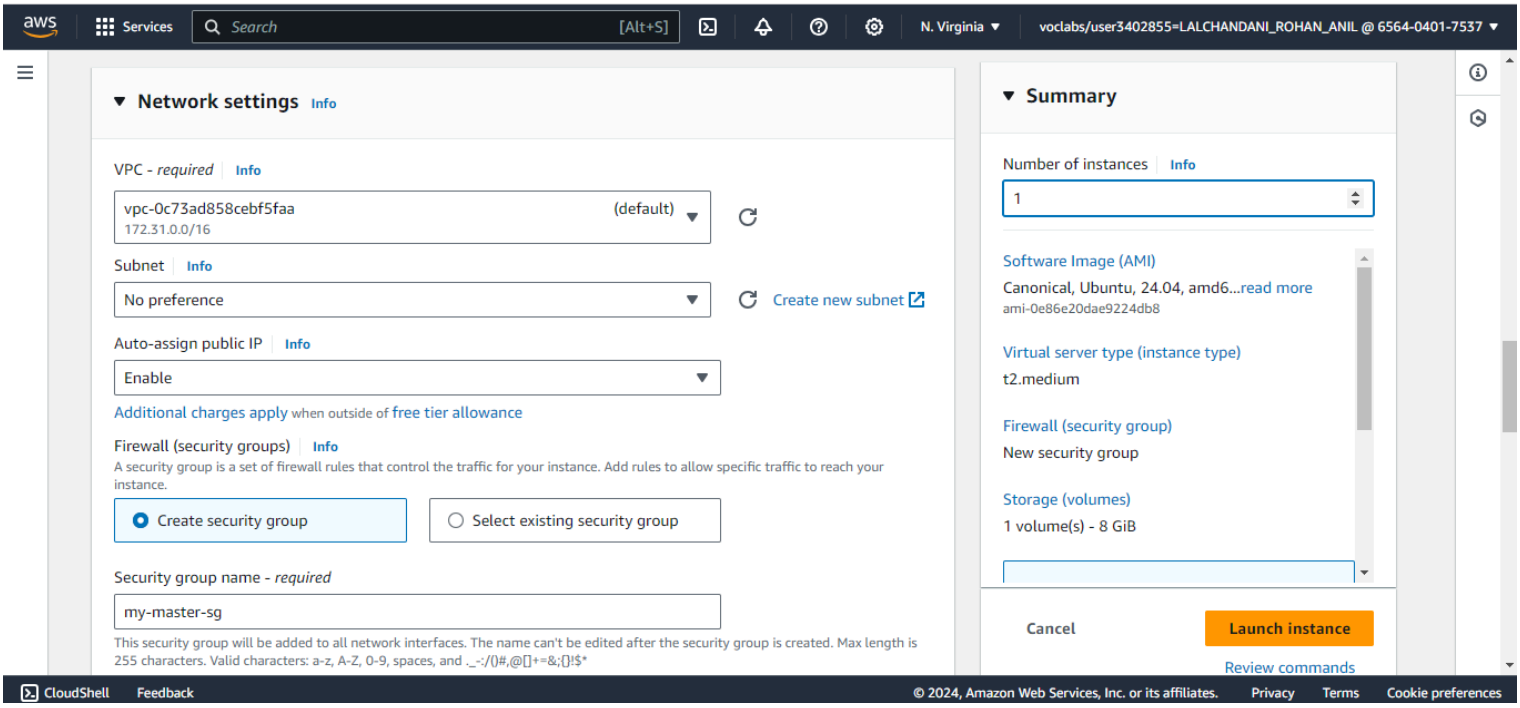
2. Select the following AMI image.



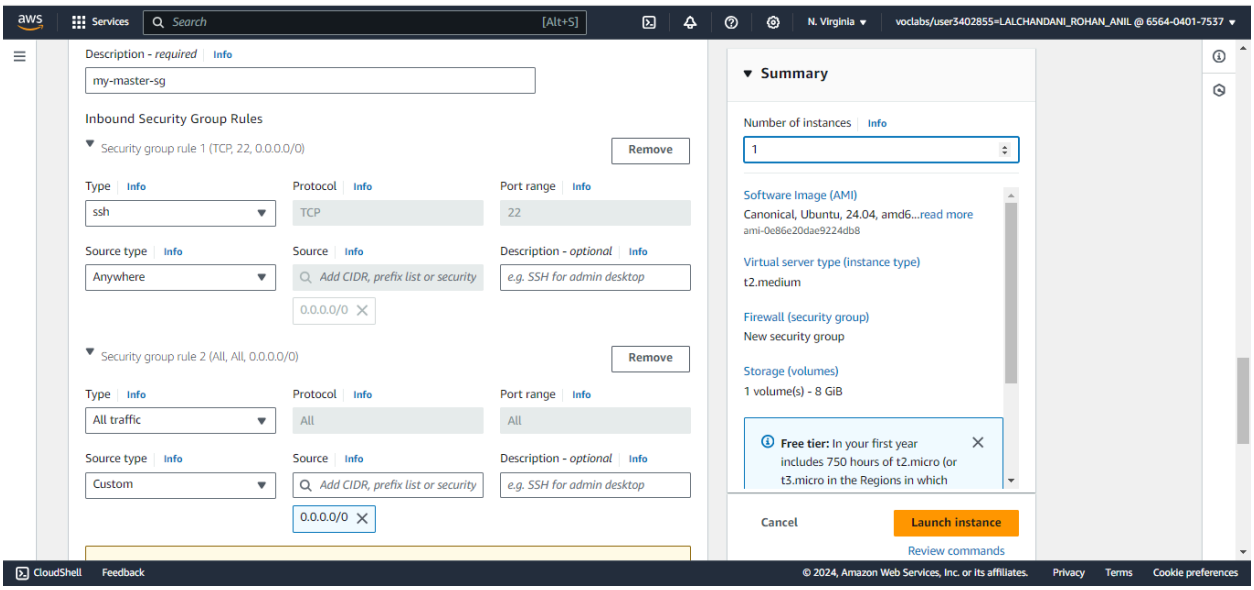
3. Select t2.medium in instance type.
Why?
t2.medium has 2 vCPUs and 4 GB of RAM, which allows it to run more pods and handle larger or more resource-intensive containers compared to **t2.micro** (1 vCPU, 1 GB of RAM).



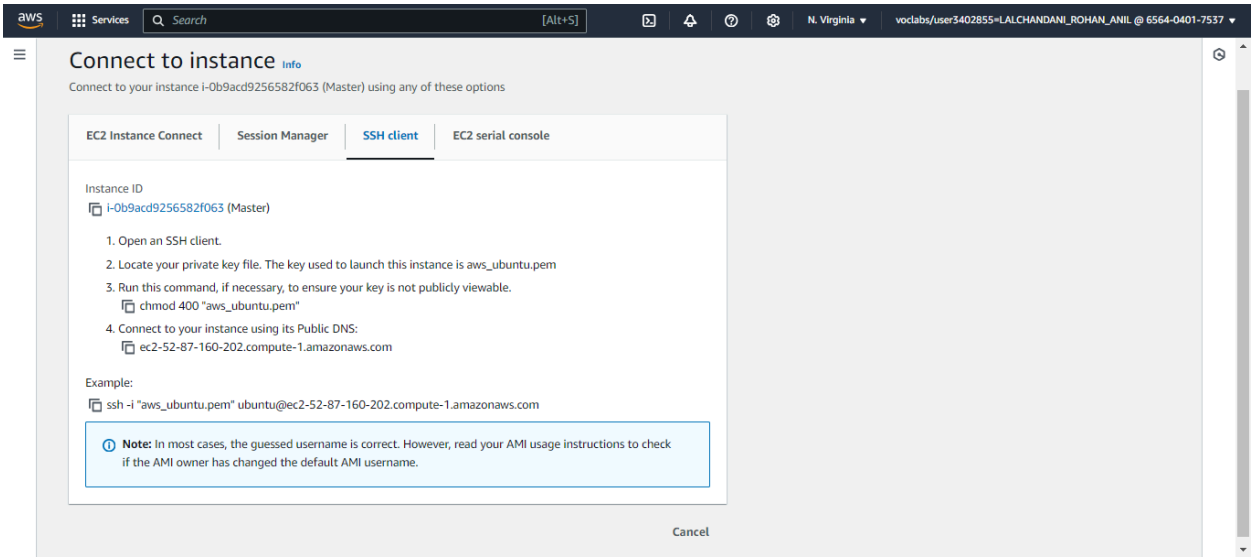
4. Create security group
In it type the group name and description as my-master-sg



5. Edit the inbound rules and add a new rule to accept “All traffic” as shown below.



6. Now connect to the instance by copying the “ssh” command and executing in Git Bash.

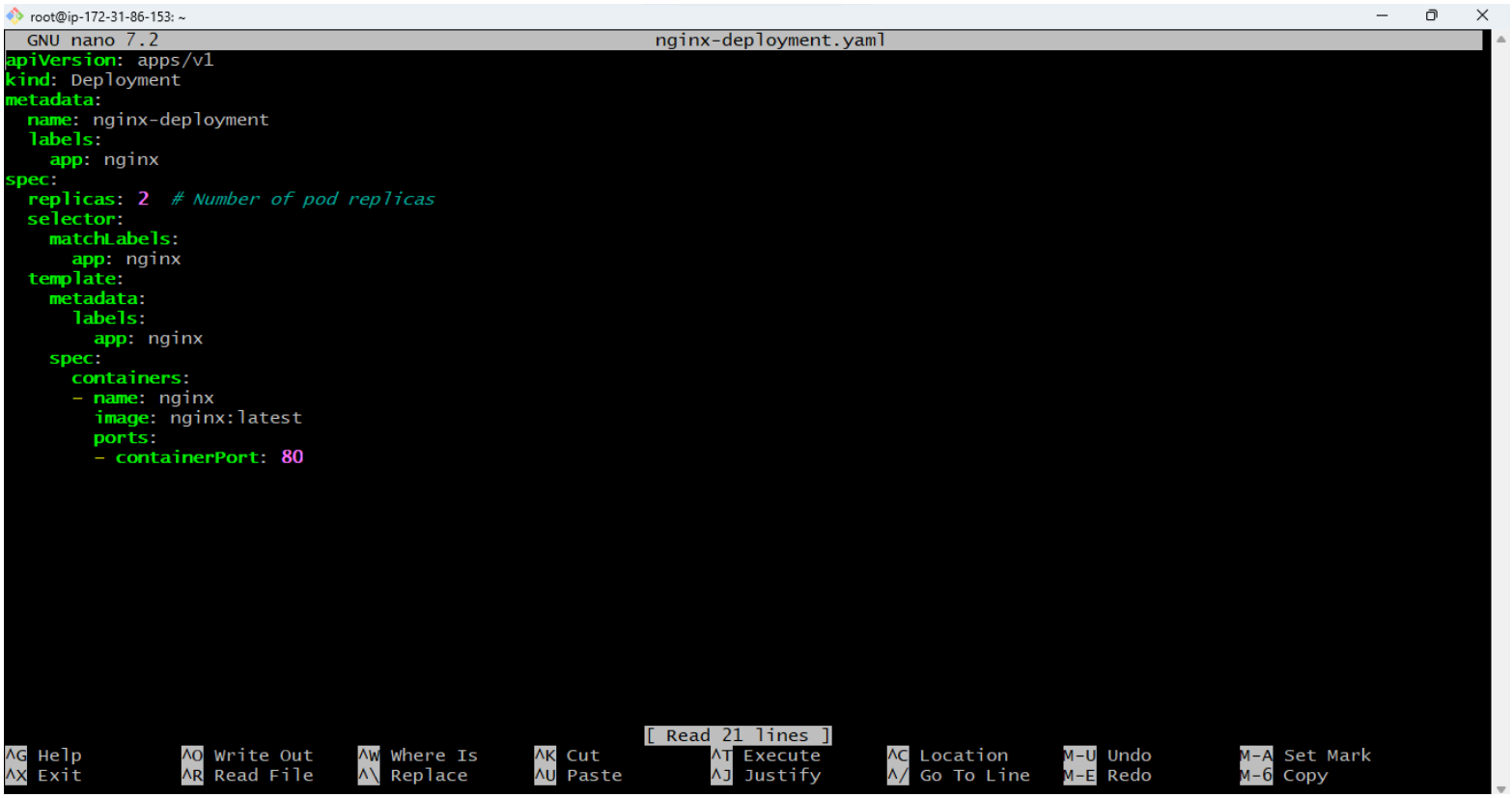


7. Install docker, kubernetes by steps did in 3rd experiment.

8. After that we have to create 2 files as shown below.

```
root@ip-172-31-86-153:~# nano nginx-deployment.yaml
root@ip-172-31-86-153:~# nano nginx-service.yaml
```

nginx-deployment.yaml file :



nginx-service.yaml file :

```
root@ip-172-31-86-153: ~
  GNU nano 7.2                                  nginx-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80  # Port on the service
      targetPort: 80  # Port on the container
  type: LoadBalancer  # For cloud environments, or use ClusterIP for internal traffic only

                                [ Read 12 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy
```

Apply the files

```
root@ip-172-31-86-153:~# kubectl apply -f nginx-deployment.yaml
kubectl apply -f nginx-service.yaml
deployment.apps/nginx-deployment created
service/nginx-service created
root@ip-172-31-86-153:~# kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2            2           12s
```

Check the status of the services and pods if they are running properly

```
root@ip-172-31-86-153:~# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-54b9c68f67-5f8vb   1/1     Running   0          23s
nginx-deployment-54b9c68f67-sbxr6   1/1     Running   0          23s
root@ip-172-31-86-153:~# kubectl get services
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP      10.96.0.1       <none>        443/TCP        30m
nginx-service   LoadBalancer   10.110.82.177   <pending>     80:30319/TCP   60s
```

Expose the port by this command

```
root@ip-172-31-86-153:~# kubectl expose deploy nginx --port 80 --target-port 80 --type NodePort
```

Check the status of the port

```
root@ip-172-31-86-153:~# kubectl get services
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP      10.96.0.1        <none>        443/TCP        111m
nginx           NodePort       10.101.242.191   <none>        80:30905/TCP   6m33s
nginx-service   LoadBalancer   10.110.82.177    <pending>     80:30319/TCP   81m
```

Now go to the browser and copy the “public dns” of the master instance created, paste it in the browser put a colon and then type the port number of the nginx service, in this case it is 30905

Example: http://52.87.160.202:30905/

Adv. Devops Experiment no. 5

Name: Rohan Lalchandani
Class: D15A Roll no.: 25

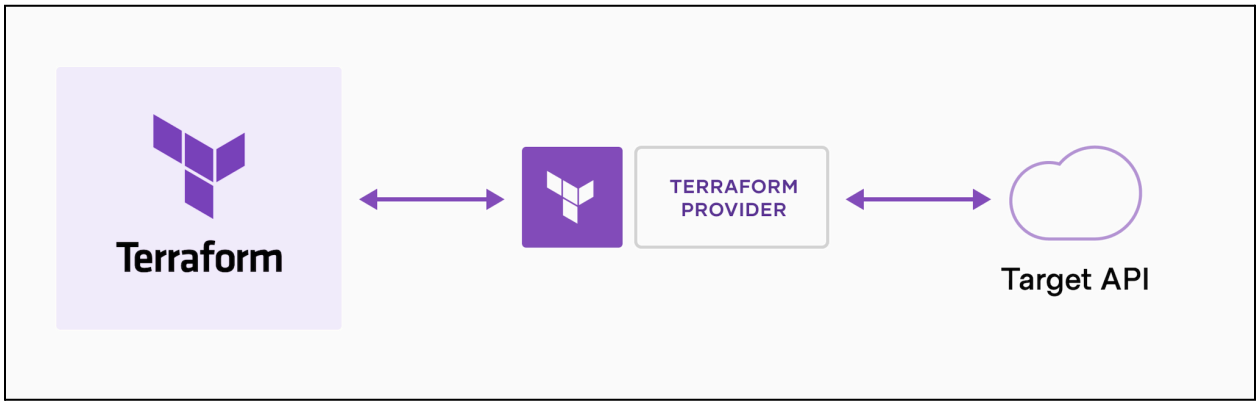
Aim: To understand terraform lifecycle, core concepts/terminologies and install it on a Linux Machine.

Theory:

Terraform:
HashiCorp Terraform is an infrastructure as code tool that lets you define both cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share. You can then use a consistent workflow to provision and manage all of your infrastructure throughout its lifecycle. Terraform can manage low-level components like compute, storage, and networking resources, as well as high-level components like DNS entries and SaaS features.

How does Terraform work?

Terraform creates and manages resources on cloud platforms and other services through their application programming interfaces (APIs). Providers enable Terraform to work with virtually any platform or service with an accessible API.



Key Features

1. Infrastructure as Code (IaC)

- **Declarative Configuration:** Users describe the desired final state of infrastructure, and Terraform determines the steps to achieve that state.
- **Version Control:** Infrastructure configurations can be versioned and treated similarly to application code, allowing for easy tracking of changes and collaboration.
- **Reusability:** Modules and configurations can be reused across different projects, promoting consistency and reducing duplication.

2. Multi-Cloud Support

- **Providers:** Terraform supports a wide range of cloud providers such as AWS, Azure, Google Cloud Platform, and many others, including on-premises solutions.
- **Abstraction:** It provides a consistent workflow across different providers, enabling users to manage heterogeneous environments seamlessly.
- **Portability:** Easy to migrate or replicate infrastructure across different cloud environments.

3. Resource Management

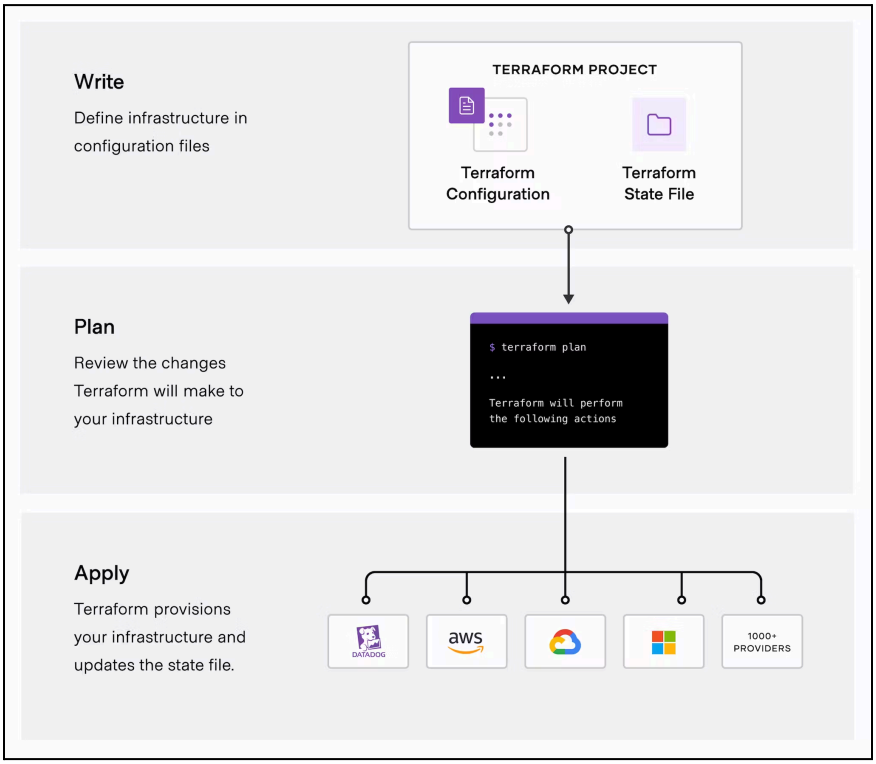
- **Dependency Graphing:** Terraform automatically understands and manages dependencies between resources, ensuring correct order of operations.
- **Plan and Apply:** The `terraform plan` command shows a preview of changes before they are applied, while `terraform apply` executes the changes, providing a safe and predictable workflow.

4. Extensibility

- **Modules:** Users can create and use modules to organize and encapsulate infrastructure configurations, promoting modularity and best practices.
- **Community and Ecosystem:** A vibrant community contributes modules, plugins, and support, enhancing Terraform's capabilities and usability.
- **Custom Plugins:** Ability to develop custom providers and plugins to extend functionality as needed.

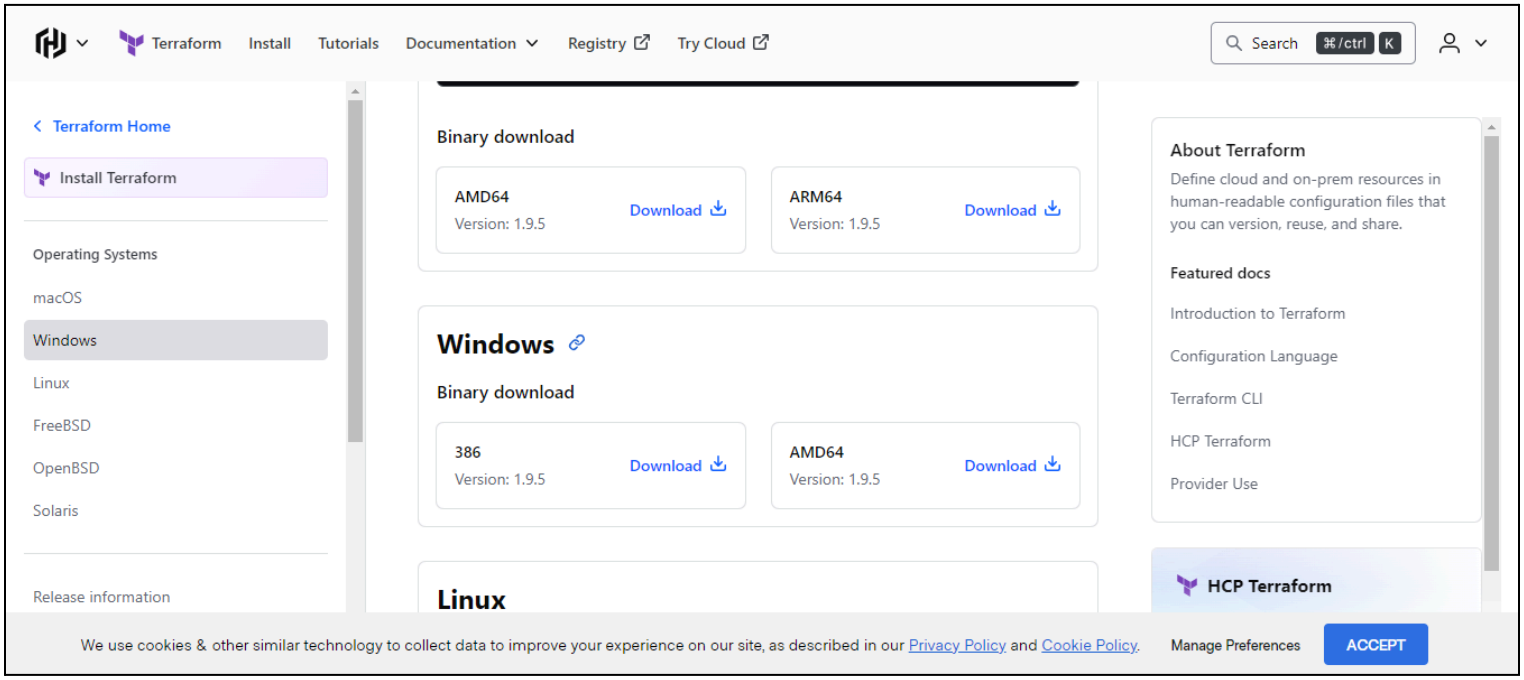
Common Use Cases

- **Provisioning Cloud Infrastructure:** Setting up and managing cloud resources like virtual machines, networks, and storage.
- **Managing Multi-Cloud Environments:** Coordinating resources across different cloud platforms for redundancy and optimization.
- **Infrastructure Migration:** Facilitating the migration of resources from one environment to another through codified configurations.

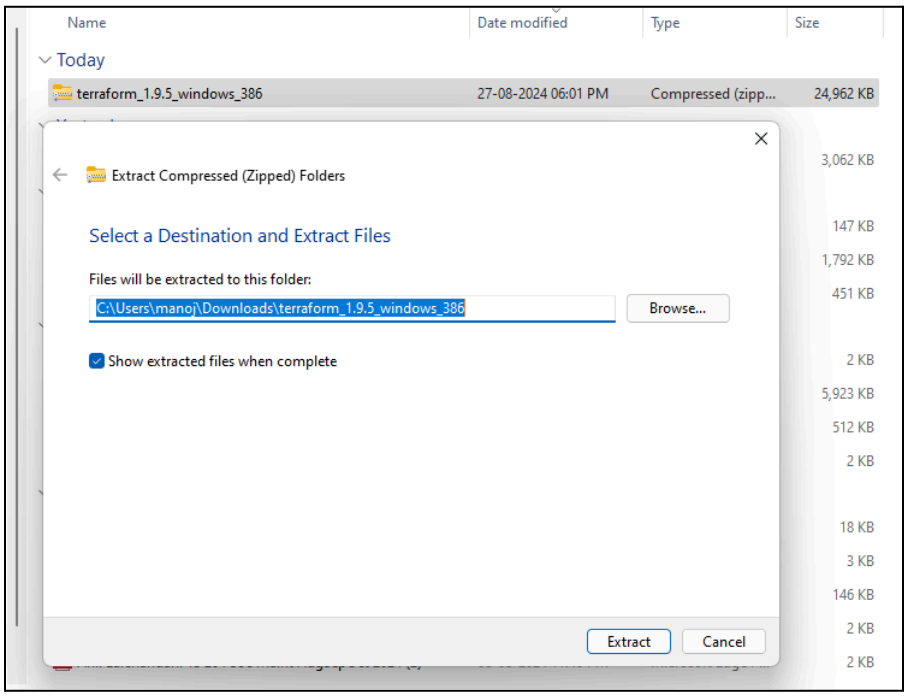


Implementation:

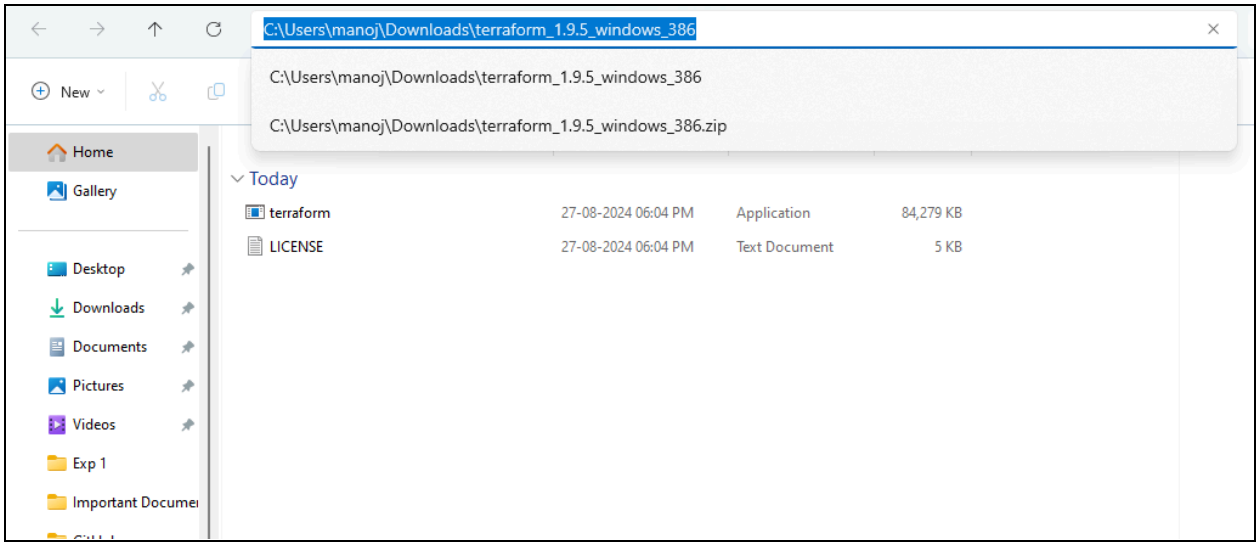
1) Go to website [“https://www.terraform.io/downloads.html”](https://www.terraform.io/downloads.html). Select your OS, in this case we are installing on Windows, download 386.



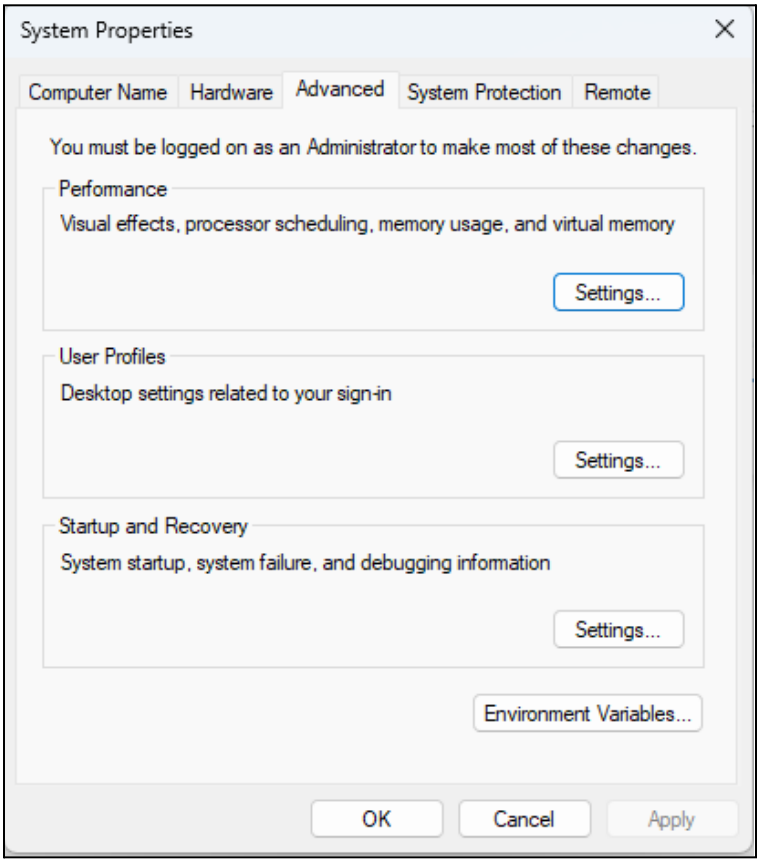
2) Extract the downloaded zip folder.



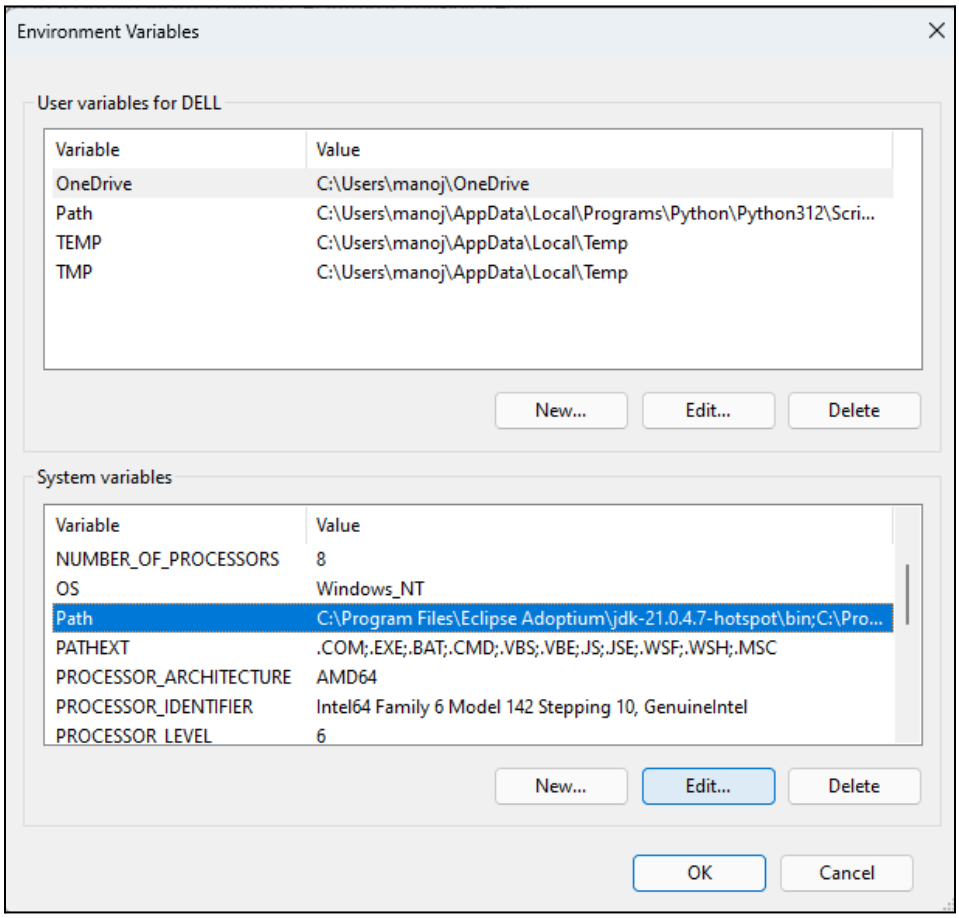
3) Go to the extracted folder and copy the path of the folder as shown below.



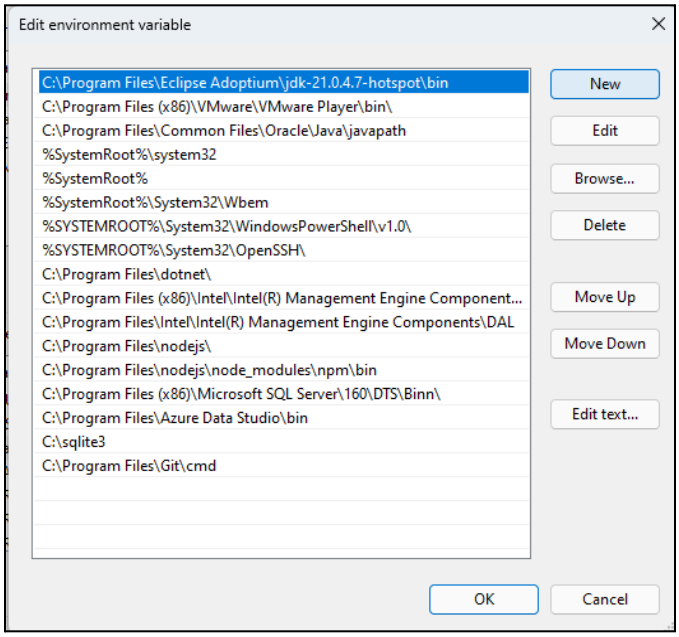
4) Search for “Environment variables” in the Start menu. In the dialog box click on environment variables.



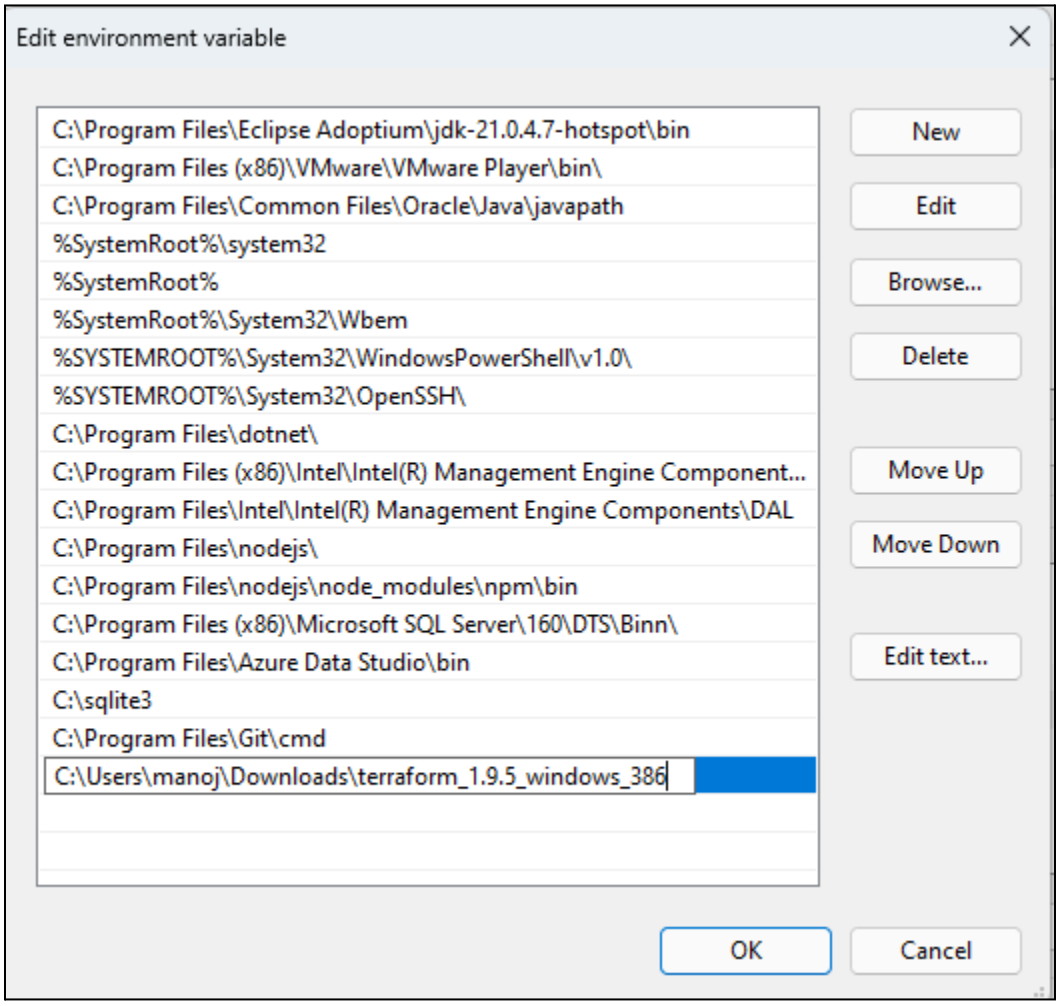
5) Click on path and click on edit as shown below.



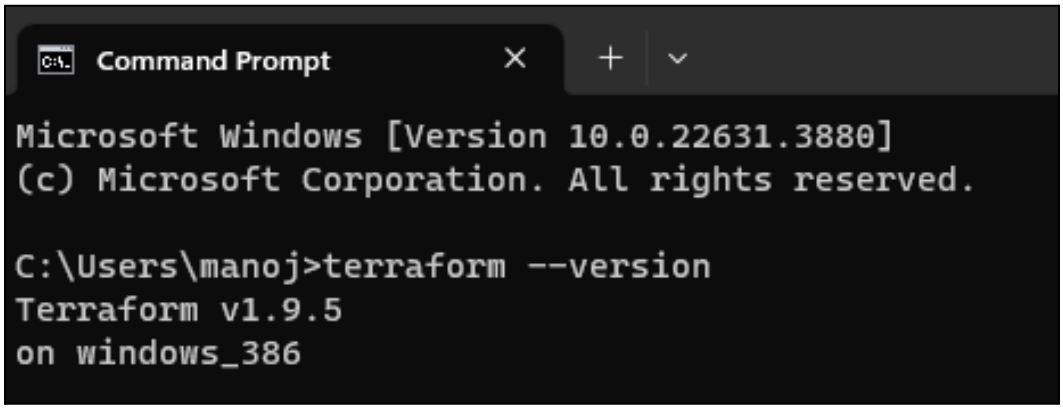
6) Click on New.



7) Paste the path of the extracted folder and click on ok.



8) Open cmd and run the command “terraform --version” to check if it is successfully installed.



Adv. Devops Experiment no. 6

Name: Rohan Lalchandani
Class: D15A Roll no.: 25

Aim: To Build, change, and destroy AWS infrastructure Using Terraform (S3 bucket or Docker).

Theory:

Terraform is an Infrastructure as Code (IaC) tool designed to manage and provision cloud and on-premises infrastructure through code. It offers a consistent and declarative way to describe and automate infrastructure deployments across various platforms.
IaC is a methodology where infrastructure is managed and provisioned using code, rather than manual processes.

1. Core Components of Terraform:

a) Providers:

Providers are plugins that Terraform uses to interact with various infrastructure services. Each provider is responsible for understanding API interactions with a specific service or platform (e.g., AWS, Azure, Google Cloud, Docker).

b) Resources

Resources are the fundamental units of infrastructure managed by Terraform. They represent components such as virtual machines, databases, or networking elements.

c) Modules

Modules are reusable containers of Terraform configuration that are used to create multiple instances of a resource or to encapsulate complex configurations. Modules can be created and shared to standardize and simplify deployments.

d) Variables

Variables allow you to parameterize your Terraform configurations. They help in customizing configurations without modifying the main configuration files directly.

e) Outputs

Outputs are used to extract information from Terraform configurations and make it available to other configurations or external systems.

2. Terraform Workflow

Terraform follows a specific workflow to manage infrastructure:

a) Write

You define your infrastructure requirements using Terraform configuration files (`.tf` files) written in HCL or JSON.

b) Plan

Terraform generates an execution plan to show what actions will be taken to reach the desired state defined in your configuration files. This step helps in previewing changes before applying them.

c) Apply

Terraform applies the changes required to reach the desired state of the configuration. This step creates, updates, or deletes infrastructure resources as necessary.

d) Destroy

Terraform removes all infrastructure defined in the configuration files. This step is useful for cleaning up resources when they are no longer needed.

3. State Management

Terraform maintains a state file (terraform.tfstate) that records the current state of your infrastructure. This state file is crucial for:

- **Tracking Resources:** It maps the resources defined in the configuration files to real-world infrastructure.
- **Planning Changes:** It helps Terraform determine what changes are needed by comparing the state file to the desired configuration.
- **Concurrency Control:** It ensures that multiple users or systems do not make conflicting changes to the infrastructure.

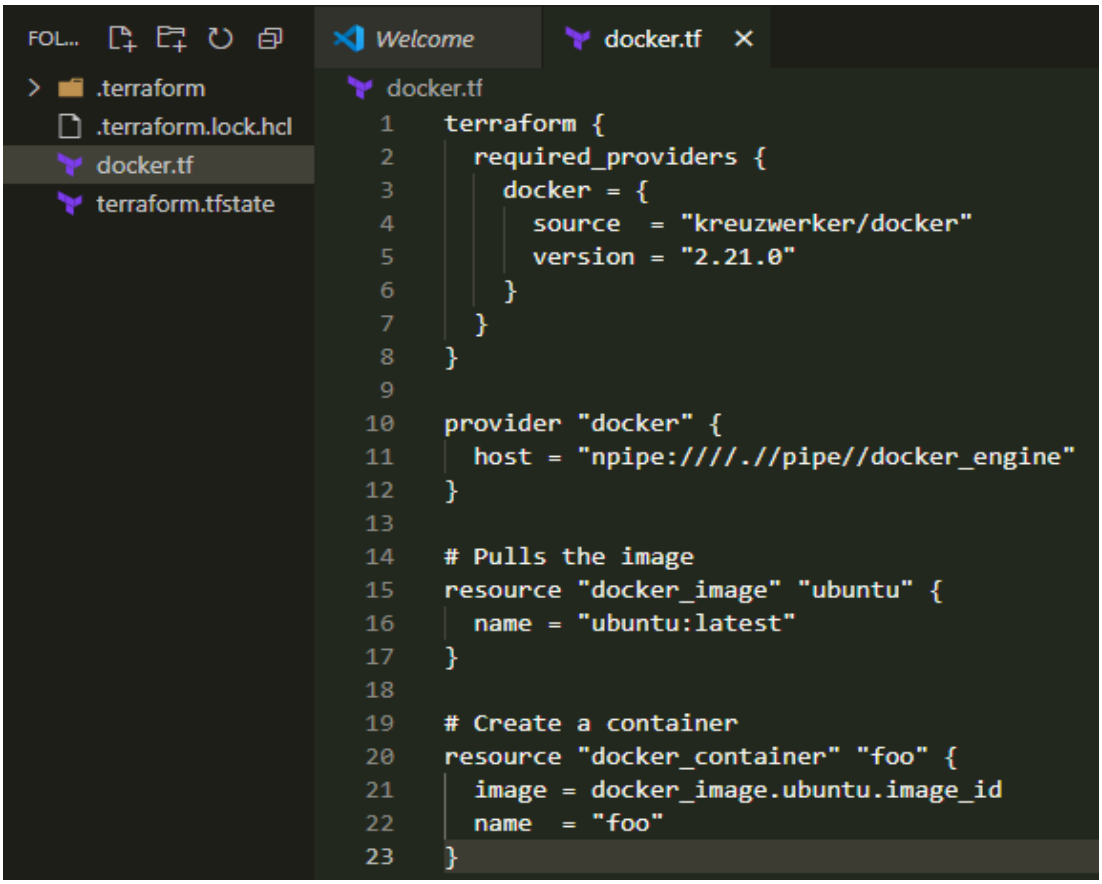
Implementation:

1. Download and install docker from https://www.docker.com/
For Windows Click on AMD64 version. Check if successfully installed in cmd using docker
docker --version

```
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\manoj>docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Common Commands:
  run         Create and run a new container from an image
  exec        Execute a command in a running container
  ps          List containers
  build       Build an image from a Dockerfile
  pull        Download an image from a registry
  push        Upload an image to a registry
  images      List images
  login       Log in to a registry
  logout      Log out from a registry
  search      Search Docker Hub for images
  version     Show the Docker version information
  info        Display system-wide information

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx
  compose*    Docker Compose
  container   Manage containers
  context     Manage contexts
  debug*      Get a shell into any image or container
```

```
C:\Users\manoj>docker --version
Docker version 27.1.1, build 6312585
```

2. Create a folder TerraformScripts and in it create a folder named docker. Open it in VS code and create a new file named docker.tf and write the following code.



