ADVANCE DEVOPS EXPERIMENT NO.1

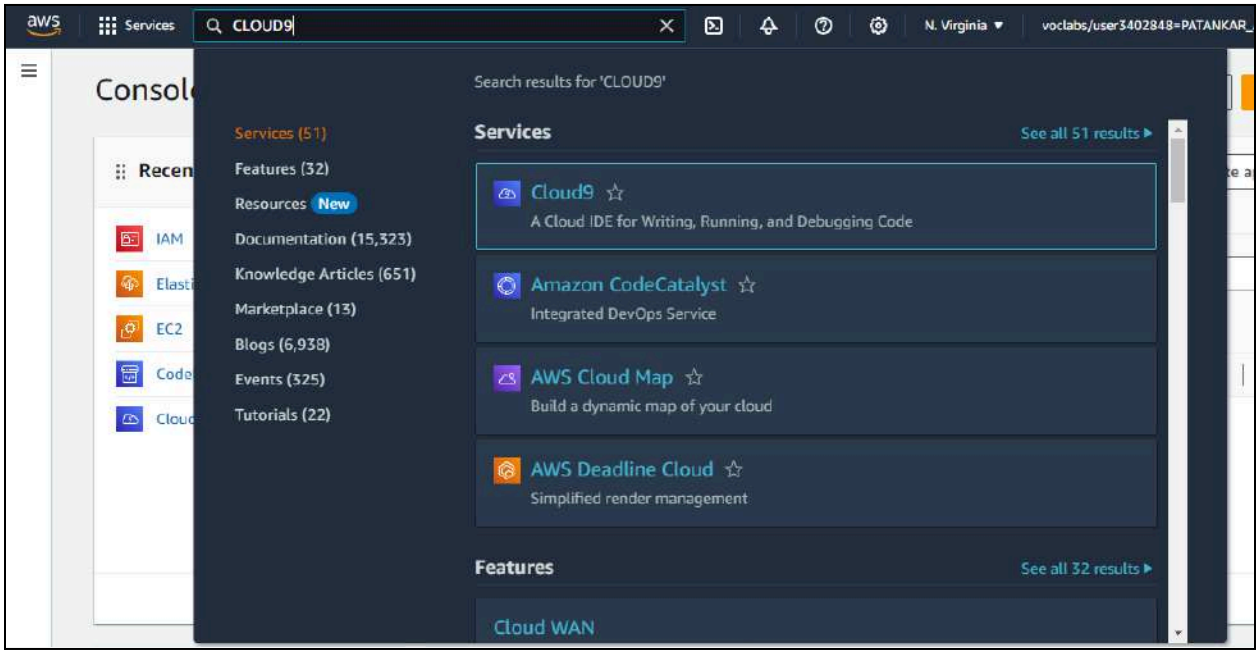
Name: Aryan Anil Patankar
Class:D15A
Roll No:34

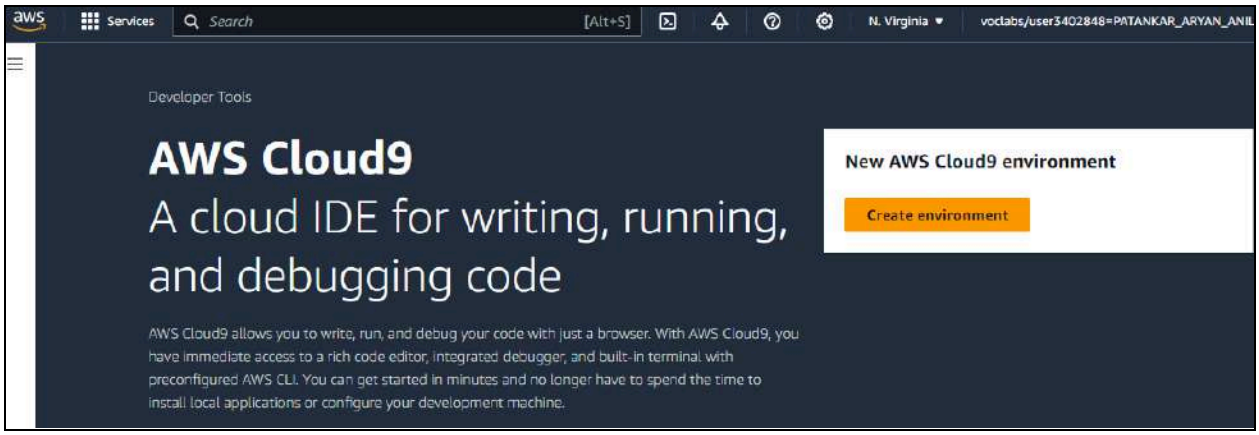
Aim:To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.

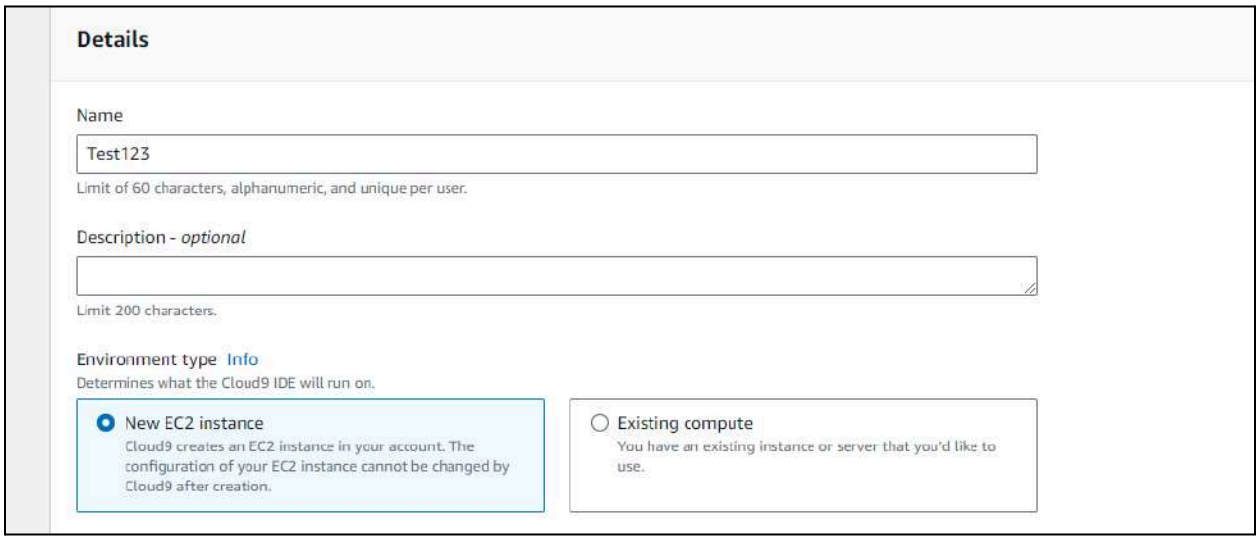
Cloud9

Steps:

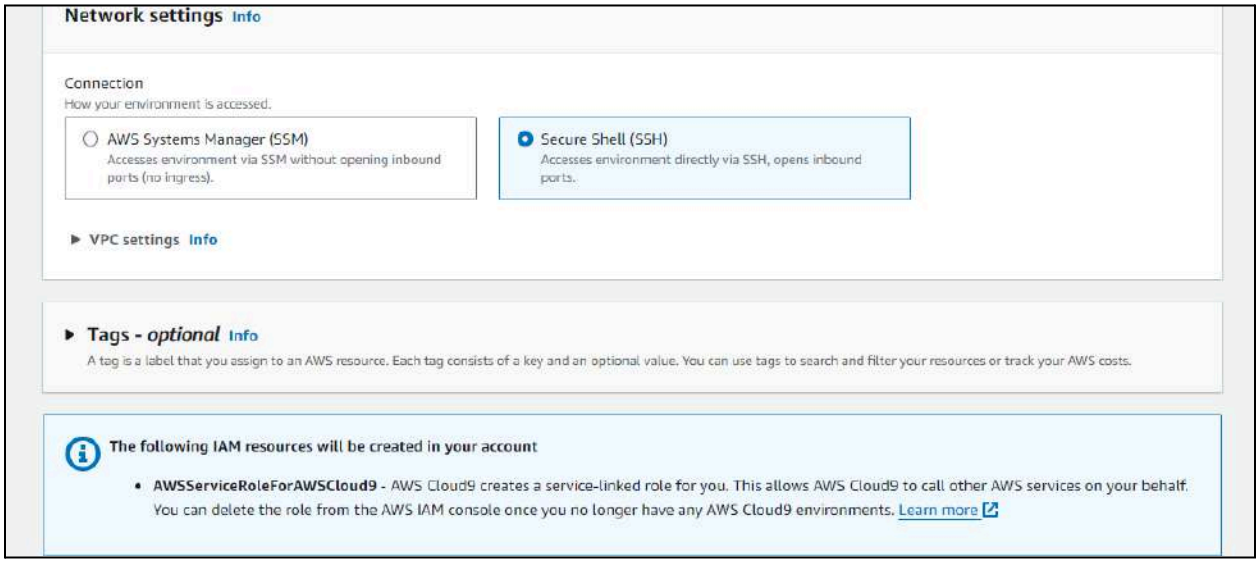
1.Open your AWS account and search for Cloud9 service inside Developer tools. Create a new Cloud9 environment by filling in the required details. Make sure you use an EC2 instance to create your environment.