3. Open the terminal, make sure the path is set to the docker folder and execute

a) terraform init

```
PS D:\TerraformScripts\Docker> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "2.21.0"...
- Installing kreuzwerker/docker v2.21.0...
- Installed kreuzwerker/docker v2.21.0 (self-signed, key ID BD080C4571C6104C)
Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

b) terraform plan

```
PS D:\TerraformScripts\Docker> terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = (known after apply)
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = (known after apply)
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
      + name             = "foo"
      + network_data     = (known after apply)
      + read_only        = false
      + remove_volumes   = true
      + restart          = "no"
      + rm               = false
      + runtime          = (known after apply)
      + security_opts    = (known after apply)
      + shm_size         = (known after apply)
      + start            = true
      + stdin_open       = false
      + stop_signal      = (known after apply)
      + stop_timeout     = (known after apply)
      + tty              = false

      + healthcheck (known after apply)

      + labels (known after apply)
    }

  # docker_image.ubuntu will be created
  + resource "docker_image" "ubuntu" {
      + id          = (known after apply)
      + image_id    = (known after apply)
      + latest      = (known after apply)
      + name        = "ubuntu:latest"
      + output      = (known after apply)
      + repo_digest = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.
```

(Extra)

c) terraform validate

```
PS D:\TerraformScripts\Docker> terraform validate
Success! The configuration is valid.
```

d) terraform apply

```
PS D:\TerraformScripts\Docker> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = [
          + "/bin/bash",
          + "-c",
          + "while true; do sleep 3600; done",
        ]
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = (known after apply)
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
      + name             = "foo"
      + network_data     = (known after apply)
```

```
      + network_data     = (known after apply)
      + read_only        = false
      + remove_volumes   = true
      + restart          = "no"
      + rm               = false
      + runtime          = (known after apply)
      + security_opts    = (known after apply)
      + shm_size         = (known after apply)
      + start            = true
      + stdin_open       = false
      + stop_signal      = (known after apply)
      + stop_timeout     = (known after apply)
      + tty              = false

      + healthcheck (known after apply)

      + labels (known after apply)
    }

  # docker_image.ubuntu will be created
  + resource "docker_image" "ubuntu" {
      + id          = (known after apply)
      + image_id    = (known after apply)
      + latest      = (known after apply)
      + name        = "ubuntu:latest"
      + output      = (known after apply)
      + repo_digest = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

```
docker_image.ubuntu: Creating...
docker_image.ubuntu: Still creating... [10s elapsed]
docker_image.ubuntu: Still creating... [20s elapsed]
docker_image.ubuntu: Still creating... [30s elapsed]
docker_image.ubuntu: Creation complete after 37s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Creating...
docker_container.foo: Creation complete after 1s [id=984860ac4d11d2b4bed665180e05f69408c3e0f24227c2b0386e77dc63568188]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

4. Docker images before terraform apply command

```
PS D:\TerraformScripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

5. Docker images after terraform apply command

```
PS D:\TerraformScripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   5 weeks ago   78.1MB
```

6. terraform destroy

```
PS D:\TerraformScripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=984860ac4d11d2b4bed665180e05f69408c3e0f24227c2b0386e77dc63568188]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # docker_container.foo will be destroyed
  - resource "docker_container" "foo" {
      - attach           = false -> null
      - command          = [
          - "/bin/bash",
          - "-c",
          - "while true; do sleep 3600; done",
        ] -> null
      - cpu_shares       = 0 -> null
      - dns              = [] -> null
      - dns_opts         = [] -> null
      - dns_search       = [] -> null
      - entrypoint       = [] -> null
      - env              = [] -> null
      - gateway          = "172.17.0.1" -> null
      - group_add        = [] -> null
      - hostname         = "984860ac4d11" -> null
      - id               = "984860ac4d11d2b4bed665180e05f69408c3e0f24227c2b0386e77dc63568188" -> null
      - image            = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - init             = false -> null
      - ip_address       = "172.17.0.2" -> null
      - ip_prefix_length = 16 -> null
      - ipc_mode         = "private" -> null
      - links            = [] -> null
      - log_driver       = "json-file" -> null
      - log_opts         = {} -> null
      - logs             = false -> null
```

```
      - log_opts           = {} -> null
      - logs               = false -> null
      - max_retry_count    = 0 -> null
      - memory             = 0 -> null
      - memory_swap        = 0 -> null
      - must_run           = true -> null
      - name               = "foo" -> null
      - network_data       = [
          - {
              - gateway                     = "172.17.0.1"
              - global_ipv6_prefix_length = 0
              - ip_address                  = "172.17.0.2"
              - ip_prefix_length            = 16
              - network_name                = "bridge"
                # (2 unchanged attributes hidden)
            },
        ] -> null
      - network_mode       = "bridge" -> null
      - privileged         = false -> null
      - publish_all_ports = false -> null
      - read_only          = false -> null
      - remove_volumes     = true -> null
      - restart            = "no" -> null
      - rm                 = false -> null
      - runtime            = "runc" -> null
      - security_opts      = [] -> null
      - shm_size           = 64 -> null
      - start              = true -> null
      - stdin_open         = false -> null
      - stop_timeout       = 0 -> null
      - storage_opts       = {} -> null
      - sysctls            = {} -> null
      - tmpfs              = {} -> null
      - tty                = false -> null
        # (8 unchanged attributes hidden)
    }
```

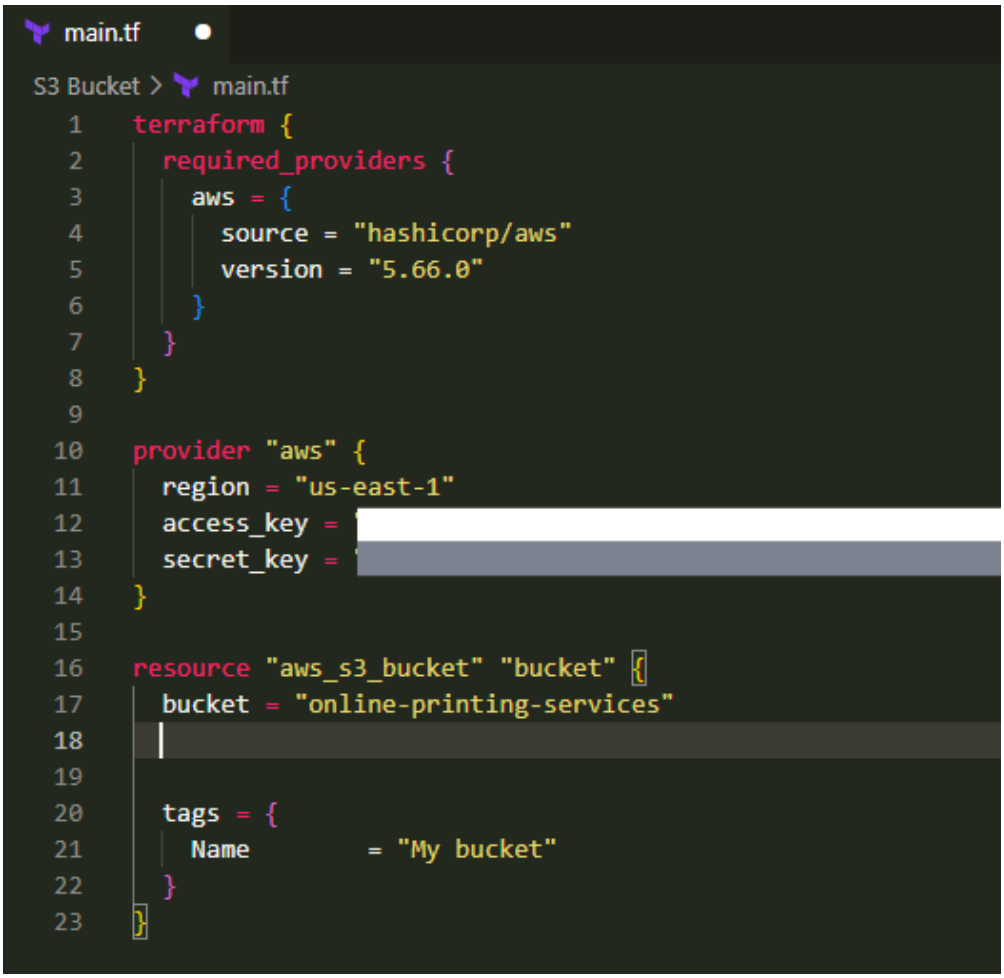
```
  # docker_image.ubuntu will be destroyed
  - resource "docker_image" "ubuntu" {
      - id          = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest" -> null
      - image_id    = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - latest      = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - name        = "ubuntu:latest" -> null
      - repo_digest = "ubuntu@sha256:8a37d68f4f73ebf3d4efafbcf66379bf3728902a8038616808f04e34a9ab63ee" -> null
    }

Plan: 0 to add, 0 to change, 2 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

docker_container.foo: Destroying... [id=984860ac4d11d2b4bed665180e05f69408c3e0f24227c2b0386e77dc63568188]
docker_container.foo: Destruction complete after 1s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
```

7. Docker images after terraform destroy

```
Destroy complete! Resources: 2 destroyed.
PS D:\TerraformScripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

S3 bucket using terraform:

1. Create another folder **“S3 Bucket”** in **“TerraformScripts”** folder open it in VS code and create a “main.tf file” and write the following code, taken from terraform documentation.
a) For access_key and secret_key, Go to your “AWS Management Console” and go to security credentials option by click on you profile name at top right corner of console.
b) Click on generate access key, copy and paste the key in the code.



2) Open terminal, move to the “S3 Bucket” folder and execute terraform init.

```
PS D:\TerraformScripts> cd '.\S3 Bucket\'
PS D:\TerraformScripts\S3 Bucket> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.66.0"...
- Installing hashicorp/aws v5.66.0...
- Installed hashicorp/aws v5.66.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

2. Execute terraform validate to check if your code is correct or not.

```
PS D:\TerraformScripts\S3 Bucket> terraform validate
Success! The configuration is valid.
```

3. terraform plan

```
PS D:\TerraformScripts\S3 Bucket> terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_s3_bucket.bucket will be created
  + resource "aws_s3_bucket" "bucket" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "OnlinePrintingServices"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags                        = {
          + "Name" = "My bucket"
        }
      + tags_all                    = {
          + "Name" = "My bucket"
        }
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + cors_rule (known after apply)

      + grant (known after apply)
```

```
      + grant (known after apply)

      + lifecycle_rule (known after apply)

      + logging (known after apply)

      + object_lock_configuration (known after apply)

      + replication_configuration (known after apply)

      + server_side_encryption_configuration (known after apply)

      + versioning (known after apply)

      + replication_configuration (known after apply)


      + replication_configuration (known after apply)


      + replication_configuration (known after apply)

      + replication_configuration (known after apply)


      + replication_configuration (known after apply)


      + replication_configuration (known after apply)

      + server_side_encryption_configuration (known after apply)

      + versioning (known after apply)

      + website (known after apply)
  }
```

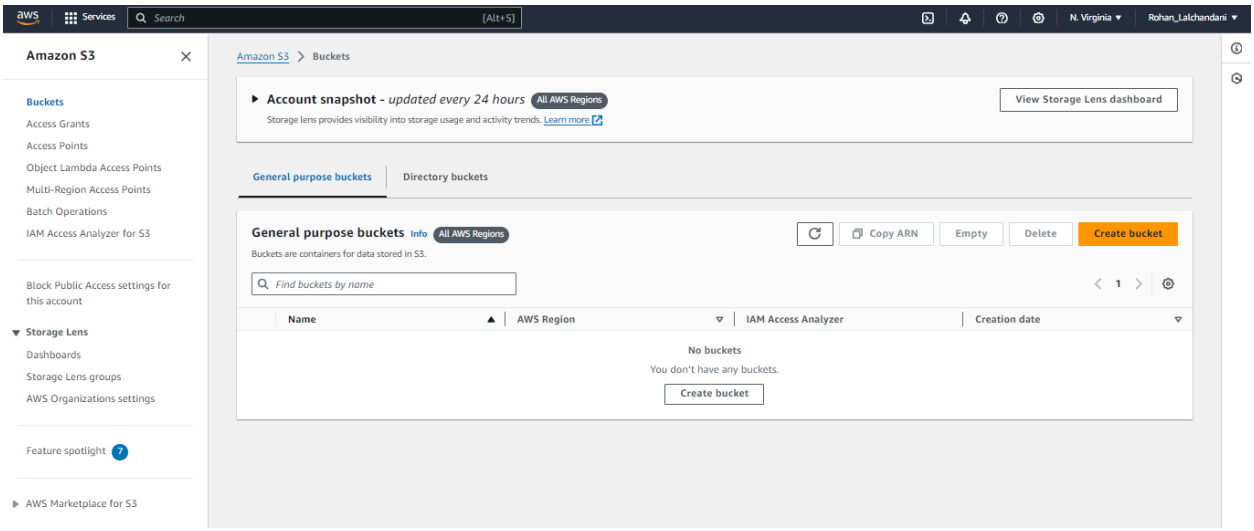
```
Plan: 1 to add, 0 to change, 0 to destroy.
```

4. terraform apply

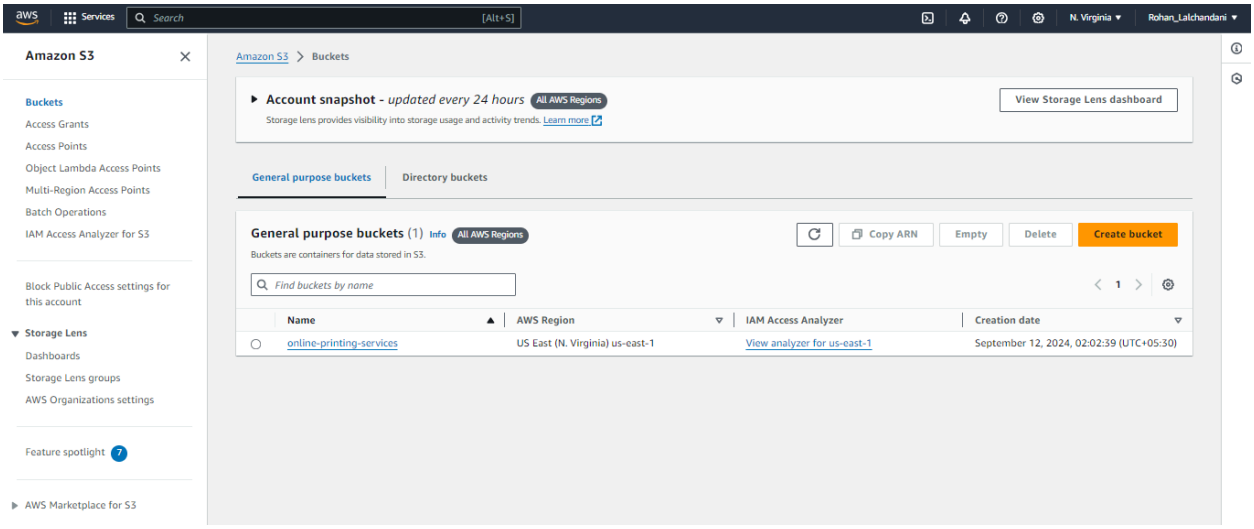
```
PS D:\TerraformScripts\S3 Bucket> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbol
  + create

Terraform will perform the following actions:

  # aws_s3_bucket.bucket will be created
  + resource "aws_s3_bucket" "bucket" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "online-printing-services"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags                        = {
          + "Name" = "My bucket"
        }
      + tags_all                    = {
          + "Name" = "My bucket"
        }
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + cors_rule (known after apply)

      + grant (known after apply)
```

```
      + grant (known after apply)

      + lifecycle_rule (known after apply)

      + logging (known after apply)

      + object_lock_configuration (known after apply)

      + replication_configuration (known after apply)

      + server_side_encryption_configuration (known after apply)

      + versioning (known after apply)

      + website (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_s3_bucket.bucket: Creating...
aws_s3_bucket.bucket: Creation complete after 7s [id=online-printing-services]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS D:\TerraformScripts\S3 Bucket>
```

5. Buckets in console before terraform apply.



6. Buckets in console after terraform apply.

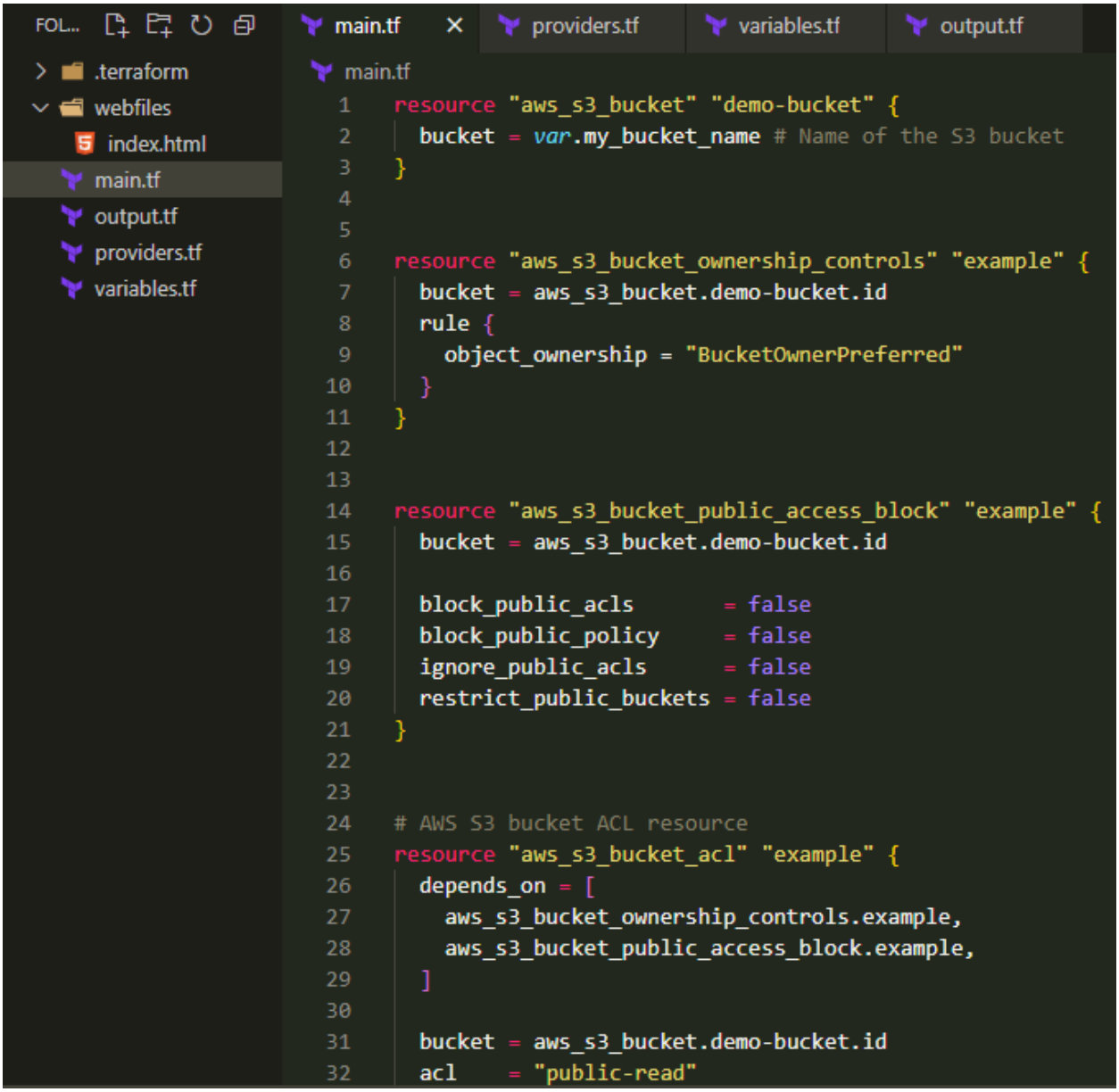


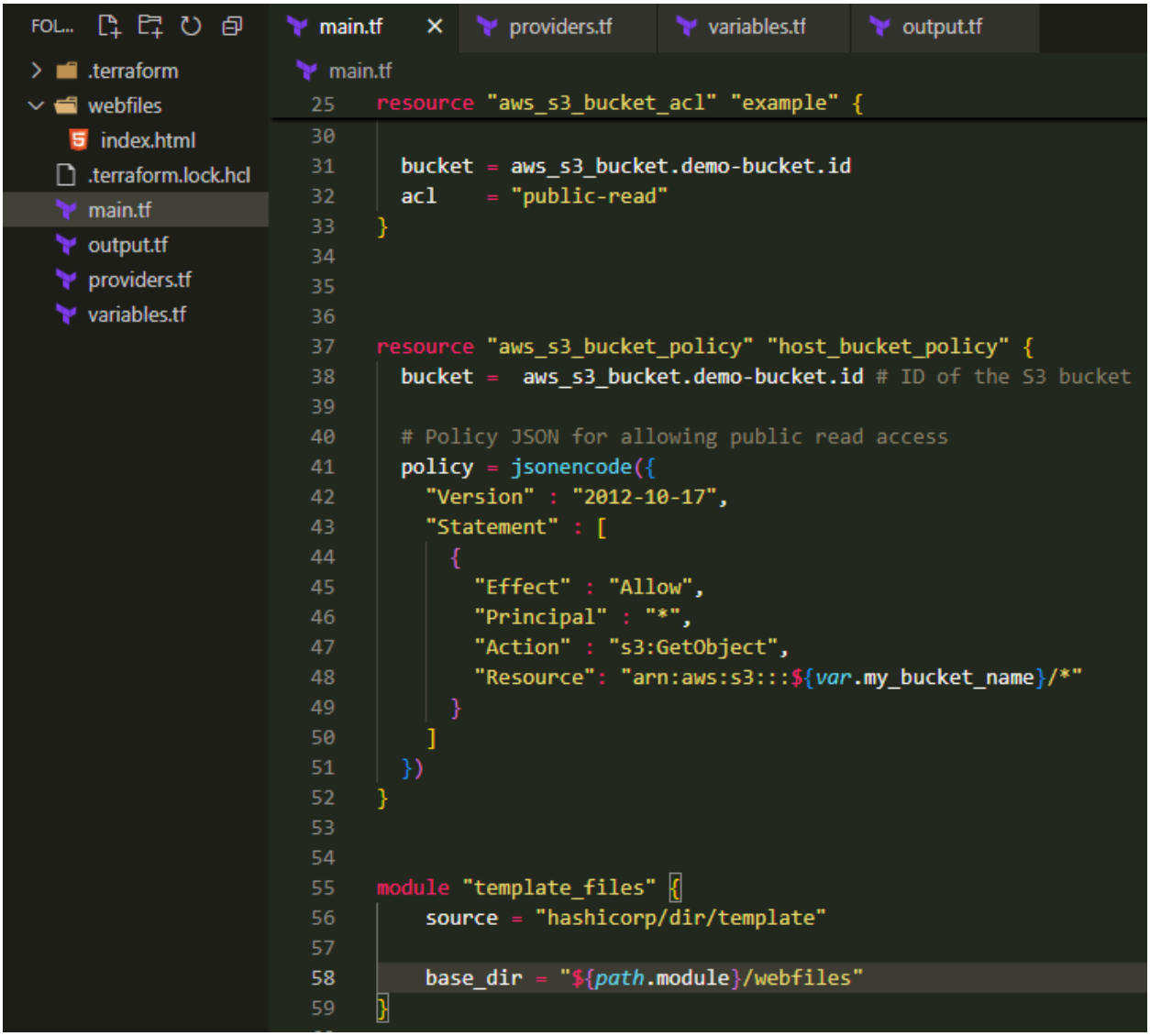
(Extra)
Hosting a website using s3 bucket in terraform

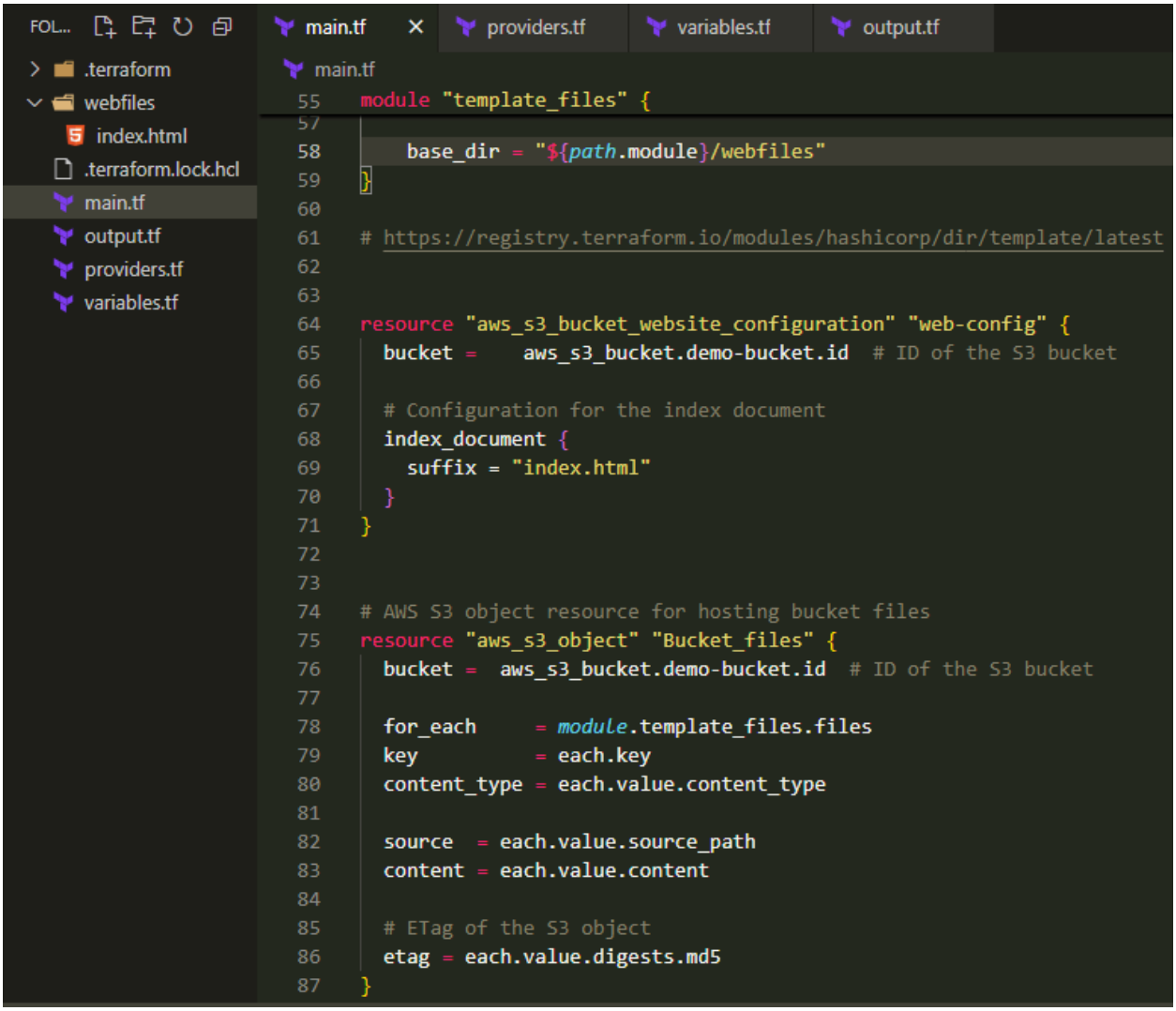
1. Create a new folder “S3 Website” under “TerraformScripts”.
Create four files “main.tf”, ”providers.tf”, ”variables.tf”, ”output.tf” and write the following code.

main.tf :
This Terraform code creates and configures an AWS S3 bucket for hosting static website files with public access.

1. **S3 Bucket**: Creates an S3 bucket (`aws_s3_bucket.demo-bucket`) using a variable for the name.
2. **Ownership Controls**: Configures ownership of the objects in the bucket, making the bucket owner the preferred owner (`BucketOwnerPreferred`).
3. **Public Access Block**: Disables blocking of public access (`aws_s3_bucket_public_access_block`), allowing public access settings.
4. **Bucket ACL**: Sets the bucket's access control list (ACL) to allow public read access (`public-read`).
5. **Bucket Policy**: Adds a policy to the bucket allowing public read access (`s3:GetObject`) to all objects in the bucket.
6. **Template Files Module**: Fetches local files from a directory for hosting using the `hashicorp/dir/template` module.
7. **Website Configuration**: Configures the bucket to host a static website, specifying `index.html` as the main document.
8. **S3 Object**: Uploads files to the S3 bucket for website hosting, with each object having its key (file name) and content.


FOL...
.terraform
webfiles
index.html
main.tf
output.tf
providers.tf
variables.tf
main.tf
providers.tf
variables.tf
output.tf
main.tf
1 resource "aws_s3_bucket" "demo-bucket" {
2 bucket = var.my_bucket_name # Name of the S3 bucket
3 }
4
5
6 resource "aws_s3_bucket_ownership_controls" "example" {
7 bucket = aws_s3_bucket.demo-bucket.id
8 rule {
9 object_ownership = "BucketOwnerPreferred"
10 }
11 }
12
13
14 resource "aws_s3_bucket_public_access_block" "example" {
15 bucket = aws_s3_bucket.demo-bucket.id
16
17 block_public_acls = false
18 block_public_policy = false
19 ignore_public_acls = false
20 restrict_public_buckets = false
21 }
22
23
24 # AWS S3 bucket ACL resource
25 resource "aws_s3_bucket_acl" "example" {
26 depends_on = [
27 aws_s3_bucket_ownership_controls.example,
28 aws_s3_bucket_public_access_block.example,
29]
30
31 bucket = aws_s3_bucket.demo-bucket.id
32 acl = "public-read"

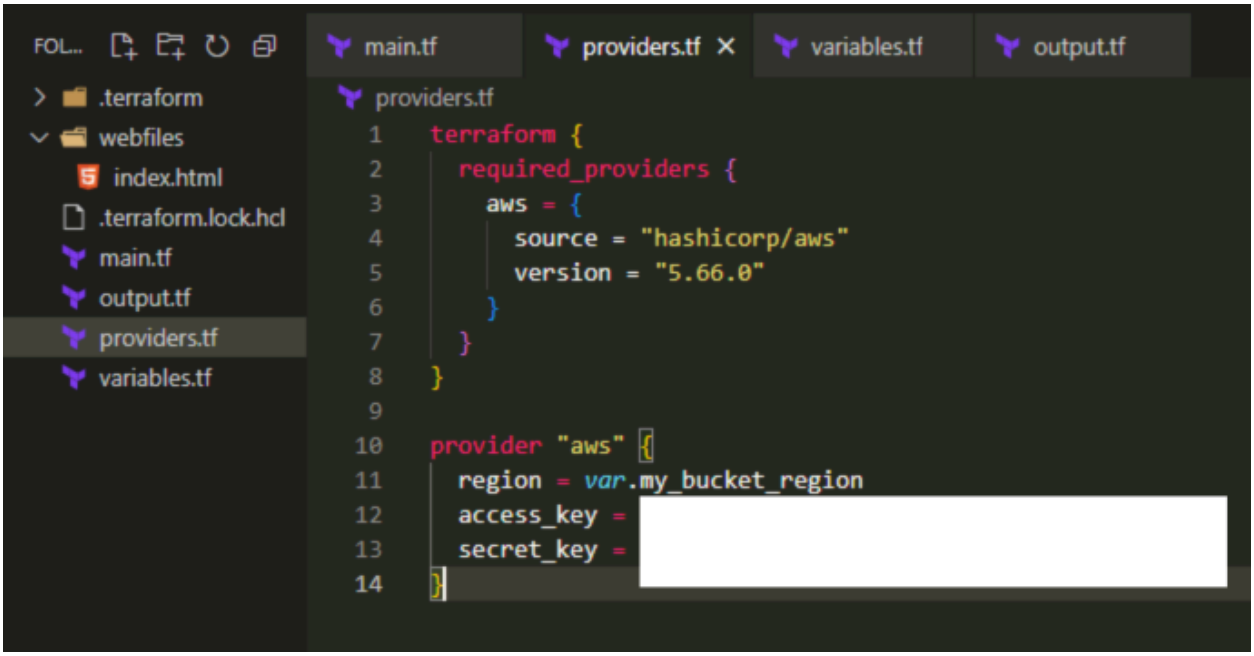

FOL...
.terraform
webfiles
index.html
.terraform.lock.hcl
main.tf
output.tf
providers.tf
variables.tf
main.tf
providers.tf
variables.tf
output.tf
main.tf
25 resource "aws_s3_bucket_acl" "example" {
30
31 bucket = aws_s3_bucket.demo-bucket.id
32 acl = "public-read"
33 }
34
35
36
37 resource "aws_s3_bucket_policy" "host_bucket_policy" {
38 bucket = aws_s3_bucket.demo-bucket.id # ID of the S3 bucket
39
40 # Policy JSON for allowing public read access
41 policy = jsonencode({
42 "Version" : "2012-10-17",
43 "Statement" : [
44 {
45 "Effect" : "Allow",
46 "Principal" : "*",
47 "Action" : "s3:GetObject",
48 "Resource": "arn:aws:s3:::${var.my_bucket_name}/*"
49 }
50]
51 })
52 }
53
54
55 module "template_files" {
56 source = "hashicorp/dir/template"
57
58 base_dir = "${path.module}/webfiles"
59 }



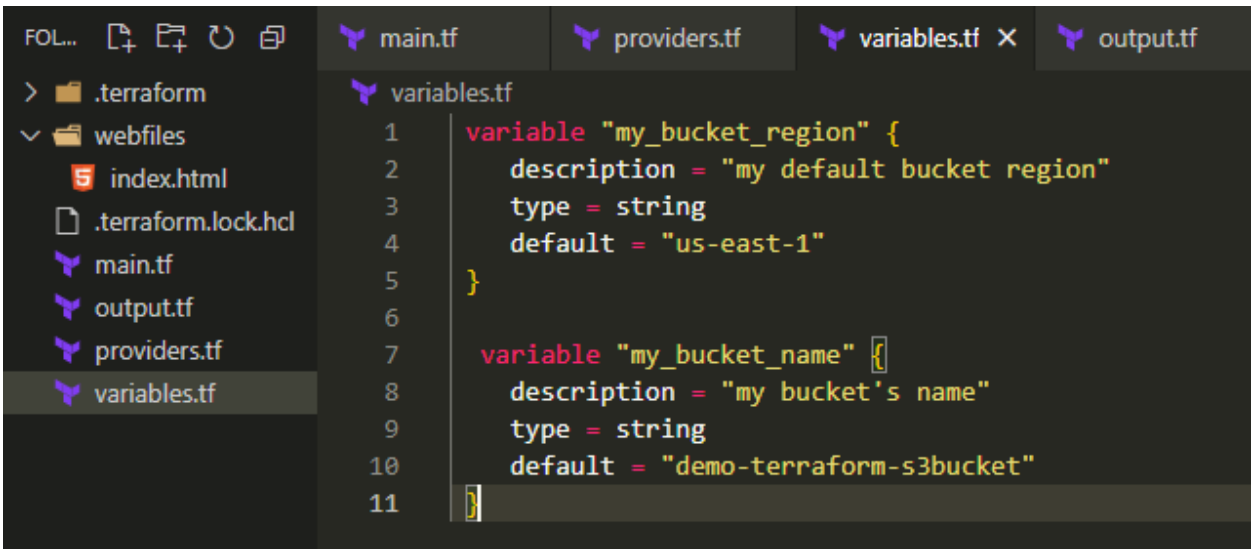
providers.tf :

This Terraform snippet is defining the AWS provider and specifying its version for the configuration. Here's what it does:

1. **Terraform Block**:
    - Specifies that the AWS provider (hashicorp/aws) version 5.66.0 is required.
2. **Provider Configuration**:
    - Configures the AWS provider with the region set by the my_bucket_region variable.
    - The access_key and secret_key fields are placeholders for AWS credentials, which are needed for authentication when interacting with AWS services.



variables.tf :
These variables allow flexible configuration, so you can easily change the region and bucket name without modifying the main code.



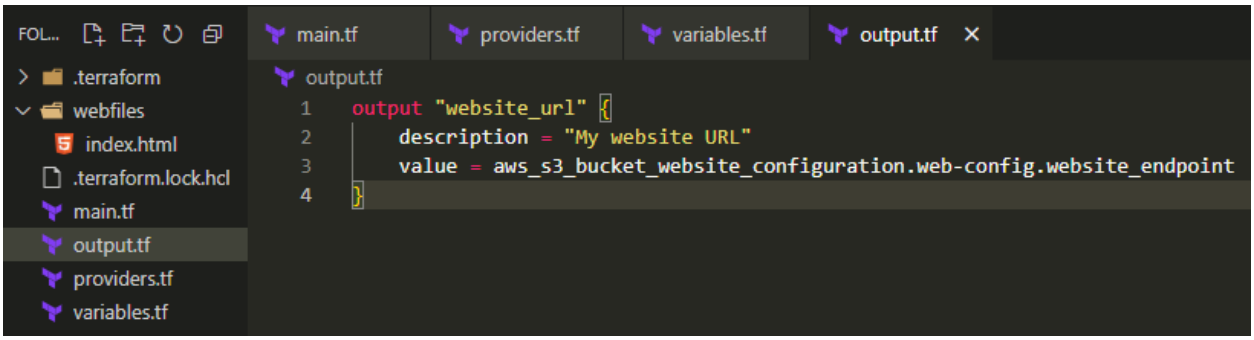
output.tf :

This **output block** in Terraform is used to display the **URL of the static website** hosted on the S3 bucket after the configuration is applied.

- **`output "website_url"`**: Defines an output variable named `website_url` that will be shown after the Terraform execution.
- **`description`**: Describes the purpose of this output, which is to display the website's URL.

- **`value`**: Uses the `website_endpoint` attribute from the `aws_s3_bucket_website_configuration` resource (referenced as `web-config`). This contains the URL of the website hosted in the S3 bucket.

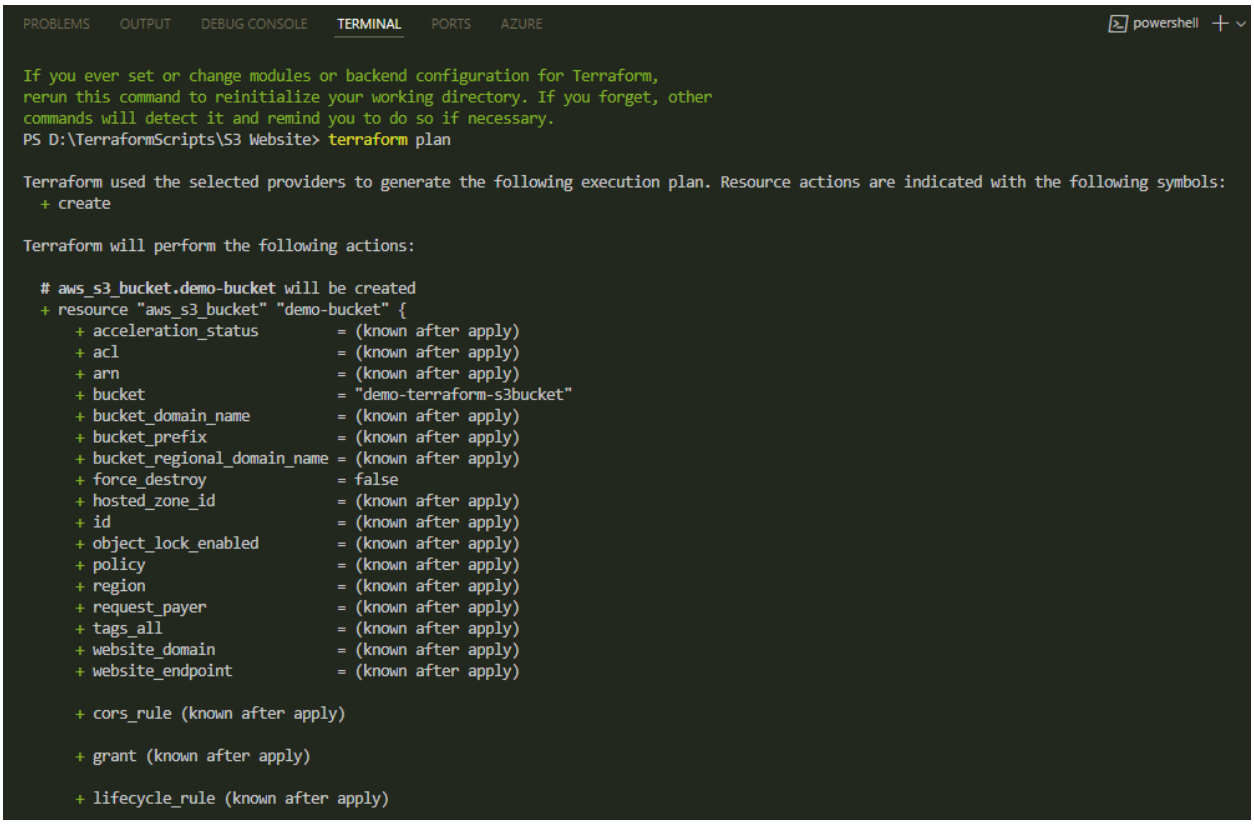
Once Terraform completes, it will print the website's URL for easy access.



2. Open terminal and run terraform init.

```
PS D:\TerraformScripts\S3 Website> terraform init
Initializing the backend...
Initializing modules...
Downloading registry.terraform.io/hashicorp/dir/template 1.0.2 for template_files...
- template_files in .terraform\modules\template_files
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.66.0"...
Initializing modules...
Downloading registry.terraform.io/hashicorp/dir/template 1.0.2 for template_files...
- template_files in .terraform\modules\template_files
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.66.0"...
Downloading registry.terraform.io/hashicorp/dir/template 1.0.2 for template_files...
- template_files in .terraform\modules\template_files
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.66.0"...
- template_files in .terraform\modules\template_files
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.66.0"...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.66.0"...
- Installing hashicorp/aws v5.66.0...

- Installing hashicorp/aws v5.66.0...
- Installed hashicorp/aws v5.66.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.
```

3. Execute terraform plan.

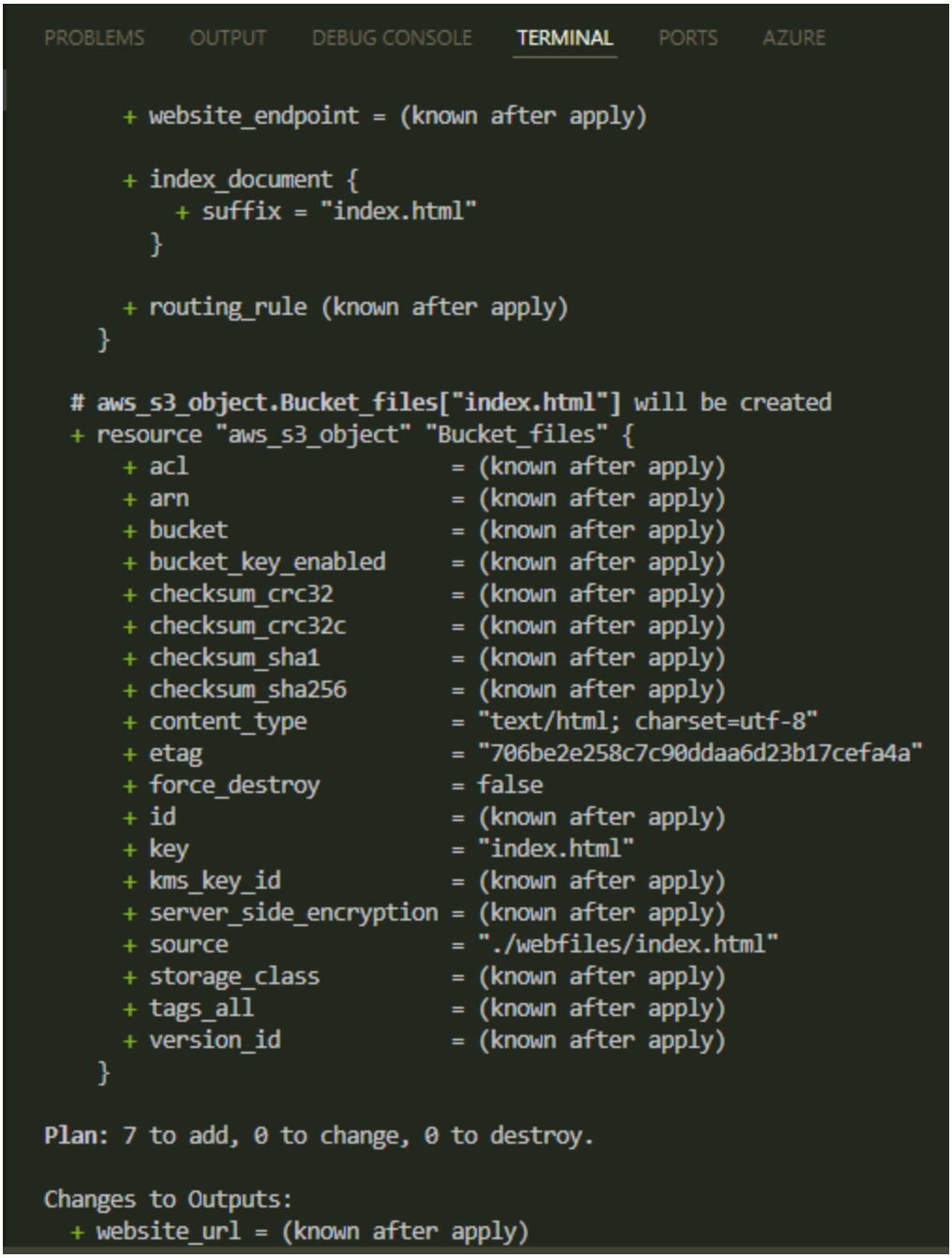


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

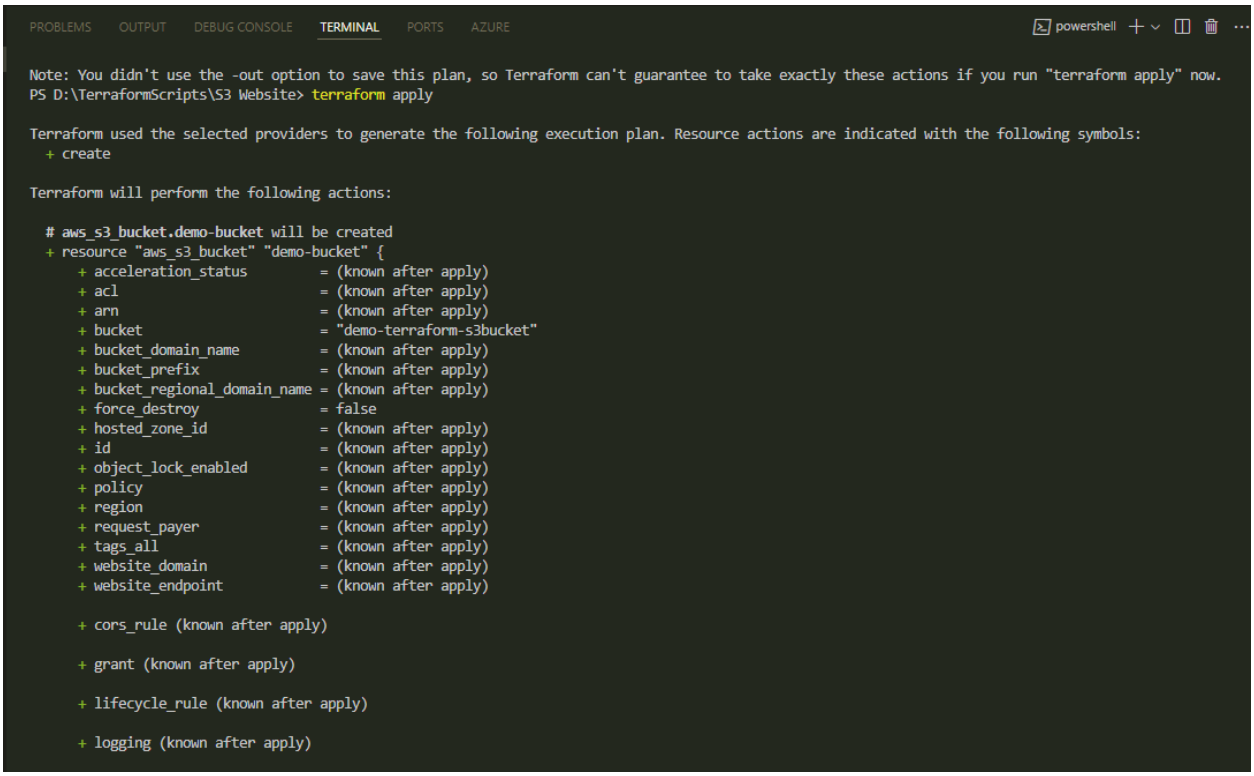
```
      + lifecycle_rule (known after apply)

      + logging (known after apply)

      + object_lock_configuration (known after apply)

      + replication_configuration (known after apply)

      + server_side_encryption_configuration (known after apply)

      + versioning (known after apply)

      + website (known after apply)
    }

  # aws_s3_bucket_acl.example will be created
  + resource "aws_s3_bucket_acl" "example" {
      + acl    = "public-read"
      + bucket = (known after apply)
      + id     = (known after apply)

      + access_control_policy (known after apply)
    }

  # aws_s3_bucket_ownership_controls.example will be created
  + resource "aws_s3_bucket_ownership_controls" "example" {
      + bucket = (known after apply)
      + id     = (known after apply)

      + rule {
          + object_ownership = "BucketOwnerPreferred"
        }
    }

  # aws_s3_bucket_policy.host_bucket_policy will be created
  + resource "aws_s3_bucket_policy" "host_bucket_policy" {
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

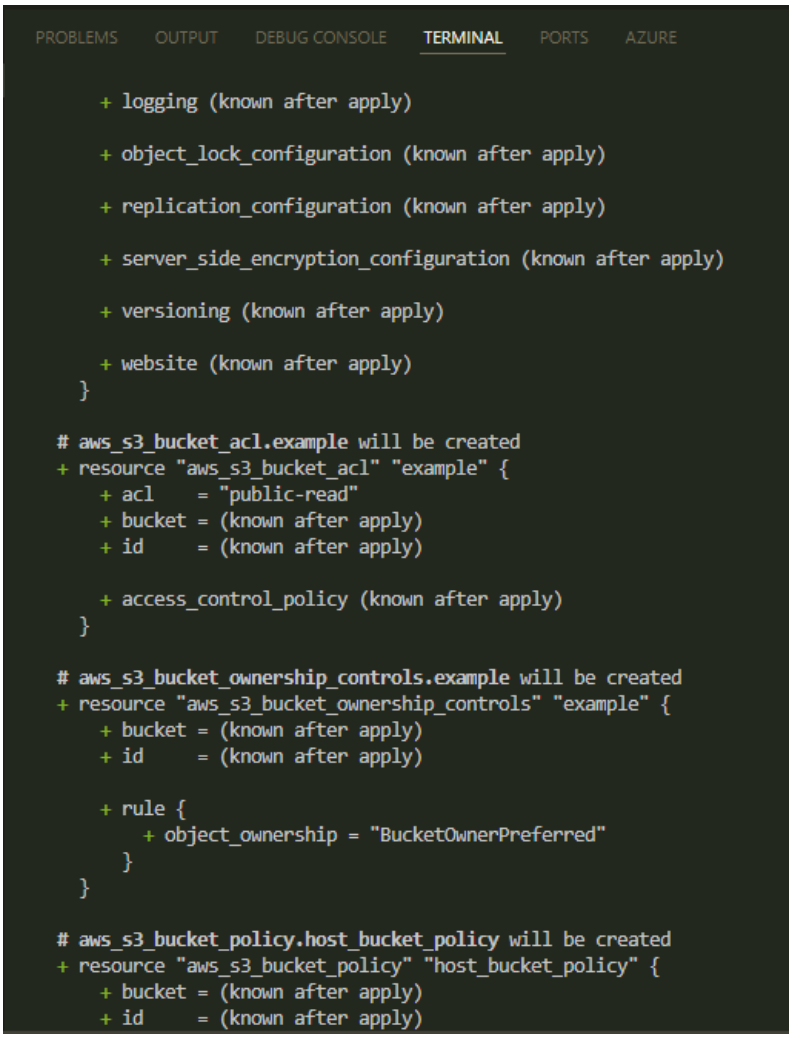
```
  + resource "aws_s3_bucket_policy" "host_bucket_policy" {
      + bucket = (known after apply)
      + id     = (known after apply)
      + policy = jsonencode(
            {
              + Statement = [
                  + {
                      + Action    = "s3:GetObject"
                      + Effect    = "Allow"
                      + Principal = "*"
                      + Resource  = "arn:aws:s3:::demo-terraform-s3bucket/*"
                    },
                ]
              + Version   = "2012-10-17"
            }
        )
    }