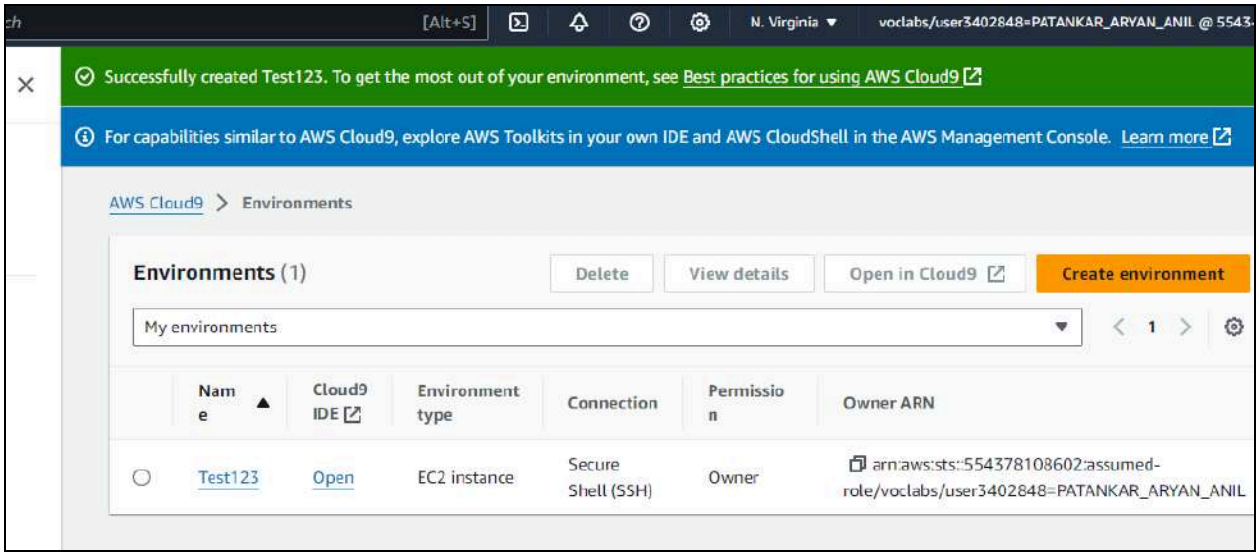


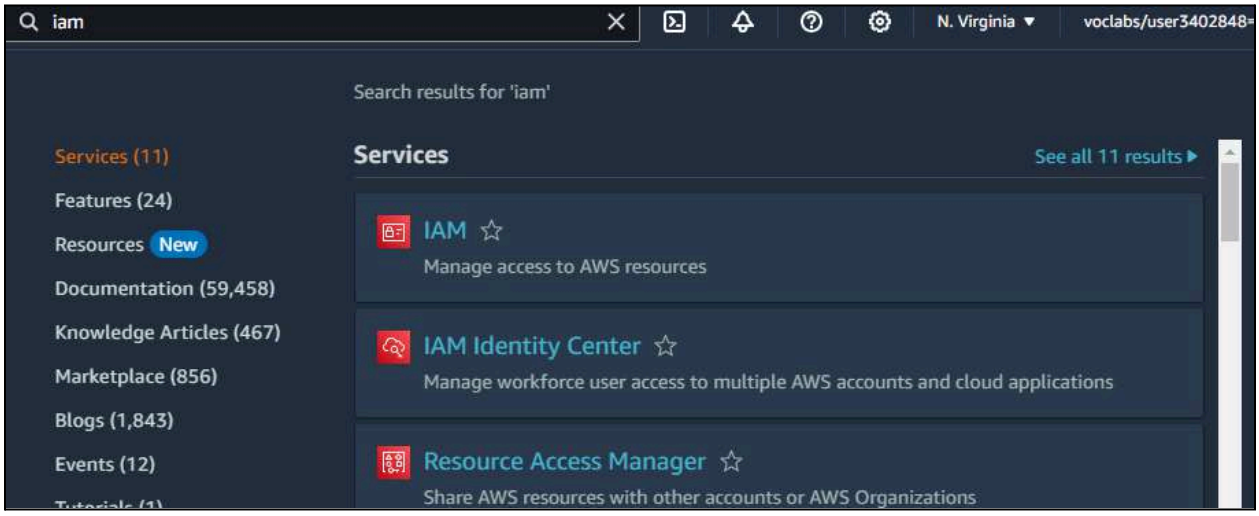

aws
Services
Search
[Alt+S]
N. Virginia
vocabs/user3402848=PATANKAR_ARYAN_ANIL
Developer Tools
AWS Cloud9
A cloud IDE for writing, running, and debugging code
AWS Cloud9 allows you to write, run, and debug your code with just a browser. With AWS Cloud9, you have immediate access to a rich code editor, integrated debugger, and built-in terminal with preconfigured AWS CLI. You can get started in minutes and no longer have to spend the time to install local applications or configure your development machine.
New AWS Cloud9 environment
Create environment

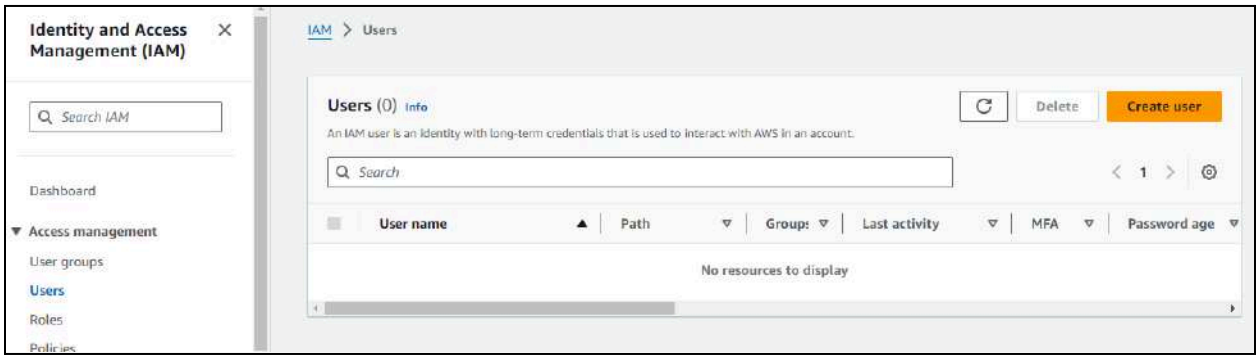

Details
Name
Test123
Limit of 60 characters, alphanumeric, and unique per user.
Description - optional
Limit 200 characters.
Environment type Info
Determines what the Cloud9 IDE will run on.
New EC2 instance
Cloud9 creates an EC2 instance in your account. The configuration of your EC2 instance cannot be changed by Cloud9 after creation.
Existing compute
You have an existing instance or server that you'd like to use.


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.
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


[Alt+S]
N. Virginia
voclabs/user3402848=PATANKAR_ARYAN_ANIL @ 5543
Successfully created Test123. To get the most out of your environment, see Best practices for using AWS Cloud9
For capabilities similar to AWS Cloud9, explore AWS Toolkits in your own IDE and AWS CloudShell in the AWS Management Console. Learn more
AWS Cloud9 > Environments
Environments (1)
Delete
View details
Open in Cloud9
Create environment
My environments
Name
Cloud9 IDE
Environment type
Connection
Permission
Owner ARN
Test123
Open
EC2 instance
Secure Shell (SSH)
Owner
arn:aws:sts::554378108602:assumed-role/voclabs/user3402848=PATANKAR_ARYAN_ANIL


iam
N. Virginia
voclabs/user3402848=
Search results for 'iam'
Services (11)
Features (24)
Resources New
Documentation (59,458)
Knowledge Articles (467)
Marketplace (856)
Blogs (1,843)
Events (12)
Services
See all 11 results
IAM
Manage access to AWS resources
IAM Identity Center
Manage workforce user access to multiple AWS accounts and cloud applications
Resource Access Manager
Share AWS resources with other accounts or AWS Organizations


Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
IAM > Users
Users (0) Info
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.
Delete
Create user
Search
User name
Path
Groups
Last activity
MFA
Password age
No resources to display

User name

aryan

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

☑ Provide user access to the AWS Management Console - *optional*

If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

Console password

○ Autogenerated password

You can view the password after you create the user.

◉ Custom password

Enter a custom password for the user.

••••••••••

- Must be at least 8 characters long
- Must include at least three of the following mix of character types: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-
  @ # $ % ^ & * ( ) _ + - (hyphen) = [ ] { } | '

☐ Show password

☑ Users must create a new password at next sign-in - Recommended

Users automatically get the IAMUserChangePassword policy to allow them to change their own password.