  # aws_s3_bucket_public_access_block.example will be created
  + resource "aws_s3_bucket_public_access_block" "example" {
      + block_public_acls       = false
      + block_public_policy     = false
      + bucket                  = (known after apply)
      + id                      = (known after apply)
      + ignore_public_acls      = false
      + restrict_public_buckets = false
    }

  # aws_s3_bucket_website_configuration.web-config will be created
  + resource "aws_s3_bucket_website_configuration" "web-config" {
      + bucket           = (known after apply)
      + id               = (known after apply)
      + routing_rules    = (known after apply)
      + website_domain   = (known after apply)
      + website_endpoint = (known after apply)
```


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
AZURE
+ website_endpoint = (known after apply)
+ index_document {
+ suffix = "index.html"
}
+ routing_rule (known after apply)
}
# aws_s3_object.Bucket_files["index.html"] will be created
+ resource "aws_s3_object" "Bucket_files" {
+ acl = (known after apply)
+ arn = (known after apply)
+ bucket = (known after apply)
+ bucket_key_enabled = (known after apply)
+ checksum_crc32 = (known after apply)
+ checksum_crc32c = (known after apply)
+ checksum_sha1 = (known after apply)
+ checksum_sha256 = (known after apply)
+ content_type = "text/html; charset=utf-8"
+ etag = "706be2e258c7c90ddaa6d23b17cefa4a"
+ force_destroy = false
+ id = (known after apply)
+ key = "index.html"
+ kms_key_id = (known after apply)
+ server_side_encryption = (known after apply)
+ source = "./webfiles/index.html"
+ storage_class = (known after apply)
+ tags_all = (known after apply)
+ version_id = (known after apply)
}
Plan: 7 to add, 0 to change, 0 to destroy.
Changes to Outputs:
+ website_url = (known after apply)

4. Execute terraform apply



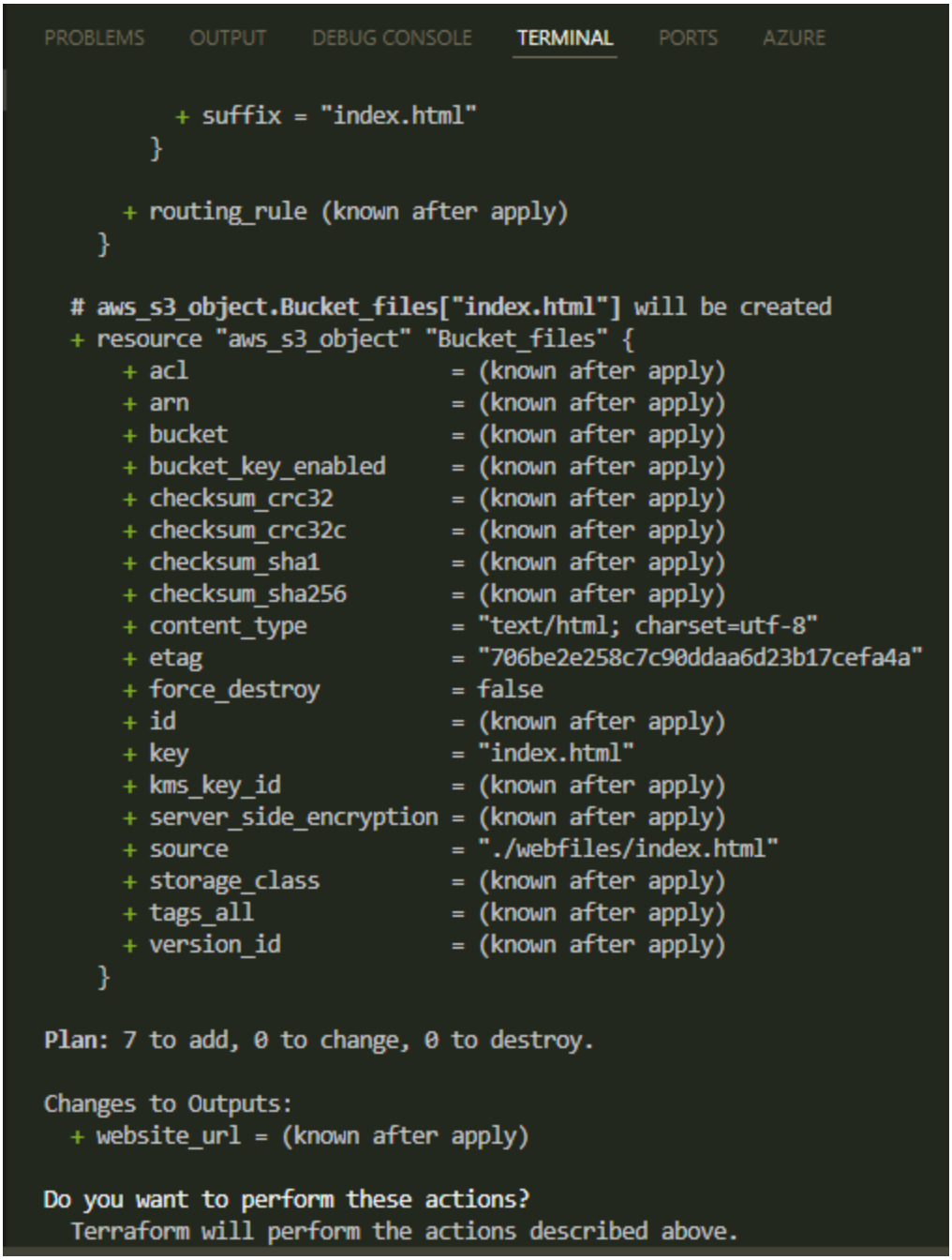
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE                                                    powershell

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
PS D:\TerraformScripts\S3 Website> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_s3_bucket.demo-bucket will be created
  + resource "aws_s3_bucket" "demo-bucket" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "demo-terraform-s3bucket"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags_all                    = (known after apply)
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + cors_rule (known after apply)

      + grant (known after apply)

      + lifecycle_rule (known after apply)

      + logging (known after apply)
```



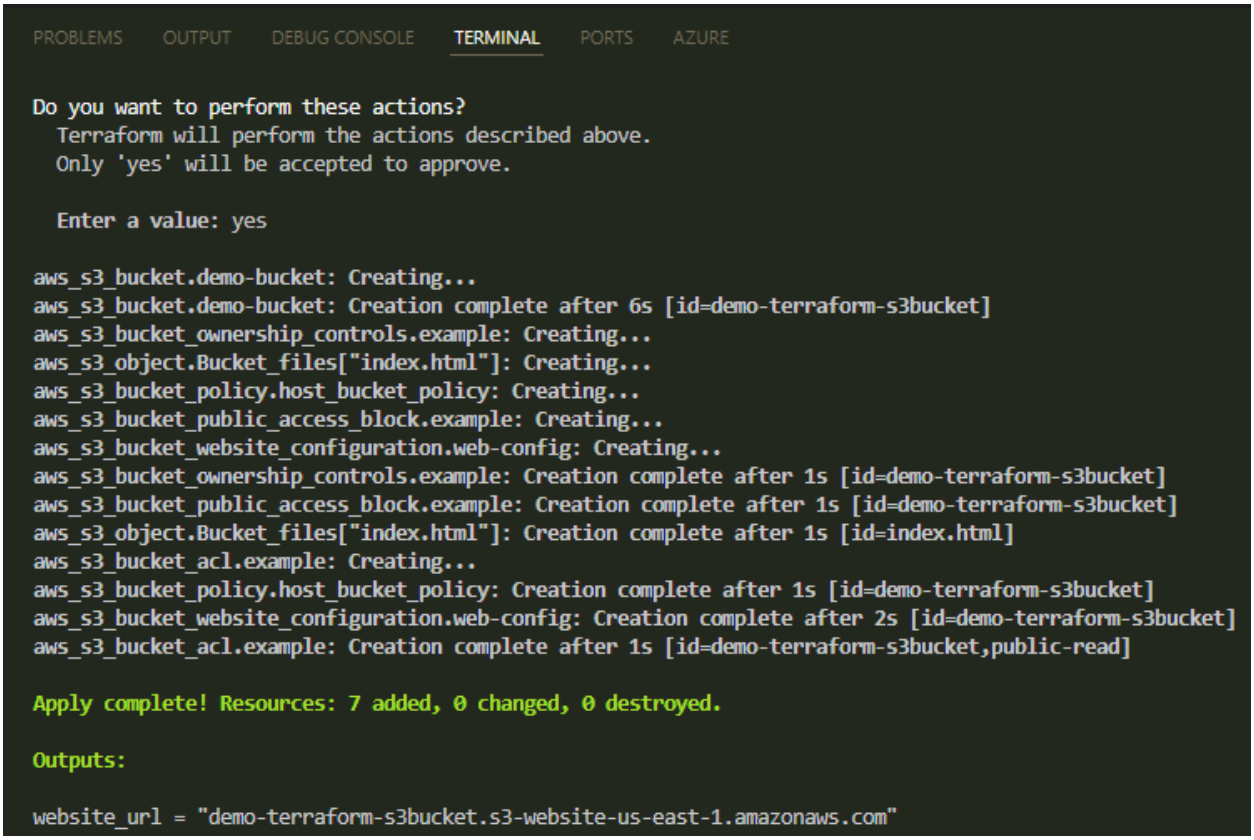
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE

      + logging (known after apply)

      + object_lock_configuration (known after apply)

      + replication_configuration (known after apply)

      + server_side_encryption_configuration (known after apply)

      + versioning (known after apply)

      + website (known after apply)
    }

  # aws_s3_bucket_acl.example will be created
  + resource "aws_s3_bucket_acl" "example" {
      + acl    = "public-read"
      + bucket = (known after apply)
      + id     = (known after apply)

      + access_control_policy (known after apply)
    }

  # aws_s3_bucket_ownership_controls.example will be created
  + resource "aws_s3_bucket_ownership_controls" "example" {
      + bucket = (known after apply)
      + id     = (known after apply)

      + rule {
          + object_ownership = "BucketOwnerPreferred"
        }
    }

  # aws_s3_bucket_policy.host_bucket_policy will be created
  + resource "aws_s3_bucket_policy" "host_bucket_policy" {
      + bucket = (known after apply)
      + id     = (known after apply)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

```
      + id     = (known after apply)
      + policy = jsonencode(
            {
              + Statement = [
                  + {
                      + Action    = "s3:GetObject"
                      + Effect    = "Allow"
                      + Principal = "*"
                      + Resource  = "arn:aws:s3:::demo-terraform-s3bucket/*"
                    },
                ]
              + Version   = "2012-10-17"
            }
        )
    }

  # aws_s3_bucket_public_access_block.example will be created
  + resource "aws_s3_bucket_public_access_block" "example" {
      + block_public_acls       = false
      + block_public_policy     = false
      + bucket                  = (known after apply)
      + id                      = (known after apply)
      + ignore_public_acls      = false
      + restrict_public_buckets = false
    }

  # aws_s3_bucket_website_configuration.web-config will be created
  + resource "aws_s3_bucket_website_configuration" "web-config" {
      + bucket           = (known after apply)
      + id               = (known after apply)
      + routing_rules    = (known after apply)
      + website_domain   = (known after apply)
      + website_endpoint = (known after apply)

      + index_document {
          + suffix = "index.html"
```



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE

          + suffix = "index.html"
        }

      + routing_rule (known after apply)
    }

  # aws_s3_object.Bucket_files["index.html"] will be created
  + resource "aws_s3_object" "Bucket_files" {
      + acl                    = (known after apply)
      + arn                    = (known after apply)
      + bucket                 = (known after apply)
      + bucket_key_enabled     = (known after apply)
      + checksum_crc32         = (known after apply)
      + checksum_crc32c        = (known after apply)
      + checksum_sha1          = (known after apply)
      + checksum_sha256        = (known after apply)
      + content_type           = "text/html; charset=utf-8"
      + etag                   = "706be2e258c7c90ddaa6d23b17cefa4a"
      + force_destroy          = false
      + id                     = (known after apply)
      + key                    = "index.html"
      + kms_key_id             = (known after apply)
      + server_side_encryption = (known after apply)
      + source                 = "./webfiles/index.html"
      + storage_class          = (known after apply)
      + tags_all               = (known after apply)
      + version_id             = (known after apply)
    }

Plan: 7 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + website_url = (known after apply)

Do you want to perform these actions?
  Terraform will perform the actions described above.
```



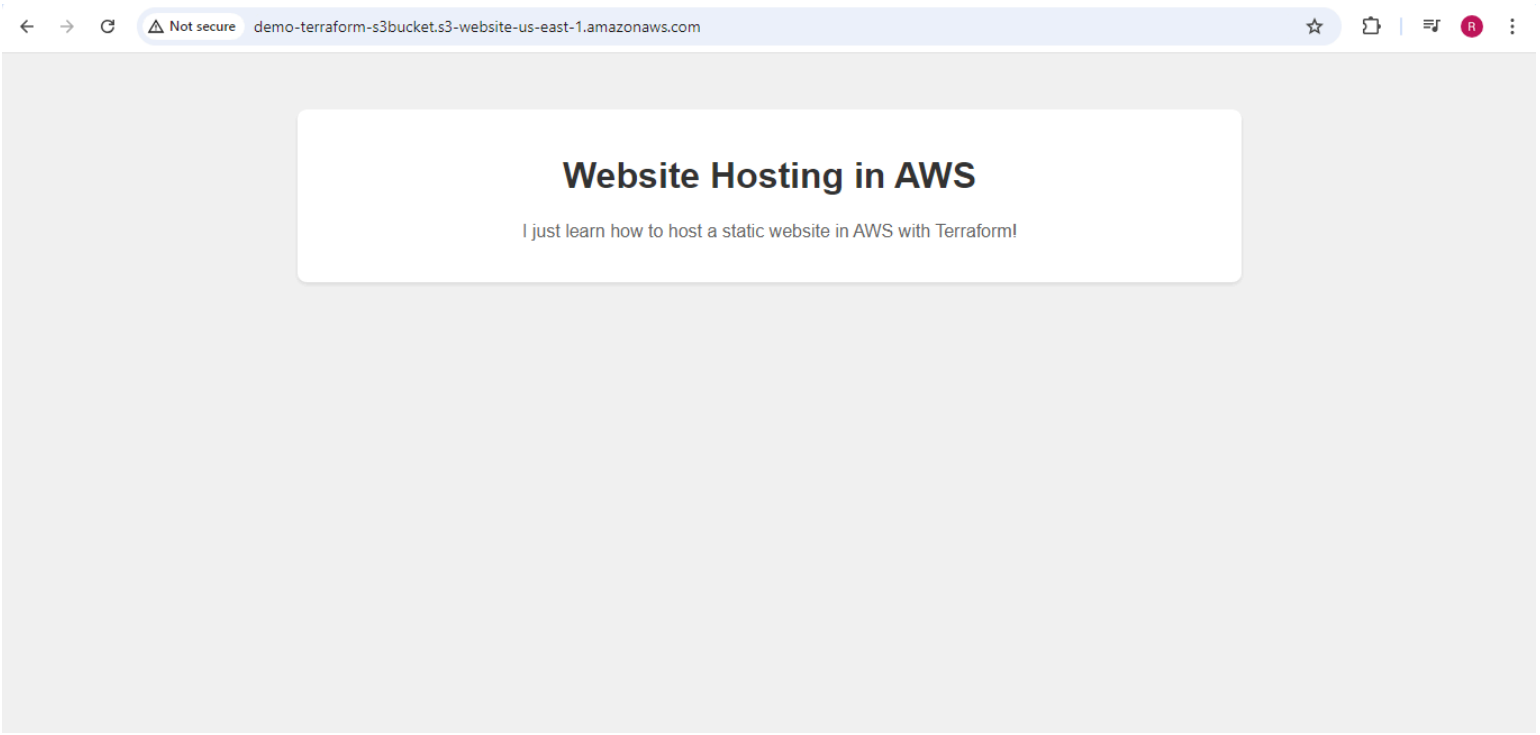
5. The “curl demo-terraform-s3bucket.s3-website-us-east-1.amazonaws.com” command is used to make an HTTP request to the specified URL, which corresponds to the static website hosted on an AWS S3 bucket. Here's what it does:

- **curl**: A command-line tool to transfer data from or to a server, commonly used to fetch web pages or APIs.

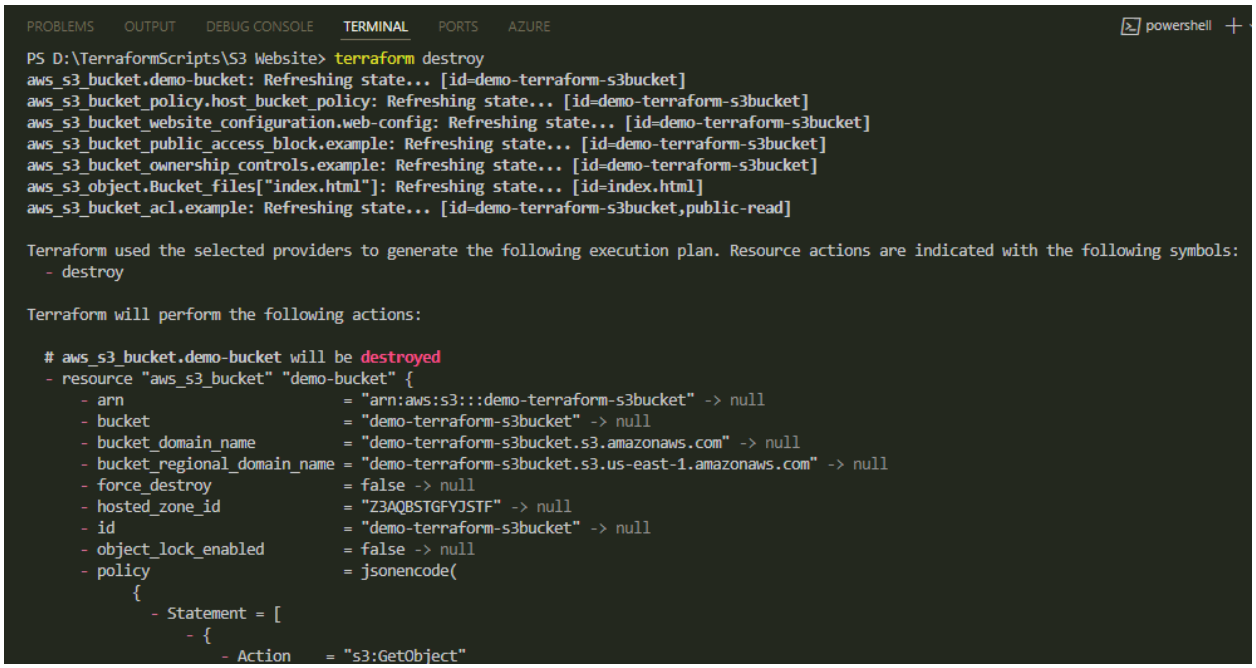
It fetches the content of the static website (likely an HTML page like `index.html`) stored in the S3 bucket and displays the response in the terminal.

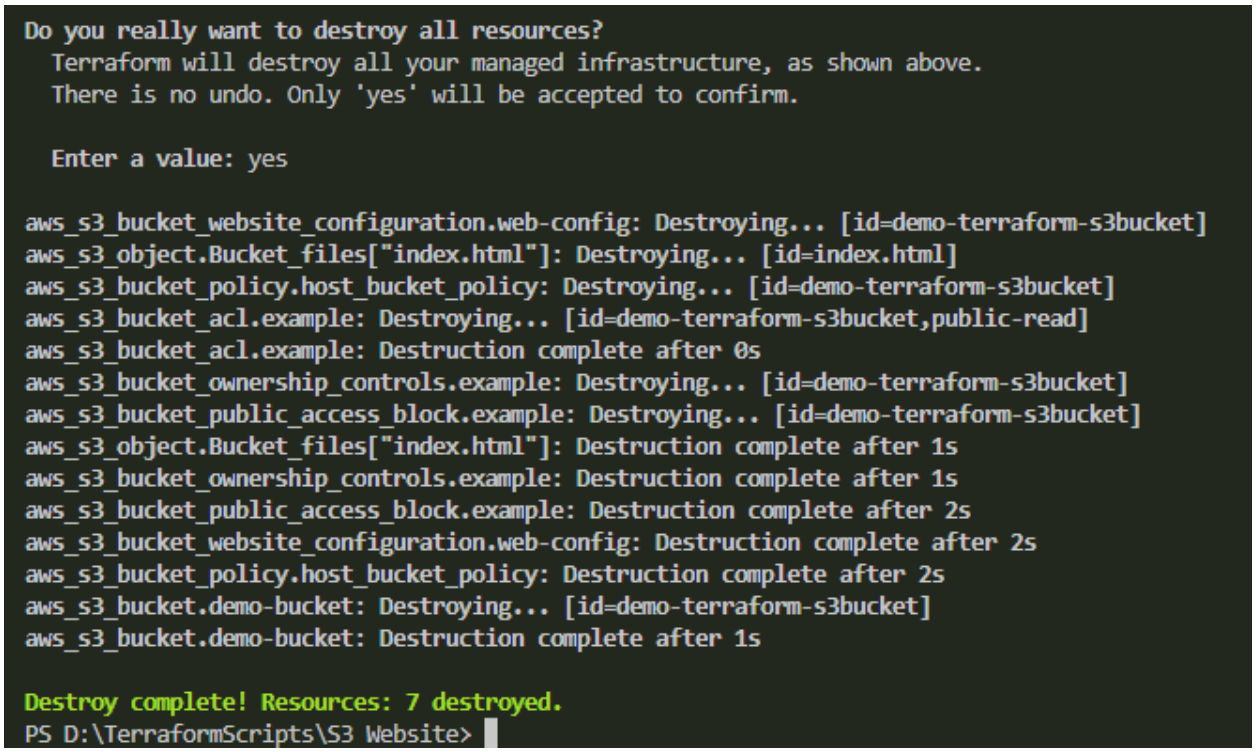
```
PS D:\TerraformScripts\S3 Website> curl demo-terraform-s3bucket.s3-website-us-east-1.amazonaws.com


StatusCode        : 200
StatusDescription : OK
Content           : <!DOCTYPE html>
                    <html lang="en">
                    <head>
                        <meta charset="UTF-8">
                        <meta name="viewport" content="width=device-width, initial-scale=1.0">
                        <title>Website Hosting in AWS</title>
                        <style...
RawContent        : HTTP/1.1 200 OK
                    x-amz-id-2: gqFshOXNgxgx0O4fpo0wFbu0gY++NLD0QFMz+hIsKG24/jwo64P6K8SKFTEpXlgTJhoYk4xm89Q=
                    x-amz-request-id: KYT3KHRDSSYFT5ZE
                    Content-Length: 990
                    Content-Type: text/html; charset=utf...
Forms             : {}
Headers           : {[x-amz-id-2, gqFshOXNgxgx0O4fpo0wFbu0gY++NLD0QFMz+hIsKG24/jwo64P6K8SKFTEpXlgTJhoYk4xm89Q=], [x-amz-request-id, KYT3KHRDSSYFT5ZE],
                    [Content-Length, 990], [Content-Type, text/html; charset=utf-8]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 990
```

6. Go to the browser and enter the url to see the hosted webpage.



7. Use “terraform destroy” to delete the bucket.





Adv. Devops Experiment no. 7

Name: Rohan Lalchandani
Class: D15A Roll no.: 25

Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Theory:

Static application security testing (SAST), or static analysis, is a testing methodology that analyzes source code to find security vulnerabilities that make your organization's applications susceptible to attack. SAST scans an application before the code is compiled.
It's also known as white box testing.

Why SAST is Important in DevOps ?

Early Detection of Vulnerabilities: SAST helps identify vulnerabilities early in the development phase, allowing developers to fix issues before the code is deployed to production.

Shifts Security Left: Incorporating SAST in DevOps practices supports the "Shift Left" approach in security, where testing starts earlier in the pipeline rather than waiting until the final stages.

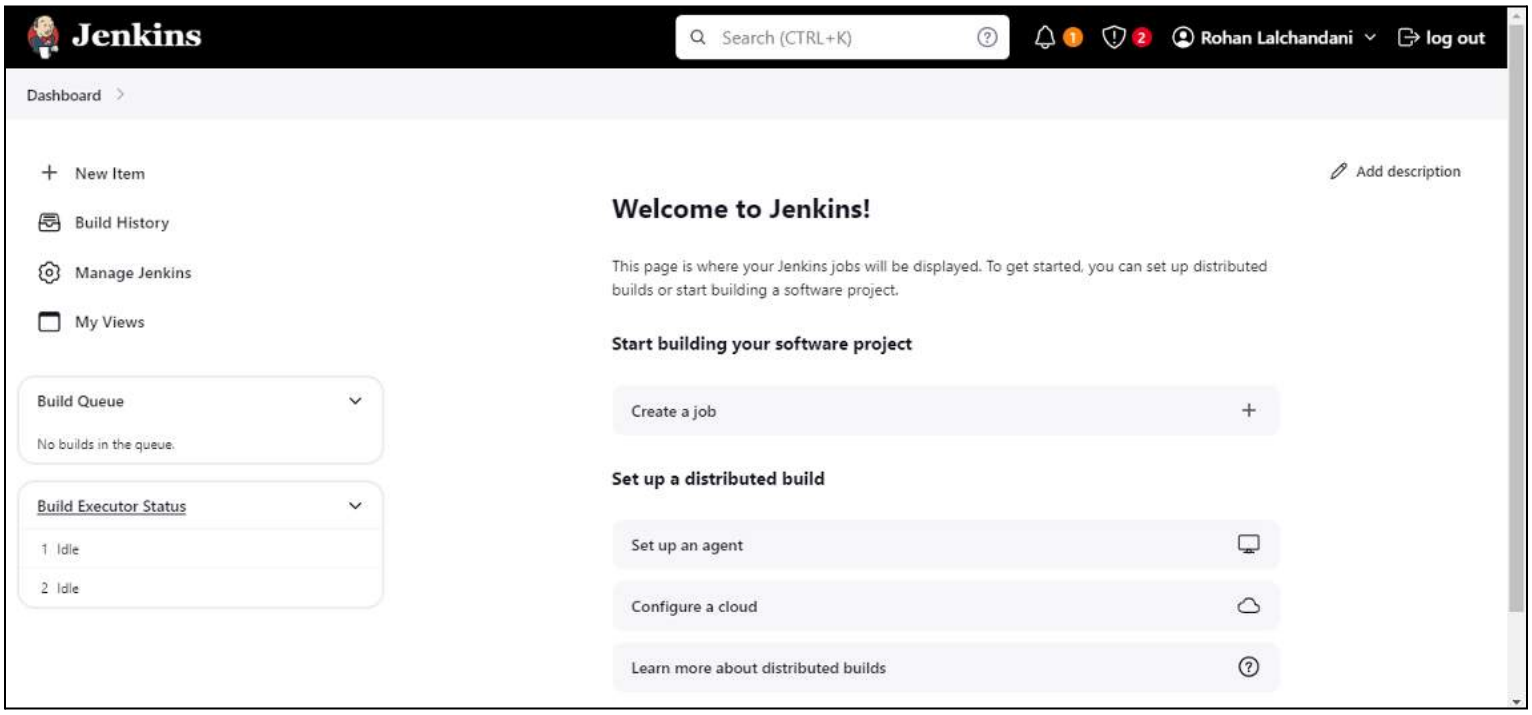
Automation-Friendly: SAST tools can be integrated with CI/CD pipelines, ensuring automated security checks on every code commit or pull request.

SAST in DevOps Workflow Example

1. **Code Commit**: Developers push code changes to a version control system like Git.
2. **Automated Build**: The CI system (e.g., Jenkins) automatically triggers a build and kicks off the SAST scan as part of the pipeline.
3. **SAST Scan**: The SAST tool scans the code and reports vulnerabilities if present. Some popular SAST tools for DevOps include:
4. **Build Failures or Warnings**: If critical vulnerabilities are found, the build fails or issues warnings, depending on the security policies in place.
5. **Developer Feedback**: Developers receive feedback, either through IDE plugins, the CI/CD dashboard, or via notifications, so they can address the issues quickly.
6. **Security Approval**: Once all critical vulnerabilities are resolved, the build proceeds, and the application can move to the next stage of deployment.

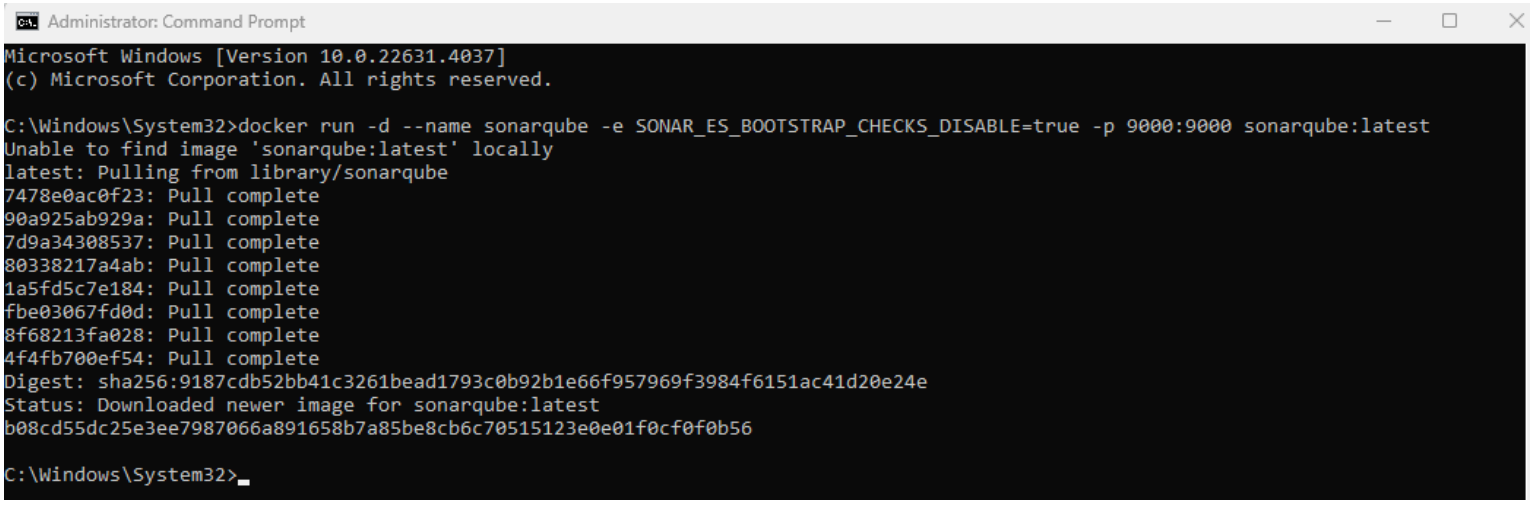
Implementation:

Step 1: Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.

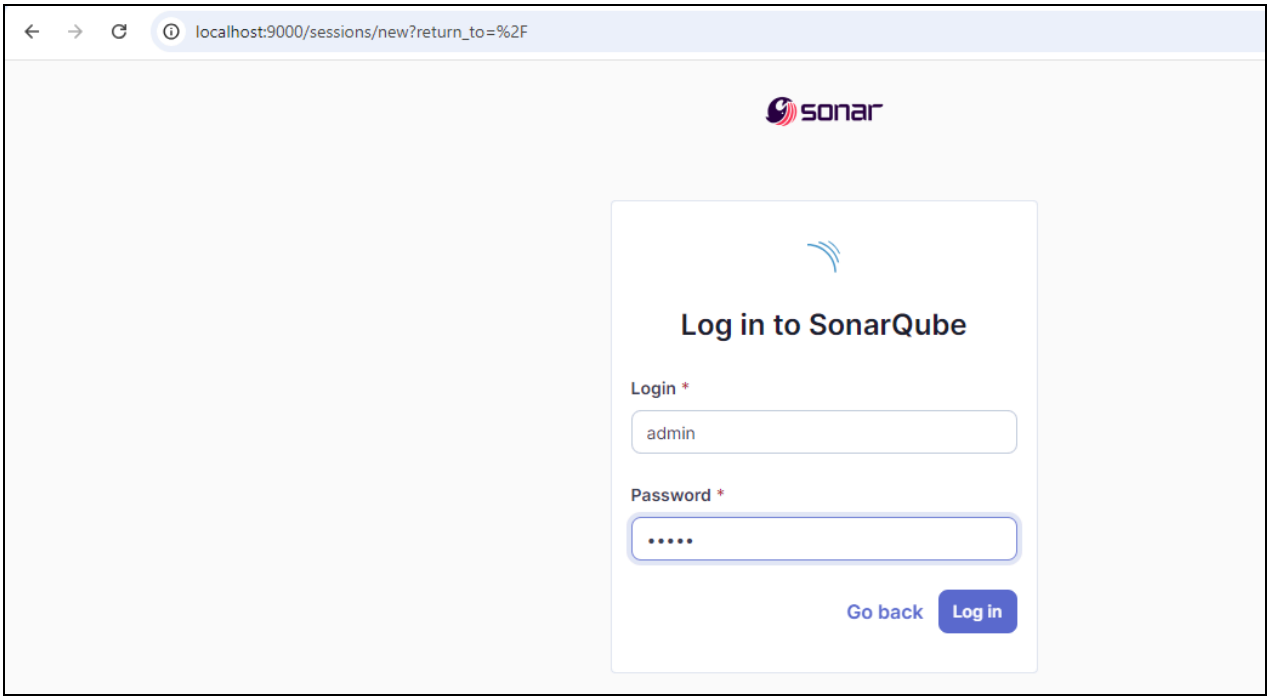


Step 2: Run Sonarqube’s image using the following command.

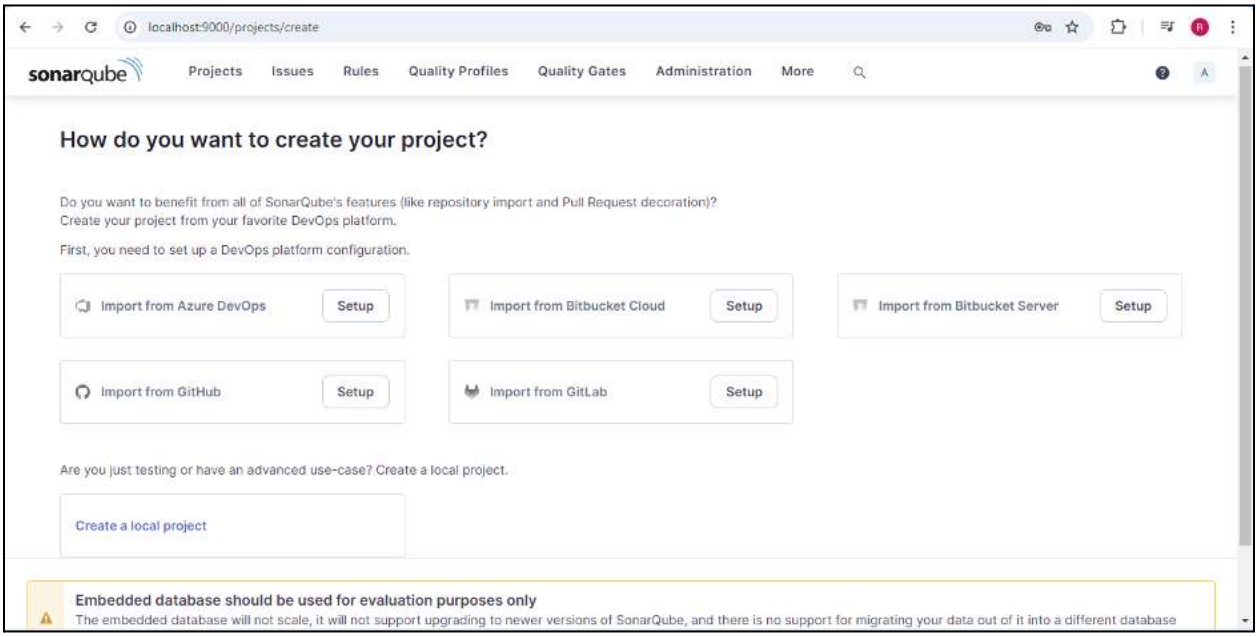
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest



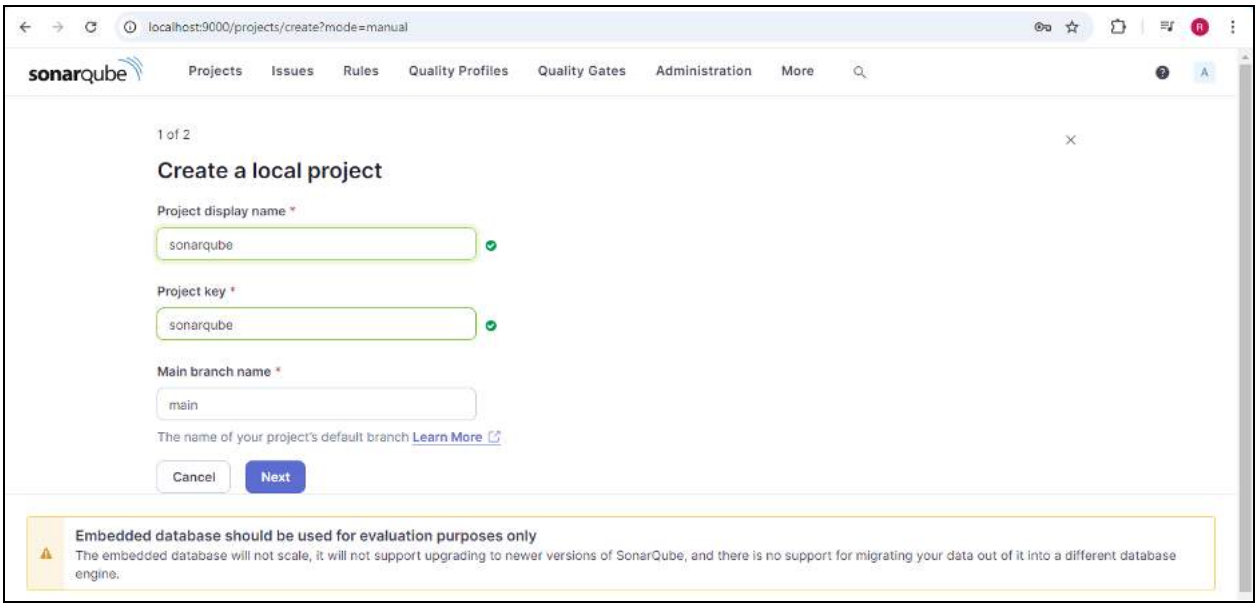
Once the container is up and running, you can check the status of SonarQube at localhost port 9000.



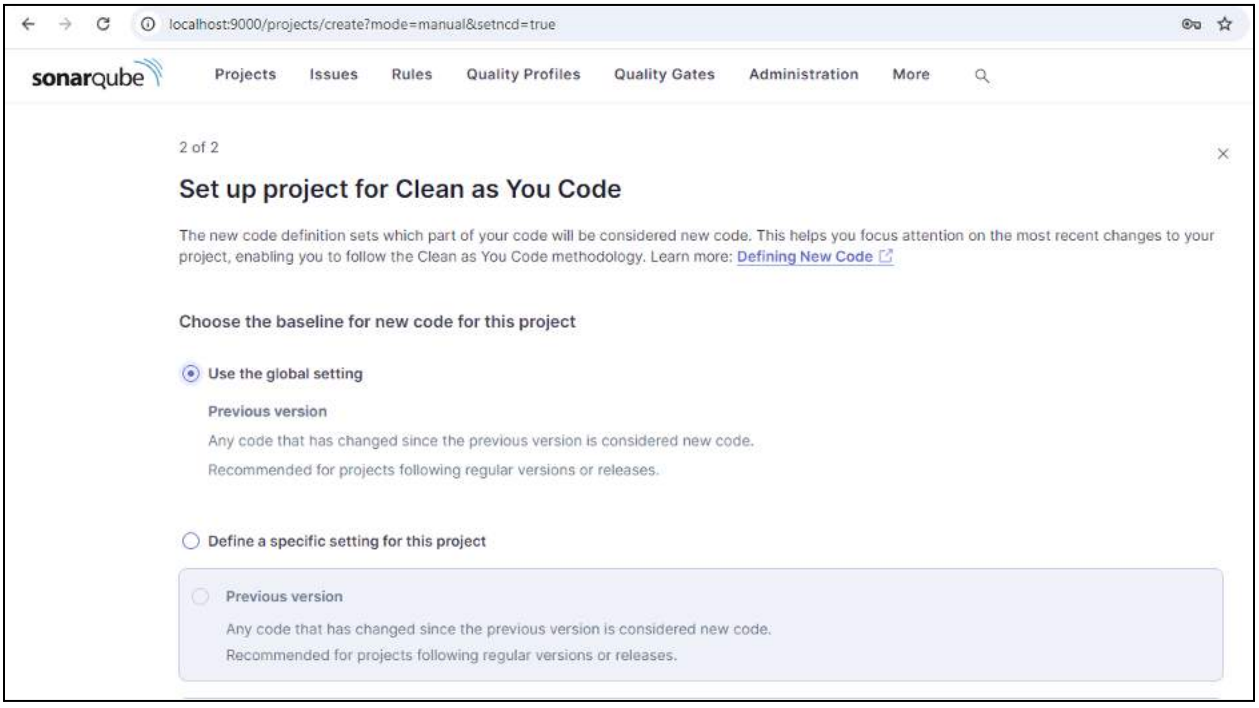
4.Login to SonarQube using username admin and password admin.



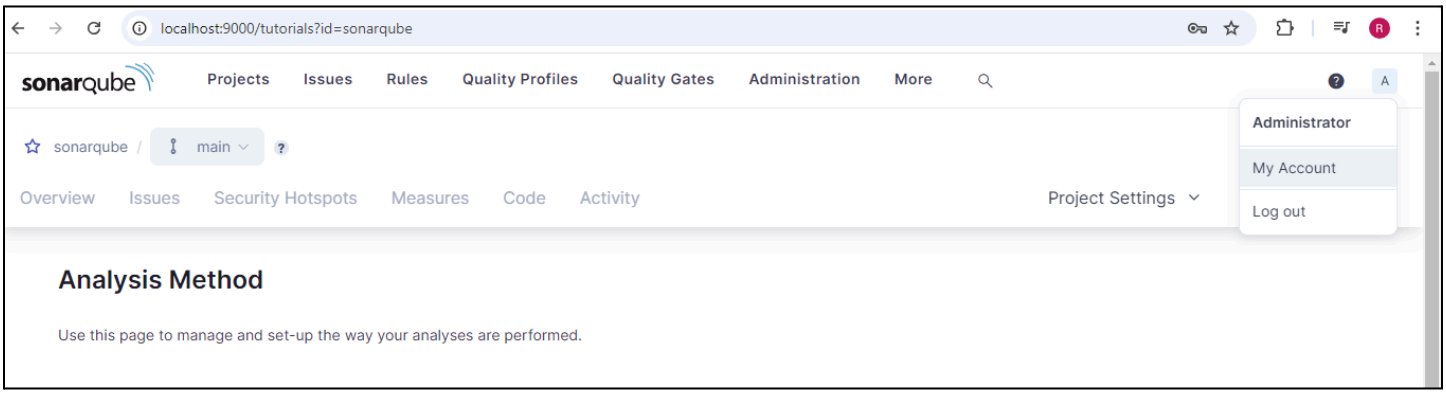
5.Create a manual project in SonarQube with the name sonarqube



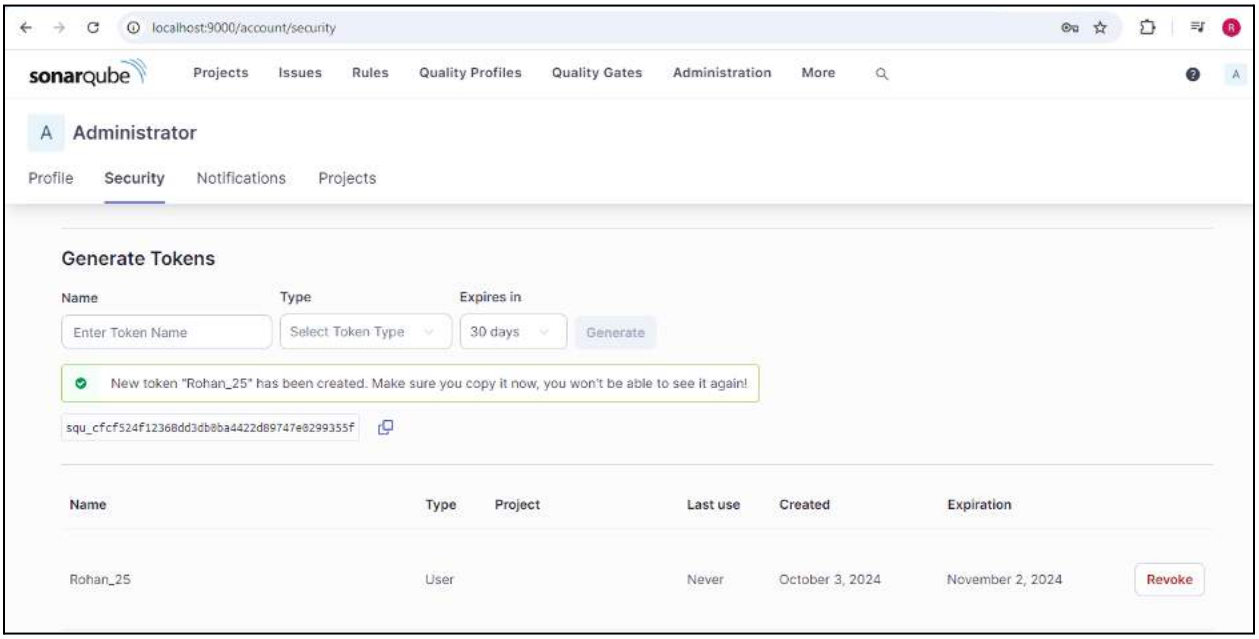
Setup the project by selecting the following options.



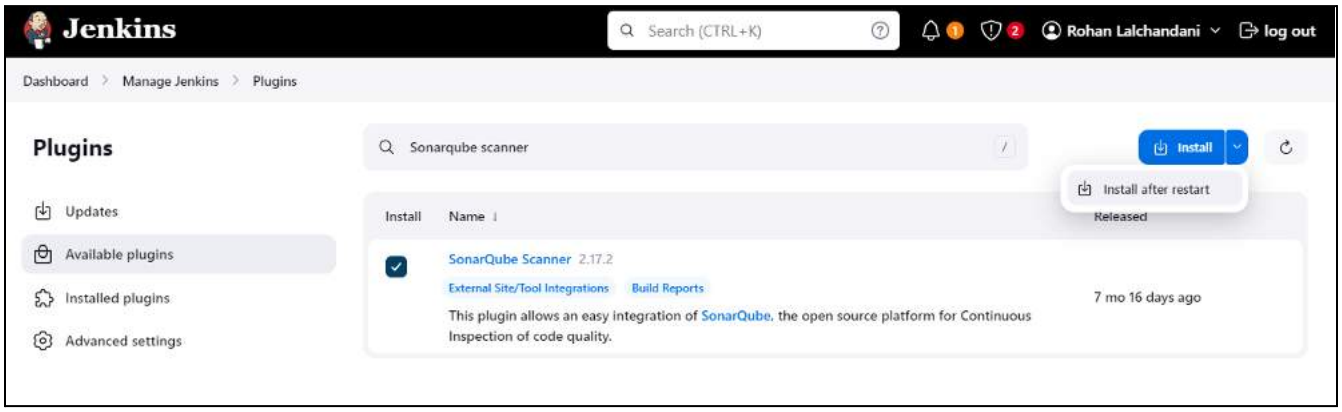
Now, for setting up sonarqube in Jenkins we need to have the login and password, for that we need to generate token.
Go to My Account > Security

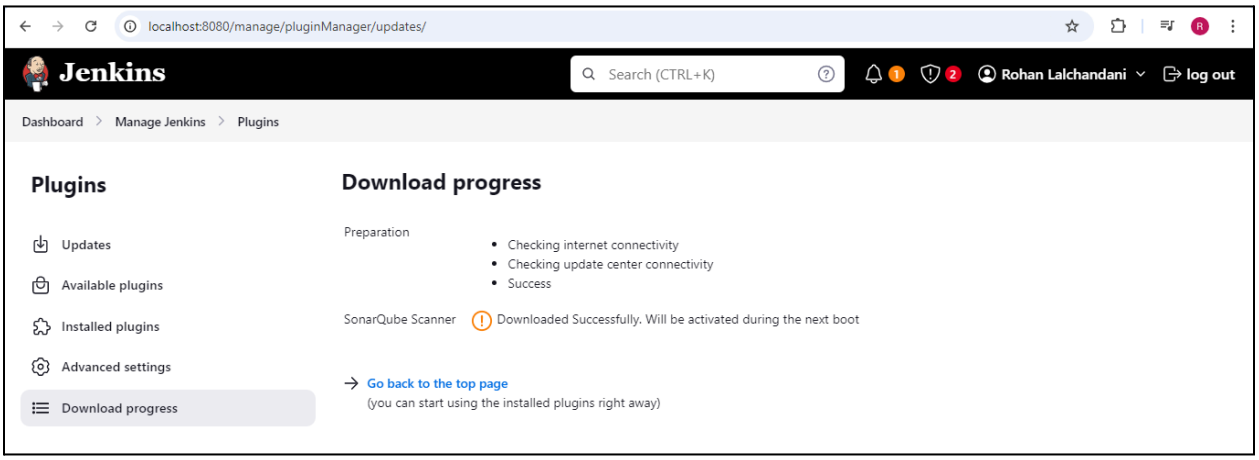


Type the details and click on generate token, copy the token and save it in a notepad.

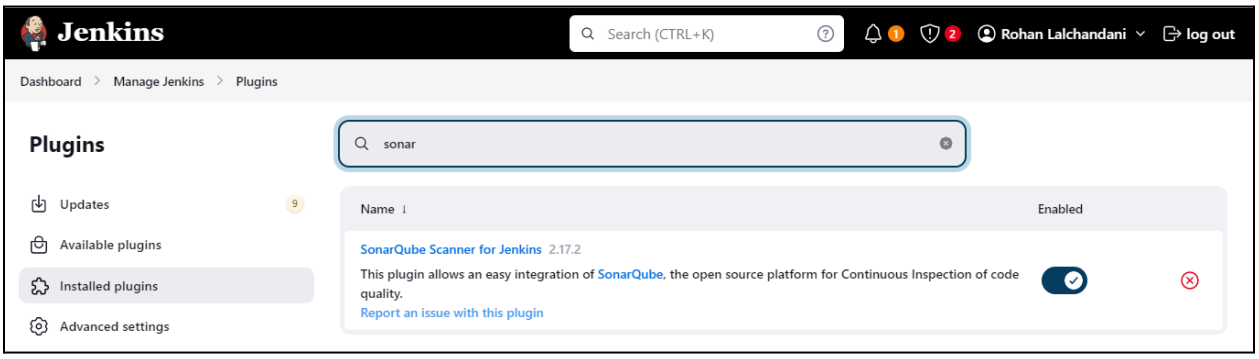


Setup the project and come back to Jenkins Dashboard. Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.

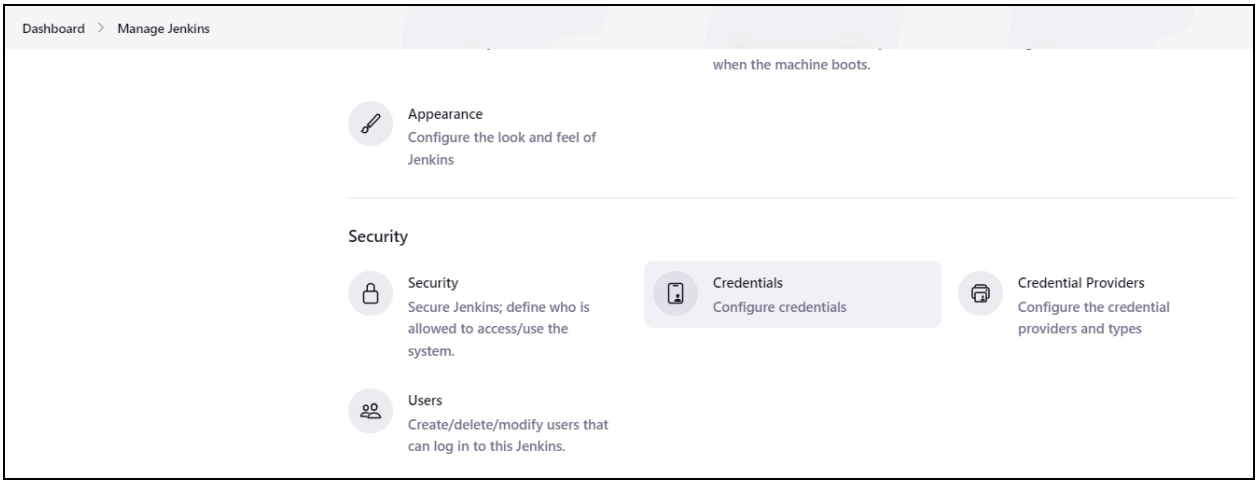




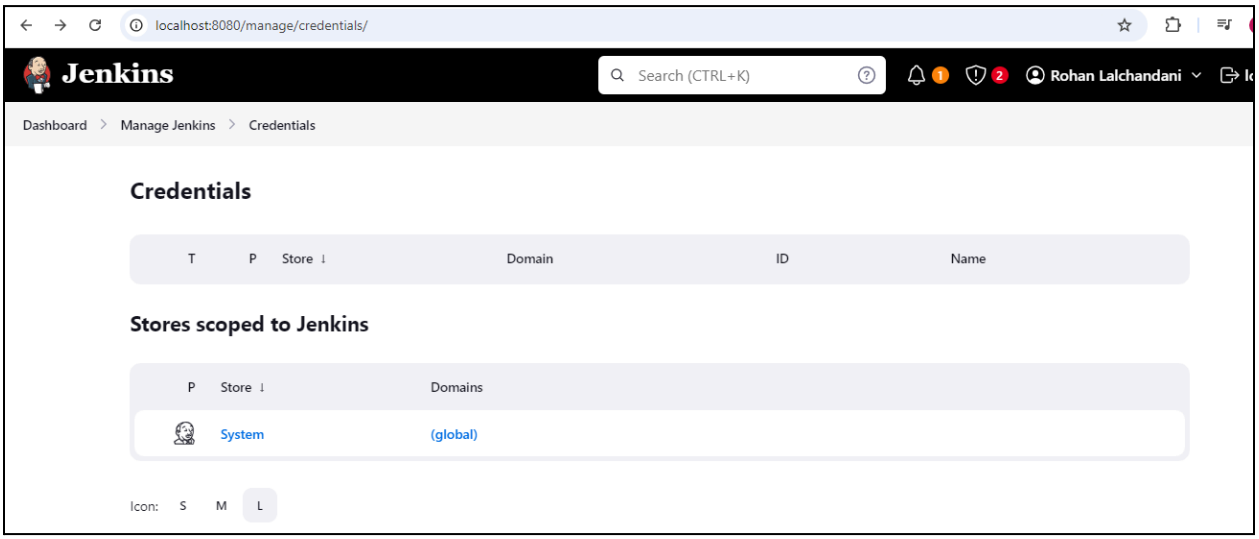
Go to Installed plugins to see if sonarqube scanner is successfully installed



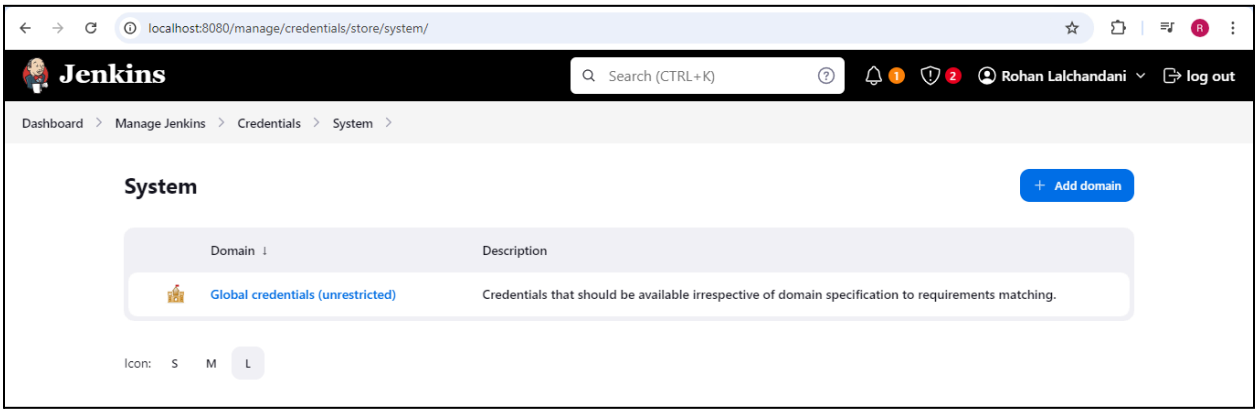
Go to Manage Jenkins > Credentials



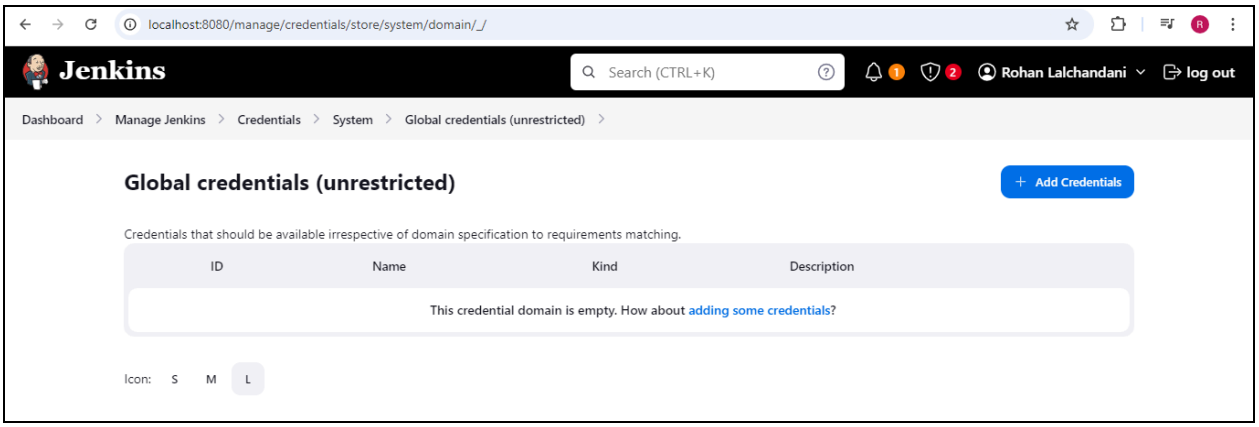
Click on System



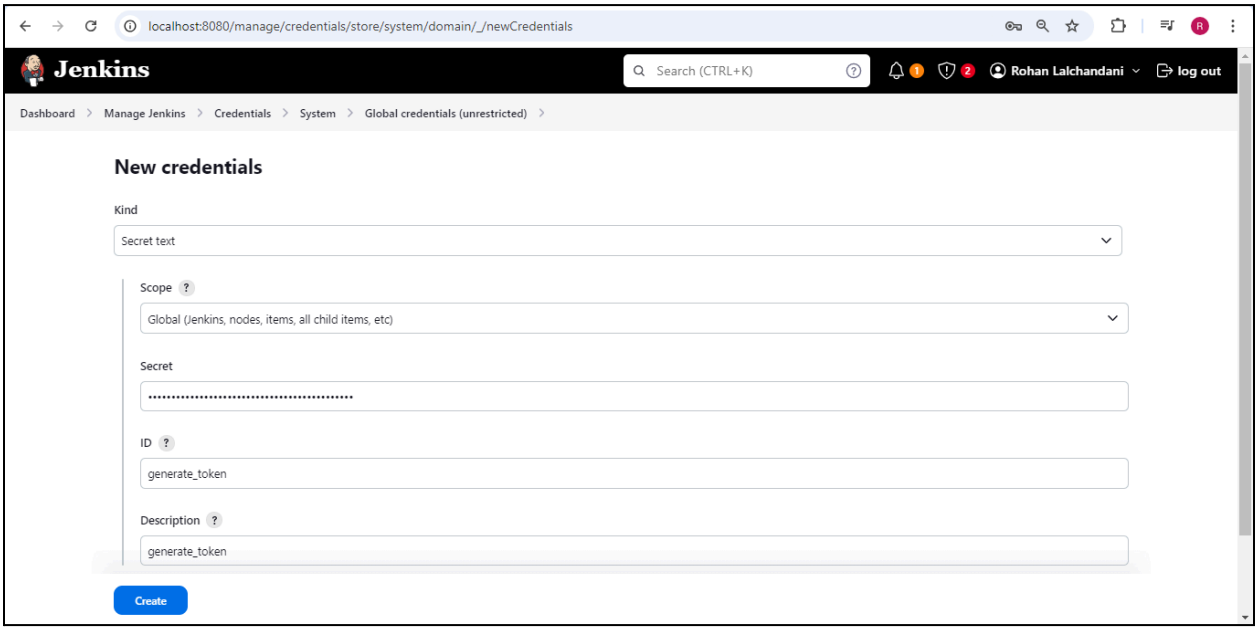
Click on Global Credentials



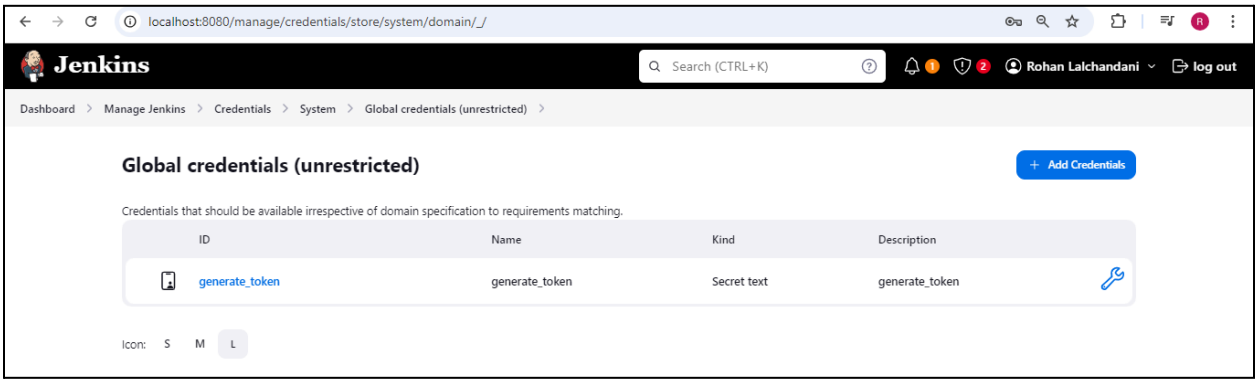
Click on Add Credentials



Now paste the generated token from sonarqube in the secret box and fill in other details as shown below.

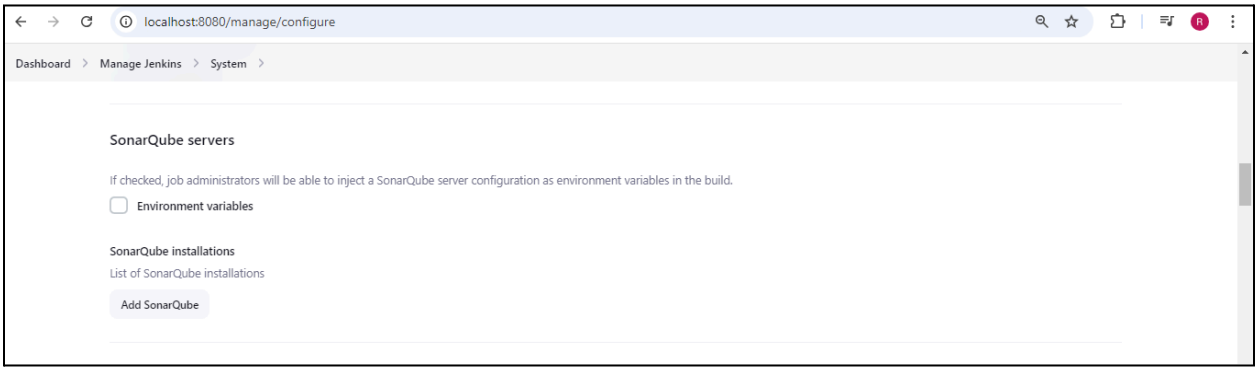


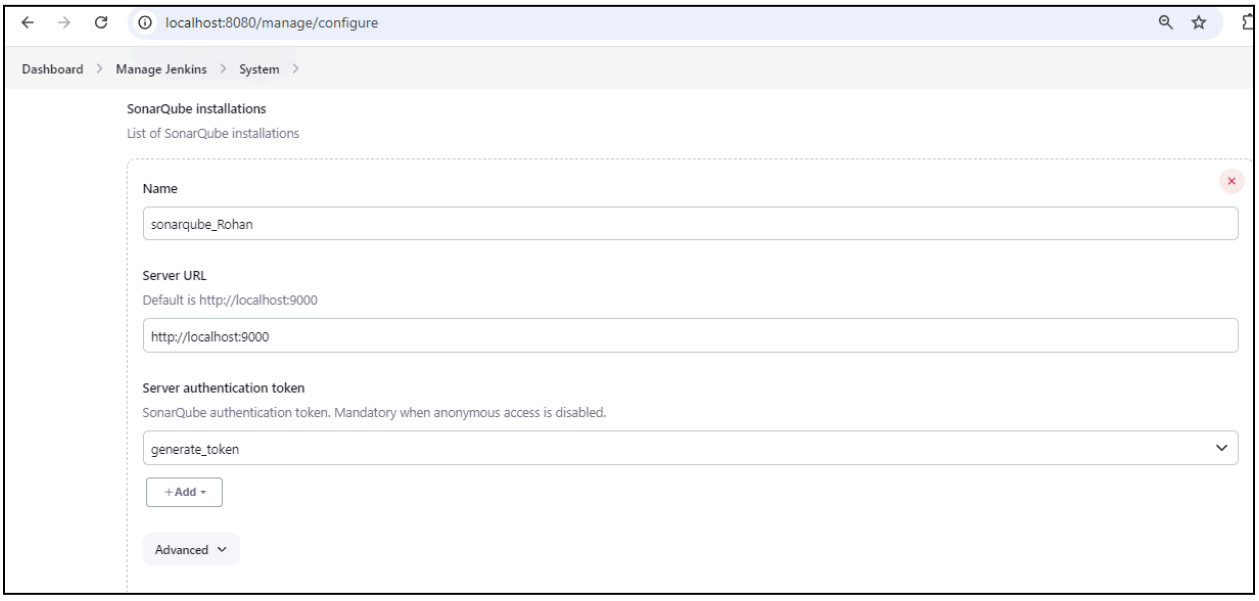
Token has been added successfully to the Jenkins



Configure SonarQube Scanner in Jenkins

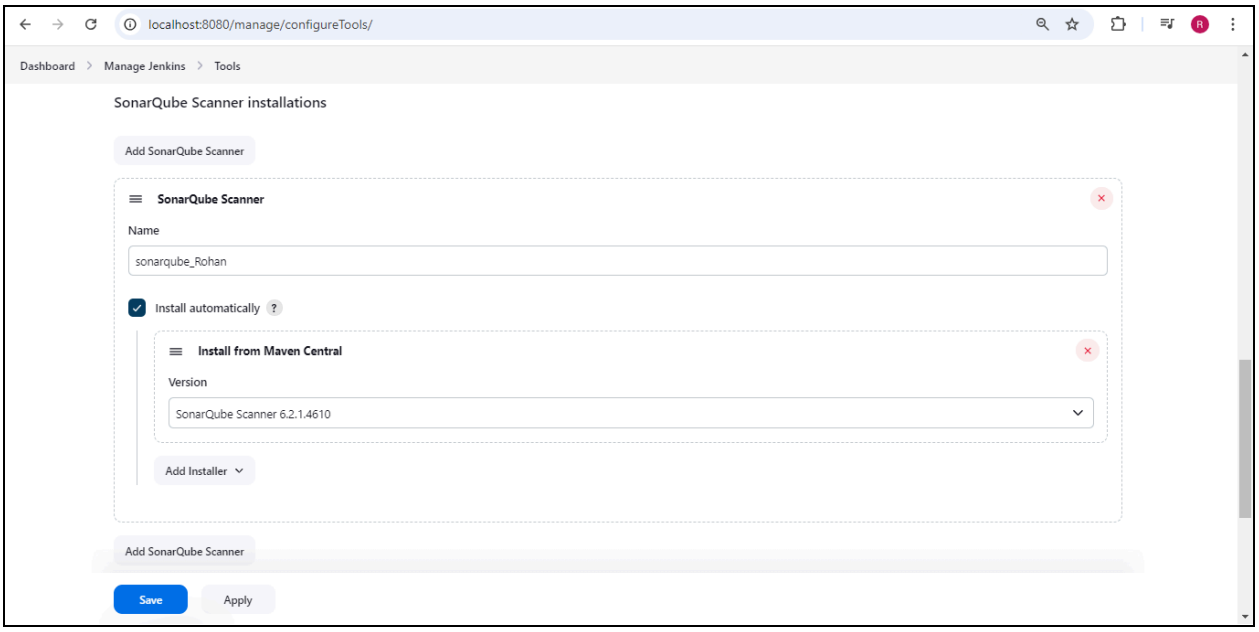
● Go to Manage Jenkins > System. Scroll down to Sonarqube servers, check on environment variables.
● Select he token from drop down menu.
● Click on save.



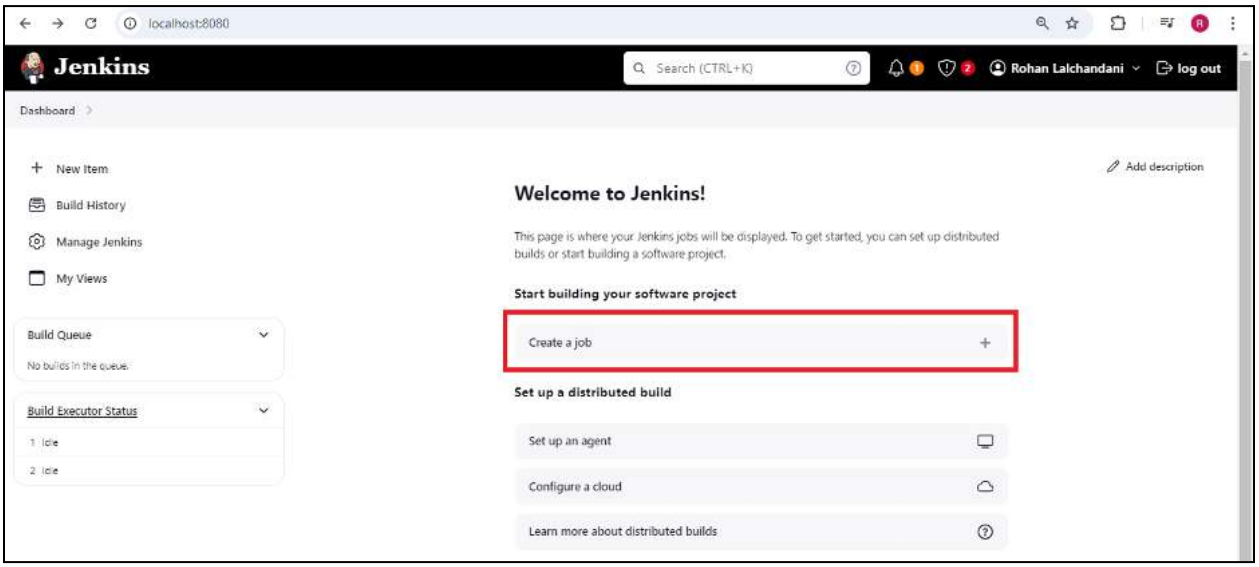


Configure SonarQube Scanner in Jenkins

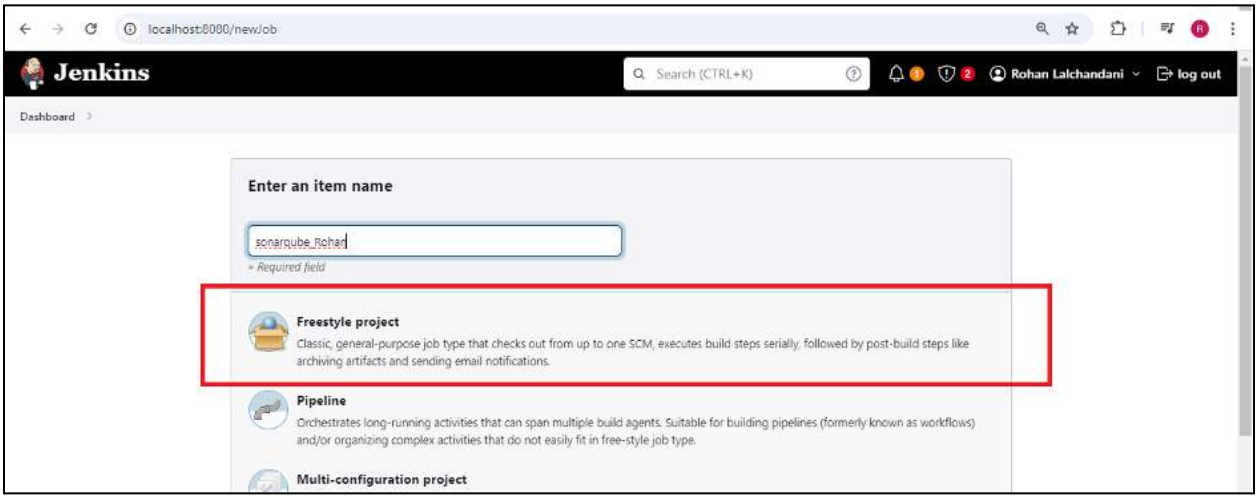
● Go to Manage Jenkins > Global Tool Configuration.
● Scroll down to SonarQube Scanner.
● Choose the latest version and select Install automatically



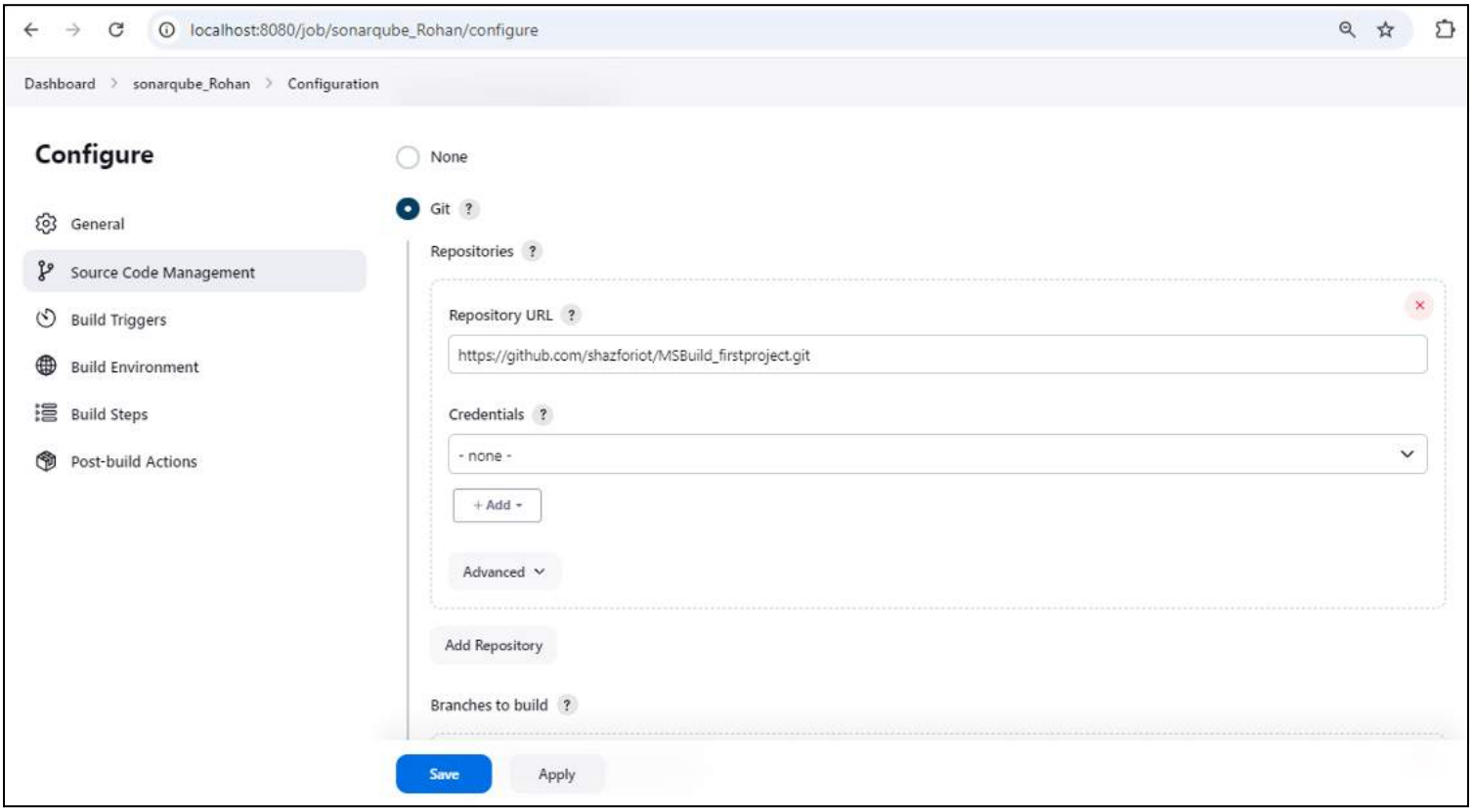
Go to your Jenkins Dashboard and Create a new Job



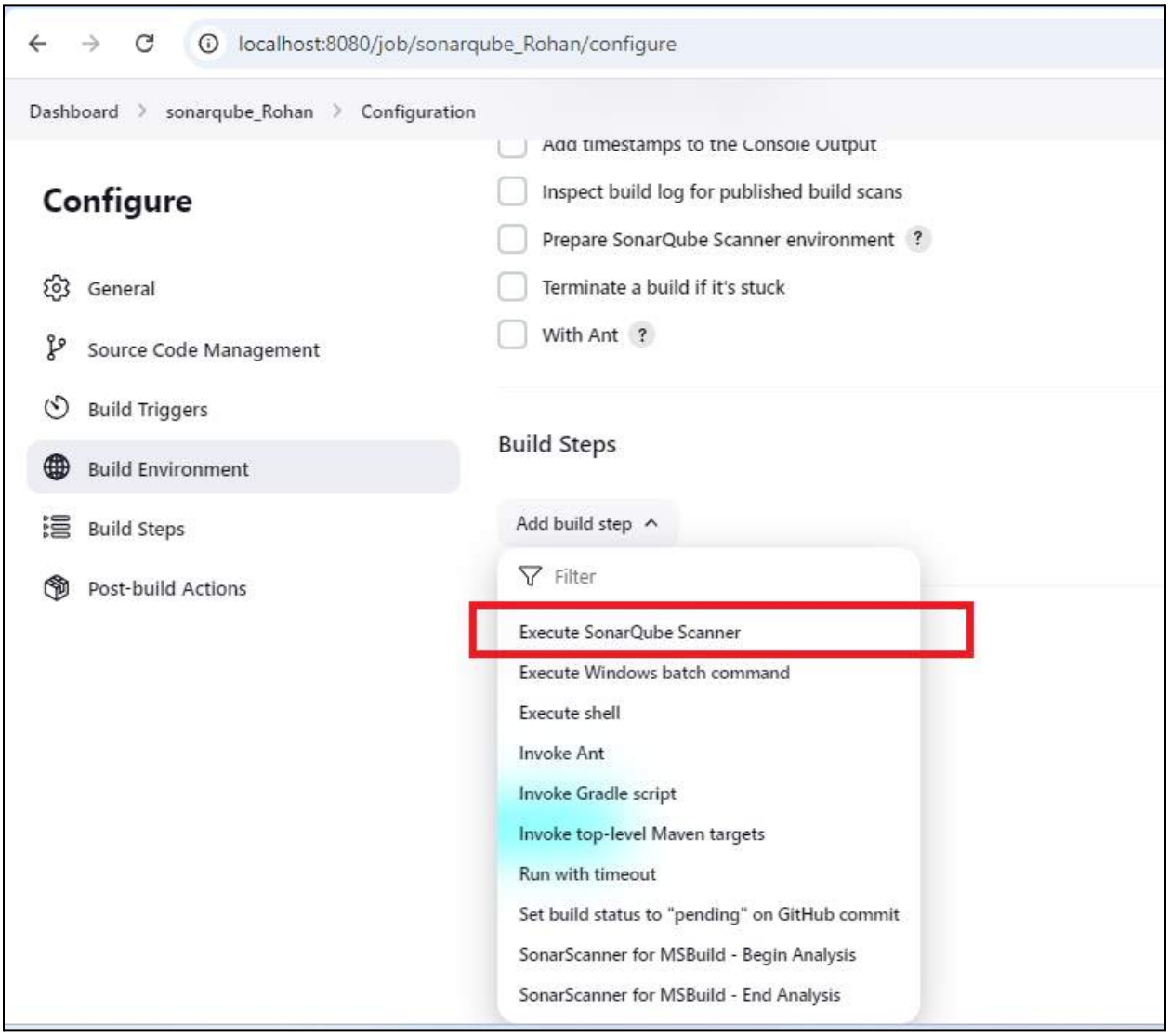
Enter name of the project and select freestyle project



Under Configure put the github link in source code management
https://github.com/shazforiot/MSBuild_firstproject.git

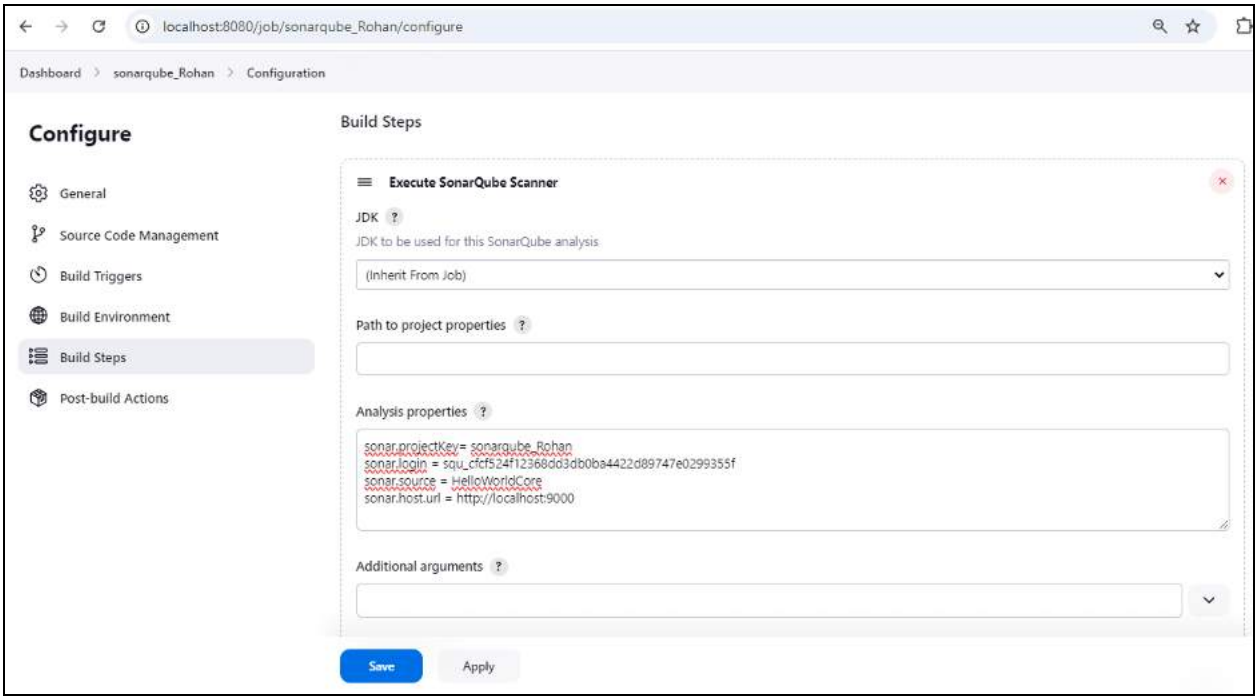


In build steps Select “Execute SonarQube scanner”

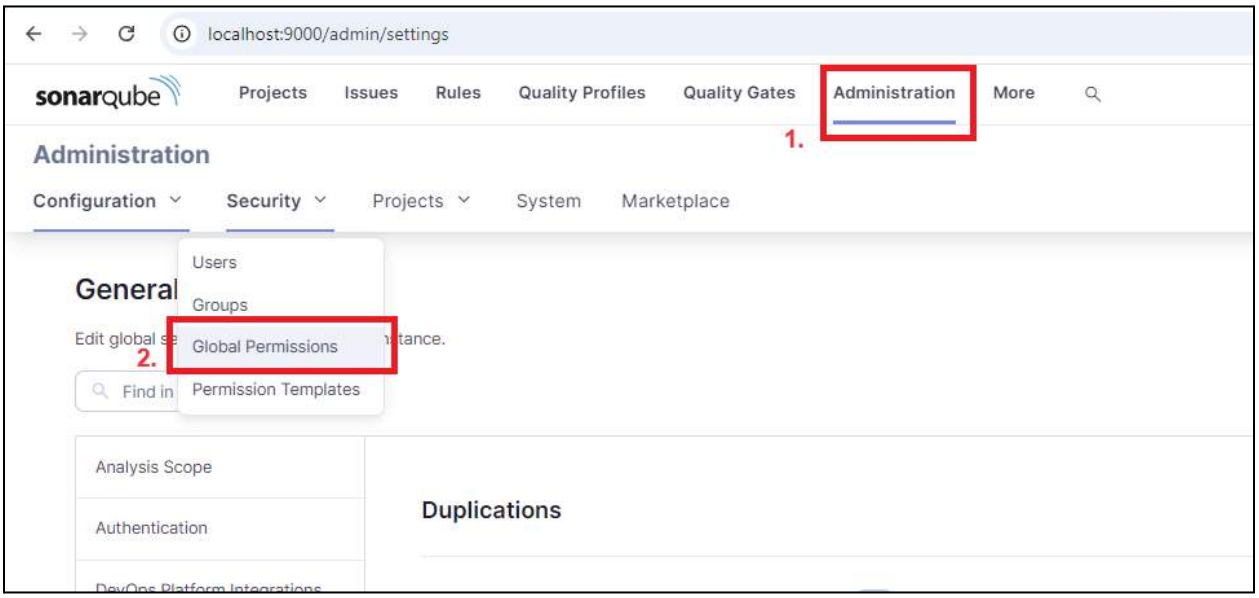


Type the following in analysis properties:

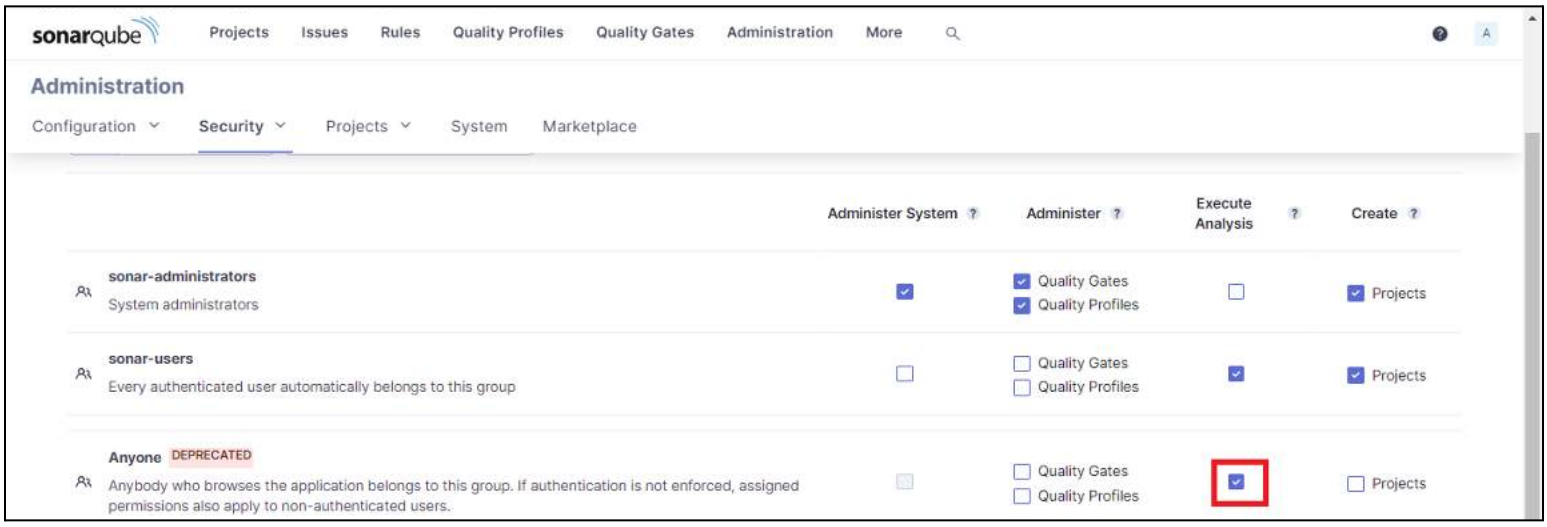
- sonar.projectKey=my_project_name
- sonar.login=your_generated_token
- sonar.sources=HelloWorldCore
- sonar.host.url=http://localhost:9000



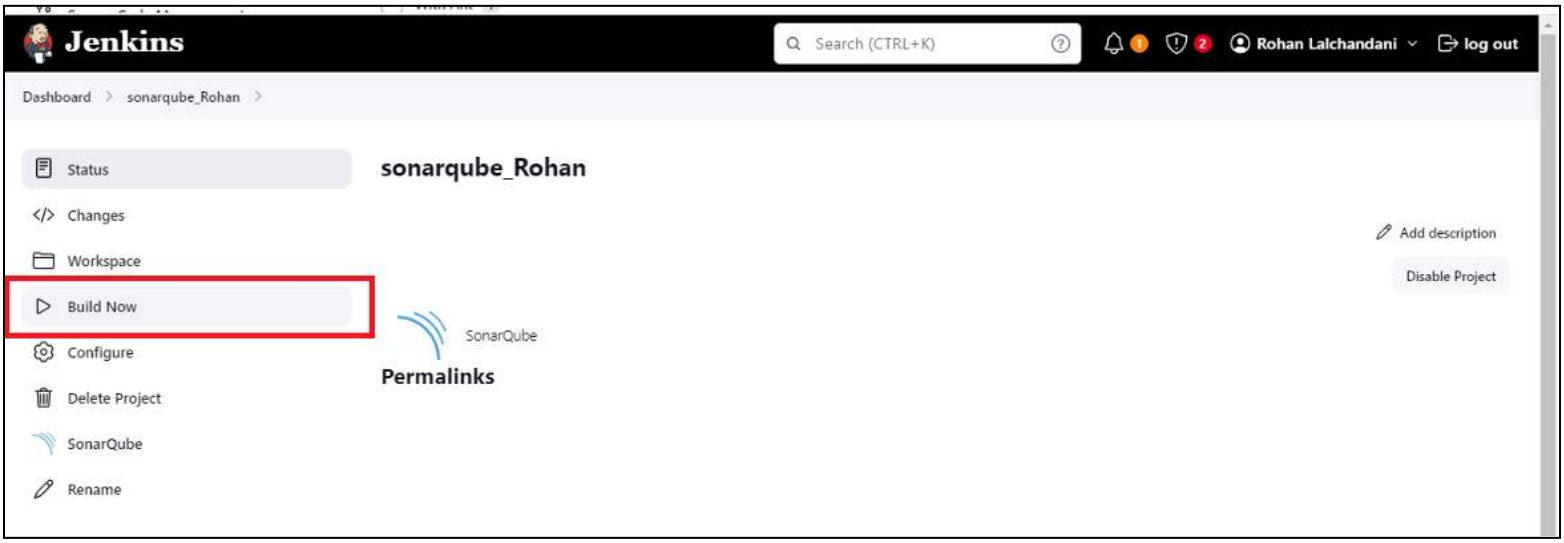
Now we need to enable access to sonarqube.
For that go to the sonarqube website
click on administration and then global permissions.



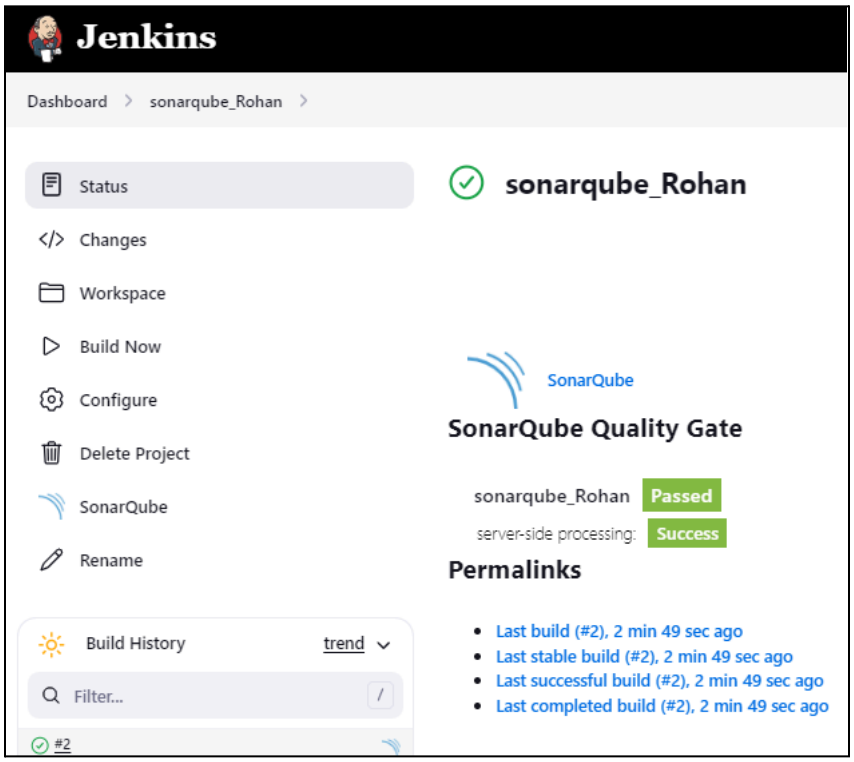
Click on Anyone in the checkbox as shown below.



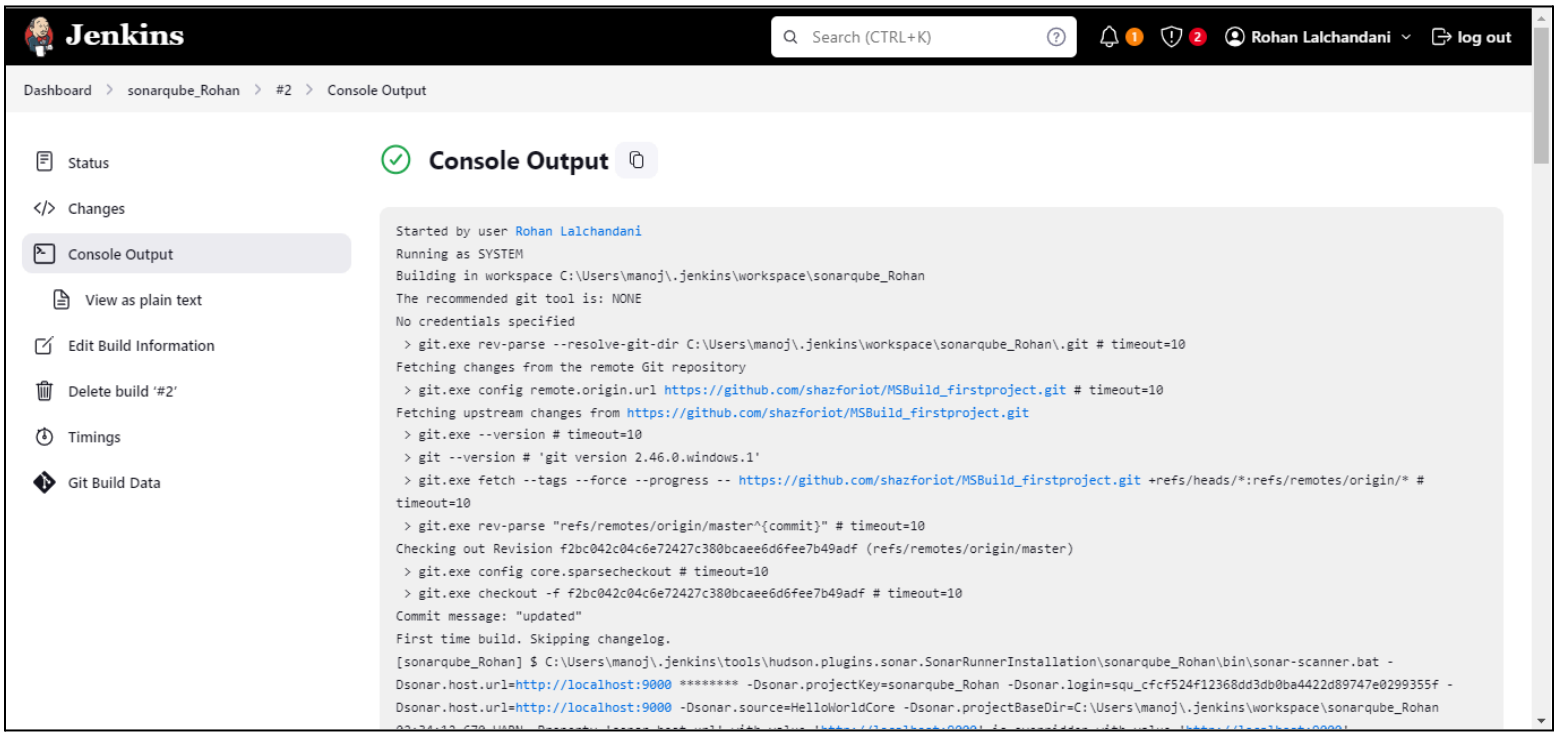
Go to the Jenkins Dashboard and click on “Build Now”



The build has been completed successfully.



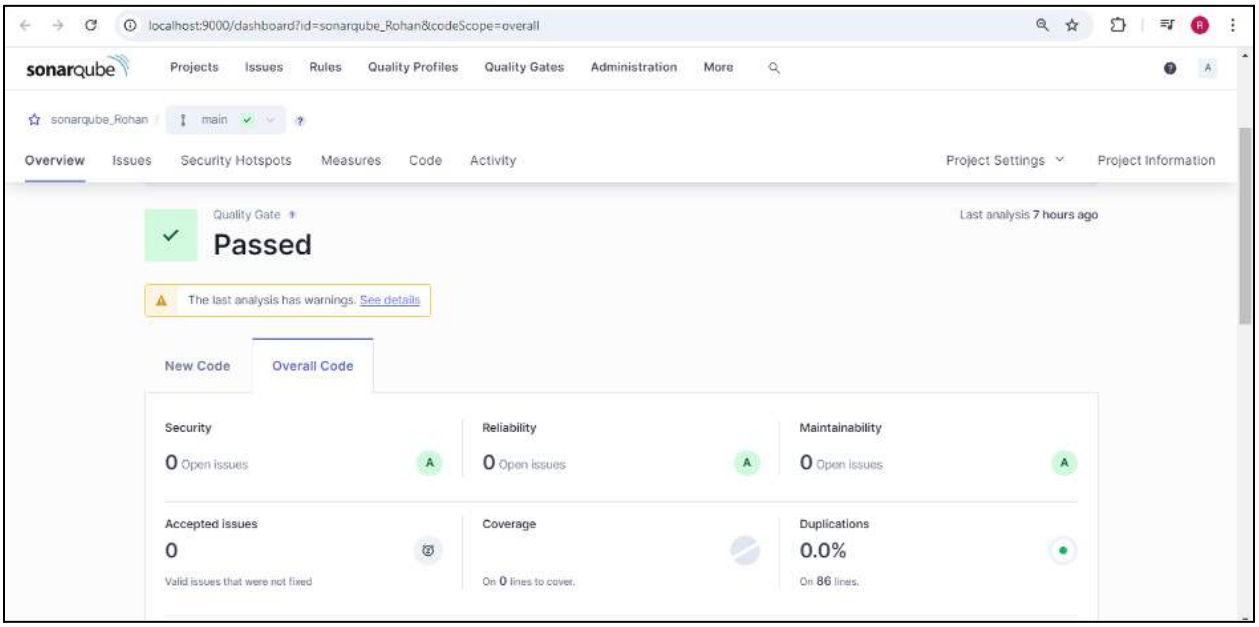
Go to the console output.



Scroll down click on the link as shown below to view the SonarQube analysis report.

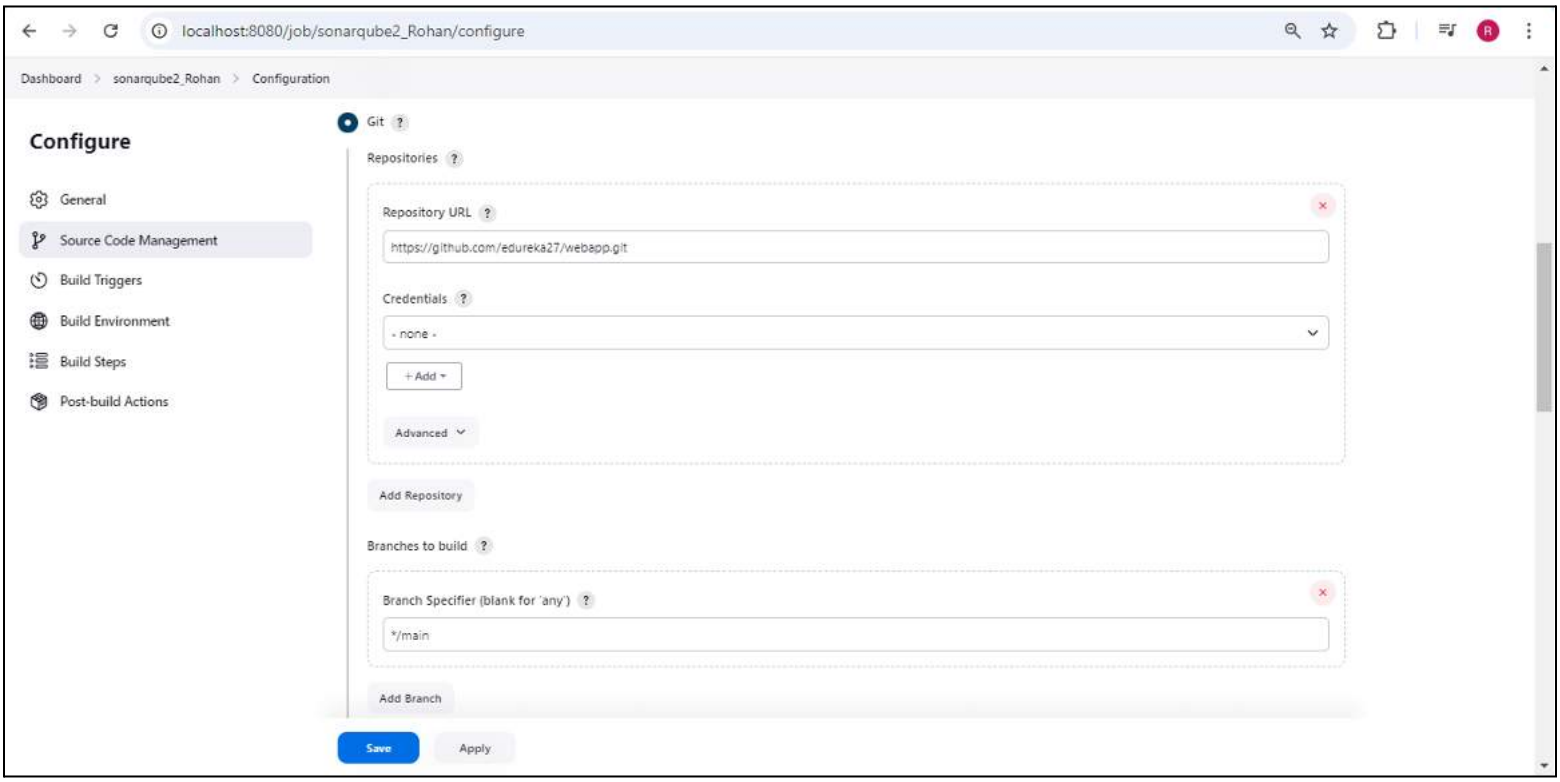
```
02:34:34.415 INFO  Analysis report compressed in 44ms, zip size=23.8 kB
02:34:34.448 INFO  Analysis report uploaded in 31ms
02:34:34.451 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube_Rohan
02:34:34.451 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
02:34:34.451 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=91e88697-16f7-4df8-9bcc-69a576460a6c
02:34:34.468 INFO  Analysis total time: 18.278 s
02:34:34.471 INFO  SonarScanner Engine completed successfully
02:34:34.558 INFO  EXECUTION SUCCESS
02:34:34.559 INFO  Total time: 21.879s
Finished: SUCCESS
```

You can see the code has successfully passed.



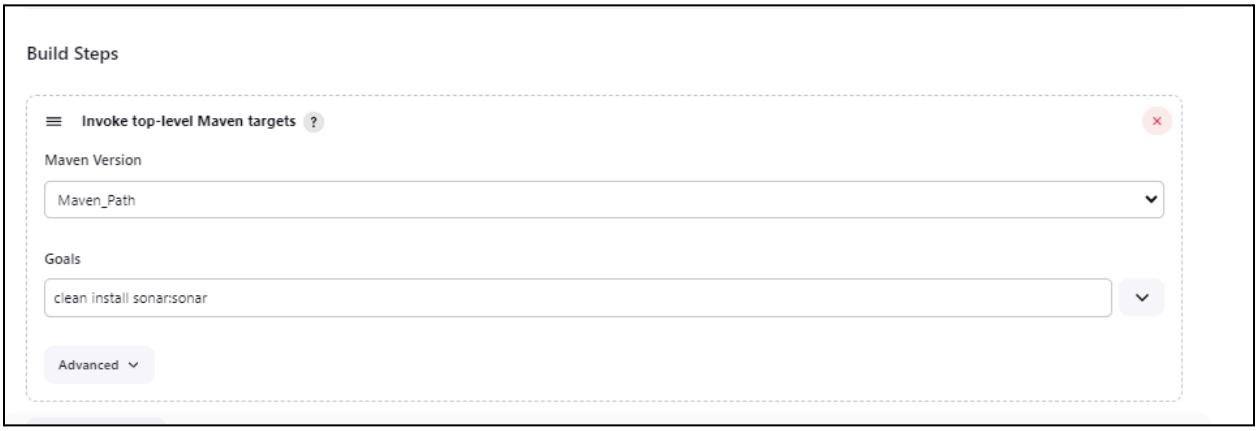
(EXTRA) : Building and Analysis of Maven project.

1. Create a maven project and paste the github link in source code management
https://github.com/edureka27/webapp.git

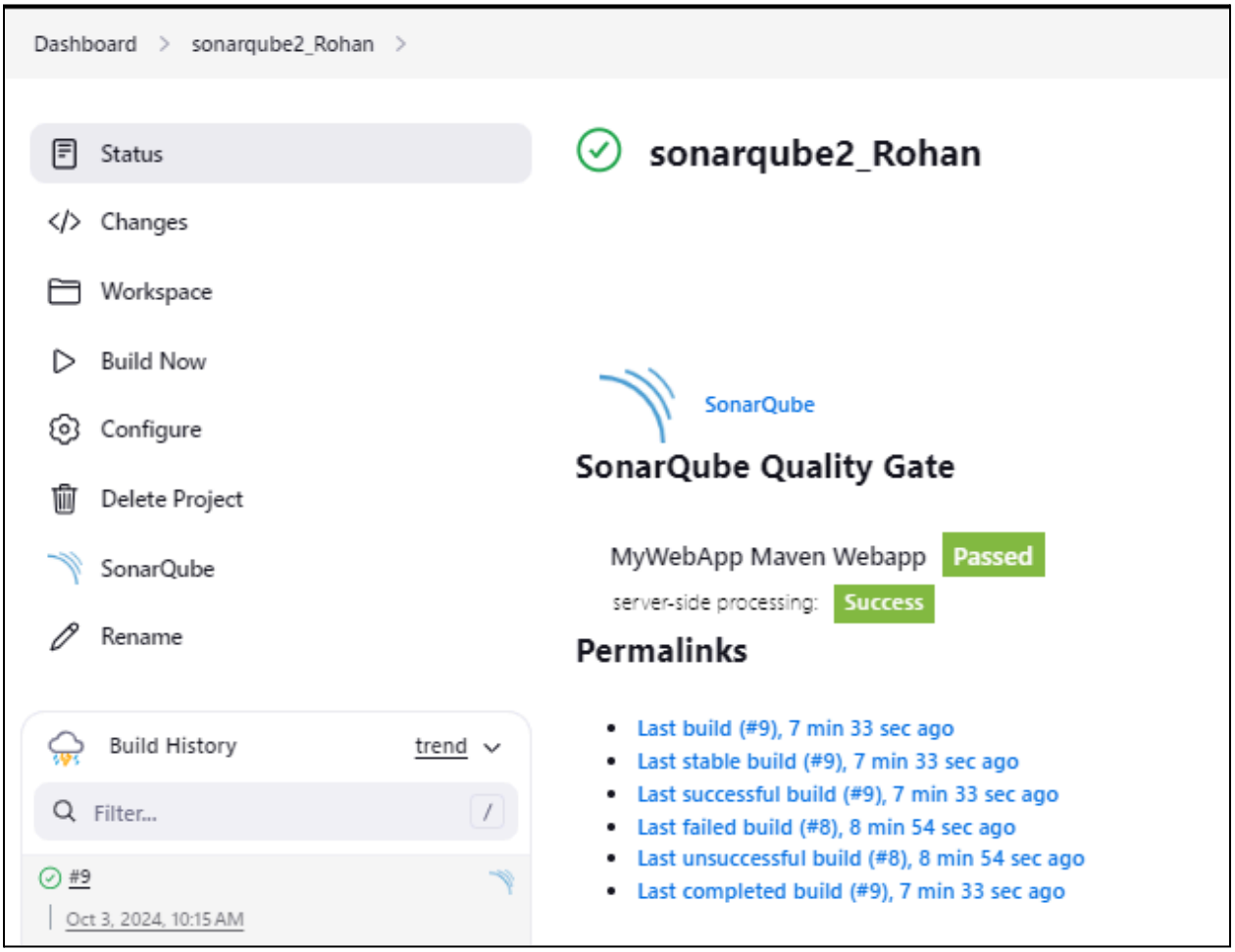


2. Check on prepare sonarqube scanner

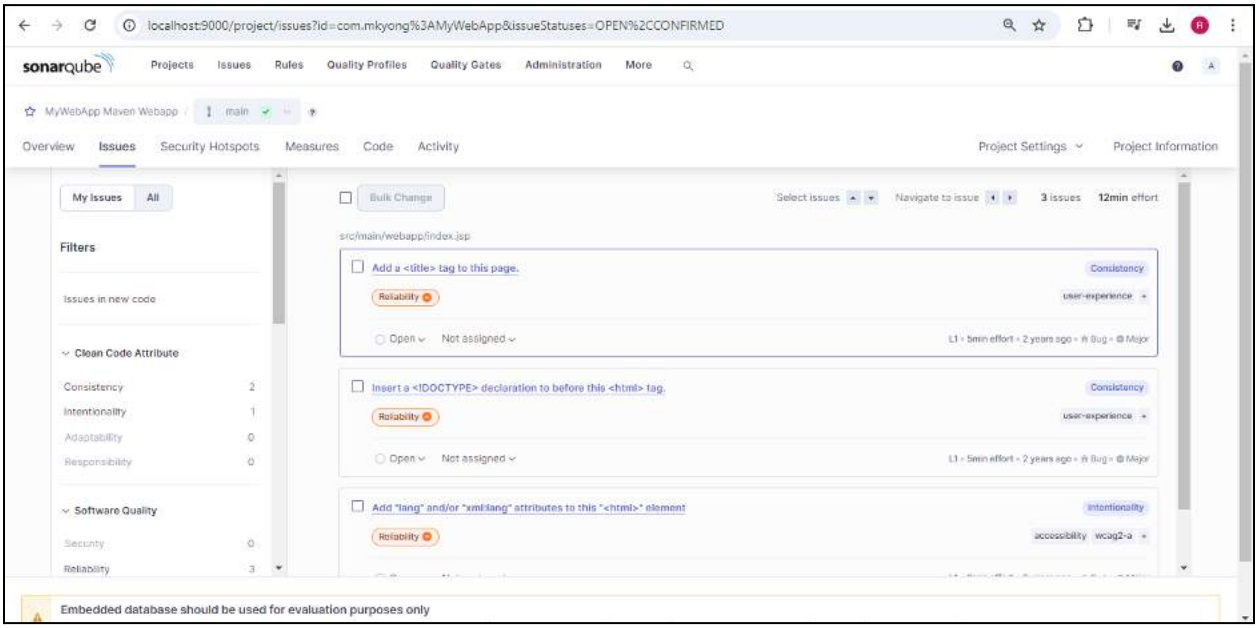
3. Give the maven_path and click on save.



Build completed successfully.



Go to the console output and scroll down click on the analysis report link and you will be able to see the sonarqube analysis report.



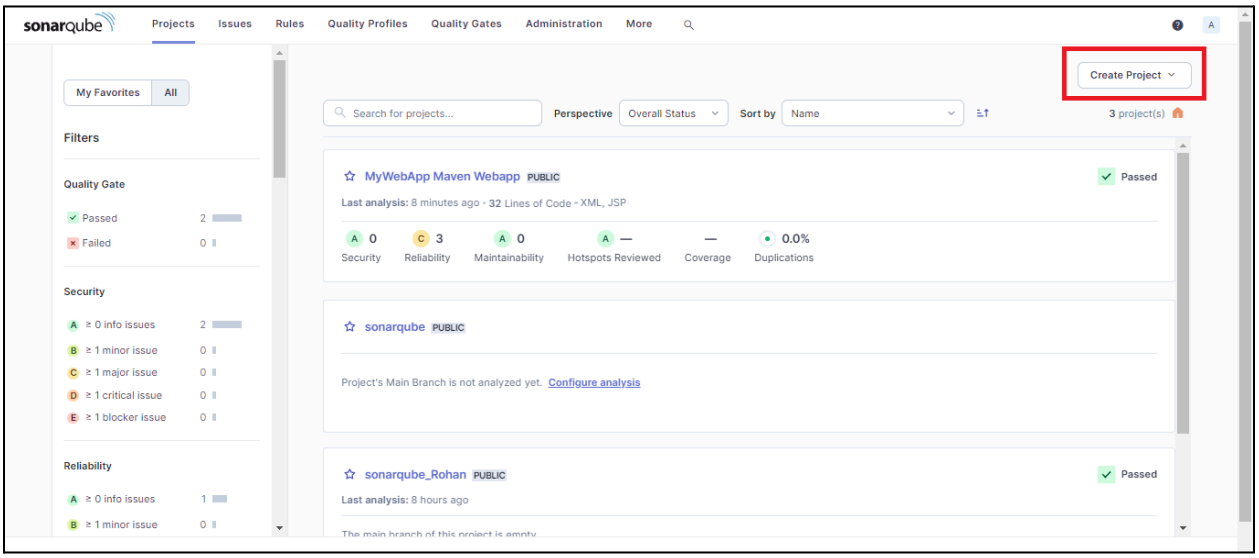
Adv. Devops Experiment no. 8

Name: Rohan Lalchandani
Class: D15A Roll no.: 25

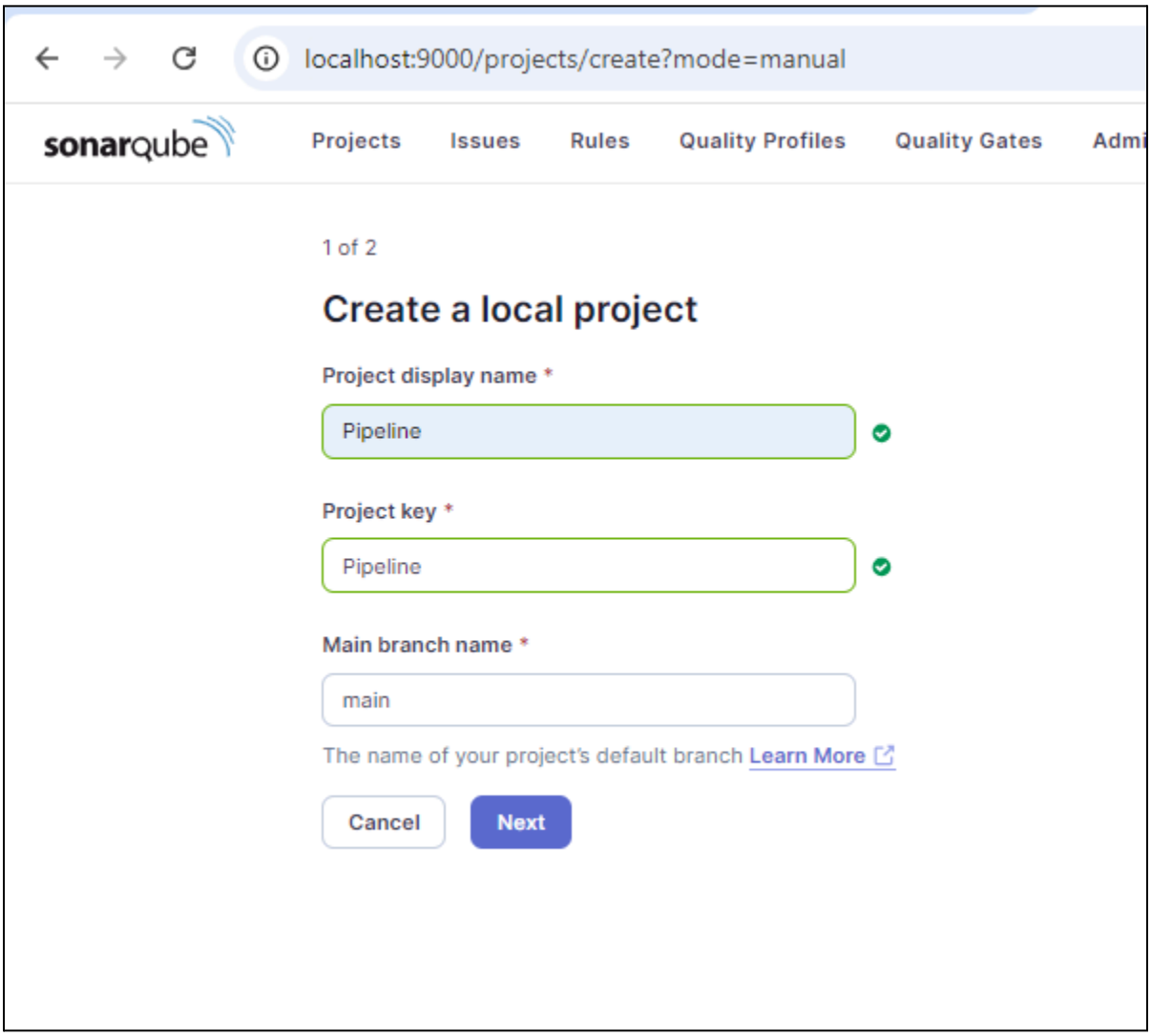
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Implementation:

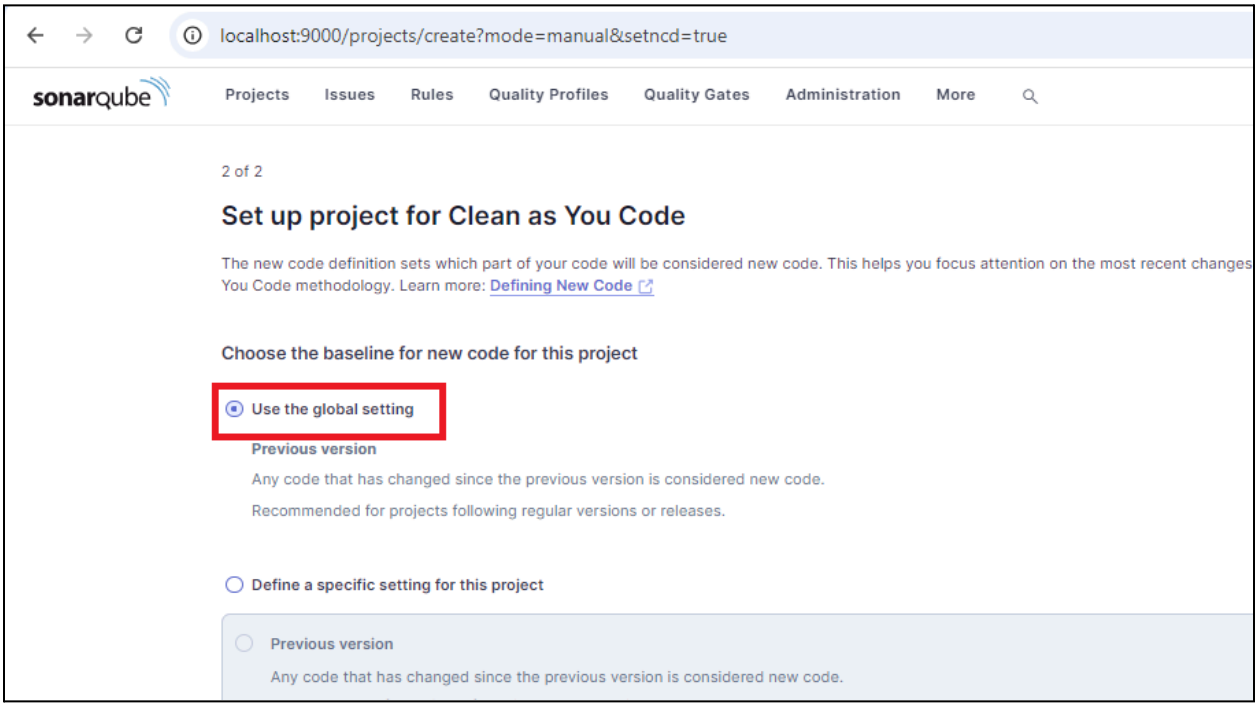
1. Download SonarQube by following steps of experiment 7, login using credentials and create a new project.



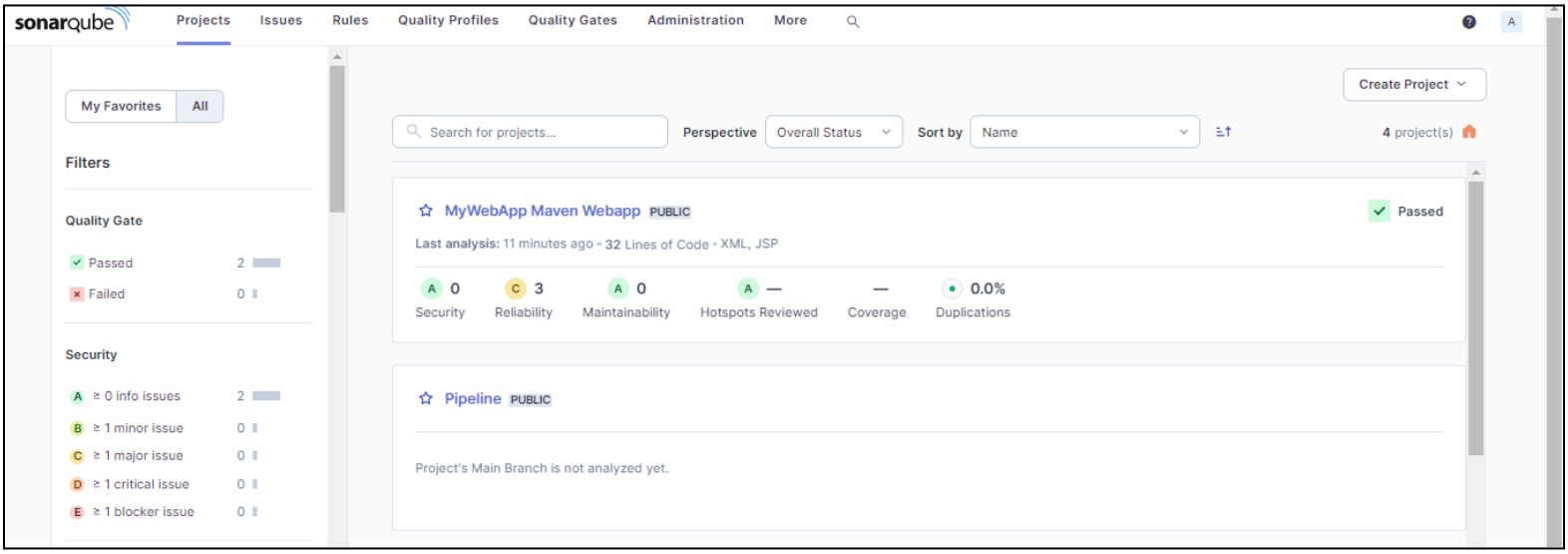
2. Type the name of the project as shown.



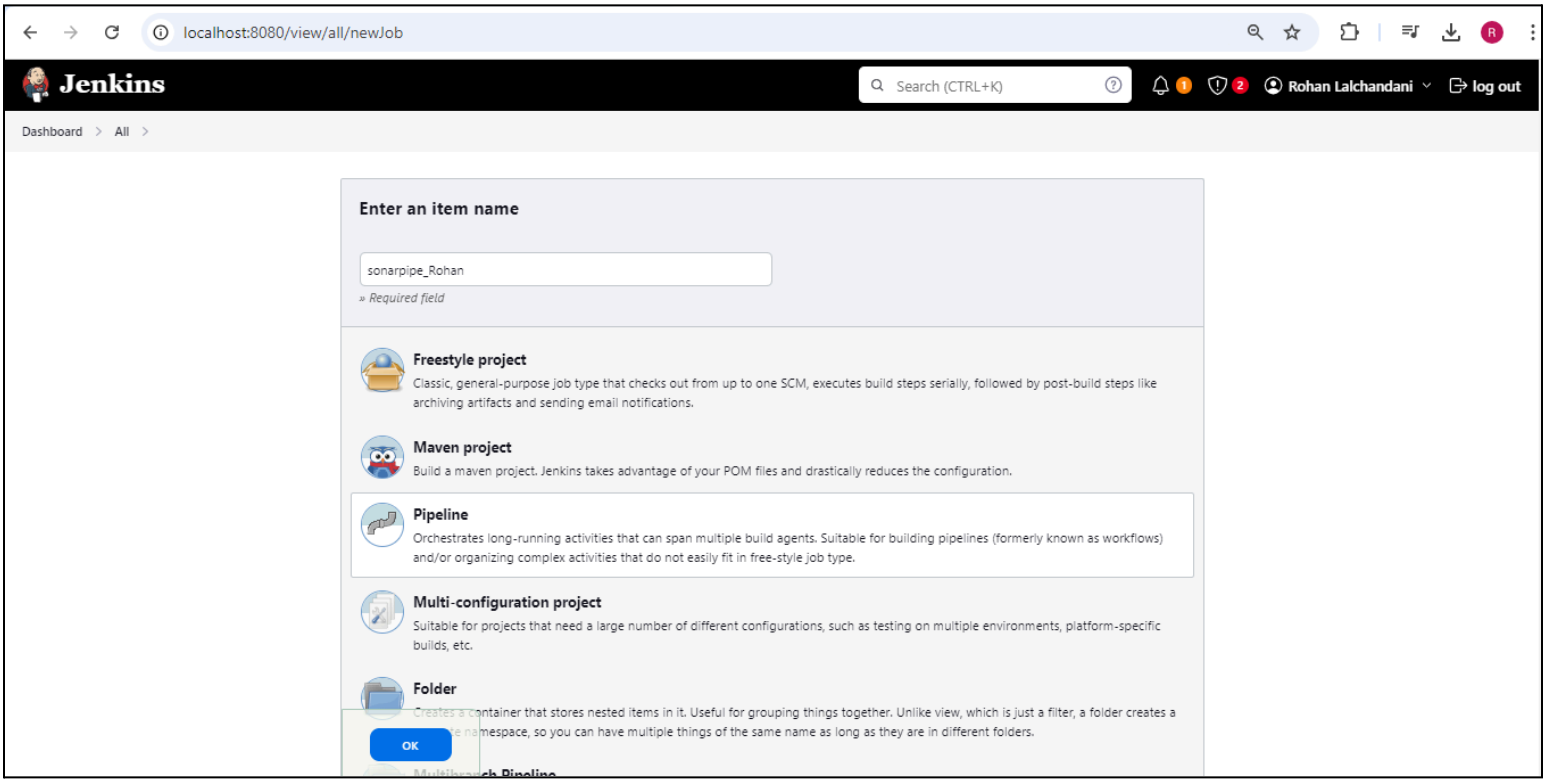
3. Use the global setting.



4. See if your project is created in the dashboard.



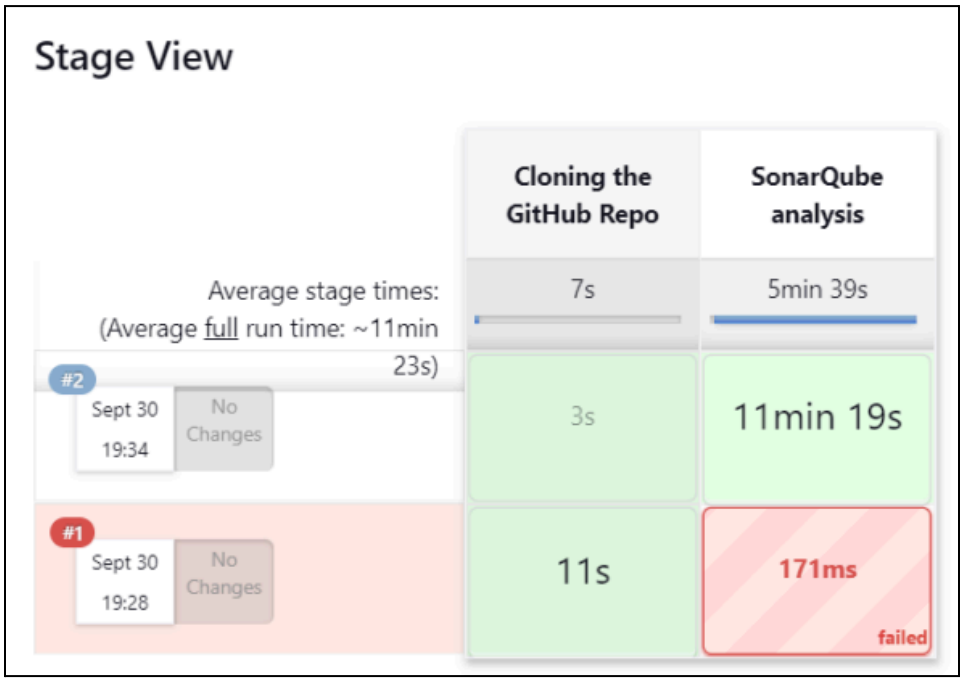
5. Now go to Jenkins, and create a pipeline.

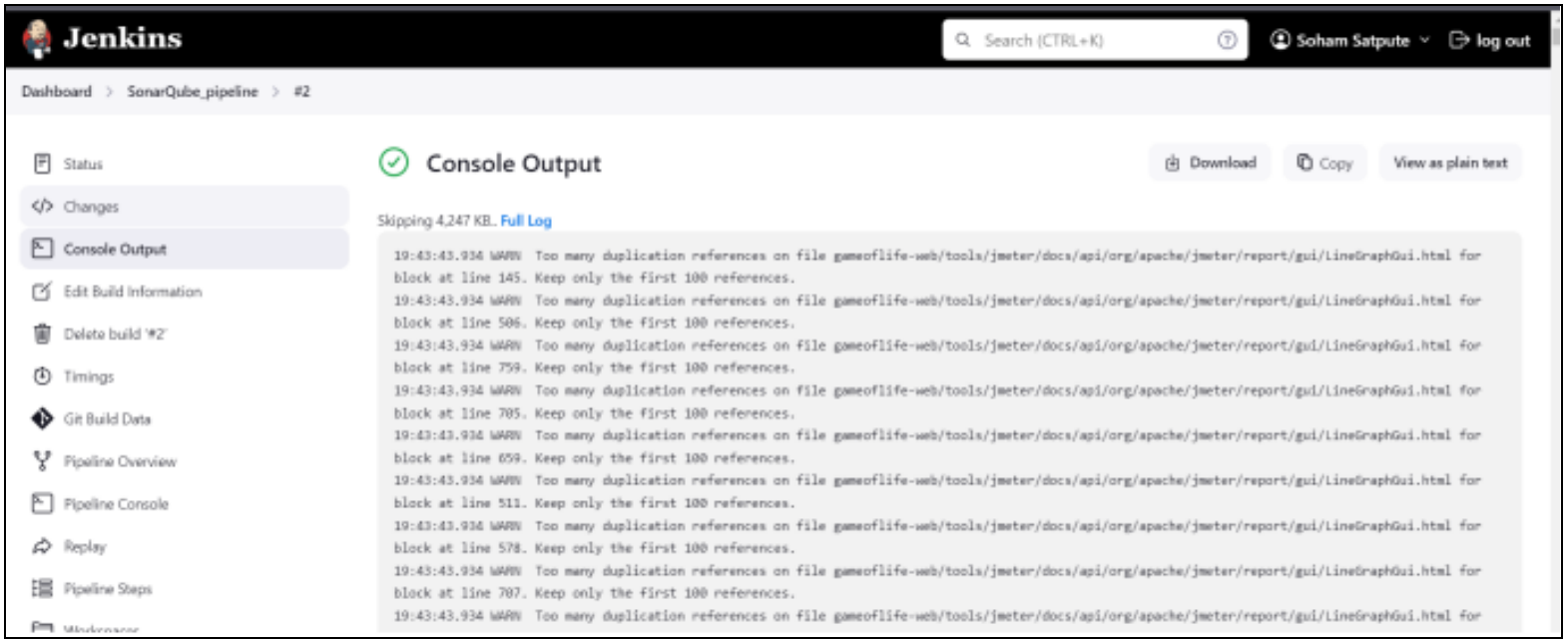


6. Enter the pipeline script

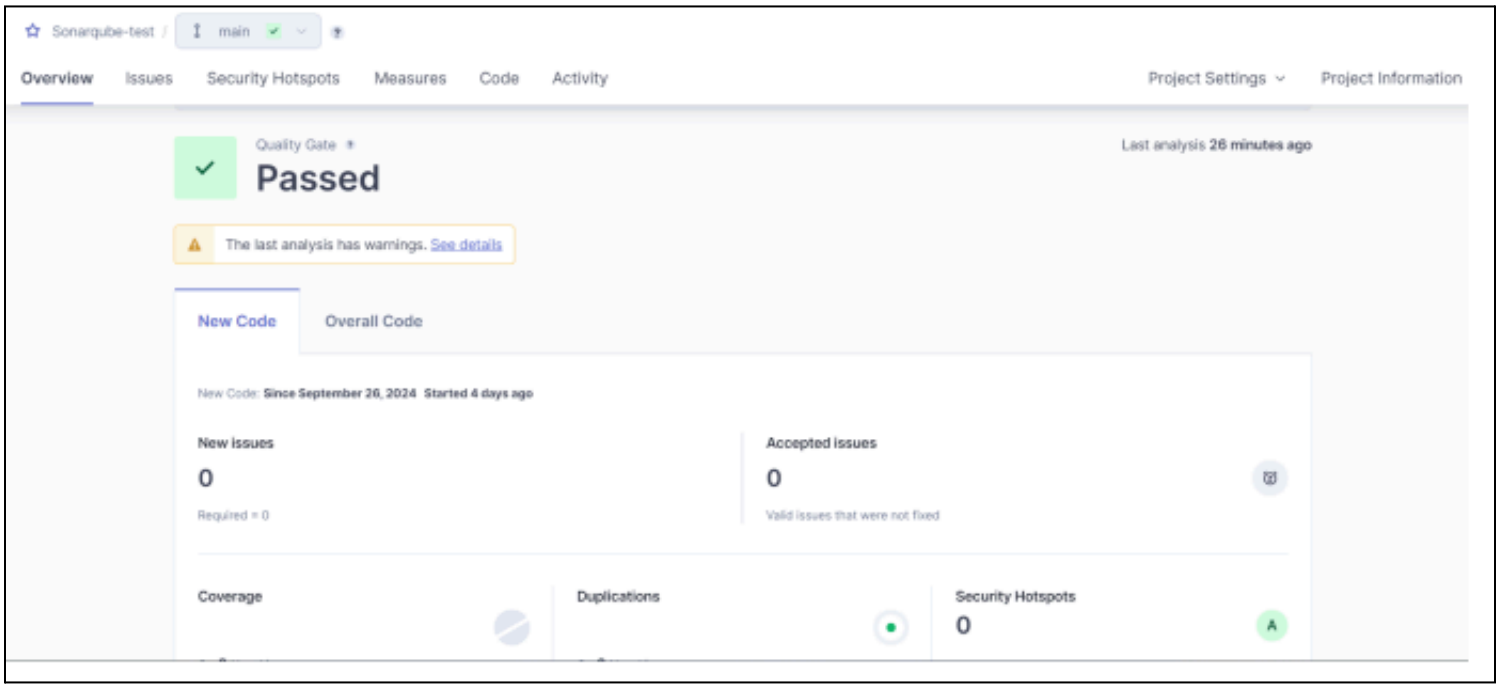
```
node {
    stage('Cloning the GitHub Repo') {
        git 'https://github.com/shazforiot/GOL.git'
    }