User details

User name	Console password type	Require password reset
aryan	Custom password	Yes

Permissions summary 1

Name	Type	Used as
IAMUserChangePassword	AWS managed	Permissions policy

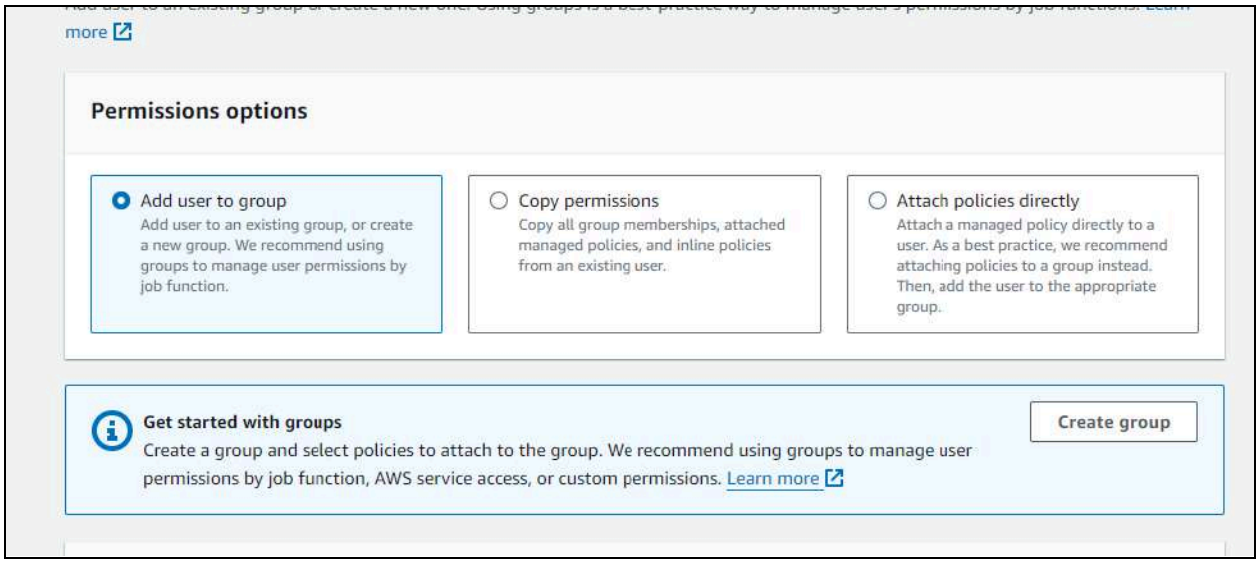
Tags - *optional*

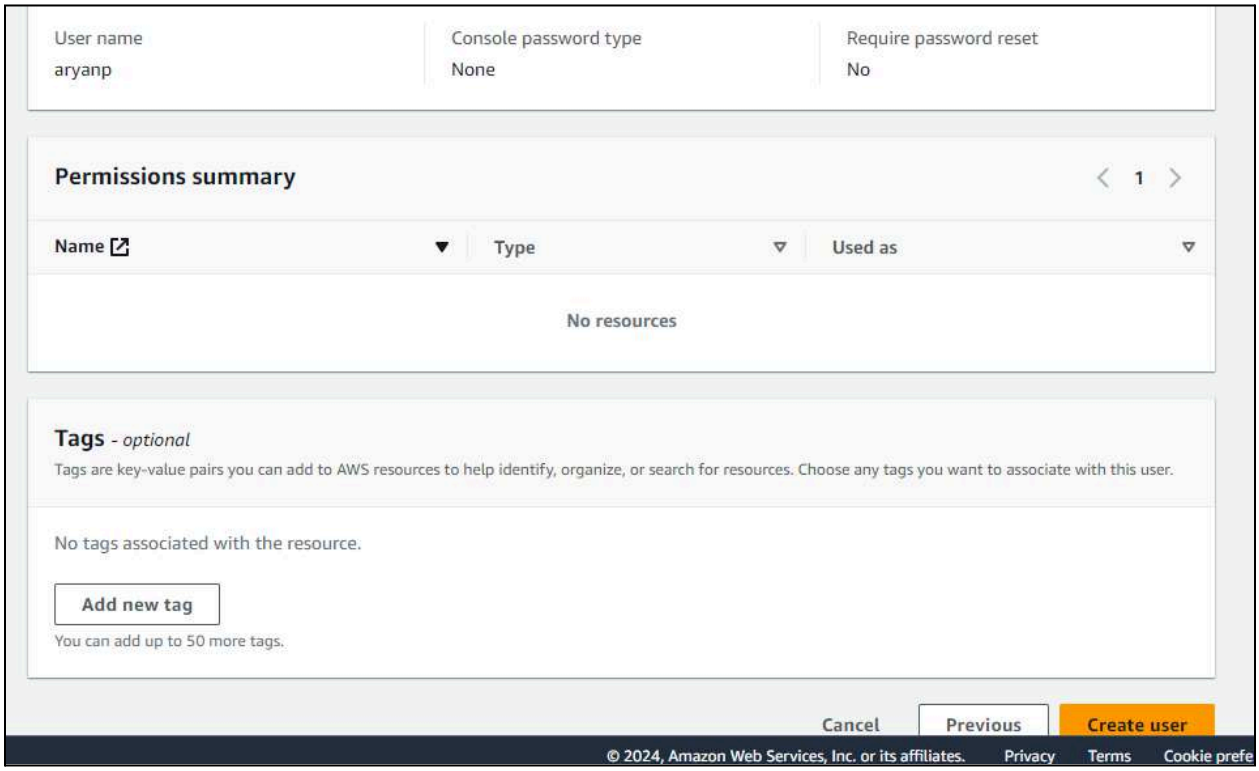
Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

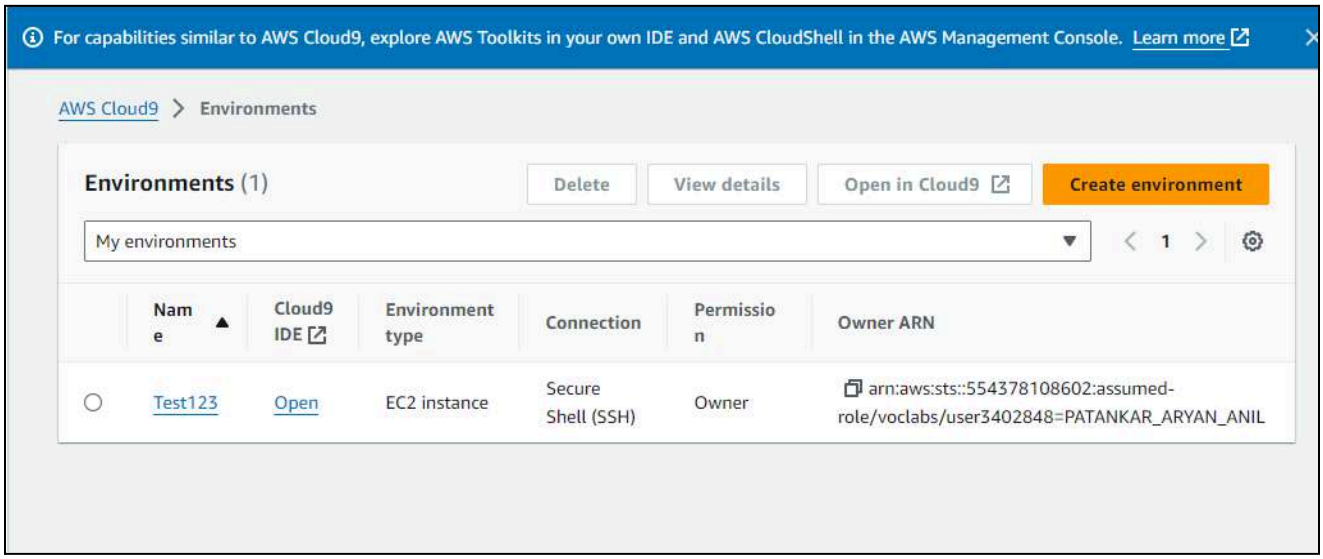
Add new tag

You can add up to 50 more tags.

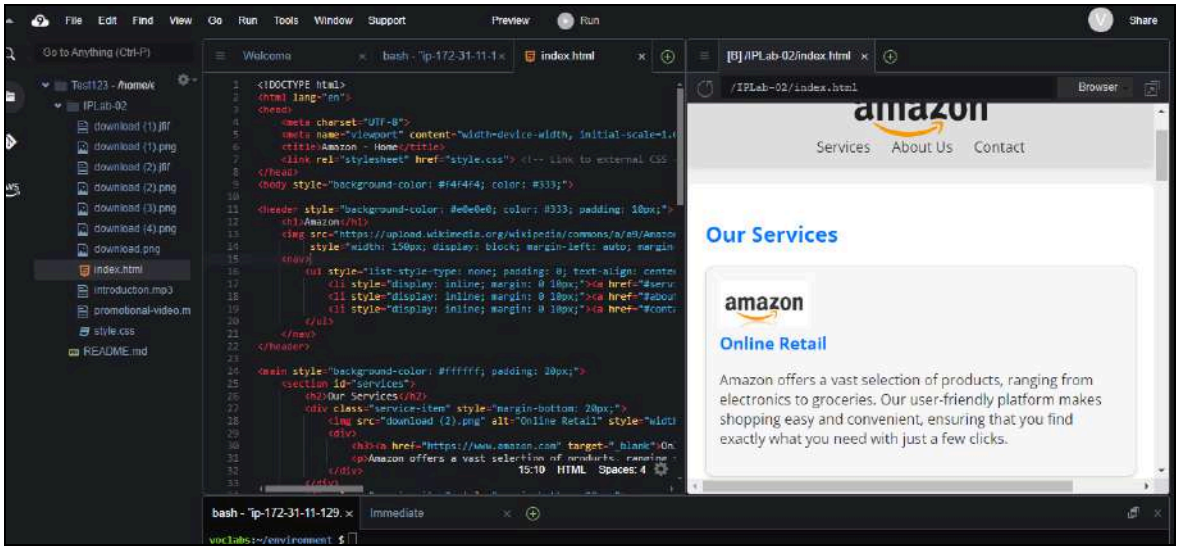

more
Permissions options
Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.
Attach policies directly
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


User name
aryanp
Console password type
None
Require password reset
No
Permissions summary
1
Name
Type
Used as
No resources
Tags - optional
Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.
No tags associated with the resource.
Add new tag
You can add up to 50 more tags.
Cancel
Previous
Create user
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie prefe

Here the environment has been successfully created



2.We have successfully set up and launched our Cloud9 environment. Over here, we can build and develop programs as per our desire. We are also allowed to collaborate with multiple other users and access shared resources.