    stage('SonarQube analysis') {
        withSonarQubeEnv('<Name_of_SonarQube_environment_on_Jenkins>') {
            sh """
            <PATH_TO_SONARQUBE_SCANNER_FOLDER>/bin/sonar-scanner \
            -D sonar.login=<SonarQube_USERNAME> \
            -D sonar.password=<SonarQube_PASSWORD> \
            -D sonar.projectKey=<Project_KEY> \
            -D sonar.exclusions=vendor/**,resources/**,**/*.java \
            -D sonar.host.url=<SonarQube_URL>  # Default: http://localhost:9000/
            """
        }
    }
}
```

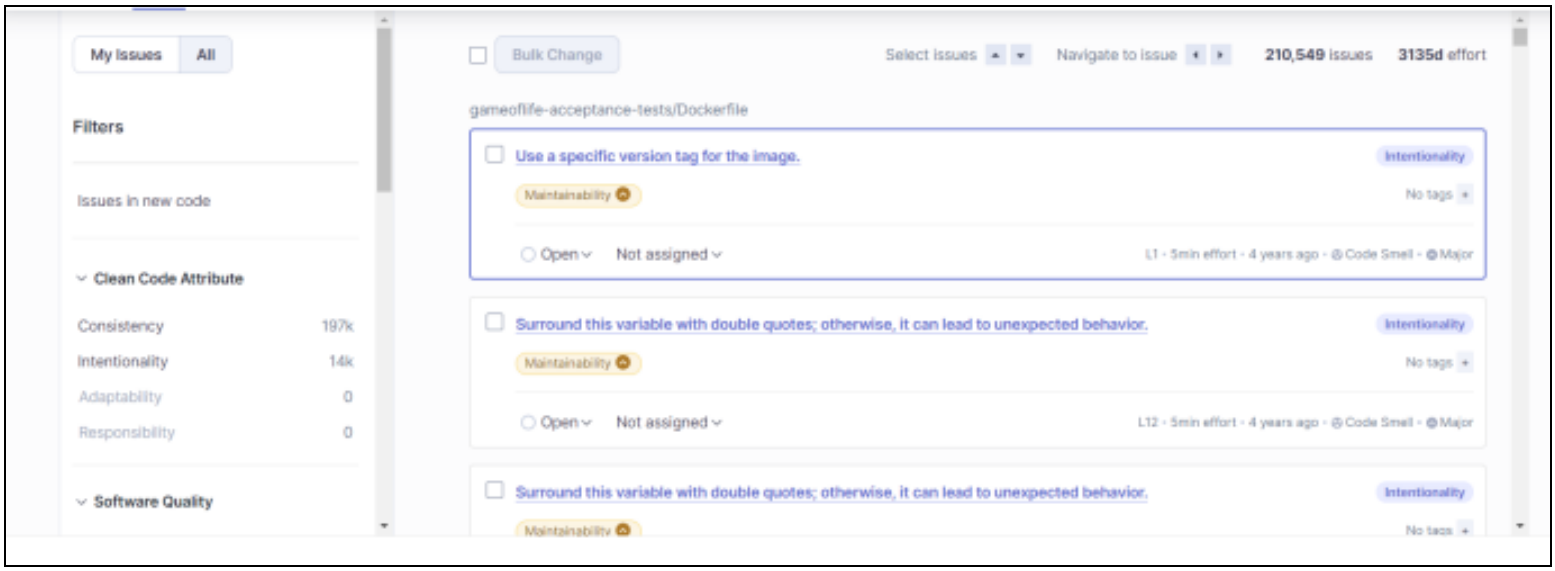
7. See the Stage view and console output the build has completed successfully.





8. See the analysis report in SonarQube.




My Issues
All
Filters
Issues in new code
Clean Code Attribute
Consistency 197k
Intentionality 14k
Adaptability 0
Responsibility 0
Software Quality
Bulk Change
Select issues
Navigate to issue
210,549 issues
3135d effort
gameoflife-acceptance-tests/Dockerfile
Use a specific version tag for the image.
Intentionality
Maintainability
No tags
Open
Not assigned
L1 · 5min effort · 4 years ago · Code Smell · Major
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Intentionality
Maintainability
No tags
Open
Not assigned
L12 · 5min effort · 4 years ago · Code Smell · Major
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Intentionality
Maintainability
No tags

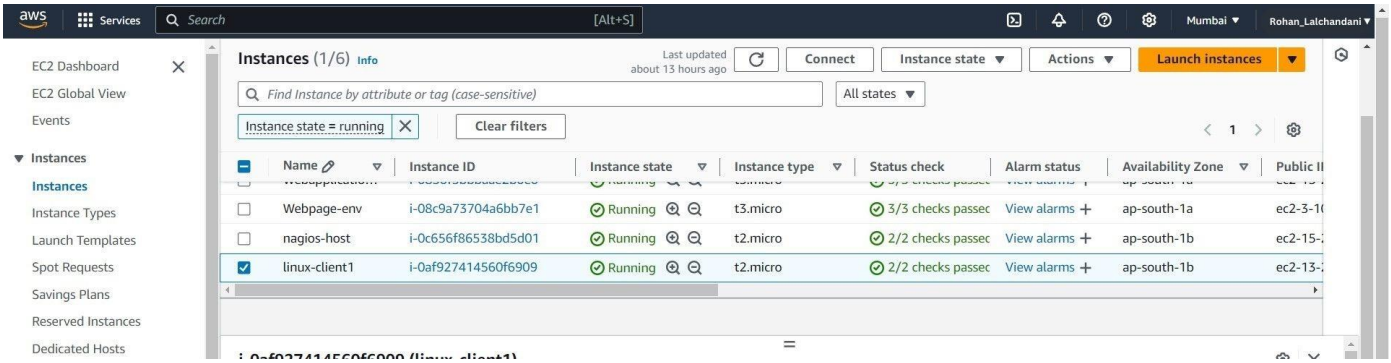
ADVANCE DEVOPS EXP 9

Name : Rohan Lalchandani

Class : D15A Roll no : 25

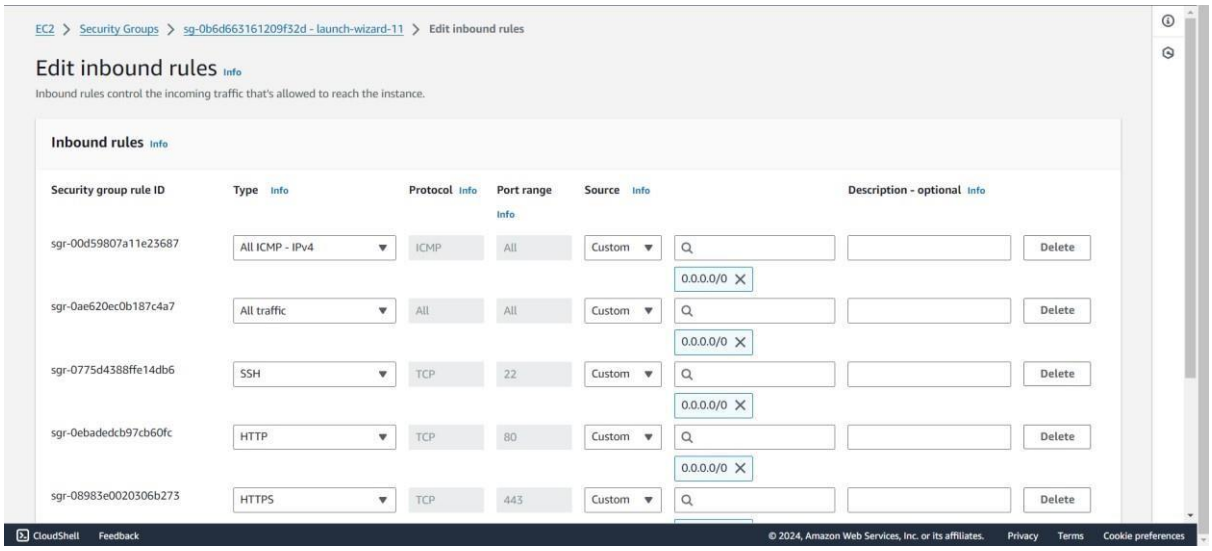
Aim :- To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.

1. Create an Amazon Linux EC2 Instance



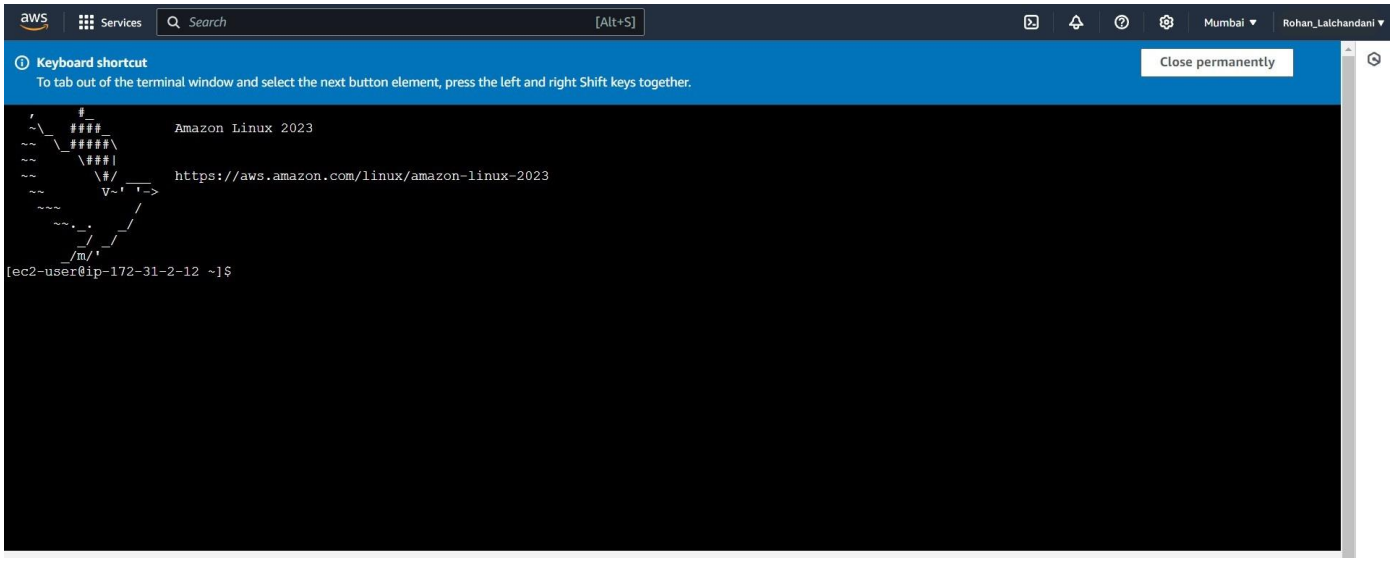
2. Configure Security Group

- Ensure HTTP, HTTPS, SSH, and ICMP are open from everywhere.
- Edit the inbound rules of the specified Security Group

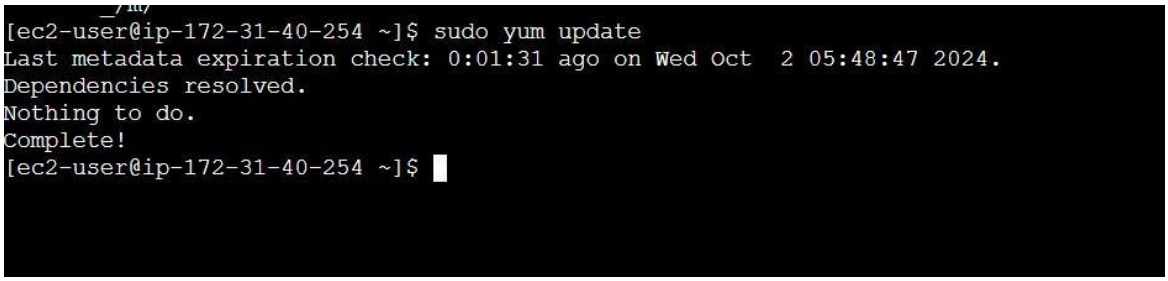


You have to edit the inbound rules of the specified Security Group for this.

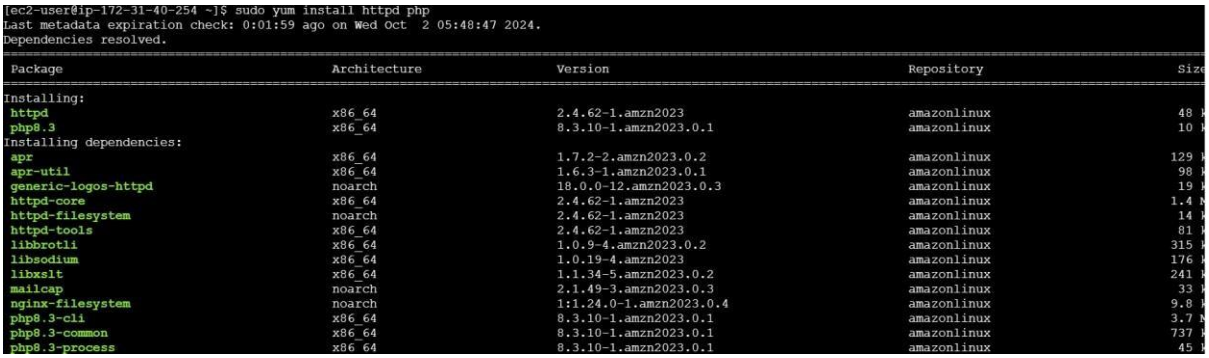
3. SSH into Your EC2 instance or simply use EC2 Instance Connect from the browser.



4. Update the package indices and install the following packages using yum sudo yum update



sudo yum install httpd php



sudo yum install gcc glibc glibc-common