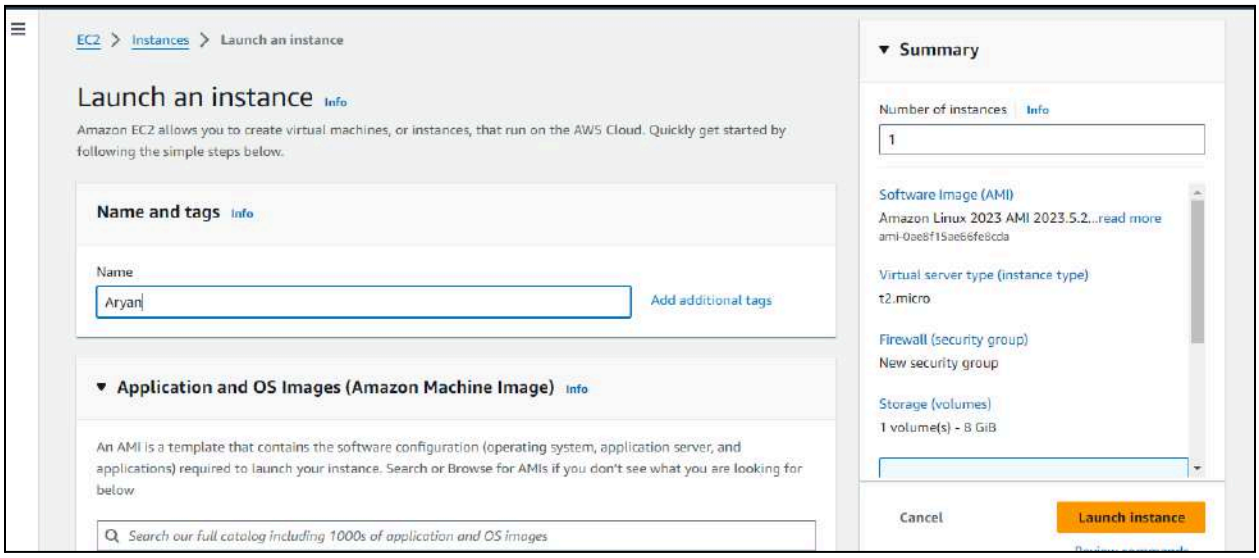


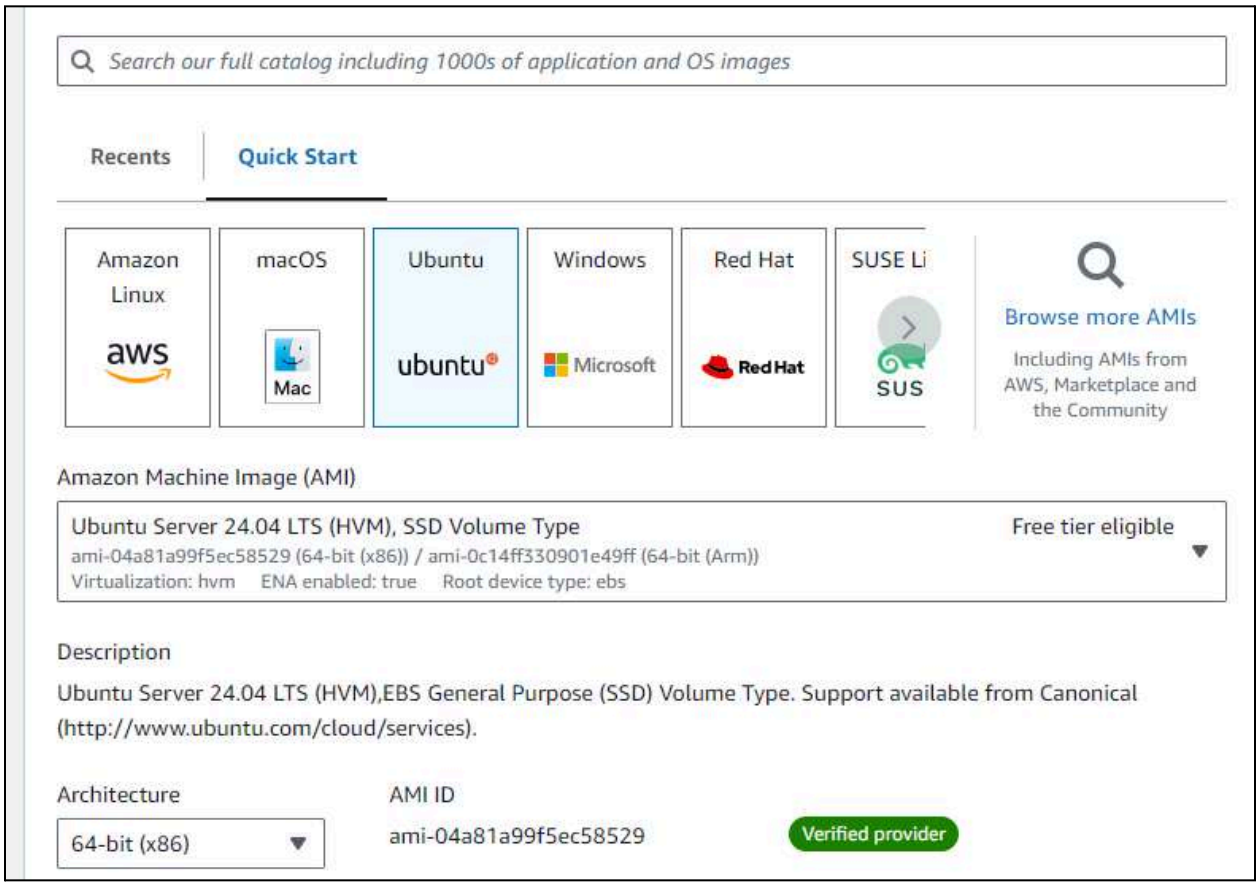
Further, we are supposed to login from another browser using the credentials of the IAM user, to access the shared cloud9 environment with us. These steps could not be completed because Cloud9 services have been disrupted and there is no access to the IAM user from the remote login.

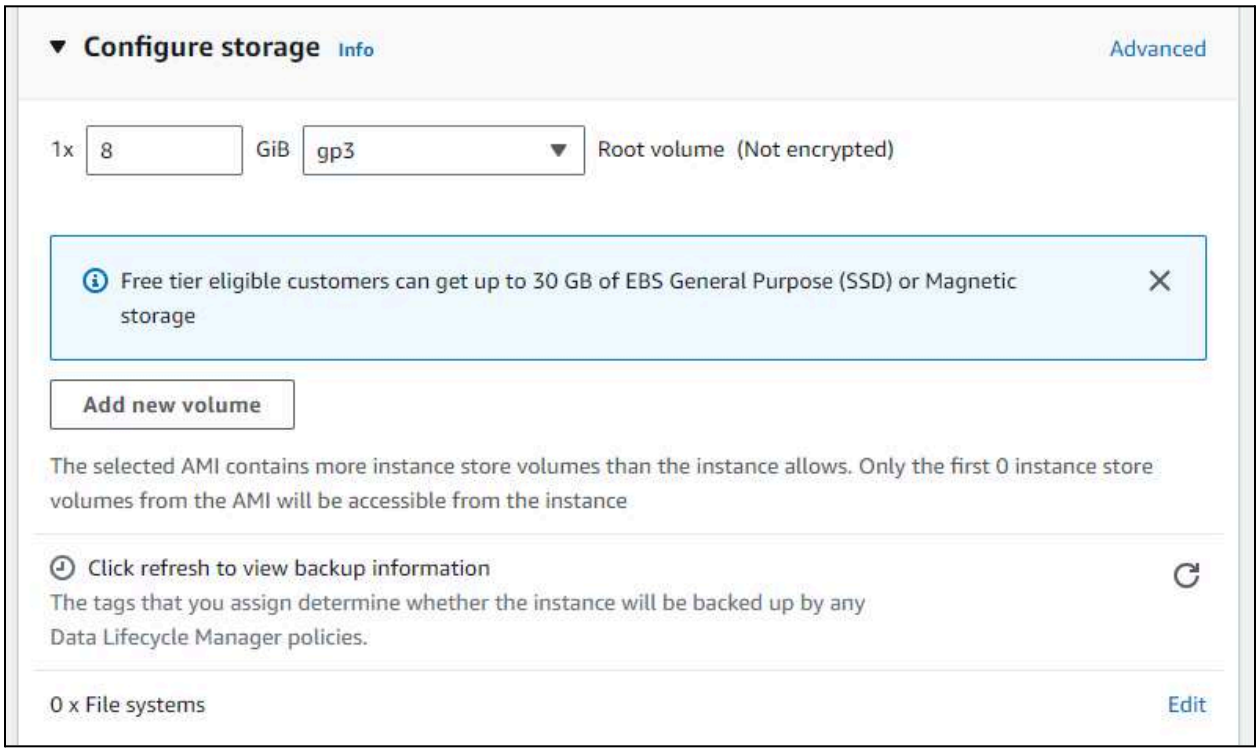
EC2 INSTANCE

Steps:

1.Create a new instance and follow the steps

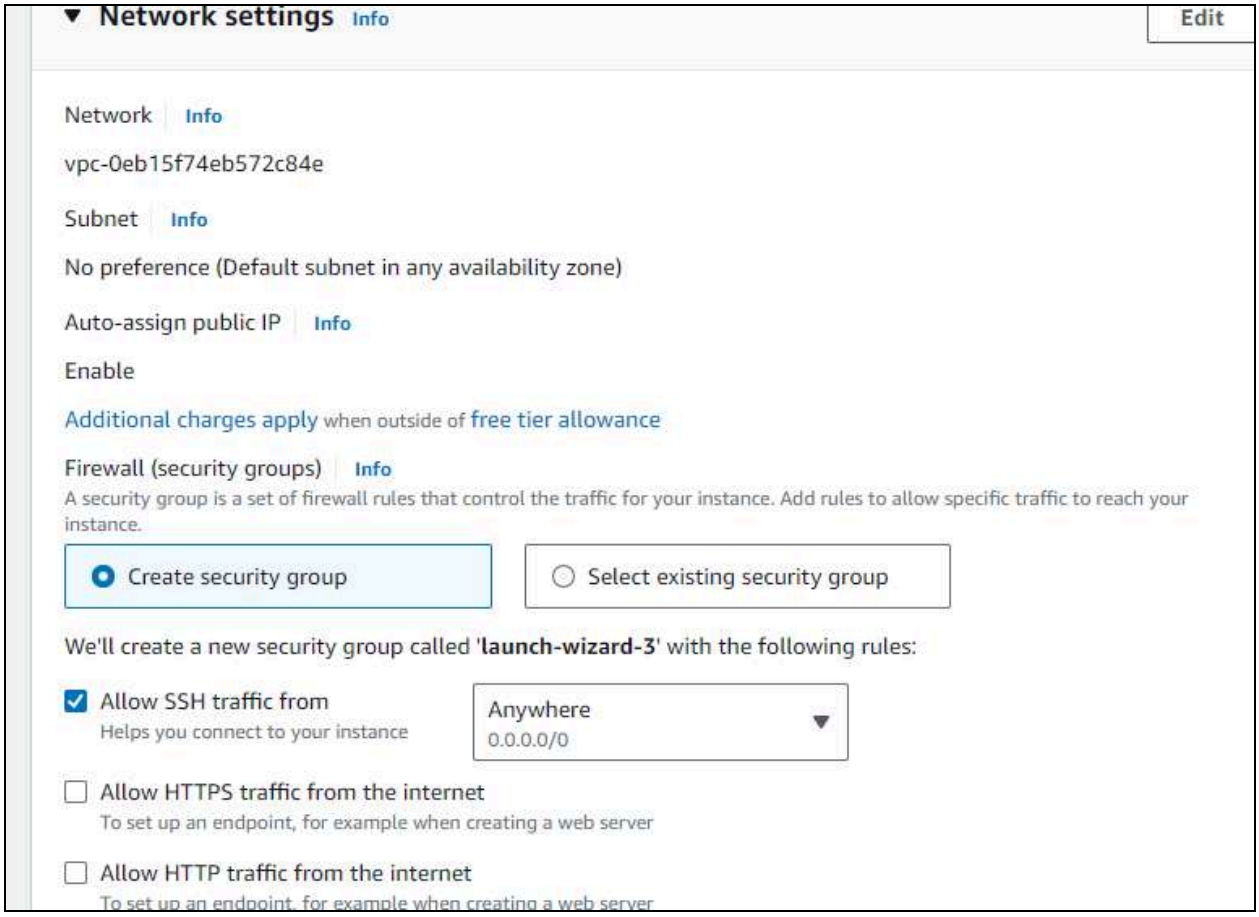


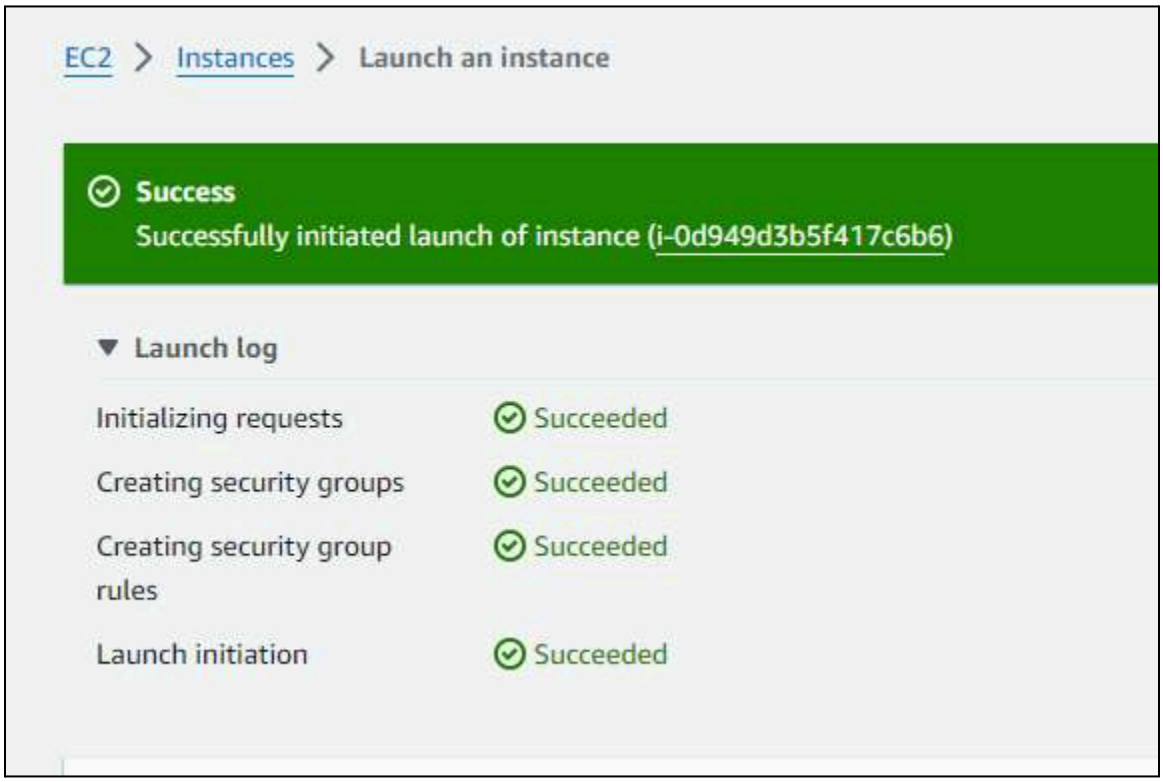


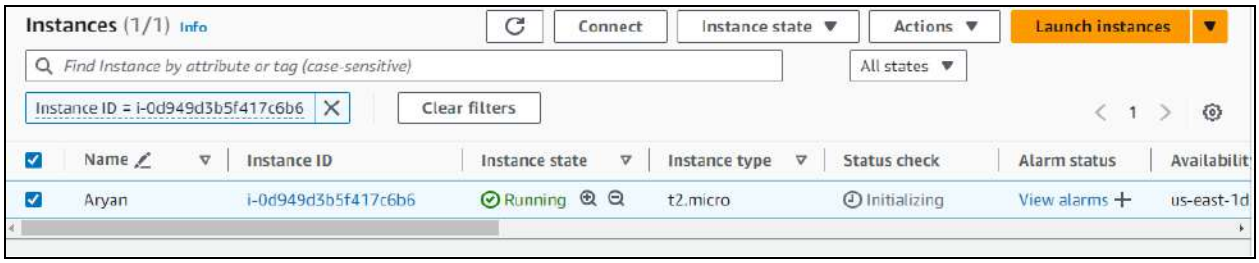

Configure storage Info
Advanced
1x 8 GiB gp3 Root volume (Not encrypted)
Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage
Add new volume
The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance
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

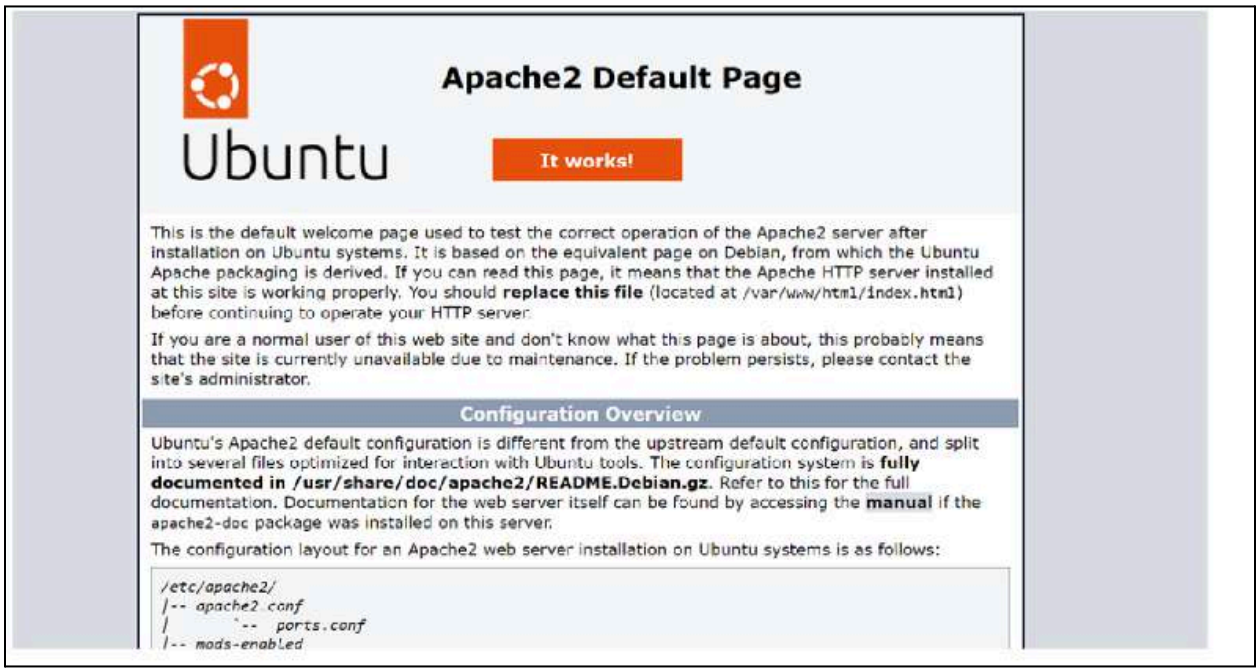

Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.
Key pair name - required
vockey
Create new key pair


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


Network settings Info
Edit
Network Info
vpc-0eb15f74eb572c84e
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:
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


EC2 > Instances > Launch an instance
Success
Successfully initiated launch of instance (i-0d949d3b5f417c6b6)
Launch log
Initializing requests
Succeeded
Creating security groups
Succeeded
Creating security group rules
Succeeded
Launch initiation
Succeeded


Instances (1/1) Info
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Instance ID = i-0d949d3b5f417c6b6
Clear filters
1
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availabilit
Aryan
i-0d949d3b5f417c6b6
Running
t2.micro
Initializing
View alarms
us-east-1d

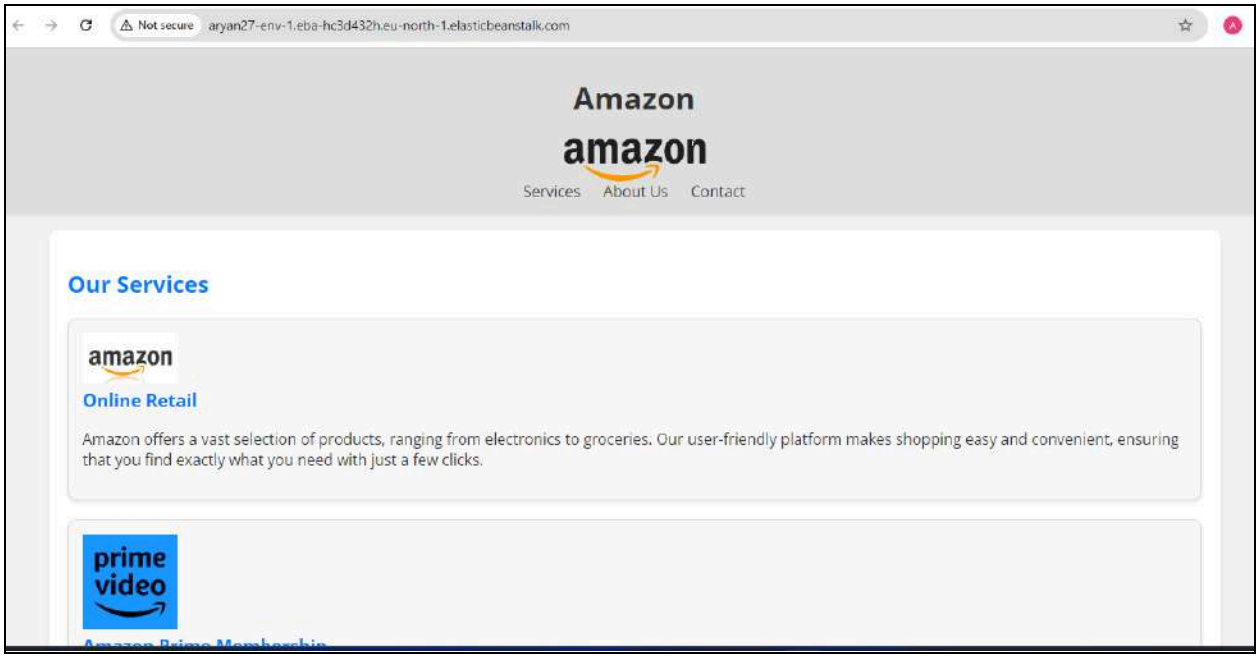

Apache2 Default Page
Ubuntu
It works!
This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should replace this file (located at /var/www/html/index.html) before continuing to operate your HTTP server.
If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.
Configuration Overview
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is fully documented in /usr/share/doc/apache2/README.Debian.gz. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the manual if the apache2-doc package was installed on this server.
The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:
/etc/apache2/
|-- apache2.conf
| `-- ports.conf
|-- mods-enabled

```
[1]+  Stopped                 nano index1.html
root@ip-172-31-36-118:/var/www/html# sudo nano index.html
root@ip-172-31-36-118:/var/www/html# sudo start apache2
sudo: start: command not found
root@ip-172-31-36-118:/var/www/html# sudo systemctl start apache2
root@ip-172-31-36-118:/var/www/html# sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-08-18 09:03:15 UTC; 22min ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 3261 (apache2)
      Tasks: 55 (limit: 1130)
     Memory: 5.3M (peak: 5.5M)
        CPU: 128ms
     CGroup: /system.slice/apache2.service
             ├─3261 /usr/sbin/apache2 -k start
             ├─3264 /usr/sbin/apache2 -k start
             └─3265 /usr/sbin/apache2 -k start

Aug 18 09:03:15 ip-172-31-36-118 systemd[1]: Starting apache2.service - The Apache HTTP Server...
Aug 18 09:03:15 ip-172-31-36-118 systemd[1]: Started apache2.service - The Apache HTTP Server.
root@ip-172-31-36-118:/var/www/html#
```

```
See "man sudo_root" for details.

ubuntu@ip-172-31-36-118:~$ :/home/ubuntu# apt install apache2
-bash: :/home/ubuntu#: No such file or directory
ubuntu@ip-172-31-36-118:~$ cls
Command 'cls' not found, but there are 20 similar ones.
ubuntu@ip-172-31-36-118:~$ sudo su
root@ip-172-31-36-118:/home/ubuntu# apt 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-s
qlite3 libaprutil1-ldap libaprutil1t64 liblua5.4-0 ssl-cert
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-bro
wser
```

i-0d949d3b5f417c6b6 (Aryan)