```
Last metadata expiration check: 0:02:41 ago on Wed Oct  2 05:48:47 2024.
Package glibc-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package glibc-common-2.34-52.amzn2023.0.11.x86_64 is already installed.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
gcc	x86_64	11.4.1-2.amzn2023.0.2	amazonlinux	32 M
Installing dependencies:				
annobin-docs	noarch	10.93-1.amzn2023.0.1	amazonlinux	92 k
annobin-plugin-gcc	x86_64	10.93-1.amzn2023.0.1	amazonlinux	887 k
cpp	x86_64	11.4.1-2.amzn2023.0.2	amazonlinux	10 M
gc	x86_64	8.0.4-5.amzn2023.0.2	amazonlinux	105 k
glibc-devel	x86_64	2.34-52.amzn2023.0.11	amazonlinux	27 k
glibc-headers-x86	noarch	2.34-52.amzn2023.0.11	amazonlinux	427 k
guile22	x86_64	2.2.7-2.amzn2023.0.3	amazonlinux	6.4 M
kernel-headers	x86_64	6.1.109-118.189.amzn2023	amazonlinux	1.4 M
libmpc	x86_64	1.2.1-2.amzn2023.0.2	amazonlinux	62 k
libtool-ltdl	x86_64	2.4.7-1.amzn2023.0.3	amazonlinux	38 k
libxcrypt-devel	x86_64	4.4.33-7.amzn2023	amazonlinux	32 k
make	x86_64	1:4.3-5.amzn2023.0.2	amazonlinux	534 k
Transaction Summary				

sudo yum install gd gd-devel

```
[ec2-user@ip-172-31-40-254 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:03:46 ago on Wed Oct  2 05:48:47 2024.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
gd	x86_64	2.3.3-5.amzn2023.0.3	amazonlinux	139 k
gd-devel	x86_64	2.3.3-5.amzn2023.0.3	amazonlinux	38 k
Installing dependencies:				
brotli	x86_64	1.0.9-4.amzn2023.0.2	amazonlinux	314 k
brotli-devel	x86_64	1.0.9-4.amzn2023.0.2	amazonlinux	31 k
bzip2-devel	x86_64	1.0.8-6.amzn2023.0.2	amazonlinux	214 k
cairo	x86_64	1.17.6-2.amzn2023.0.1	amazonlinux	684 k
cmake-filesystem	x86_64	3.22.2-1.amzn2023.0.4	amazonlinux	16 k
fontconfig	x86_64	2.13.94-2.amzn2023.0.2	amazonlinux	273 k
fontconfig-devel	x86_64	2.13.94-2.amzn2023.0.2	amazonlinux	128 k
fonts-filesystem	noarch	1:2.0.5-12.amzn2023.0.2	amazonlinux	9.5 k
freetype	x86_64	2.13.2-5.amzn2023.0.1	amazonlinux	423 k
freetype-devel	x86_64	2.13.2-5.amzn2023.0.1	amazonlinux	912 k
glib2-devel	x86_64	2.74.7-689.amzn2023.0.2	amazonlinux	486 k
google-noto-fonts-common	noarch	20201206-2.amzn2023.0.2	amazonlinux	15 k
google-noto-sans-vf-fonts	noarch	20201206-2.amzn2023.0.2	amazonlinux	492 k
graphite2	x86_64	1.3.14-7.amzn2023.0.2	amazonlinux	97 k
graphite2-devel	x86_64	1.3.14-7.amzn2023.0.2	amazonlinux	21 k
harfbuzz	x86_64	7.0.0-2.amzn2023.0.1	amazonlinux	868 k
harfbuzz-devel	x86_64	7.0.0-2.amzn2023.0.1	amazonlinux	404 k
harfbuzz-icu	x86_64	7.0.0-2.amzn2023.0.1	amazonlinux	18 k

5. **Create a new Nagios User with its password. You'll have to enter the password twice for confirmation.** sudo adduser -m nagios sudo passwd nagios

```
[ec2-user@ip-172-31-40-254 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-40-254 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password is a palindrome
Retype new password:
Sorry, passwords do not match.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-40-254 ~]$
```

6. **Create a new user group** sudo groupadd nagcmd

```
[ec2-user@ip-172-31-40-254 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-40-254 ~]$
```

7. **Use these commands so that you don't have to use sudo for Apache and Nagios**

sudo usermod -a -G nagcmd nagios sudo
usermod -a -G nagcmd apache

```
[ec2-user@ip-172-31-40-254 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-40-254 ~]$ sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-40-254 ~]$
```

8. **Create a new directory for Nagios downloads** mkdir ~/downloads cd ~/downloads

```
[ec2-user@ip-172-31-40-254 ~]$ mkdir ~/downloads
cd ~/downloads
[ec2-user@ip-172-31-40-254 downloads]$
```

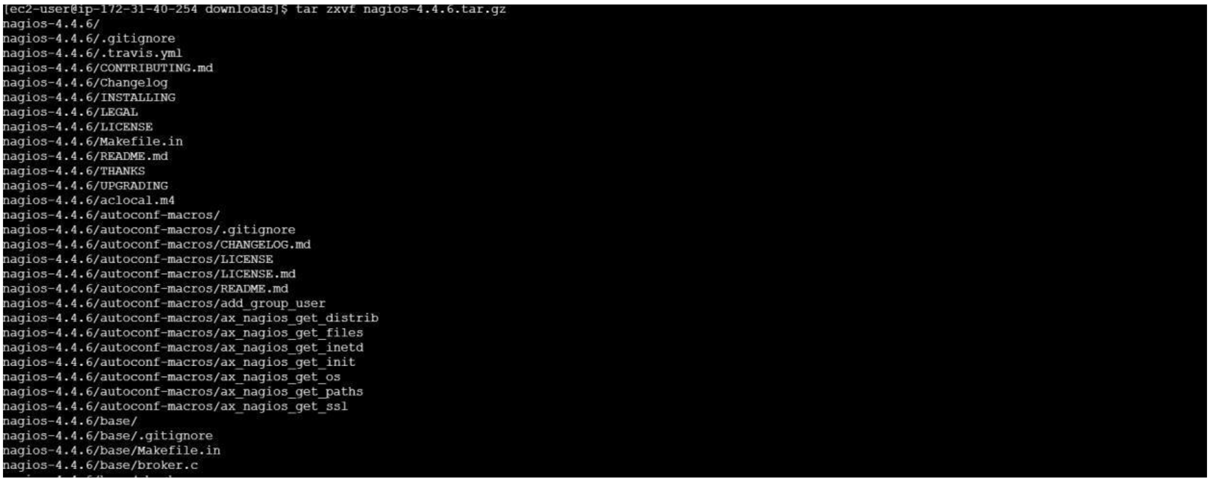
9. **Use wget to download the source zip files.**

Wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.gz wget

https://nagios-plugins.org/download/nagios-plugins-2.3.3.tar.gz

10. **Use tar to unzip and change to that directory.**

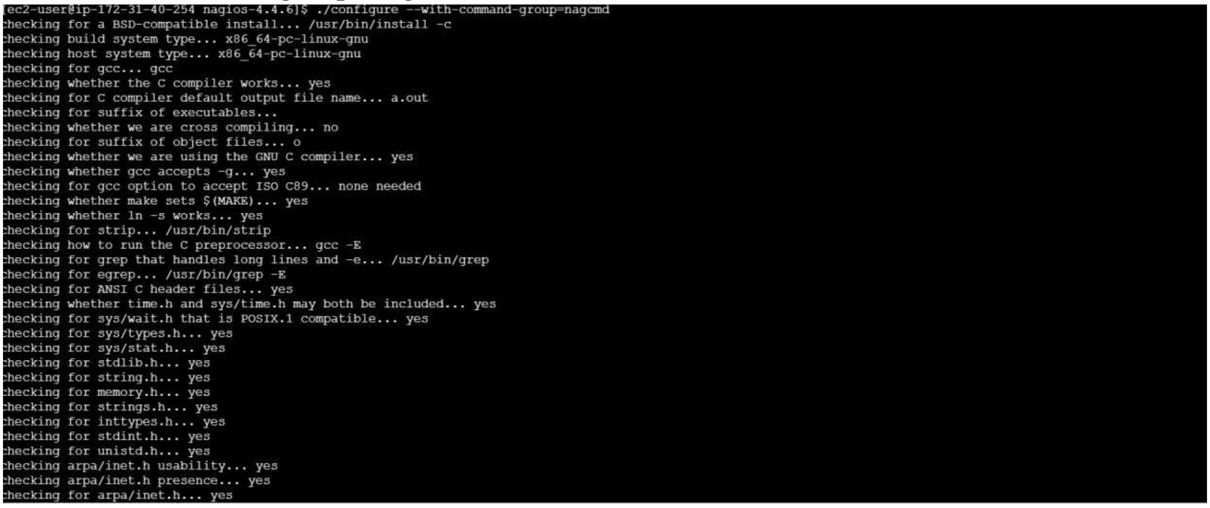
tar zxvf nagios-4.4.6.tar.gz cd
nagios-4.4.6



```
[ec2-user@ip-172-31-40-254 downloads]$ tar zxvf nagios-4.4.6.tar.gz
nagios-4.4.6/
nagios-4.4.6/.gitignore
nagios-4.4.6/.travis.yml
nagios-4.4.6/CONTRIBUTING.md
nagios-4.4.6/Changelog
nagios-4.4.6/INSTALLING
nagios-4.4.6/LEGAL
nagios-4.4.6/LICENSE
nagios-4.4.6/Makefile.in
nagios-4.4.6/README.md
nagios-4.4.6/THANKS
nagios-4.4.6/UPGRADING
nagios-4.4.6/aclocal.m4
nagios-4.4.6/autoconf-macros/
nagios-4.4.6/autoconf-macros/.gitignore
nagios-4.4.6/autoconf-macros/CHANGELOG.md
nagios-4.4.6/autoconf-macros/LICENSE
nagios-4.4.6/autoconf-macros/LICENSE.md
nagios-4.4.6/autoconf-macros/README.md
nagios-4.4.6/autoconf-macros/add_group_user
nagios-4.4.6/autoconf-macros/ax_nagios_get_distrib
nagios-4.4.6/autoconf-macros/ax_nagios_get_files
nagios-4.4.6/autoconf-macros/ax_nagios_get_inetd
nagios-4.4.6/autoconf-macros/ax_nagios_get_init
nagios-4.4.6/autoconf-macros/ax_nagios_get_os
nagios-4.4.6/autoconf-macros/ax_nagios_get_paths
nagios-4.4.6/autoconf-macros/ax_nagios_get_ssl
nagios-4.4.6/base/
nagios-4.4.6/base/.gitignore
nagios-4.4.6/base/Makefile.in
nagios-4.4.6/base/broker.c
```

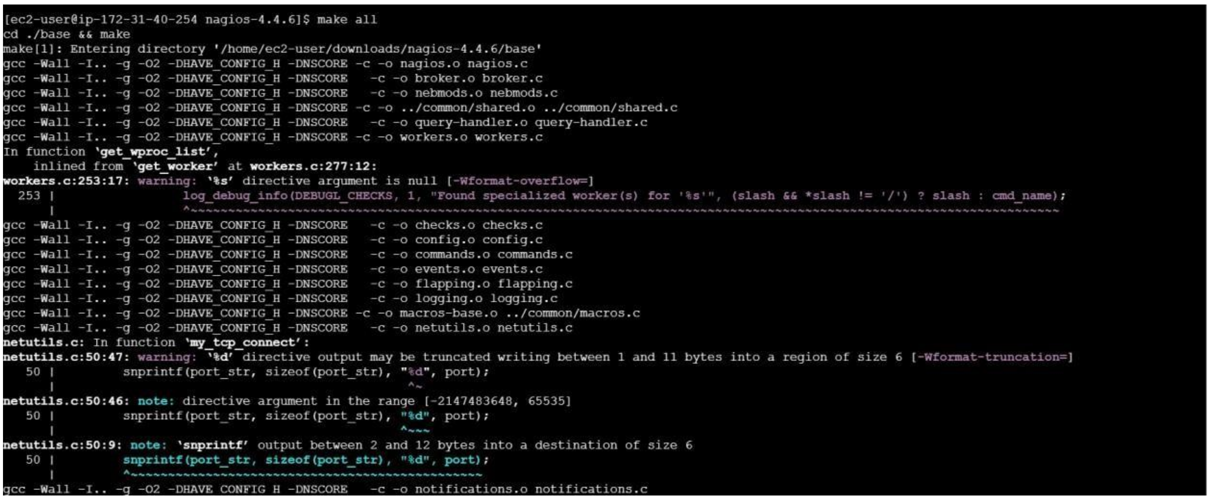
11. Run the configuration script with the same group name you previously created.

./configure --with-command-group=nagcmd

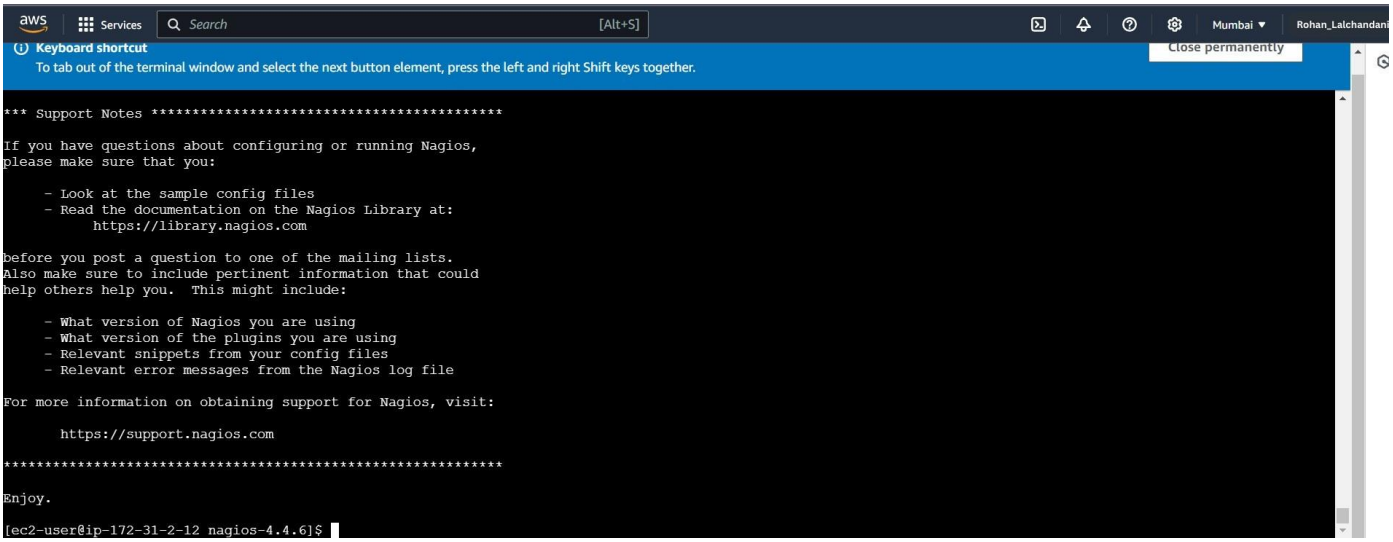


```
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$ ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for ANSI C header files... yes
checking whether time.h and sys/time.h may both be included... yes
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for memory.h... yes
checking for strings.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
checking for unistd.h... yes
checking arpa/inet.h usability... yes
checking arpa/inet.h presence... yes
checking for arpa/inet.h... yes
```

12. Compile the source code. make all



```
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$ make all
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.4.6/base'
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o nagios.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o broker.o broker.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o nebmods.o nebmods.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o ../common/shared.o ../common/shared.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o query-handler.o query-handler.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o workers.o workers.c
In function 'get_wproc_list',
    inlined from 'get_worker' at workers.c:277:12:
workers.c:253:17: warning: '%s' directive argument is null [-Wformat-overflow=]
  253 |             log_debug_info(DEBUGL_CHECKS, 1, "Found specialized worker(s) for '%s'", (slash && *slash != '/') ? slash : cmd_name);
      |             ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o checks.o checks.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o config.o config.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o commands.o commands.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o events.o events.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o flapping.o flapping.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o logging.o logging.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o macros-base.o ../common/macros.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o netutils.o netutils.c
netutils.c: In function 'my_tcp_connect':
netutils.c:50:47: warning: '%d' directive output may be truncated writing between 1 and 11 bytes into a region of size 6 [-Wformat-truncation=]
   50 |         snprintf(port_str, sizeof(port_str), "%d", port);
      |                                               ^~
netutils.c:50:46: note: directive argument in the range [-2147483648, 65535]
   50 |         snprintf(port_str, sizeof(port_str), "%d", port);
      |                                              ^~~~
netutils.c:50:9: note: 'snprintf' output between 2 and 12 bytes into a destination of size 6
   50 |         snprintf(port_str, sizeof(port_str), "%d", port);
      |         ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o notifications.o notifications.c
```

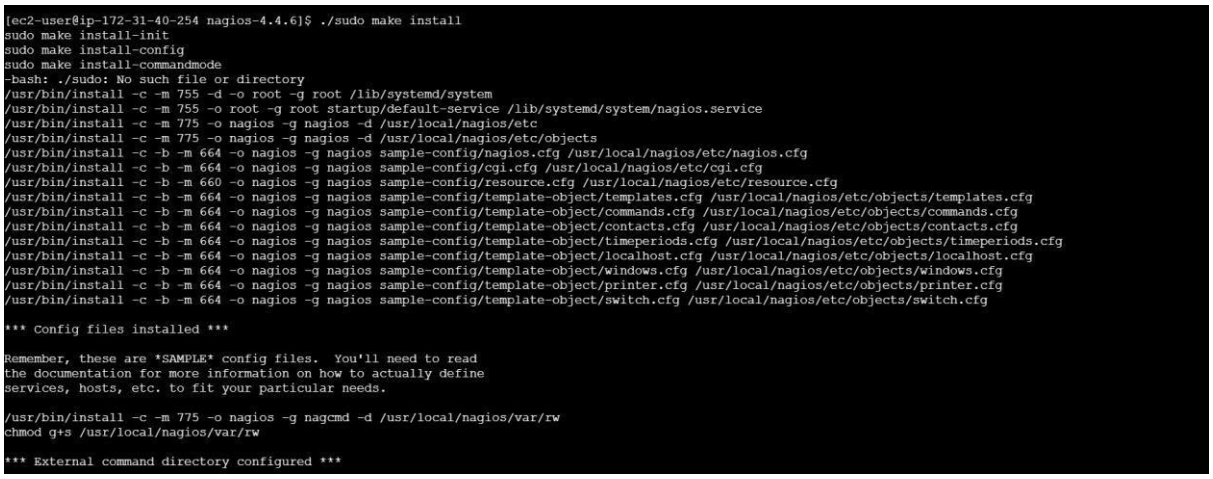


13. Install binaries, init script and sample config files. Lastly, set permissions on the external command directory.

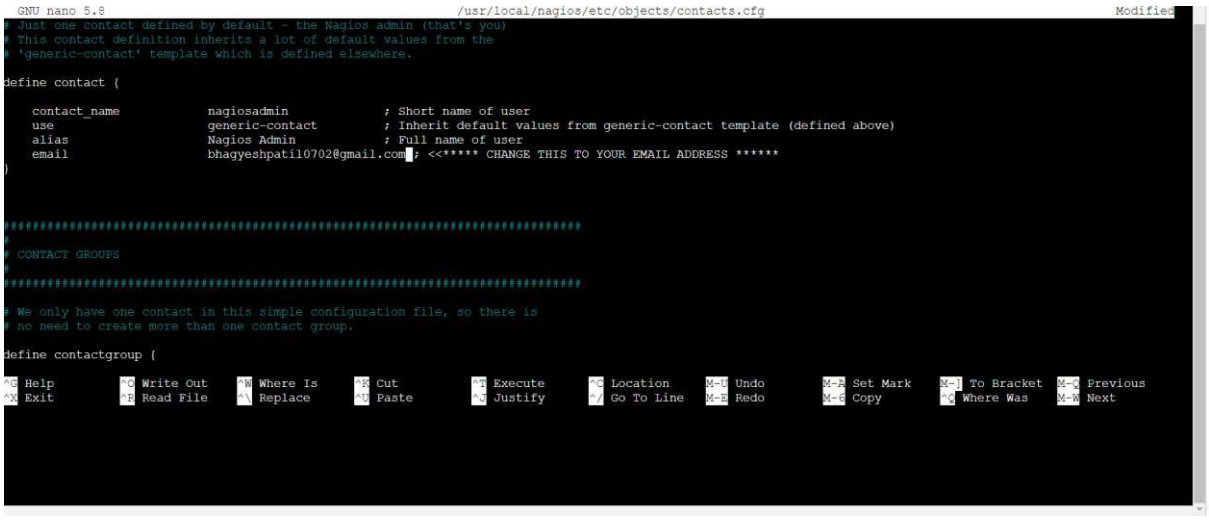
./sudo make install sudo make

install-init sudo make installconfig

sudo make installcommandmode



14. Edit the config file and change the email address. sudo nano /usr/local/nagios/etc/objects/contacts.cfg



15. Configure the web interface.

sudo make install-webconf

```
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

*** Nagios/Apache conf file installed ***

[ec2-user@ip-172-31-40-254 nagios-4.4.6]$
```

16. Create a nagiosadmin account for nagios login along with password. You'll have to specify the password twice. sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

```
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$
```

17. Restart Apache sudo systemctl restart httpd

```
Adding password for user nagiosadmin
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$ sudo systemctl restart httpd
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$
```

cd ~/downloads tar zxvf nagiosplugins-2.3.3.tar.gz cd nagiosplugins-2.3.3

```
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$ sudo systemctl restart httpd
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$ cd ~/downloads
tar zxvf nagios-plugins-2.3.3.tar.gz
 cd nagios-plugins-2.3.3
nagios-plugins-2.3.3/
nagios-plugins-2.3.3/perlmods/
nagios-plugins-2.3.3/perlmods/Config-Tiny-2.14.tar.gz
nagios-plugins-2.3.3/perlmods/parent-0.226.tar.gz
nagios-plugins-2.3.3/perlmods/Test-Simple-0.98.tar.gz
nagios-plugins-2.3.3/perlmods/Makefile.in
nagios-plugins-2.3.3/perlmods/version-0.9903.tar.gz
nagios-plugins-2.3.3/perlmods/Makefile.am
nagios-plugins-2.3.3/perlmods/Module-Runtime-0.013.tar.gz
nagios-plugins-2.3.3/perlmods/Module-Metadata-1.000014.tar.gz
nagios-plugins-2.3.3/perlmods/Params-Validate-1.08.tar.gz
nagios-plugins-2.3.3/perlmods/Class-Accessor-0.34.tar.gz
nagios-plugins-2.3.3/perlmods/Try-Tiny-0.18.tar.gz
nagios-plugins-2.3.3/perlmods/Module-Implementation-0.07.tar.gz
nagios-plugins-2.3.3/perlmods/Makefile
nagios-plugins-2.3.3/perlmods/Perl-OSType-1.003.tar.gz
nagios-plugins-2.3.3/perlmods/install_order
nagios-plugins-2.3.3/perlmods/Nagios-Plugin-0.36.tar.gz
nagios-plugins-2.3.3/perlmods/Math-Calc-Units-1.07.tar.gz
nagios-plugins-2.3.3/perlmods/Module-Build-0.4007.tar.gz
nagios-plugins-2.3.3/ABOUT-NLS
nagios-plugins-2.3.3/configure.ac
nagios-plugins-2.3.3/Makefile.in
nagios-plugins-2.3.3/config.h.in
nagios-plugins-2.3.3/ChangeLog
nagios-plugins-2.3.3/AUTHORS
nagios-plugins-2.3.3/lib/
nagios-plugins-2.3.3/lib/parse_ini.h
nagios-plugins-2.3.3/lib/extra_opts.c
nagios-plugins-2.3.3/lib/Makefile.in
```

18. Go back to the downloads folder and unzip the plugins zip file.

./configure --with-nagios-user=nagios --with-nagiosgroup=nagios make

sudo make install

```
[ec2-user@ip-172-31-80-22 nagios-plugins-2.3.3]$ ./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
sudo make install
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether to disable maintainer-specific portions of Makefiles... yes
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
```

19. Compile and install plugins sudo chkconfig --add nagios sudo chkconfig nagios on

sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg sudo systemctl

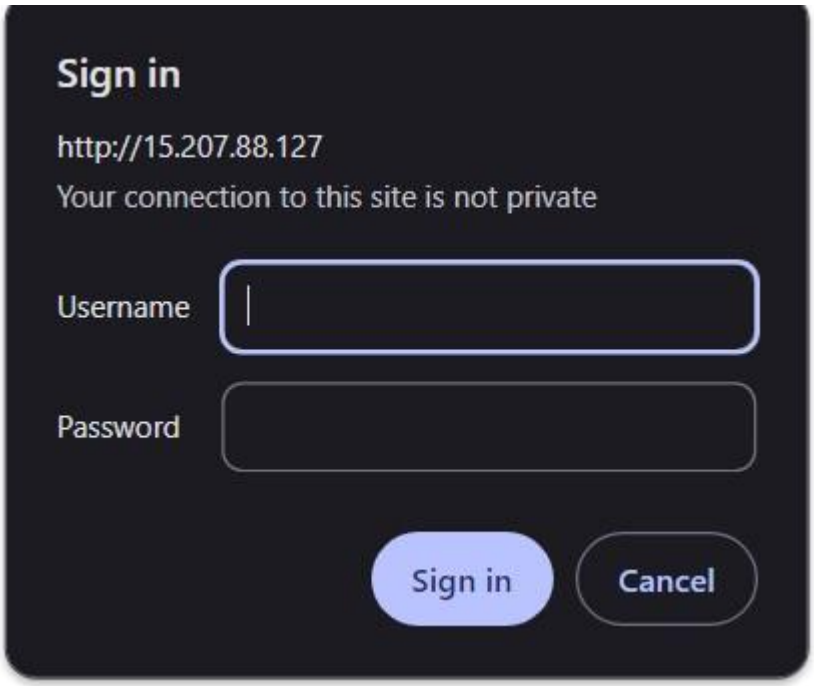
start nagios

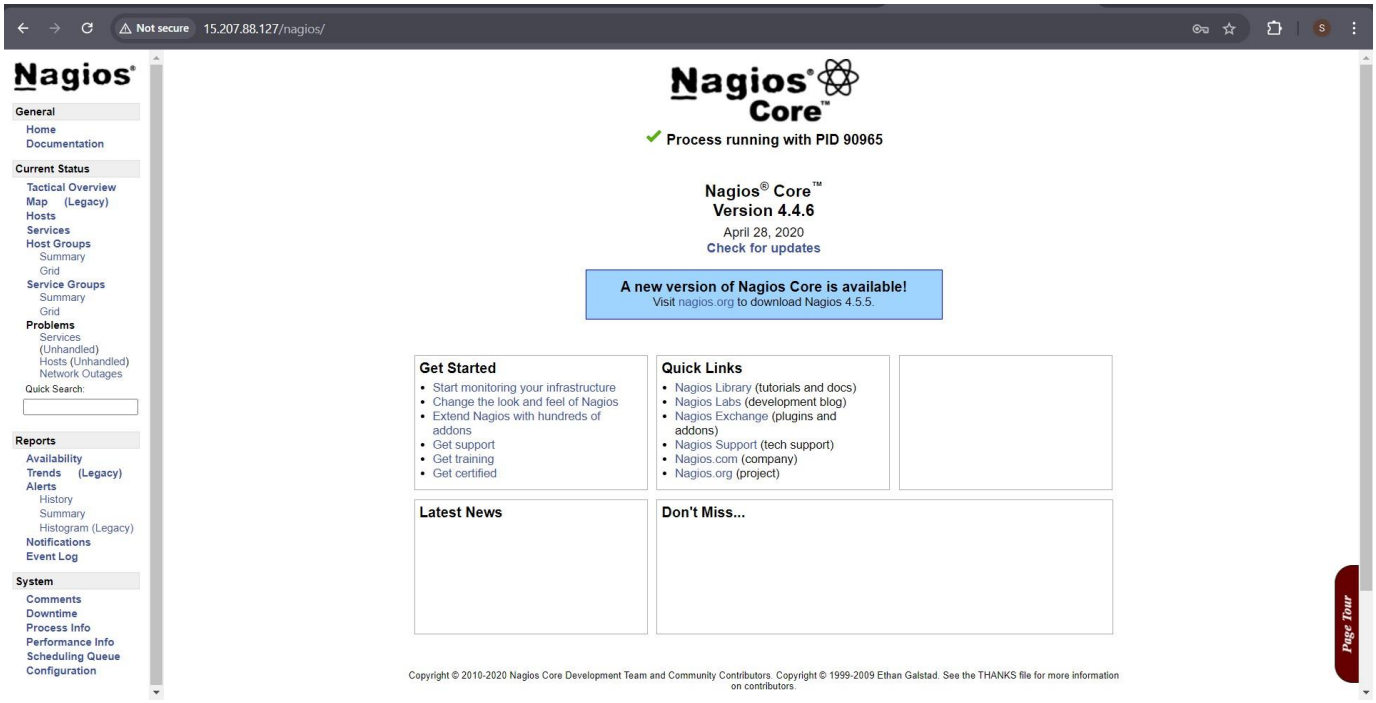
```
sudo chkconfig nagios on
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
sudo systemctl start nagios
Error reading information on service nagios: No such file or directory
Note: Forwarding request to 'systemctl enable nagios.service'.
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.