PublicIPs: 52.207.231.96 PrivateIPs: 172.31.36.118

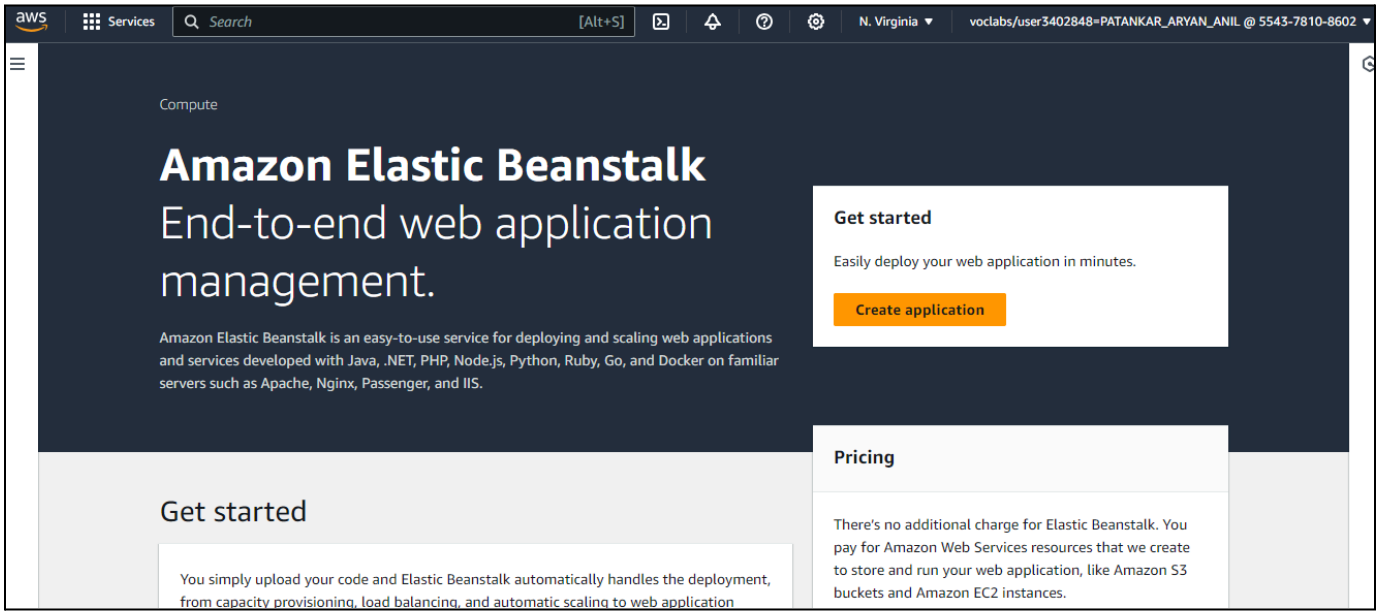

Not secure
aryan27-env-1.eba-hc3d432h.eu-north-1.elasticbeanstalk.com
Amazon
amazon
Services
About Us
Contact
Our Services
amazon
Online Retail
Amazon offers a vast selection of products, ranging from electronics to groceries. Our user-friendly platform makes shopping easy and convenient, ensuring that you find exactly what you need with just a few clicks.
prime video

ADVANCE DEVOPS EXPERIMENT NO.2

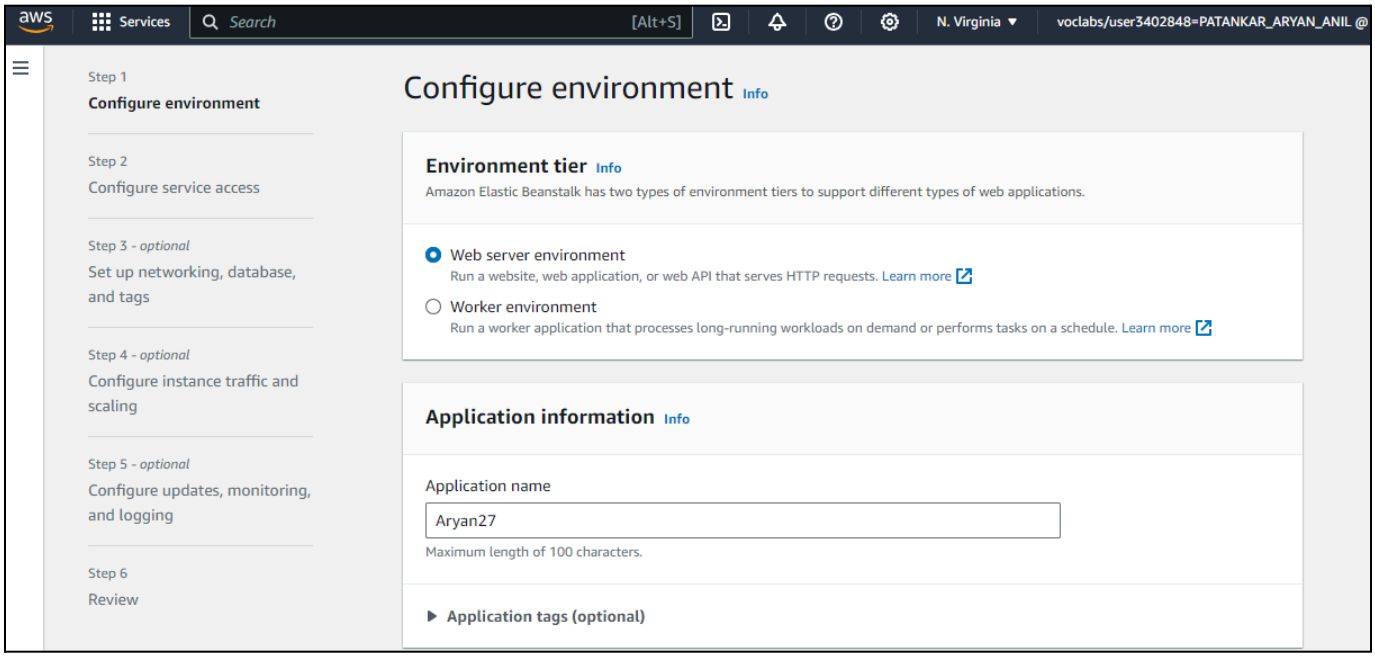
Name: Aryan Anil Patankar
Class:D15A
Roll No:34

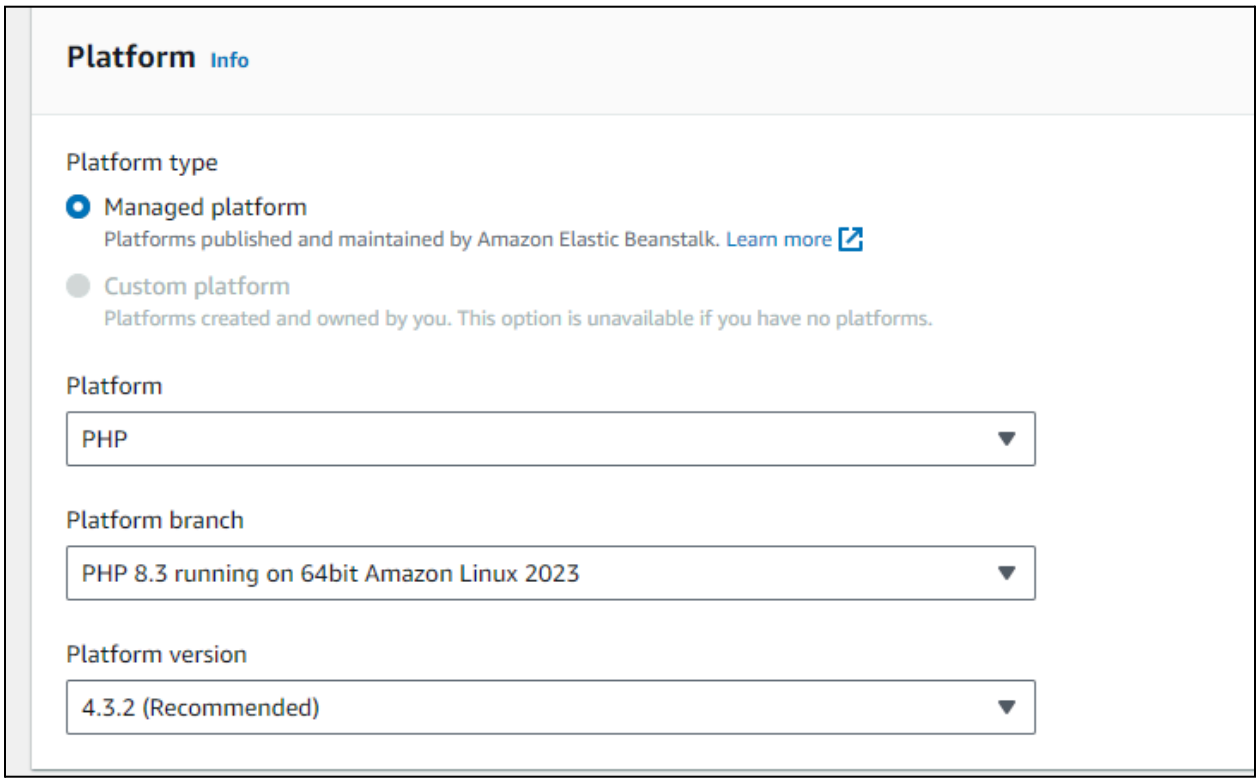
Aim: To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

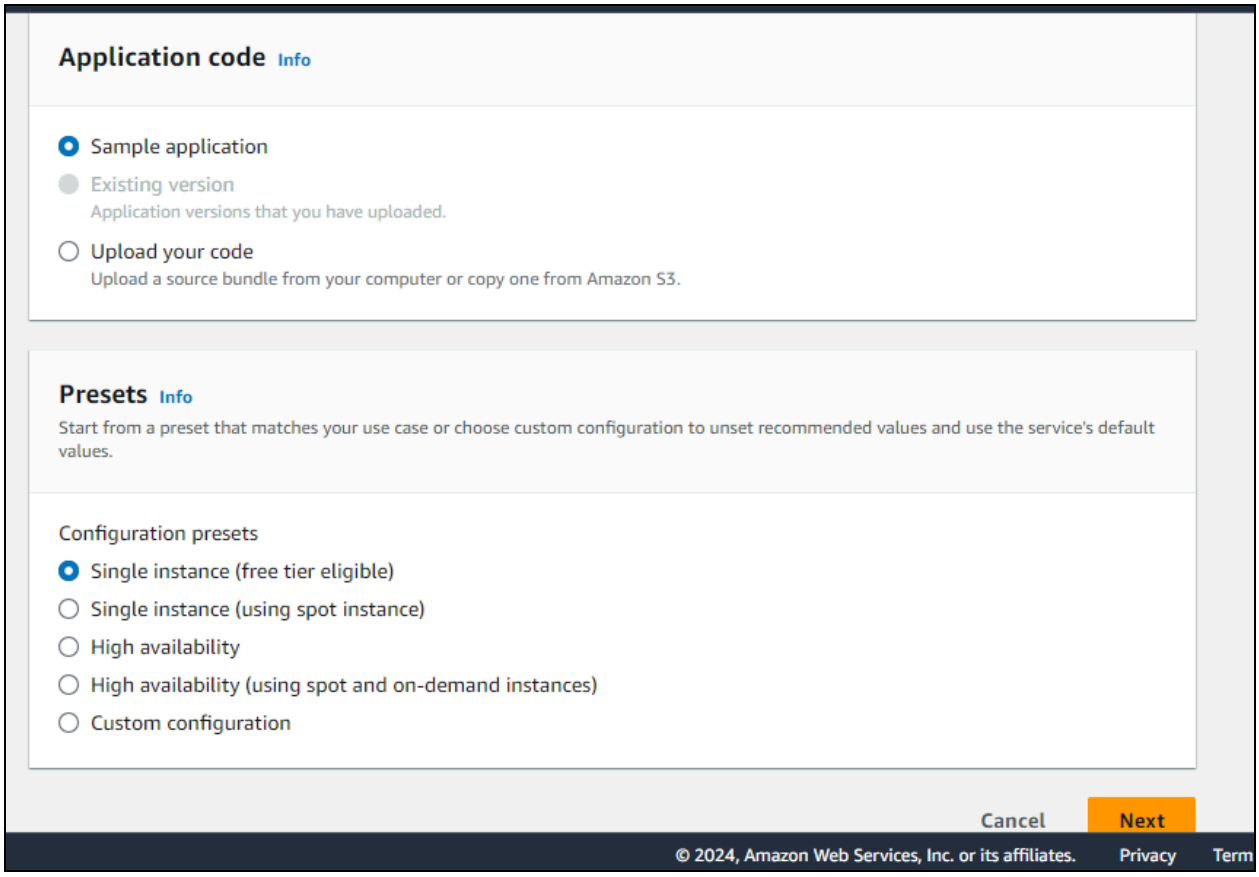
Step 1: Search for **Elastic Beanstalk** in the search bar next to services section and you would see the following page.



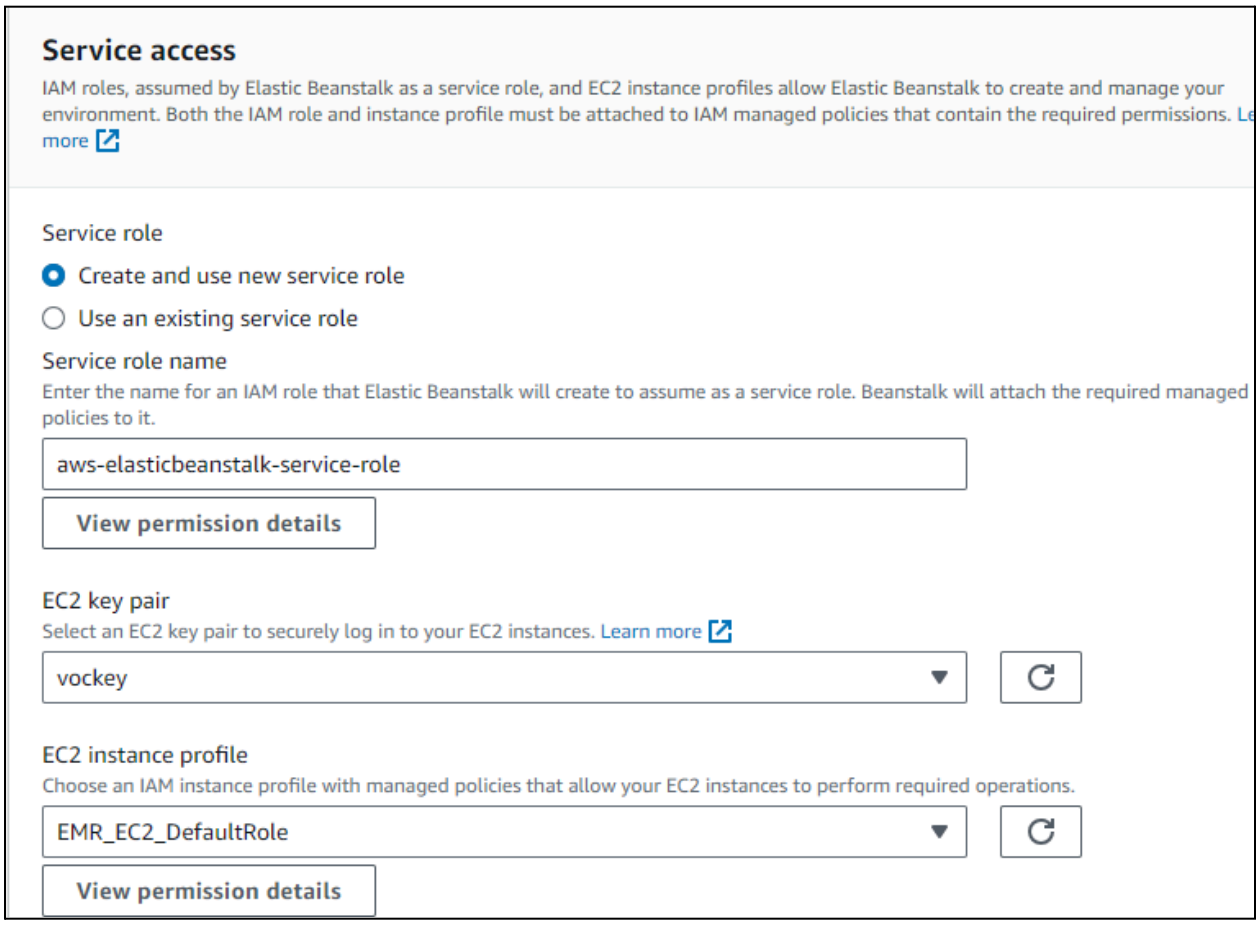
Step 2: Create a new application and proceed with the following settings






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
Cancel
Next
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Term

Step 3: Create a new service role as given below,if an existing service role with the same name does not exist.Proceed with the steps given below.



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

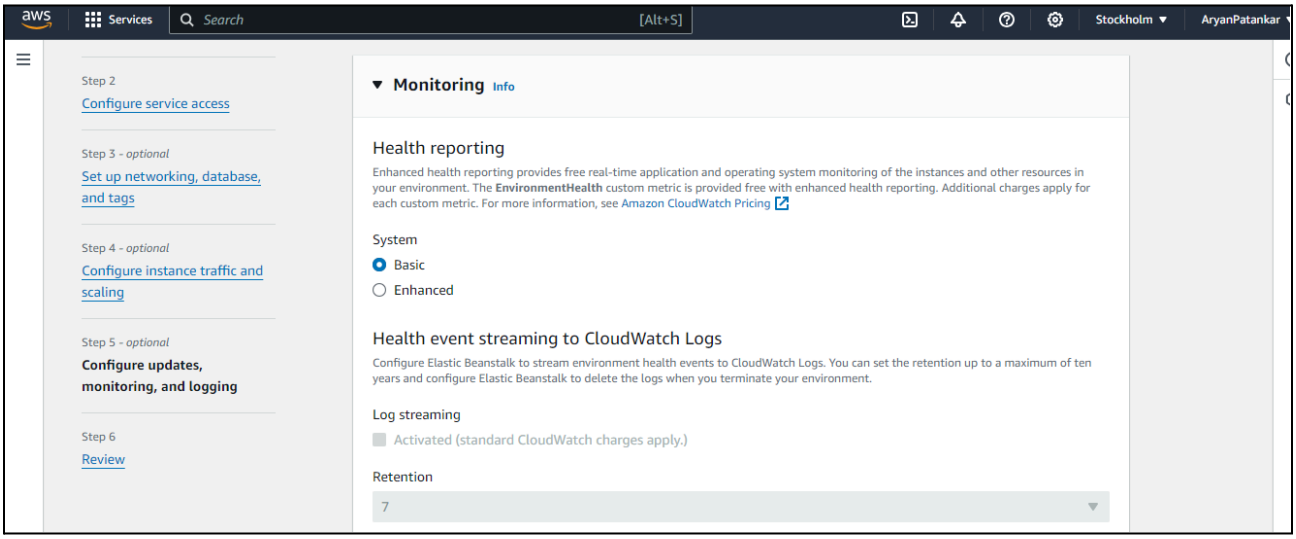
Input/output operations per second for a provisioned IOPS (SSD) volume.