Nagios Core 4.4.6
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2020-04-28
License: GPL

Website: https://www.nagios.org
Reading configuration data...
   Read main config file okay...
   Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
       Checked 8 services.
       Checked 1 hosts.
       Checked 1 host groups.
```

20. Check the status of Nagios

```
Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-45-178 nagios-plugins-2.3.3]$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Wed 2024-10-02 05:37:36 UTC; 14s ago
       Docs: https://www.nagios.org/documentation
    Process: 67990 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 67991 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 67992 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 2.0M
        CPU: 16ms
     CGroup: /system.slice/nagios.service
             ├─67992 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─67993 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─67994 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─67995 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─67996 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─67997 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 02 05:37:36 ip-172-31-45-178.ec2.internal nagios[67992]: qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successfully initialized
Oct 02 05:37:36 ip-172-31-45-178.ec2.internal nagios[67992]: qh: core query handler registered
```

23. Open up your browser and look for http://<your_public_ip_address>/nagios




Not secure
15.207.88.127/nagios/
Nagios
General
Home
Documentation
Current Status
Tactical Overview
Map (Legacy)
Hosts
Services
Host Groups
Summary
Grid
Service Groups
Summary
Grid
Problems
Services (Unhandled)
Hosts (Unhandled)
Network Outages
Quick Search:
Reports
Availability
Trends (Legacy)
Alerts
History
Summary
Histogram (Legacy)
Notifications
Event Log
System
Comments
Downtime
Process Info
Performance Info
Scheduling Queue
Configuration
Nagios Core
Process running with PID 90965
Nagios® Core™
Version 4.4.6
April 28, 2020
Check for updates
A new version of Nagios Core is available!
Visit nagios.org to download Nagios 4.5.5.
Get Started
Start monitoring your infrastructure
Change the look and feel of Nagios
Extend Nagios with hundreds of addons
Get support
Get training
Get certified
Quick Links
Nagios Library (tutorials and docs)
Nagios Labs (development blog)
Nagios Exchange (plugins and addons)
Nagios Support (tech support)
Nagios.com (company)
Nagios.org (project)
Latest News
Don't Miss...
Page Tour
Copyright © 2010-2020 Nagios Core Development Team and Community Contributors. Copyright © 1999-2009 Ethan Galstad. See the THANKS file for more information on contributors.

Advance DevOps Exp 10

Name : Rohan Lalchandani

Class : D15A Roll no : 25

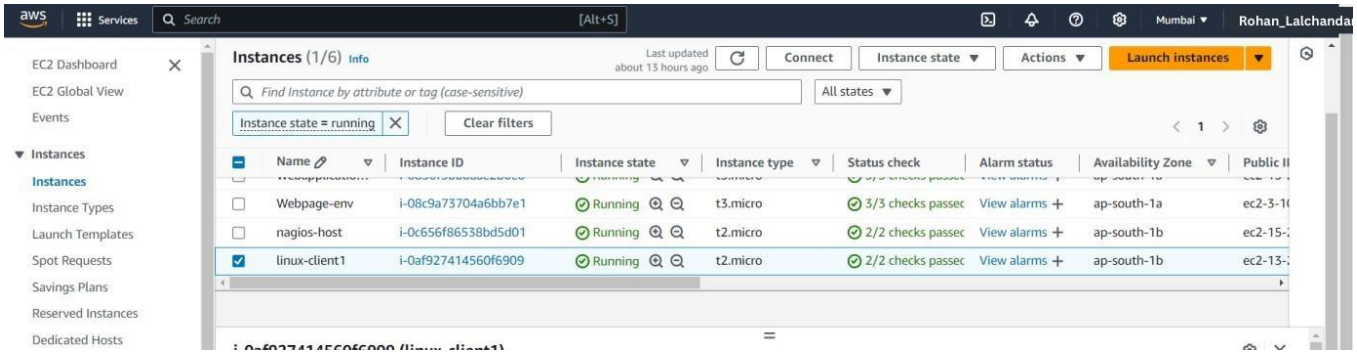
Aim: To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

Procedure:-

Check if the nagios service is running by executing following command sudo systemctl status nagios

```
ubuntu@ip-172-31-89-161:~$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-09-28 16:08:58 UTC; 1min 2s ago
       Docs: https://www.nagios.org/documentation
    Process: 15743 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 15753 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 15764 (nagios)
      Tasks: 6 (limit: 1130)
     Memory: 2.4M (peak: 3.2M)
        CPU: 29ms
     CGroup: /system.slice/nagios.service
             ├─15764 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─15765 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─15766 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─15767 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─15768 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─15769 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Sep 28 16:08:58 ip-172-31-89-161 nagios[15764]: qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successfully initialized
Sep 28 16:08:58 ip-172-31-89-161 nagios[15764]: qh: core query handler registered
Sep 28 16:08:58 ip-172-31-89-161 nagios[15764]: qh: echo service query handler registered
Sep 28 16:08:58 ip-172-31-89-161 nagios[15764]: qh: help for the query handler registered
Sep 28 16:08:58 ip-172-31-89-161 nagios[15764]: wproc: Successfully registered manager as @wproc with query handler
Sep 28 16:08:58 ip-172-31-89-161 nagios[15764]: wproc: Registry request: name=Core Worker 15765;pid=15765
Sep 28 16:08:58 ip-172-31-89-161 nagios[15764]: wproc: Registry request: name=Core Worker 15766;pid=15766
Sep 28 16:08:58 ip-172-31-89-161 nagios[15764]: wproc: Registry request: name=Core Worker 15767;pid=15767
```

Now, create a new EC2 instance on AWS



Now perform the following commands on nagios-host EC2 instance.On the server, run this command

ps -ef | grep nagios

```
ubuntu@ip-172-31-89-161:~$ ps -ef | grep nagios
nagios     15764       1  0 16:08 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
nagios     15765   15764  0 16:08 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     15766   15764  0 16:08 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     15767   15764  0 16:08 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     15768   15764  0 16:08 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     15769   15764  0 16:08 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
ubuntu     15957    1342  0 16:13 pts/0    00:00:00 grep --color=auto nagios
ubuntu@ip-172-31-89-161:~$
```

Sudo su

mkdir /usr/local/nagios/etc/objects/monitorhosts

mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts

```
ubuntu@ip-172-31-89-161:~$ sudo su
mkdir /usr/local/nagios/etc/objects/monitorhosts
mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
root@ip-172-31-89-161:/home/ubuntu#
```

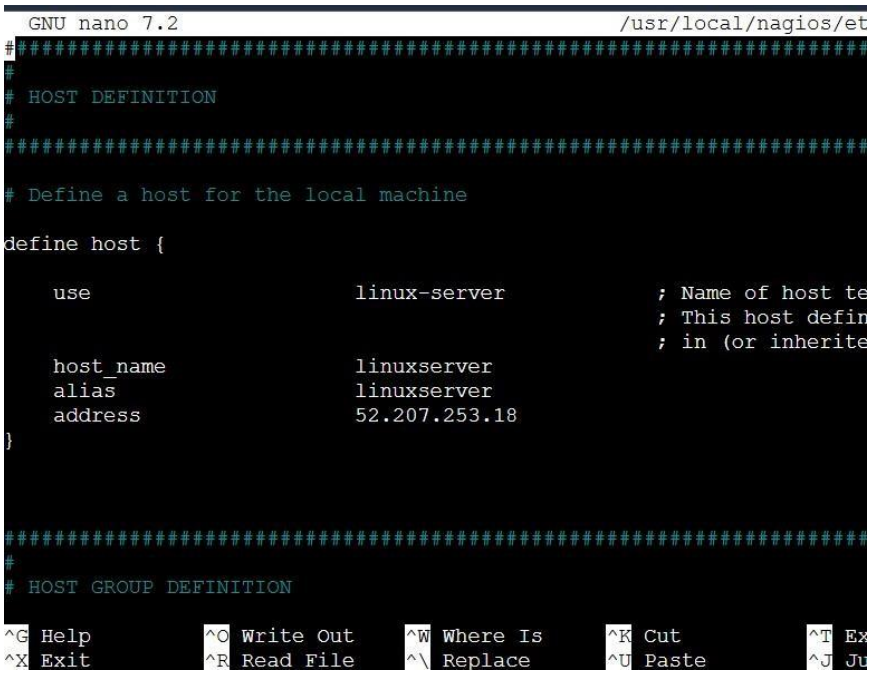
Copy localhost.cfg file to the mentioned location

cp /usr/local/nagios/etc/objects/localhost.cfg/usr/local/nagios/etc/objects/monitorhosts/linuxhosts

```
root@ip-172-31-89-161:/usr/local/nagios/etc/objects# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
cp: cannot create regular file '/usr/local/nagios/etc/objects/monitorhosts/linuxhosts': No such file or directory
root@ip-172-31-89-161:/usr/local/nagios/etc/objects# sudo mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
root@ip-172-31-89-161:/usr/local/nagios/etc/objects# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
root@ip-172-31-89-161:/usr/local/nagios/etc/objects#
```

Open the nano editor for localhost.cfg file and make these changes. Add the Ip address of the linux-client for the address field.

nano/usr/local/nagios/etc/objects/monitorhosts/linuxhosts/localhost.cfg



Note - Here replace hostname with linuxserver

nano /usr/local/nagios/etc/nagios.cfg

Add the following line to the nagios.cfg file

cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/

```
# Definitions for monitoring a router/switch
#cfg_file=/usr/local/nagios/etc/objects/switch.cfg

# Definitions for monitoring a network printer
#cfg_file=/usr/local/nagios/etc/objects/printer.cfg


# You can also tell Nagios to process all config files (with a .cfg
# extension) in a particular directory by using the cfg_dir
# directive as shown below:

#cfg_dir=/usr/local/nagios/etc/servers
#cfg_dir=/usr/local/nagios/etc/printers
#cfg_dir=/usr/local/nagios/etc/switches
#cfg_dir=/usr/local/nagios/etc/routers

cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/
```

After making the changes in nagios.cfg file now check validate the file by typing the following command in the terminal.

/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

```
License: GPL

Website: https://www.nagios.org
Reading configuration data...
   Read main config file okay...
   Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
        Checked 16 services.
        Checked 2 hosts.
        Checked 2 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 24 commands.
        Checked 5 time periods.
        Checked 0 host escalations.
        Checked 0 service escalations.
Checking for circular paths...
        Checked 2 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
root@ip-172-31-89-161:/usr/local/nagios/etc/objects/monitorhosts/linuxhosts#
```

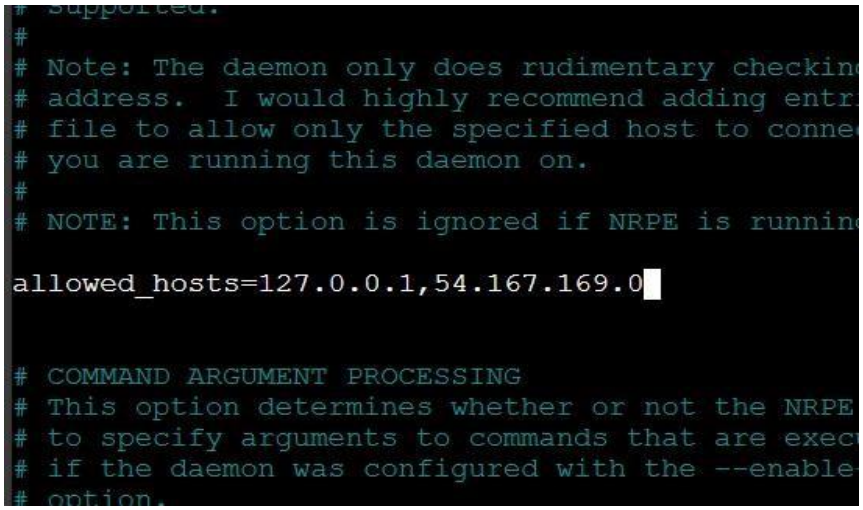
Now restart the service by using this command

service nagios restart

```
root@ip-172-31-89-161:/usr/local/nagios/etc/objects/monitorhosts/linuxhosts# service nagios restart
root@ip-172-31-89-161:/usr/local/nagios/etc/objects/monitorhosts/linuxhosts# systemctl status nagios
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-09-28 17:36:35 UTC; 19s ago
       Docs: https://www.nagios.org/documentation
    Process: 1870 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 1872 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 1874 (nagios)
      Tasks: 8 (limit: 1130)
     Memory: 3.0M (peak: 3.2M)
        CPU: 24ms
     CGroup: /system.slice/nagios.service
             ├─1874 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─1875 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─1876 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─1877 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─1878 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─1879 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─1880 /usr/local/nagios/libexec/check_ping -H 52.207.253.18 -w 3000.0,80% -c 5000.0,100% -p 5
             └─1881 /usr/bin/ping -n -U -w 30 -c 5 52.207.253.18

Sep 28 17:36:35 ip-172-31-89-161 nagios[1874]: qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successfully initialized
Sep 28 17:36:35 ip-172-31-89-161 nagios[1874]: qh: core query handler registered
Sep 28 17:36:35 ip-172-31-89-161 nagios[1874]: qh: echo service query handler registered
Sep 28 17:36:35 ip-172-31-89-161 nagios[1874]: qh: help for the query handler registered
Sep 28 17:36:35 ip-172-31-89-161 nagios[1874]: wproc: Successfully registered manager as @wproc with query handler
Sep 28 17:36:35 ip-172-31-89-161 nagios[1874]: wproc: Registry request: name=Core Worker 1875;pid=1875
lines 1-26
```

Now using this command update the apt repository of ubuntu (linux-client), install gcc, nagios-nrpe-server and nagios-plugin sudo apt update -y sudo apt install gcc -y sudo apt install -y nagios-nrpe-server nagios-plugins

Now open nrpe.cfg file and add the ip address of the nagios host as shown. To open the nrpe.cfg file copy this command.

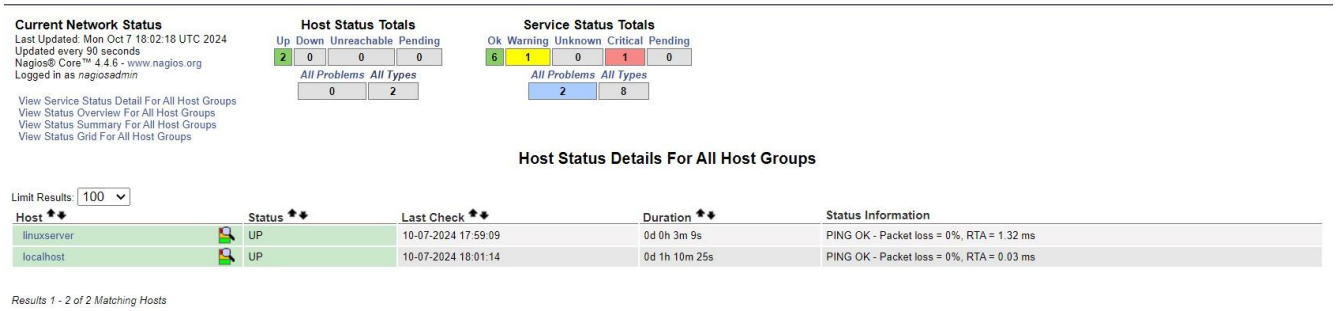


sudo nano /etc/nagios/nrpe.cfg

Now restart nrpe server by using this command

sudo systemctl restart nagios-nrpe-server

Now, check nagios dashboard, you should see linuxserver up and running, if not



Current Network Status
Last Updated: Mon Oct 7 18:02:18 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.4.6 - www.nagios.org
Logged in as *nagiosadmin*

View Service Status Detail For All Host Groups
View Status Overview For All Host Groups
View Status Summary For All Host Groups
View Status Grid For All Host Groups

Host Status Totals

Up	Down	Unreachable	Pending
2	0	0	0

All Problems	All Types
0	2

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
6	1	0	1	0

All Problems	All Types
2	8

Host Status Details For All Host Groups

Limit Results: 100

Host	Status	Last Check	Duration	Status Information
linuxserver	UP	10-07-2024 17:59:09	0d 0h 3m 9s	PING OK - Packet loss = 0%, RTA = 1.32 ms
localhost	UP	10-07-2024 18:01:14	0d 1h 10m 25s	PING OK - Packet loss = 0%, RTA = 0.03 ms

Results 1 - 2 of 2 Matching Hosts

Adv. DevOps Experiment 11

Name : Rohan Lalchandani

Class : D15A Roll no : 25

Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs. Theory:

AWS Lambda

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS). Users of AWS Lambda create functions, self-contained applications written in one of the supported languages and runtimes, and upload them to AWS Lambda, which executes those functions in an efficient and flexible manner.The Lambda functions can perform any kind of computing task, from serving web pages and processing streams of data to calling APIs and integrating with other AWS services.

The concept of "serverless" computing refers to not needing to maintain your own servers to run these functions. AWS Lambda is a fully managed service that takes care of all the infrastructure for you. And so "serverless" doesn't mean that there are no servers involved: it just means that the servers, the operating systems, the network layer and the rest of the infrastructure have already been taken care of so that you can focus on writing application code.

Features of AWS Lambda

- AWS Lambda easily scales the infrastructure without any additional configuration. It reduces the operational work involved.
- It offers multiple options like AWS S3, CloudWatch, DynamoDB, API Gateway, Kinesis, CodeCommit, and many more to trigger an event.
- You don't need to invest upfront. You pay only for the memory used by the lambda function and minimal cost on the number of requests hence cost-efficient. ● AWS Lambda is secure. It uses AWS IAM to define all the roles and security policies.
- It offers fault tolerance for both services running the code and the function. You do not have to worry about the application down. **Packaging Functions**

Lambda functions need to be packaged and sent to AWS. This is usually a process of compressing the function and all its dependencies and uploading it to an S3 bucket.

And letting AWS know that you want to use this package when a specific event takes place. To help us with this process we use the Serverless Stack Framework (SST). We'll go over this in detail later on in this guide.

Execution Model

The container (and the resources used by it) that runs our function is managed completely by AWS. It is brought up when an event takes place and is turned off if it is not being used. If additional requests are made while the original event is being served, a new container is brought up to serve a request. This means that if we are undergoing a usage spike, the cloud provider simply creates multiple instances of the container with our function to serve those requests.

This has some interesting implications. Firstly, our functions are effectively stateless. Secondly, each request (or event) is served by a single instance of a Lambda function. This means that you are not going to be handling concurrent requests in your code.

AWS brings up a container whenever there is a new request. It does make some optimizations here. It will hang on to the container for a few minutes (5 - 15mins depending on the load) so it can respond to subsequent requests without a cold start.

Stateless Functions

The above execution model makes Lambda functions effectively stateless. This means that every time your Lambda function is triggered by an event it is invoked in a completely new environment. You don't have access to the execution context of the previous event.

However, due to the optimization noted above, the actual Lambda function is invoked only once per container instantiation. Recall that our functions are run inside containers. So when a function is first invoked, all the code in our handler function gets executed and the handler function gets invoked. If the container is still available for subsequent requests, your function will get invoked and not the code around it.

For example, the createNewDbConnection method below is called once per container instantiation and not every time the Lambda function is invoked. The myHandler function on the other hand is called on every invocation.

Common Use Cases for Lambda

Due to Lambda's architecture, it can deliver great benefits over traditional cloud computing setups for applications where:

1. Individual tasks run for a short time;
2. Each task is generally self-contained;
3. There is a large difference between the lowest and highest levels in the workload of the application. Some of the most common use cases for AWS Lambda that fit these criteria are: Scalable APIs. When building APIs using AWS Lambda, one execution of a Lambda function can serve a single HTTP request.

Different parts of the API can be routed to different Lambda functions via Amazon API Gateway. AWS Lambda automatically scales individual functions according to

the demand for them, so different parts of your API can scale differently according to current usage levels. This allows for cost-effective and flexible API setups.

Data processing. Lambda functions are optimized for event-based data processing. It is easy to integrate AWS Lambda with data sources like Amazon DynamoDB and trigger a Lambda function for specific kinds of data

events. For example, you could employ Lambda to do some work every time an item in DynamoDB is created or updated, thus making it a good fit for things like notifications, counters and analytics.

Steps to create an AWS Lambda function

Step 1: Create a Lambda Function

1.Choose a Function Creation Method: Select Author from scratch.

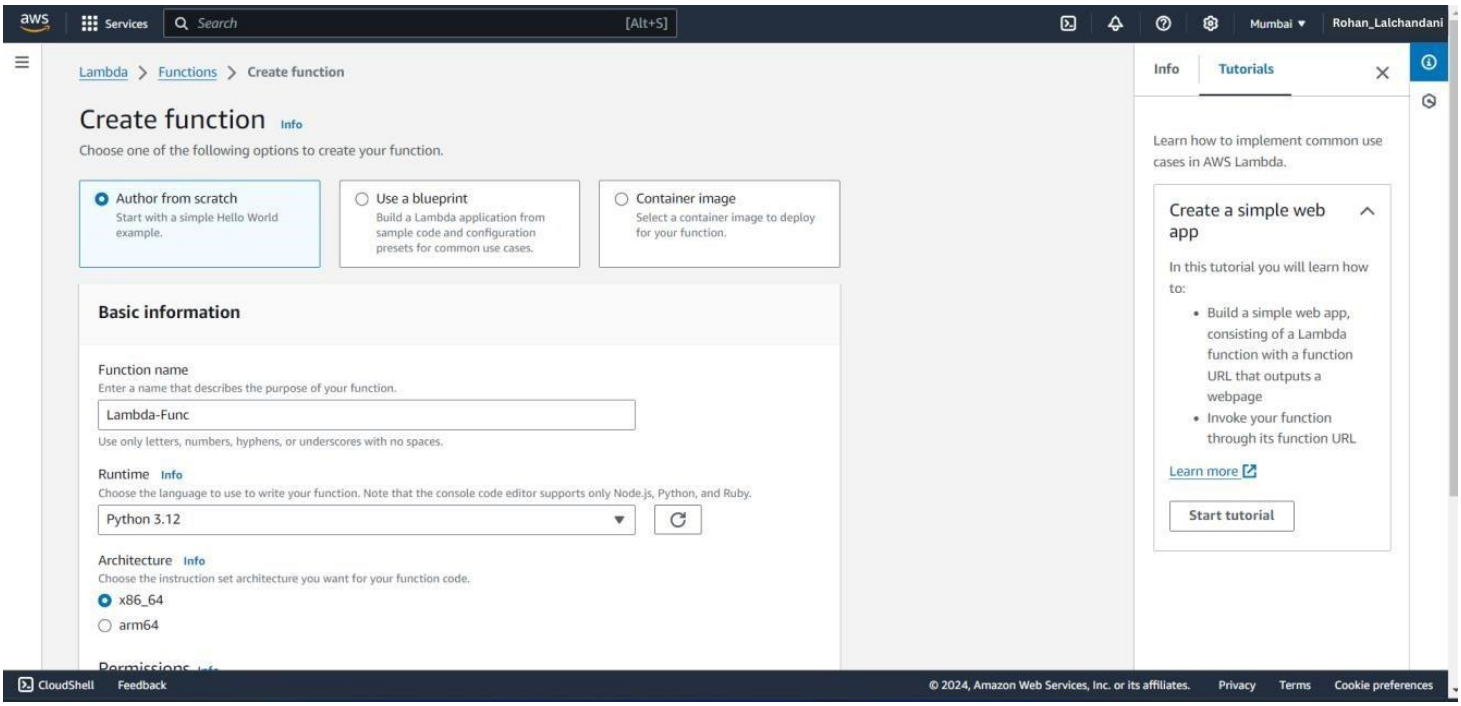
2. Configure the Function:

Function name: Enter a name for your function (e.g., MyFirstLambda).

Runtime: Choose Python 3.x (the latest available version).

Permissions:Choose Create a new role with basic Lambda permissions (this creates a role with the necessary permissions).

3. Click on Create function.



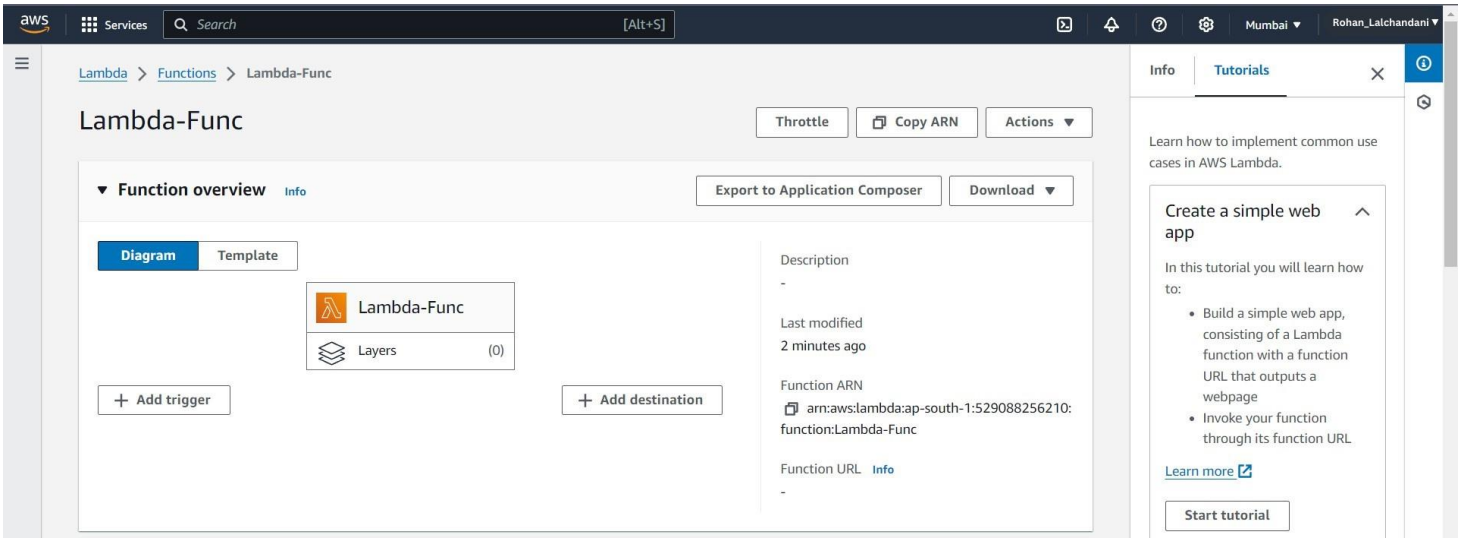
Step 2: Write Your Lambda Function Code

In the Function code section, you will see a code editor. Replace the default code with the following Python code:

python Copy code def lambda_handler(event, context): # This function returns a greeting message name = event.get('name', 'World') return {

'statusCode': 200, 'body': f'Hello, {name}!' }

This function reads a name from the event and returns a greeting message. If no name is provided, it defaults to "World".

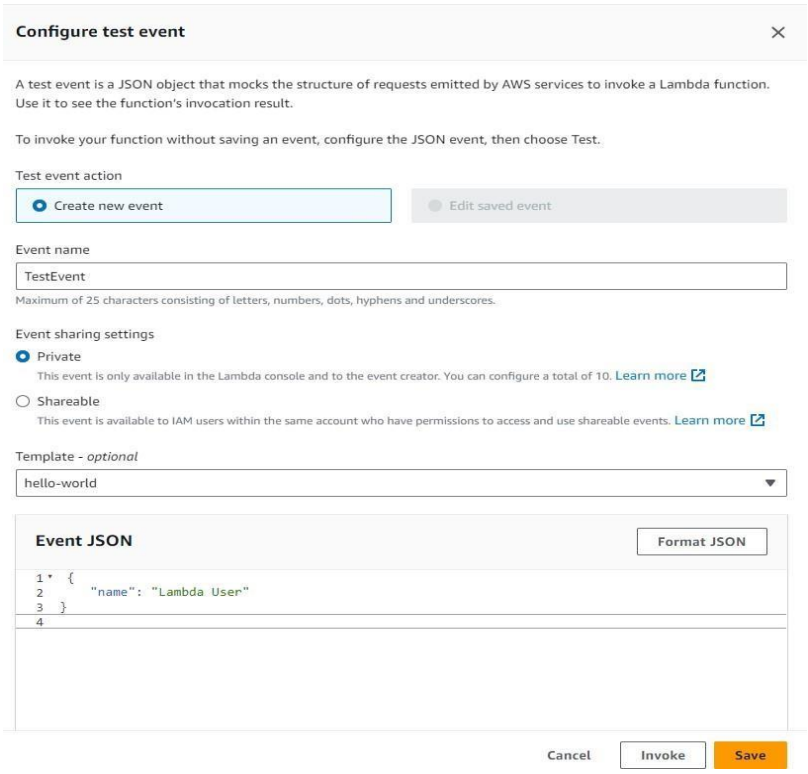


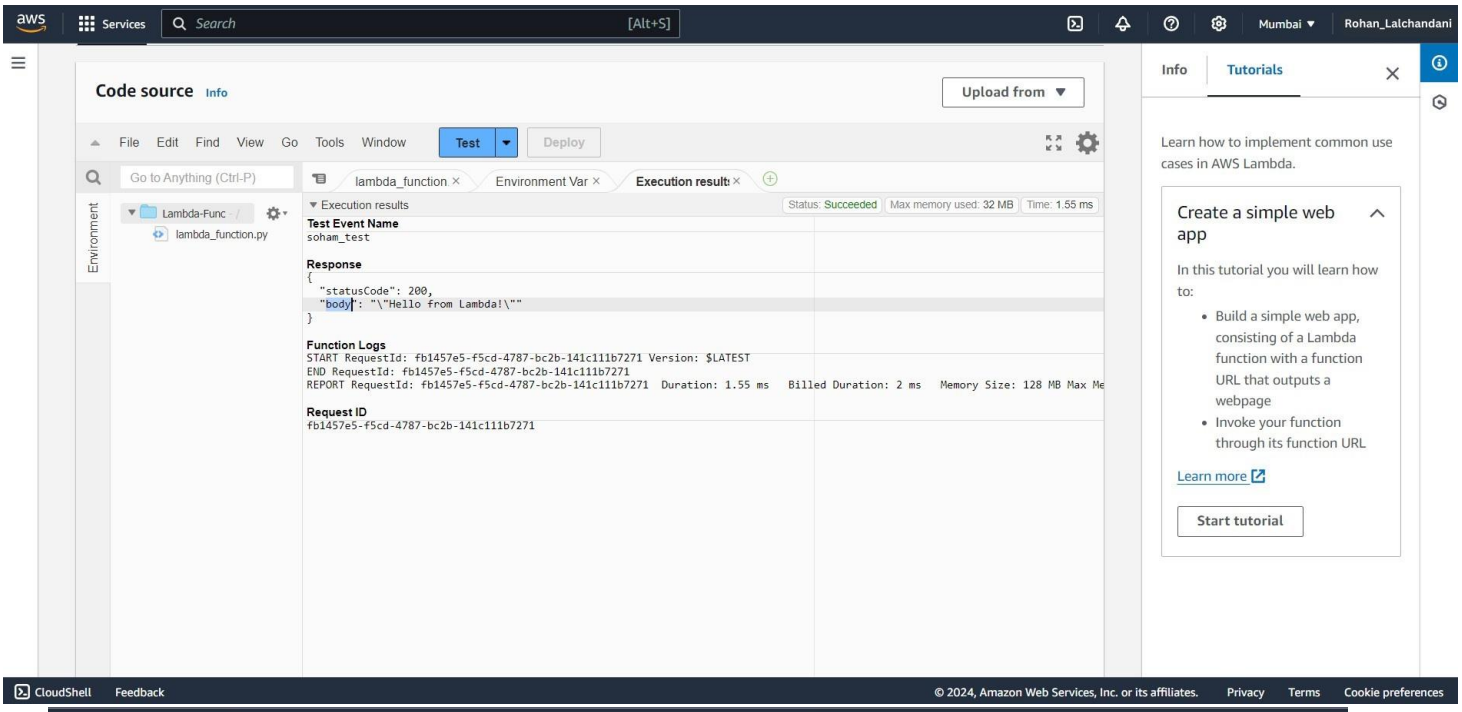
Step3: 1. Configure a Test Event:
Click on the Test button.
In the Configure test event dialog, give your event a name (e.g., TestEvent). Replace the default JSON with the following:

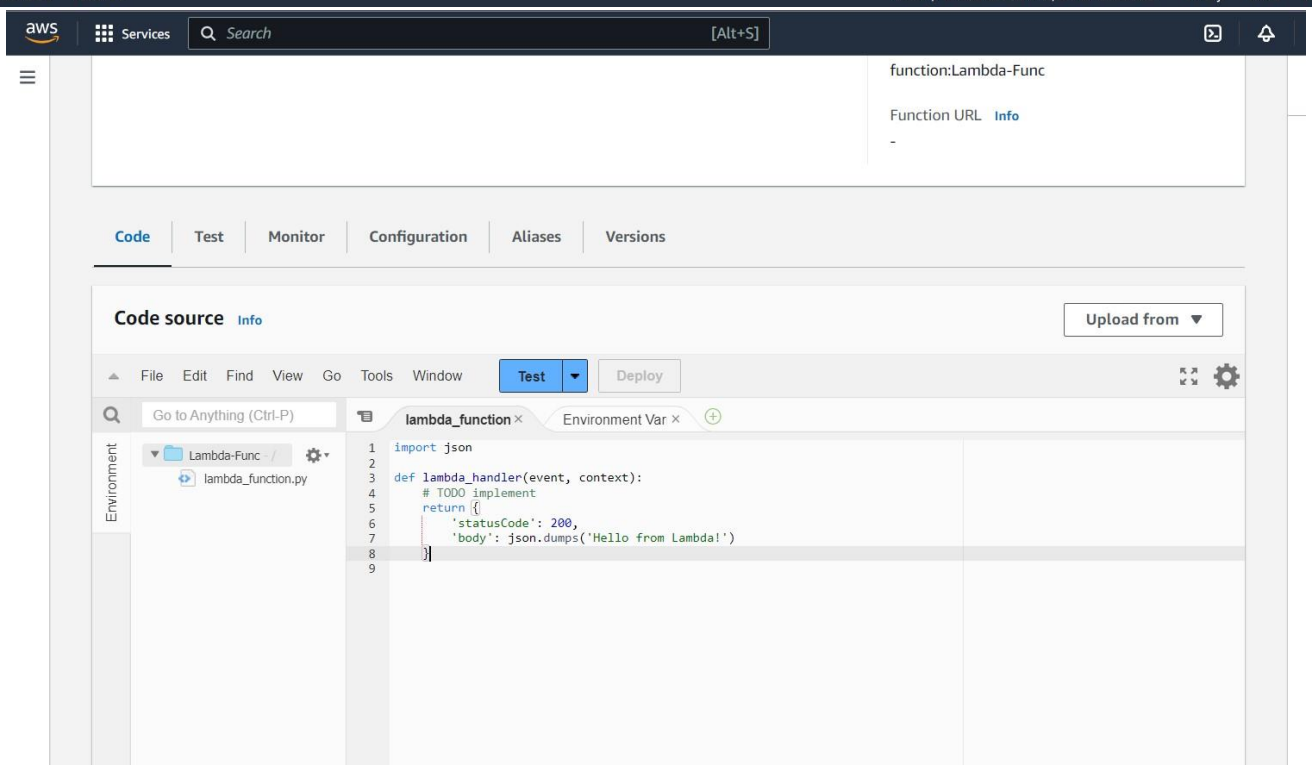
```
{
   "name": "Lambda User"
}
```

2. Run the Test:
Click on the Test button again to execute your Lambda function.
You should see the execution results below the code editor, including the response: json
Copy code

```
{
   "statusCode": 200,
   "body": "Hello, Lambda User!"
}
```







Conclusion:

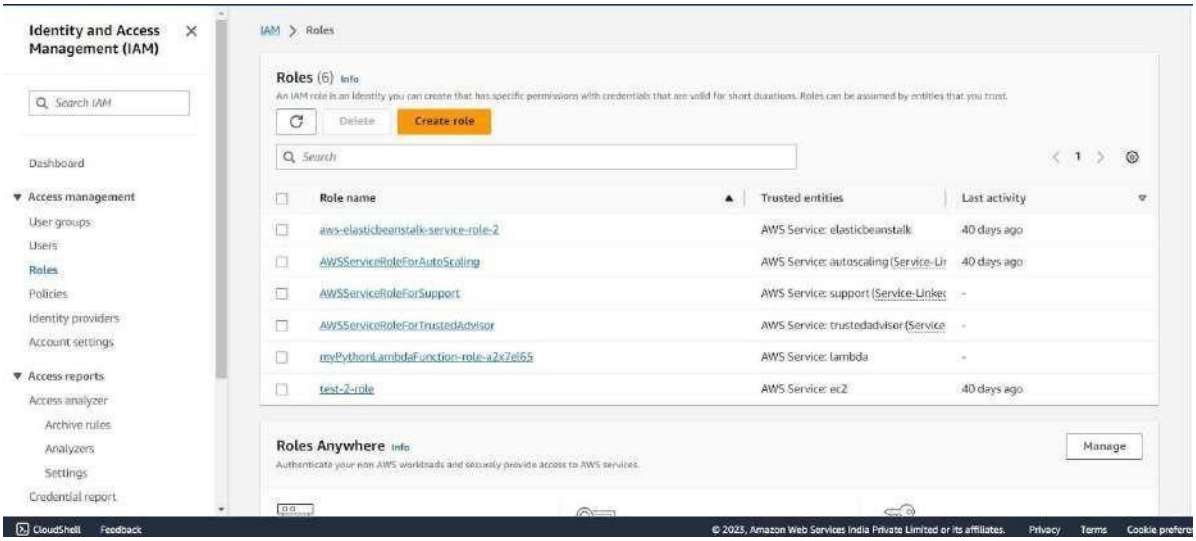
AWS Lambda is a serverless computing service that allows you to run code without managing servers, making it highly scalable, cost-effective, and easy to use. It automatically manages the compute resources, executes your code in response to specific events such as API calls, file uploads, or database updates, and scales based on the demand

Adv. DevOps Exp. 12

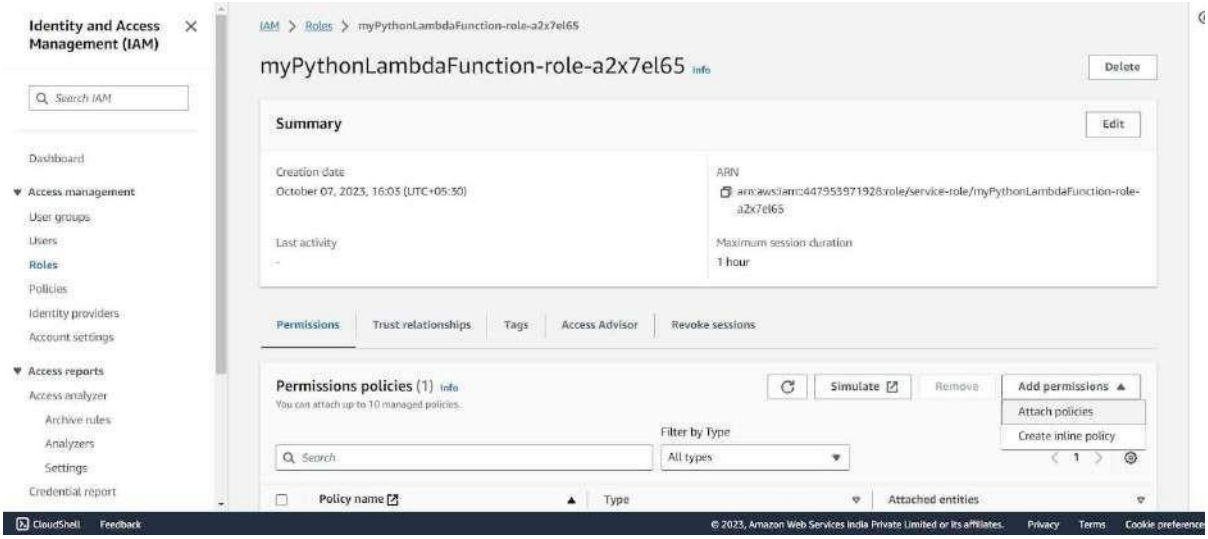
Name : Rohan Lalchandani

Class : D15A Roll no : 25

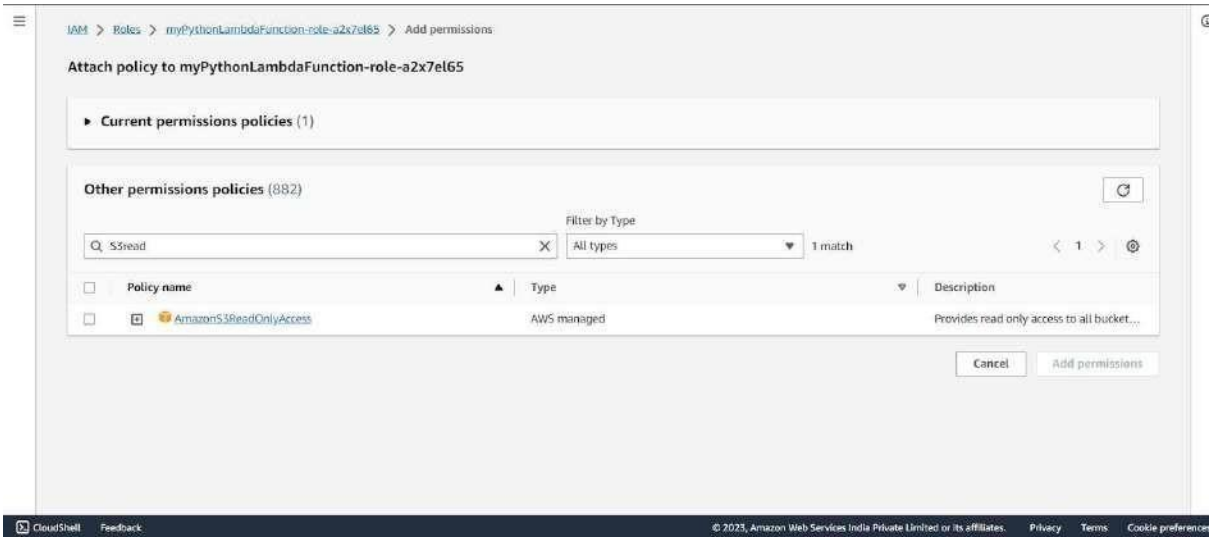
Step 1: Open the IAM (user)



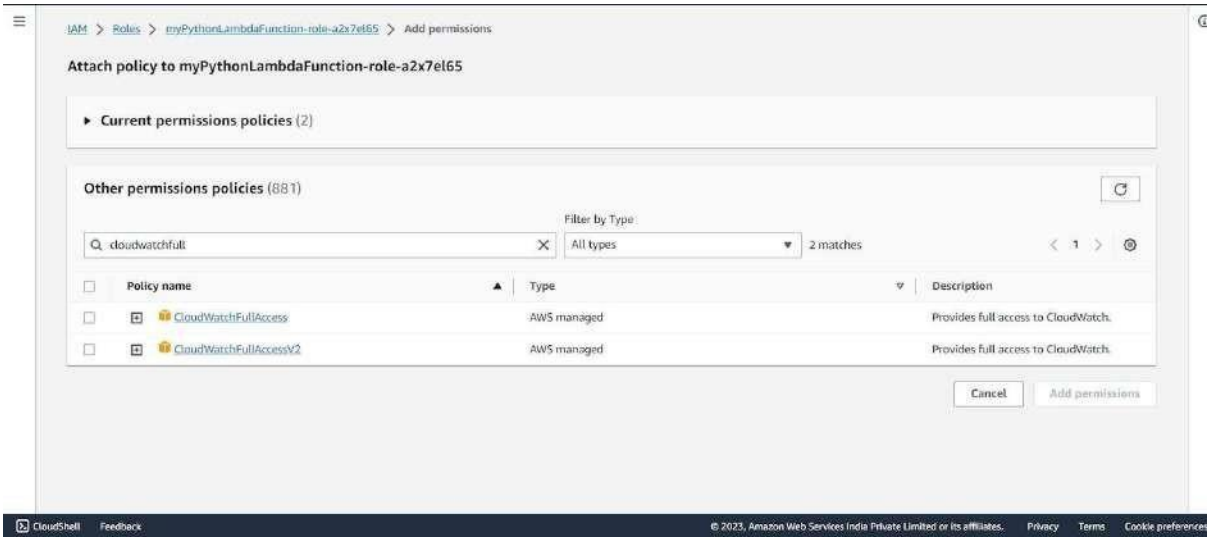
Step 2: Under Attach Policies, add S3-ReadOnly and CloudWatchFull permissions to this role.



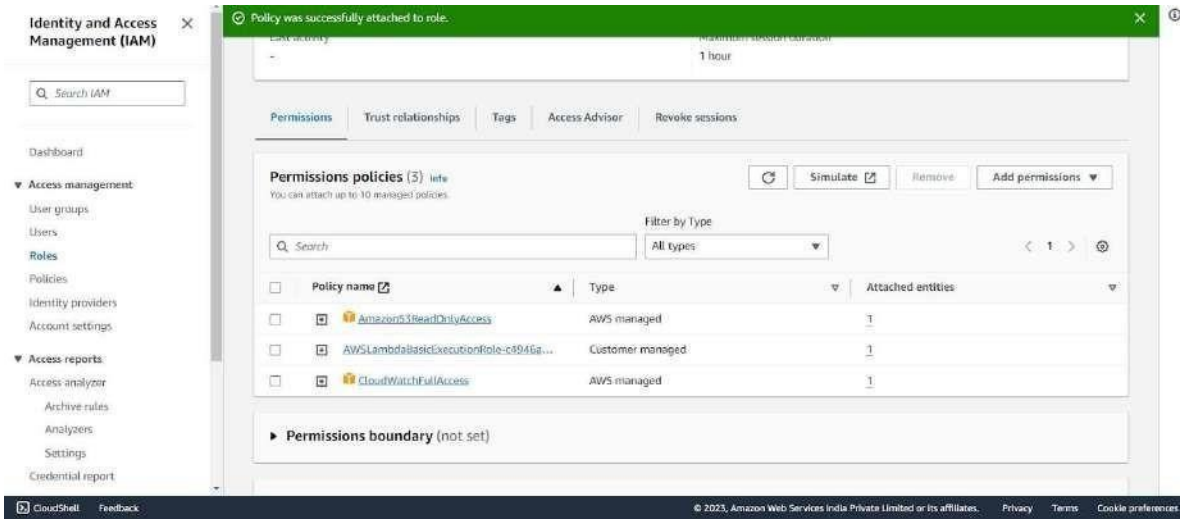
S3-ReadOnly



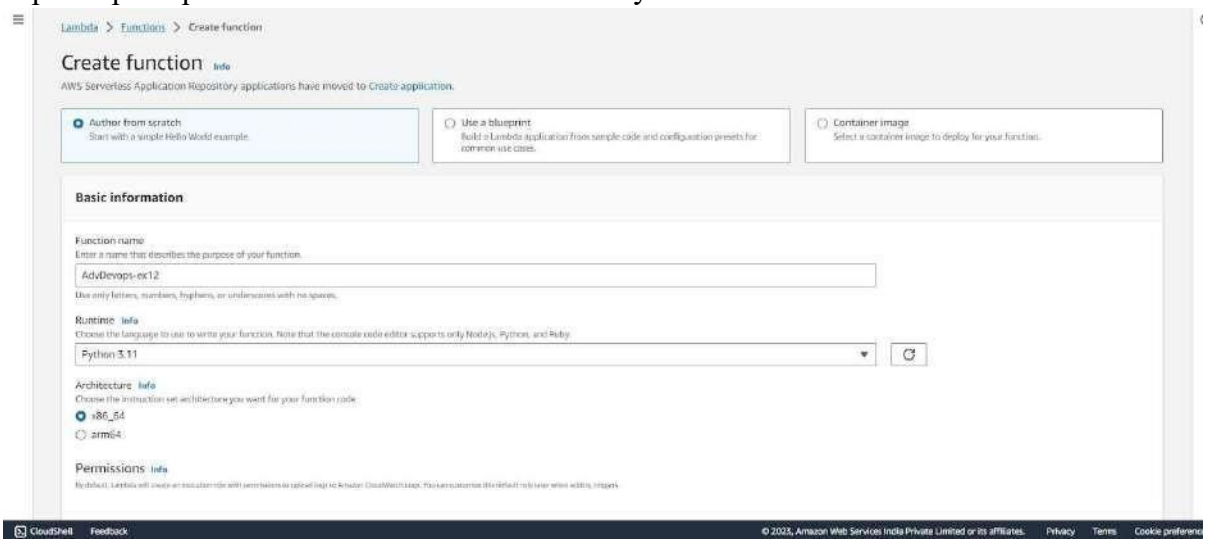
CloudWatchFull



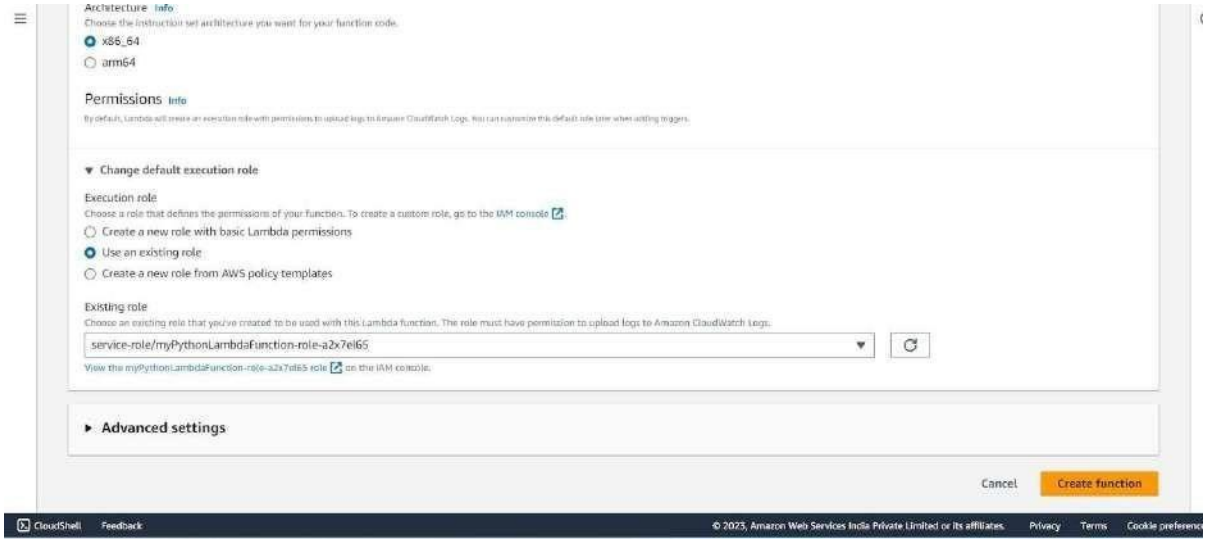
After successful attachment of policy you will see something like this you will be able to see the updated policies.



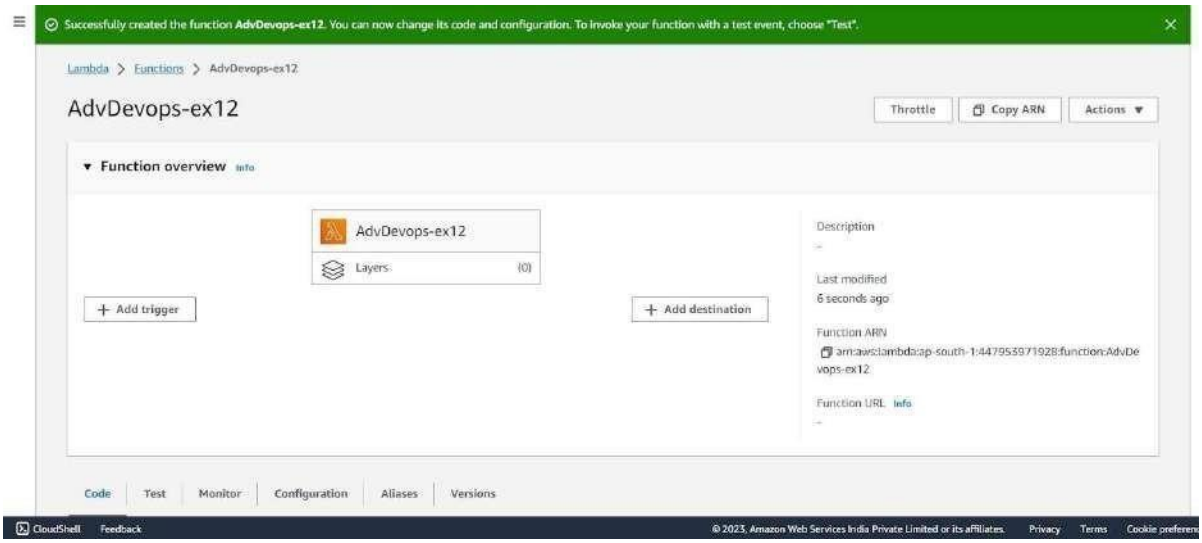
Step 3: Open up AWS Lambda and create a new Python function.



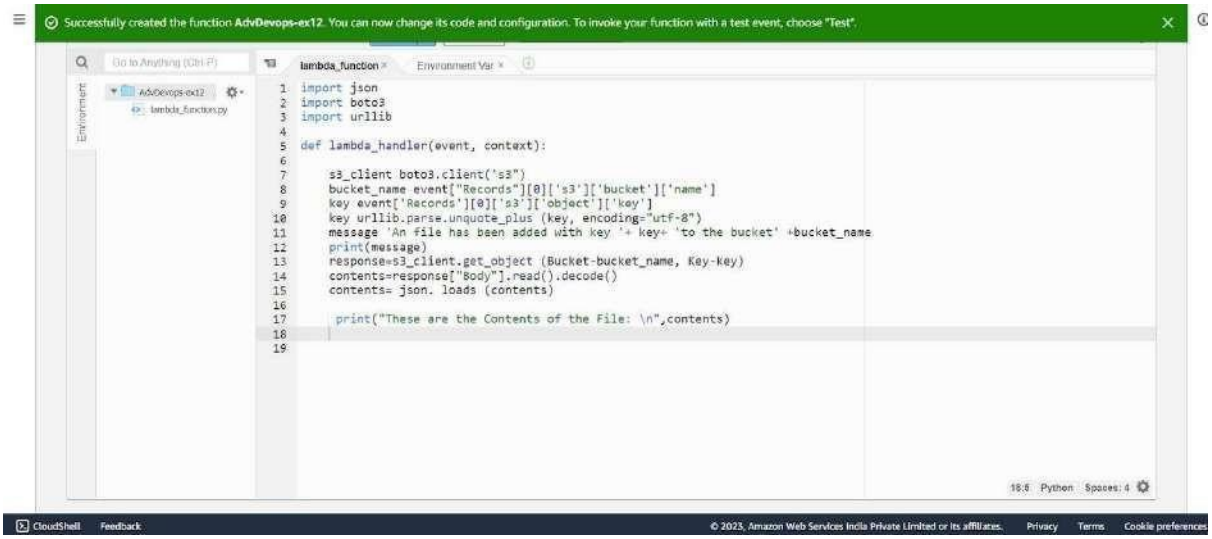
Under Execution Role, choose the existing role, then select the one which was previously created and to which we just added permissions.



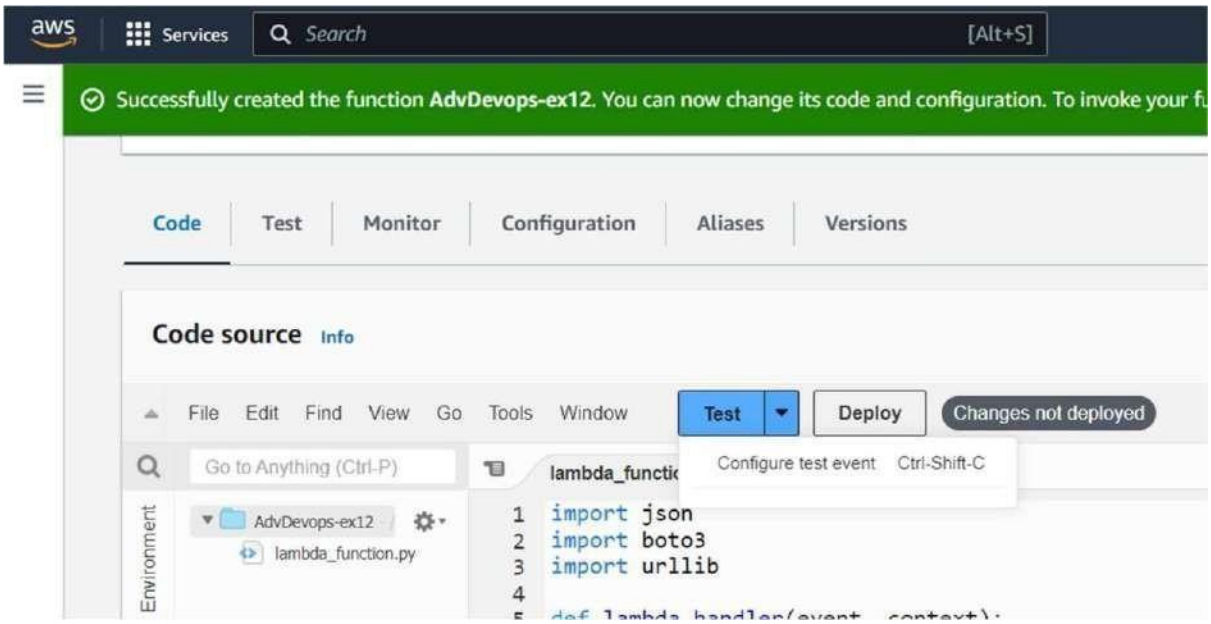
Step 4: The function is up and running.

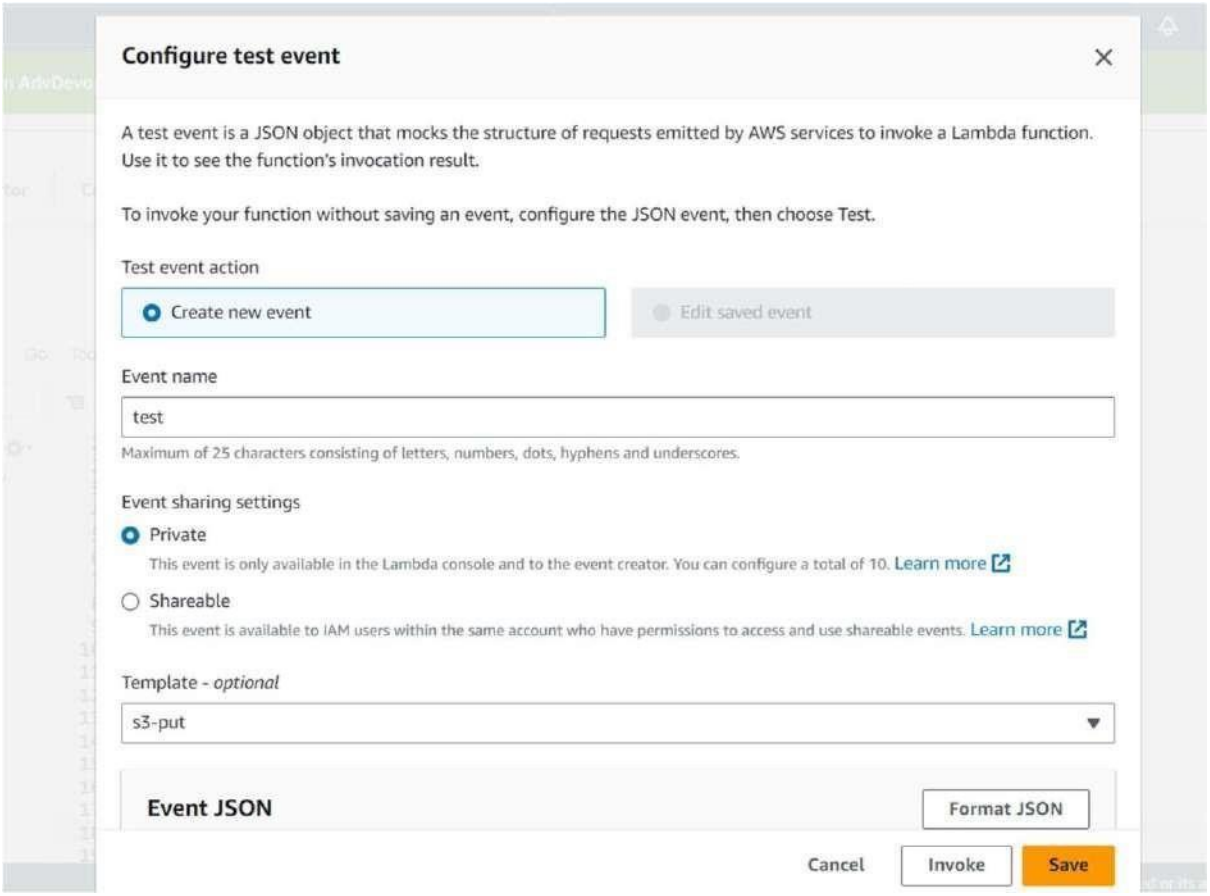


Step 5: Make the following changes to the function and click on the deploy button. This code basically logs a message and logs the contents of a JSON file which is uploaded to an S3 Bucket and then deploy the code.



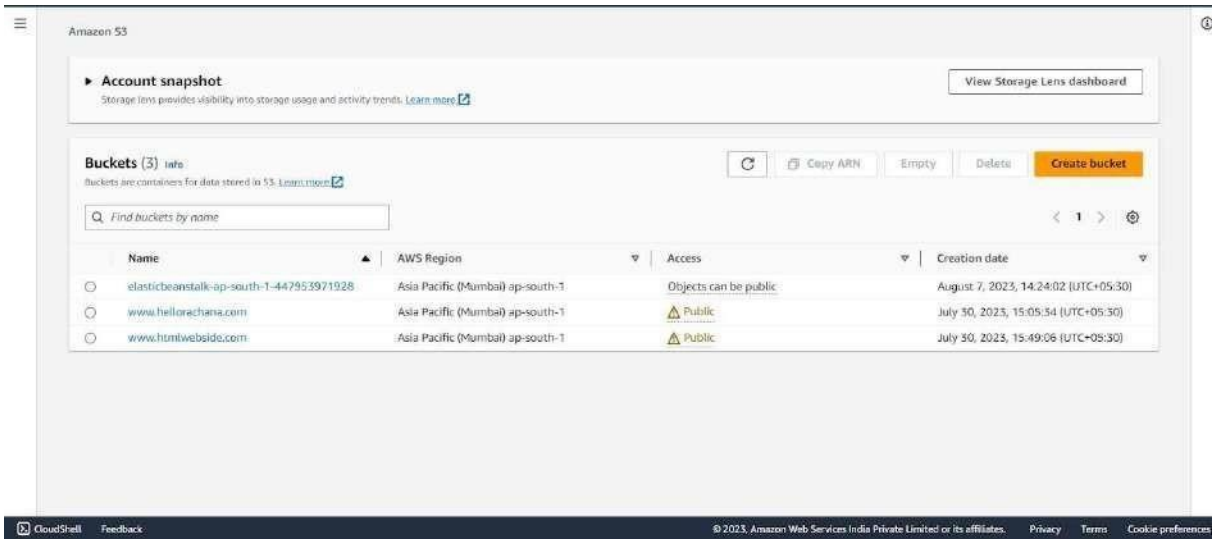
Step 6: Click on Test and choose the ‘S3 Put’ Template.



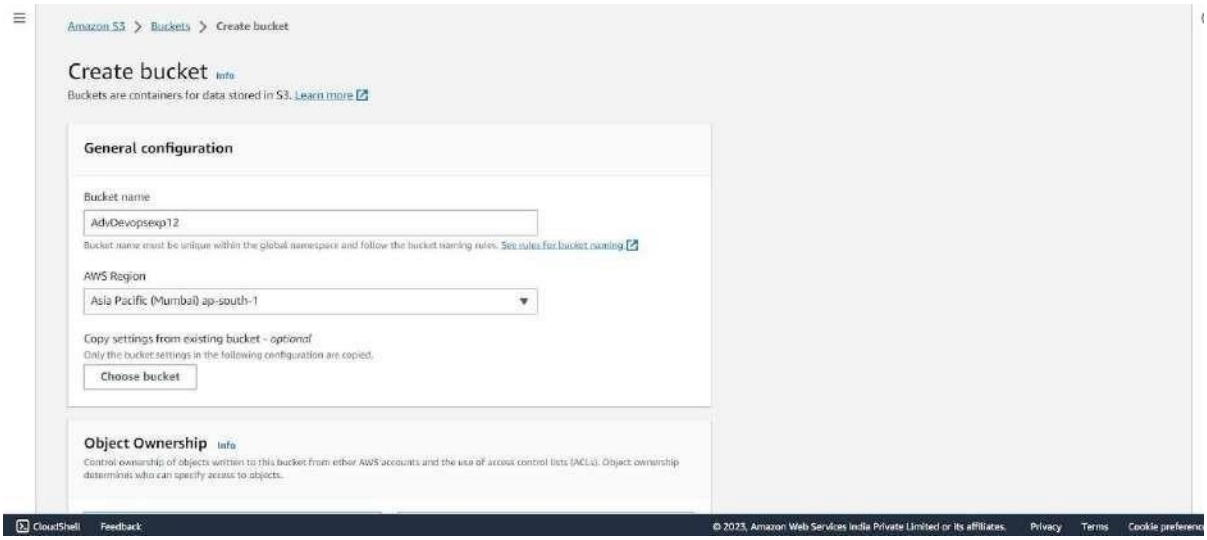


And Save it.

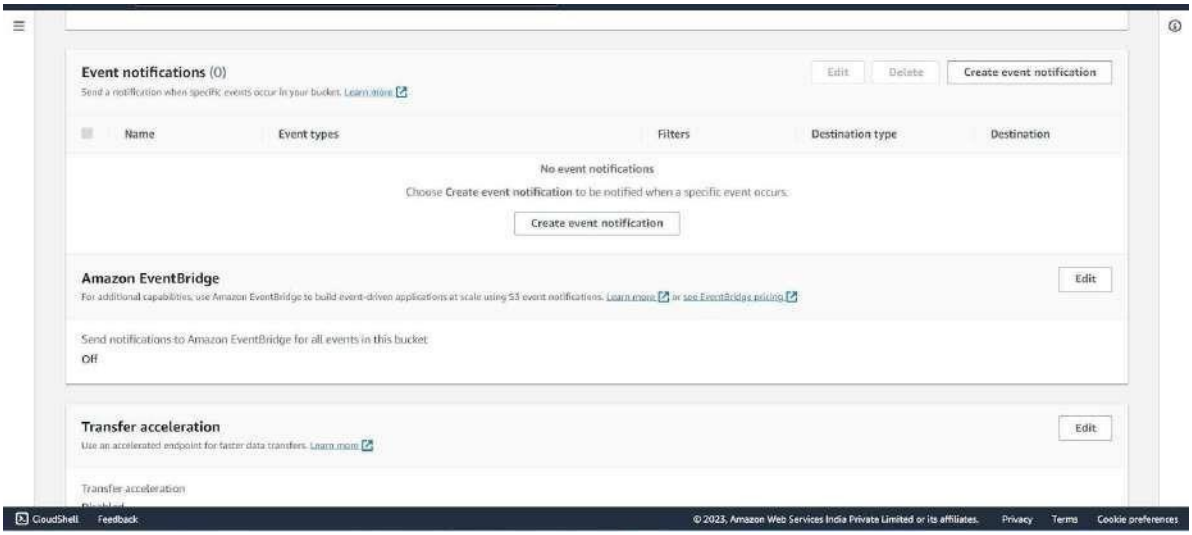
Step 7: Open up the S3 Console and create a new bucket.



Step 8: With all general settings, create the bucket in the same region as the function.

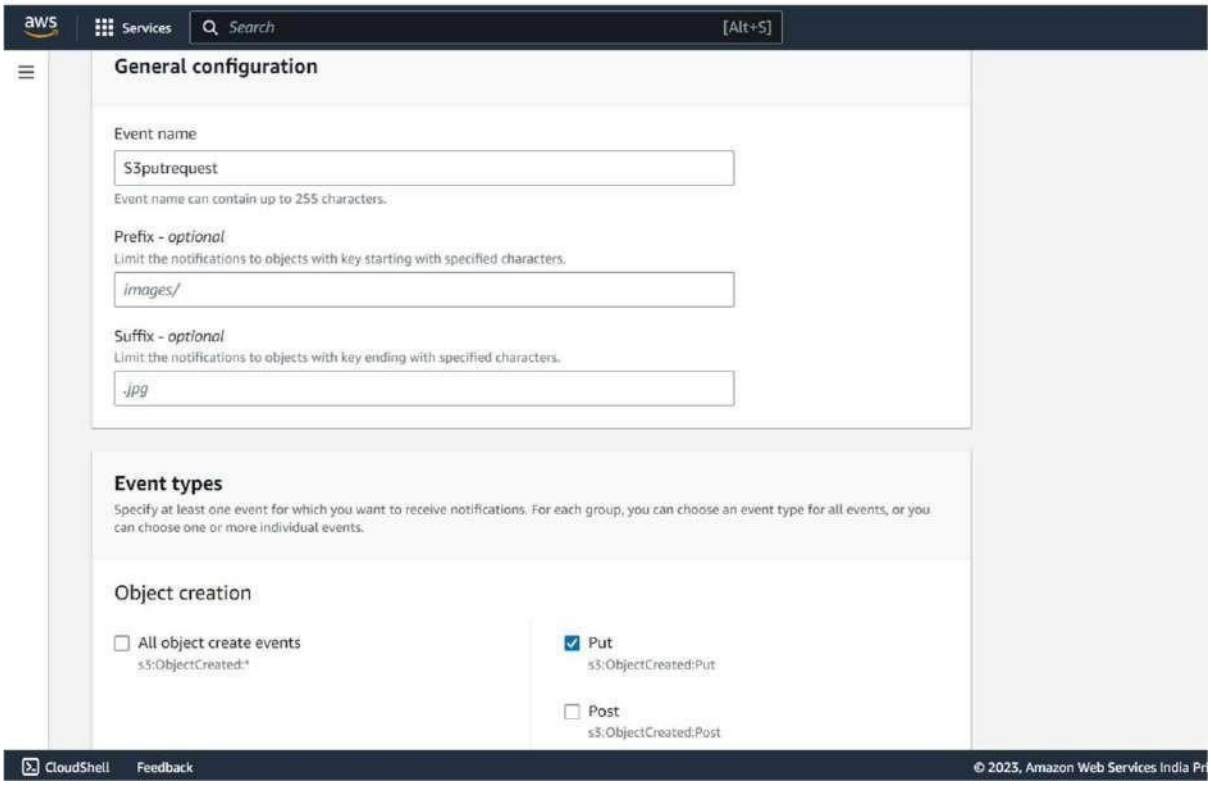


Step 9: Click on the created bucket and under properties, look for events.

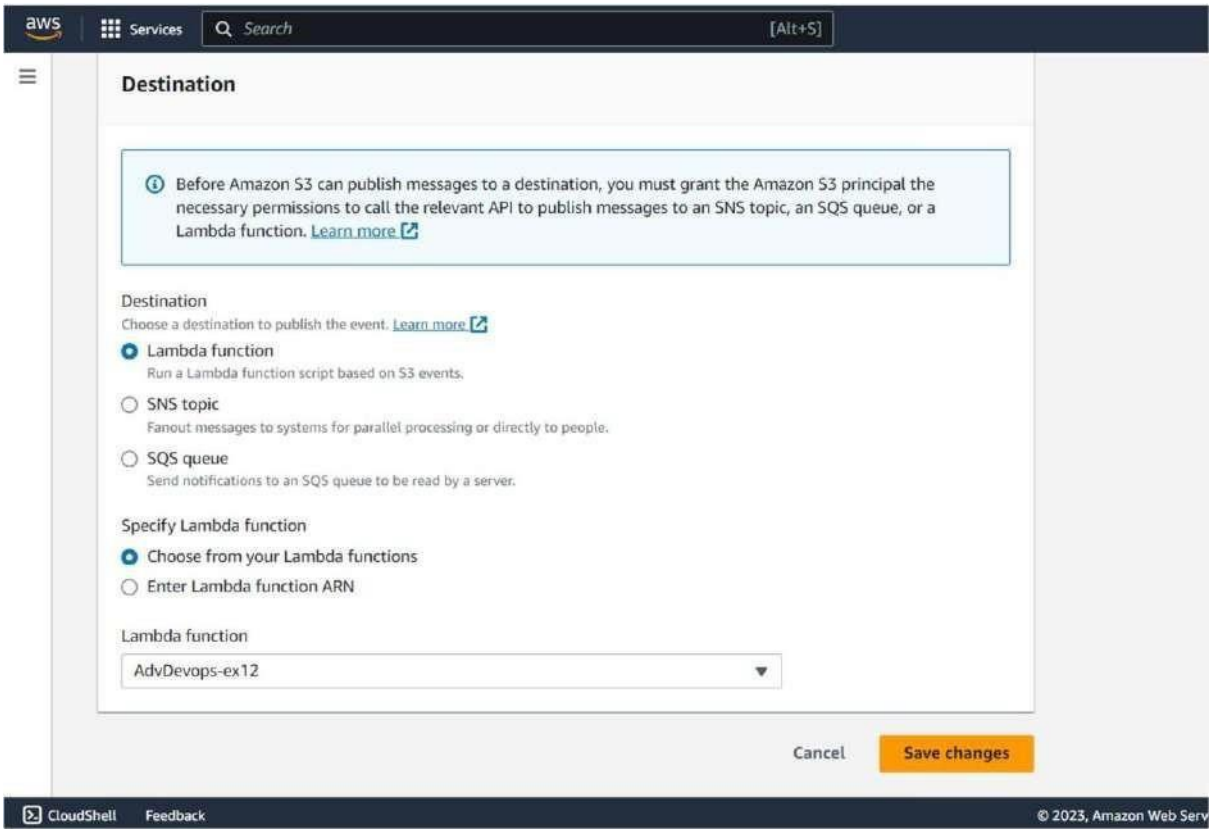


Click on Create Event Notification.

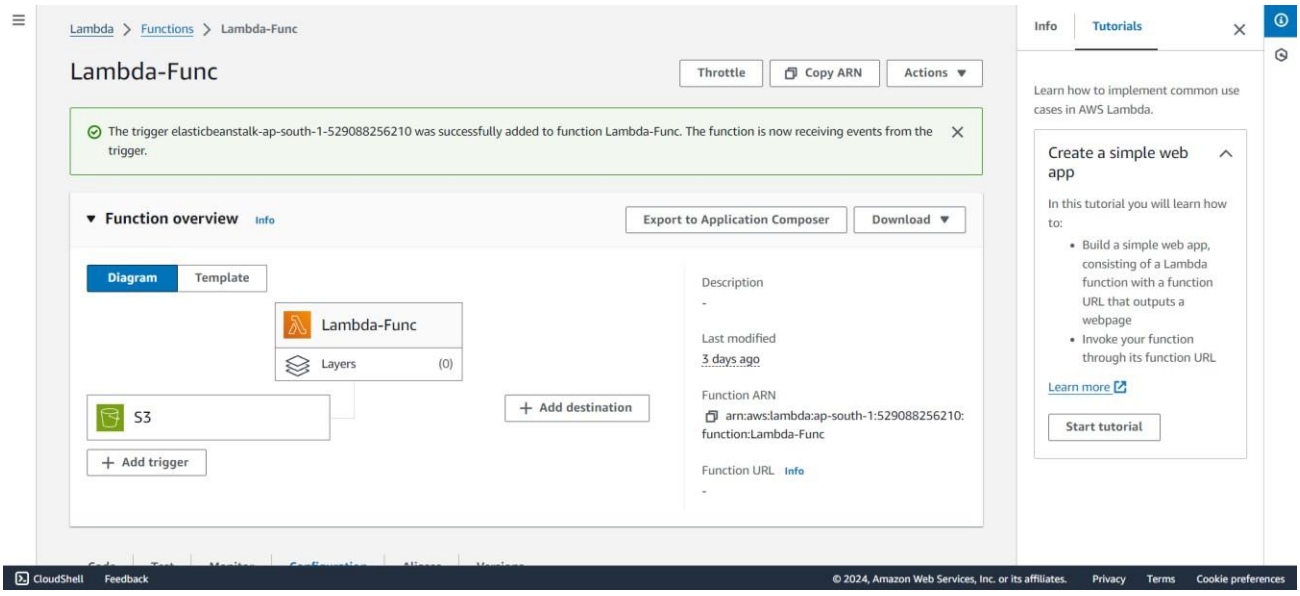
Step 10: Mention an event name and check Put under event types.



Choose Lambda function as destination and choose your lambda function and save the changes.



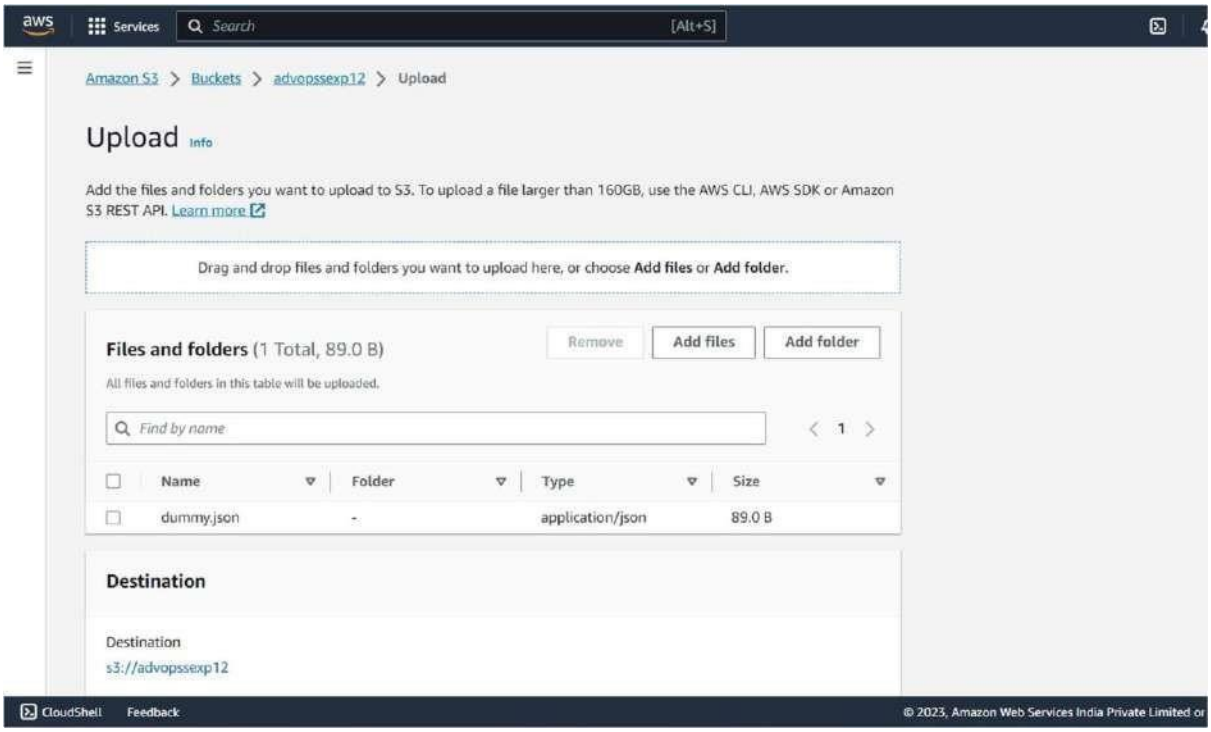
Step 11: Refresh the Lambda function console and you should be able to see an S3 Trigger in the overview.



Step 12: Now, create a dummy JSON file locally.

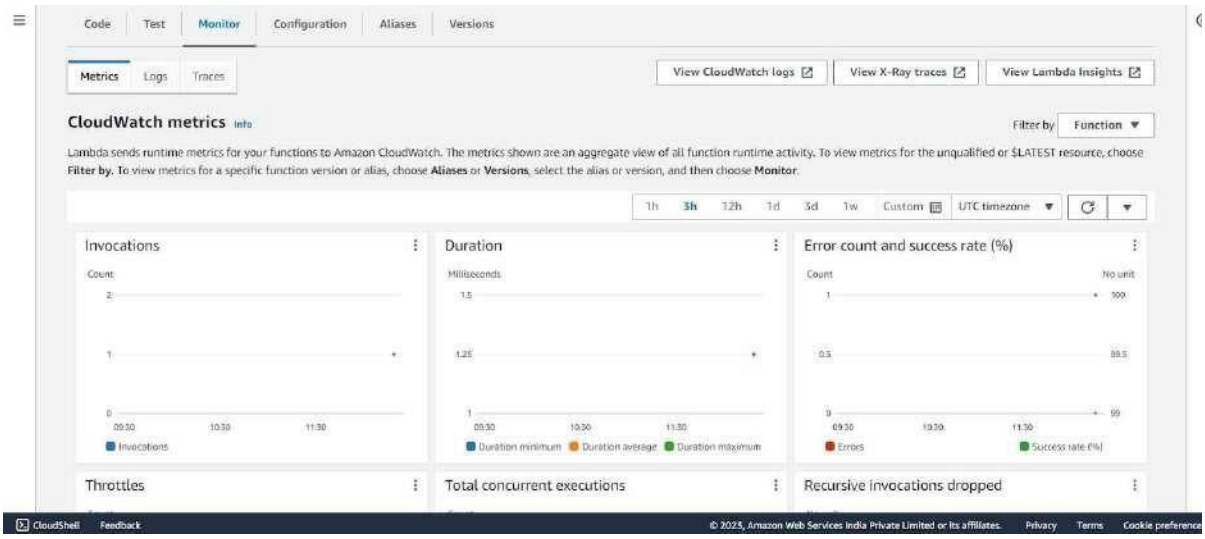
Step 13: Go back to your S3 Bucket and click on Add Files to upload a new file.

Step 14: Select the dummy data file from your computer and click Upload.

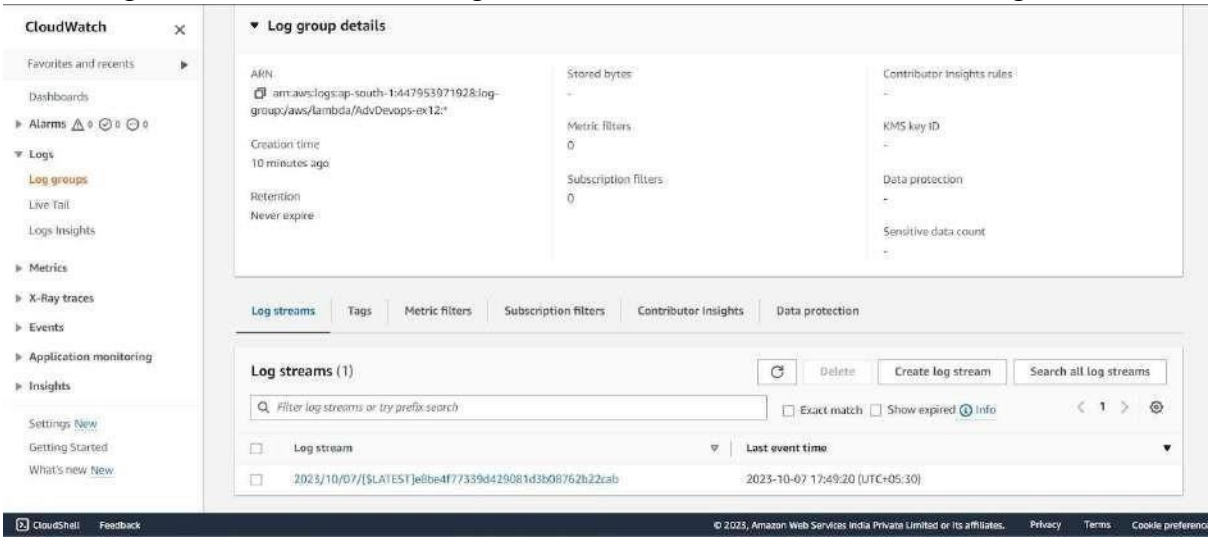


Step 15: After this make the necessary changes in the Test configuration file which we created it previously by replacing the Bucket Name and the ARN of Bucket.

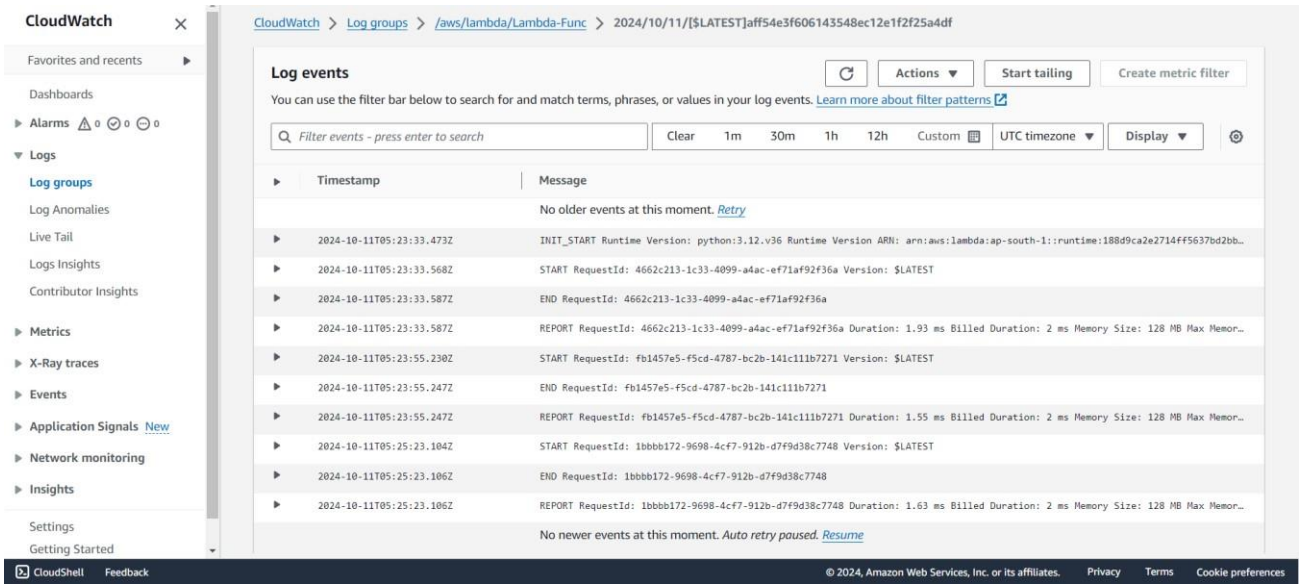
Step 16: Go back to your Lambda function , Refresh it and check the Monitor tab.



Under Log streams, click on View logs in Cloudwatch to check the Function logs.



Step 17: Click on this log Stream that was created to view what was logged by your function.



Conclusion: Thus, we have created a Lambda function which logs “An Image has been added” once you add an object to a specific bucket in S3.

~~Name : Roh~~

Name : Rohan Lalchandani
Class: D15A Roll no. 25
Subject : Adv. DevOps ~~Exp~~ (Assignment - 1)
Date: 27/09/2024

03/09

45

1. Video Streaming using S3 Bucket.

Step 1: Set up your S3 Bucket.

1.(a) Sign in to AWS Management Console
(b) Navigate to S3 & click on "Create Bucket".
(c) Name your bucket (eg my-video-streaming-bucket) and choose a region.
(d) Ensure the bucket name is unique across all of S3

2. Configure Bucket Permissions
(a) Set up the bucket to be private to secure your content
(b) Use AWS Identity & Access Management (IAM) to create a user with permissions to access S3
(c) optionally, configure a bucket policy or access control lists (ACLs) if you want public access for streaming.

~~Step 2 : Configure Bucket Permissions :~~
Step 2 : Upload your Video files.

1. Upload Video files
(a) In S3 bucket, click on "Upload" and choose video files
(b) Consider using formats like MP4 for better compatibility with streaming.

2. Set Metadata for streaming (optional)

FOR EDUCATIONAL USE

Step 3 : Set up Video Streaming :

1. Use Amazon cloudfront

(a) Set up a cloudfront distribution to serve your video content more efficiently.

(b) Point the cloudfront origin to your S3 bucket

(c) Configure caching behaviours based on re

2. Set protocol

(a) Choose whether to use HLS (HTTP live streaming) or DASH (Dynamic Adaptive Streaming over HTTP)

(b) Use AWS Media Convert to convert your video files into streaming - compatible formats if nece

Step 4 : Integrate Video Player

1. Select a video player.

(a) Choose a web video player - that supports streaming (eg video.js, JW Player).

2. Embed the player.

(a) Use HTML to embed the video player on your website

(b) Set the source of the video to your cloudfront URL or S3 URL.

Step 5 : Test your setup

1. Test video playback :

Access website & play video to ensure everything works correctly.

Sundaram

Discuss BMW & Hot Star case studies using AWS.

BMW Case Study on AWS

BMV used aws to accelerate innovation, enhance customer experiences and create data-driven ~~se~~ serives. Their use case centers on the BMW Connected Drive & BMV Cloud Data Hub.

Key Components

1. BMW Connected Drive :

Connected Serives : AWS allows BMW to offer real-time connected serives (including in-car navigation, telematics & driver assistance.

2. Data Processing : BMW uses AWS to process millions of requests daily, analyzing large amounts of data generated by vechiles in real-time.

3. Scalability : The use of amazon S3, Amazon EC2, and other AWS serives enable BMW to scale infrastructure based on growing number of connected vechiles

2. BMW Cloud Data Hub :

1. Global Data Platform : BMW built its Cloud Data Hub on AWS to consolidate manage & analyse vechiles & ~~se~~ user data across the ~~globe~~ globe.

2. Amazon S3 & Amazon Redshift :

BMW uses Amazon S3 for scalable storage & amazon redshift for analysing vast datasets related to vechile performance.

FOR EDUCATIONAL USE

Benefits :
1. Reduced Development Time
2. Global Scale & ~~Availability~~ Availability
3. Cost-effectiveness.

Hotstar (Disney + Hotstar) : Case Study on A

Hotstar, one of India's largest streaming platforms used AWS to stream high definition of vido to millions of concurrent users especially during live events such as IPL cricket matches. Their main challenge was scalability during the events, where millions of users could be streaming simultaneously.

Key Components :
1. Scalability During peak events
   - Elastic Compute (Amazon EC2) :
Hotstar uses Amazon EC2 for elastic, scalable computer power, especially during peak streaming events like IPL & cricket matches.

2. Auto Scaling :
AWS auto scaling allows hotstar to dynamically adjust its infrastructure based on user demand.

Sundaram
FOR EDUCATIONAL USE

2. Content Delivery Using Amazon EC2.

1 Global CDN: Hotstar uses Amazon Cloudfront to deliver video content worldwide, ensuring low latency & high throughput.

2 Edge locations: By caching content at AWS edge locations across the globe, Hotstar can reduce the load on its servers & deliver content more efficiently.

3: Data Analytics & ML

(a) Amazon Kinesis: Uses ~~am~~ this to monitor real time viewing statistics & engagement metrics during live streams.

(b) Amazon S3 & Redshift:
Hotstar ~~uses~~ stores user data and analytics in amazon S3 and processes this data using Amazon Redshift to derive insights on viewer preferences.

Benefits:
1) Handling Massive Traffic
2) Low Latency Streaming
3) Faster Time to Market
4) Improved User ~~Exp~~ Experience.

Q.3. Why Kubernetes ?

Kubernetes (often abbreviated as K8s) is an open-source platform designed to automate deploying, scaling and operating containerized applications. Containers help package applications and their dependencies, ensuring consistency across environments. Kubernetes takes containerization further by orchestrating these containers, making it easier to manage large-scale distributed systems.

Advantages :

1. Scalability : It allows horizontal scaling of containerized applications, making it easy to handle fluctuating workloads & large-scale deployments.
2. Portability : Kubernetes is platform-agnostic, meaning applications can run on any cloud provider or on-premises environment without significant modifications.
3. Self Healing : Kubernetes automatically monitors the health of nodes & containers, restarting failed containers & redistributing workloads to ensure service continuity.
4. Automation : Kubernetes automates various operational tasks such as container scheduling, scaling & load balancing reducing manual intervention & simplifying management.

Sundaram®
FOR EDUCATIONAL USE

Disadvantages :

1. Complexity : Kubernetes has a steep learning curve & requires considerable expertise to manage effectively. The system is complex especially for small teams or beginners in container orchestration.
2. Overhead : Running Kubernetes adds operational overhead, requiring significant resources to maintain infrastructure like control planes, nodes & networks.
3. Monitoring & Debugging :

Debugging issues in Kubernetes can be difficult due to its distributed nature.

How Addidas uses Kubernetes ?

Addidas, the global sportswear company adopted Kubernetes as a part of its digital transformation strategy, especially to modernize its ecommerce platform and build a cloud-native infrastructure.

Here's how Addidas uses Kubernetes :

1. Microservices architecture :-

Addidas moved from a monolithic application architecture to microservices using Kubernetes. This allows them to develop, deploy & scale individual services independently.

FOR EDUCATIONAL USE

2. Scalability -
Addidas e-commerce platform experiences high traff during product launches or sales. Kubernetes he scale applications on-demand by automatically increas decreasing resources based on traffic load.

3. Cloud Native & Multi-Cloud ~~Strategy~~ Strategy
Addidas uses Kubernetes to maintain cloud-native infrastructure. This abstraction layer helps them be cloud-agnostic, allowing deployments across multiple cloud providers.

Q.4. What is Nagious ?

Nagious is an open-source ~~no~~ monitoring tool designed to track performance, availability and health of IT infrastructure, including servers, network devices, applications & services. It helps administrators identify & resolve issues before they affect end users ensuring continuous service availability

Key features :
1) Monitoring of infrastructure
2) Alerts
3) Reporting & logs
4) Plugins

Sundaram
FOR EDUC

How Nagios is used in E-services ?

E-services refer to digital platforms / services over the internet such as e-commerce platforms, online banking or cloud-based applications.

1. Monitoring Service ~~to~~ Availability :
(a) Service Uptime monitoring - It continously monitors the availability of e-services like web servers (eg. Apache, nginx)
(b) Web Application monitoring - For e-services, Nagios checks critial components.
2. Network Monitoring
(a) Tracking network health - It monitors network devices (routers, switches, etc) - bandwidth, usage & response time to ensure good network performance.
3. Performance Monitoring
4. Security Monitoring
5. Alerting & Notification Systems.

Name : Rohan Lalchandani Sub : Adv. DevOps (Assignment - 2)
Class : D15 A Roll no. 25
NAME: STD.: DIV.:
DATE :
PAGE :
55

Create a REST API with serverless framework.
Creating REST API with serverless framework is an efficient way to deploy serverless applications that can scale automatically without managing servers. and serverless applications (i) Serverless framework :
A powerful tool that deployment of services and serverless applications across various cloud providers such as AWS, Azure and Google Cloud.
(ii) Serverless architecture :
This design model allows developers to build applications without worrying about underlying infrastructure enabling focus on code & business logic.
(iii) REST API : Representational state transfer is architecture style for designing network applications

Steps :
1) Install serverless framework
2) Creating a Node.js serverless project
3) Create a REST API resource
4) Deploy the service.
5) Testing the API.
6) Storing data in Dynamo DB
7) Adding more functionalities like to list all, get candidates by ID
8) AWS IAM permissions : You need to ensure serverless framework serverless framework is given right permissions to interact with AWS resources.
9) Monitoring & maintainance :
After deployment serverless framework provides service information like deployed endpoints, API key, log streams.

For Educational Use
WORLD STAR

NAME: STD.: DIV.: DATE : PAGE :

Q.2. Case study for Sonar Qube.

Ans 1) Sonarqube is an open-source platform for testing project quality. Use Sonarqube to analyze code.

ii) It detects bugs, code smells and security vul in project across various programming languages

1) Profile creation in Sonar Qube
Quality profiles are essential configuration that define rules applied during code analysis. Each project has a quality profile for every language with default being 'Sonar way' profil

2) Using Sonar Cloud to analyze github code
Sonarcloud is counterpart Cloud-based part of Sonar Qube that directly integrate with Github, Bi azure, etc. To get started with ~~Sonar Qu~~ Sonar Cloud via ~~G~~ Github Signup via Sonarcloud product page and connect your github organizat personal account. Once connected, sonarcloud ~~wo~~ mir your github ~~set~~ setup with each project ~~corr~~ corresponding to github repository. Automatic analys happens ~~at~~ automatically ~~so~~ in sonarcloud.

3) Sonarlint in Java IDE:
Sonarlint ~~is~~ is an IDE that performs ~~on~~-the-fly code analysis as you write code. It helps develop detect bugs, security vulnerabilities and code smells directly in development environment suc

WORLD STAR For Educational Use

Intellig, Idea or Eclipse.

4) Analyzing Python project with Sonarqube :
Sonarqube supports python test coverage, reporting but it requires 3rd party tool like coverage.py to generate the coverage part.
To enable coverage adjust your blid process so that process tool runs before sonarscanner and ensure the report file is saved in different paths.

5) Analyzing Node.js project with SonarQube.
For Node.js project sonarqube can analyze Javascript and Typescript code. Similar to python setup you can configure Sonarqube to analyze node.js projects by installing appropriate plugins.

Implementing "a self service" infrastructure model using terraform can transform how large organizations manage their infrastructure independently, organizations can enhance efficiency, reduce bottlenecks and ensure compliance with established needs.

Benefits of using terraform :
1) Modularity & reusability
2) Standardization
3) Increased efficiency
4) Integration with tiketing systems

WORLD STAR For Educational Use