100 IOPS

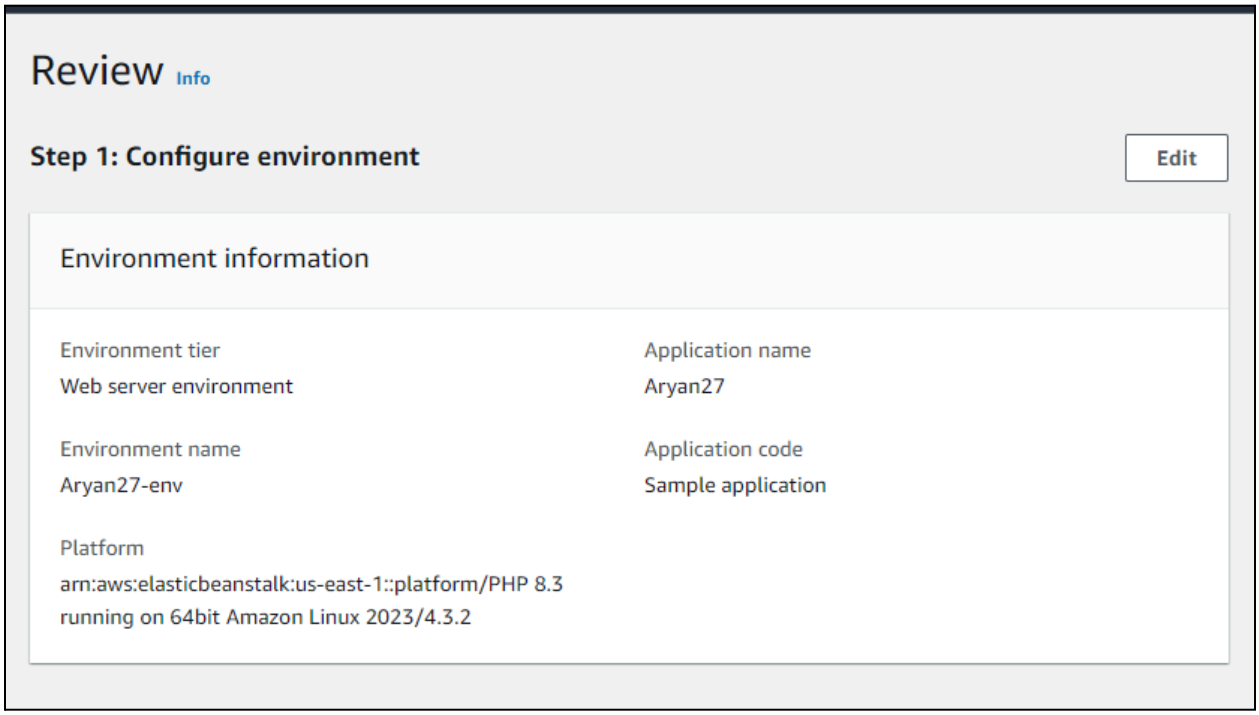
Throughput

The desired throughput to provision for the Amazon EBS root volume attached to your environment's EC2 instance

125 MiB/s



Step 4: Review each step along with the selected options and verify that the correct options have been chosen.



Step 2: Configure service access

Edit

Service access Info

Configure the service role and EC2 instance profile that Elastic Beanstalk uses to manage your environment. Choose an EC2 key pair to securely log in to your EC2 instances.

Service role	EC2 instance profile
arn:aws:iam::405894863107:role/service-role/aws-elasticbeanstalk-service-role	aws-elasticbeanstalk-ec2-role

Step 3: Set up networking, database, and tags

Edit

Networking, database, and tags Info

Configure VPC settings, and subnets for your environment's EC2 instances and load balancer. Set up an Amazon RDS database that's integrated with your environment.

Network

VPC	Public IP address	Instance subnets
vpc-0bf7d7d872a737f13	false	subnet-035fe38d8d742329e,subnet-0a7c9c6dedec1325d

Step 5: Configure updates, monitoring, and logging

Edit

Updates, monitoring, and logging Info

Define when and how Elastic Beanstalk deploys changes to your environment. Manage your application's monitoring and logging settings, instances, and other environment resources.

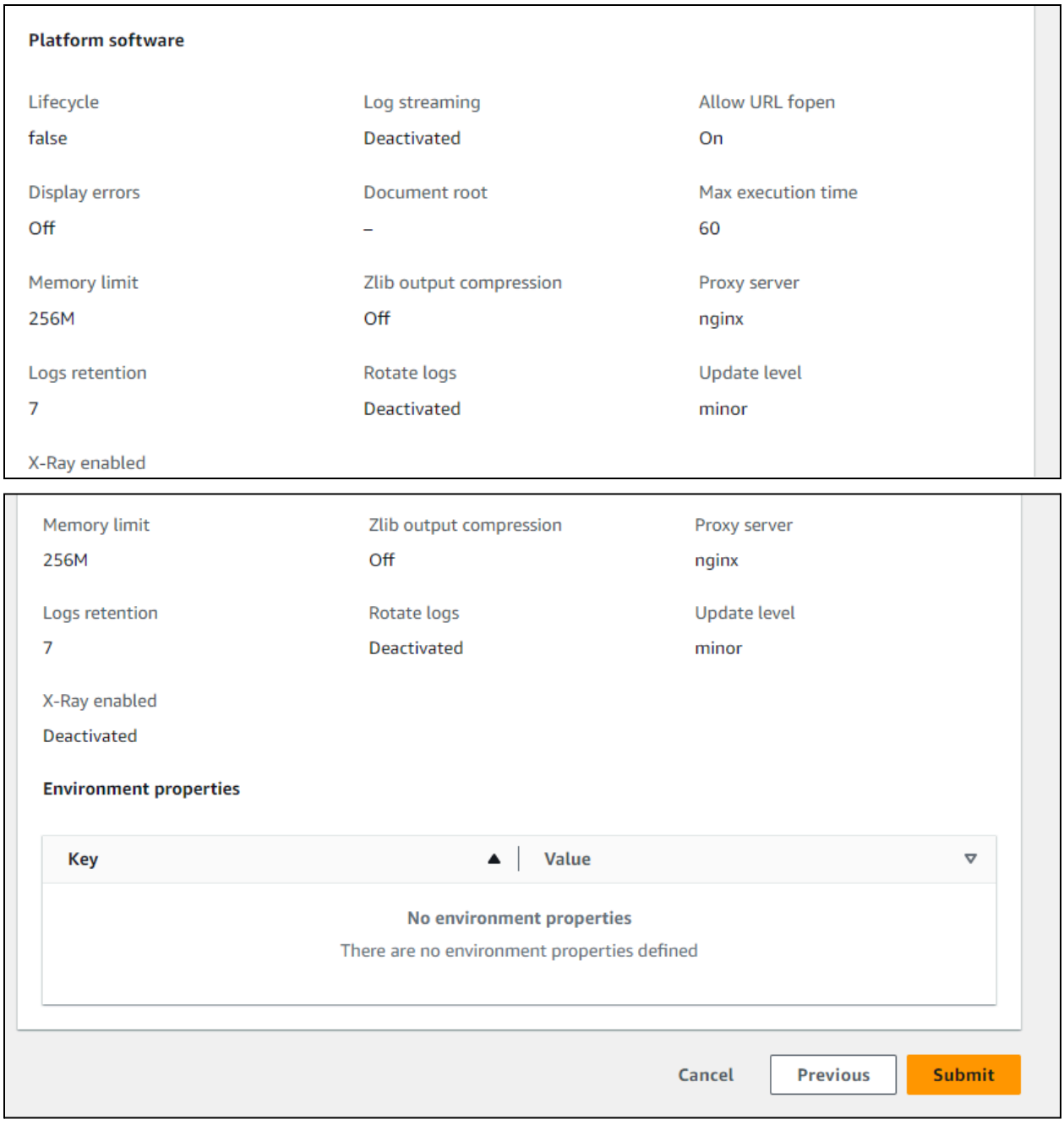
Monitoring

System	Cloudwatch custom metrics - instance	Cloudwatch custom metrics - environment
enhanced	—	—

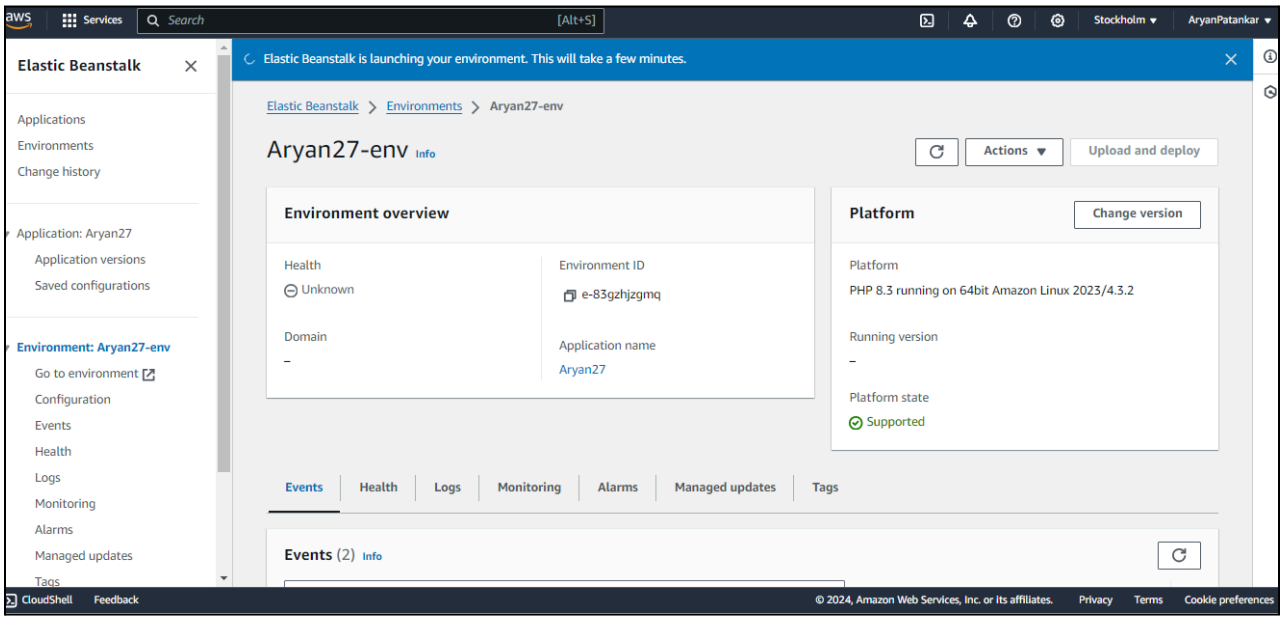
Log streaming	Retention	Lifecycle
Deactivated	7	false

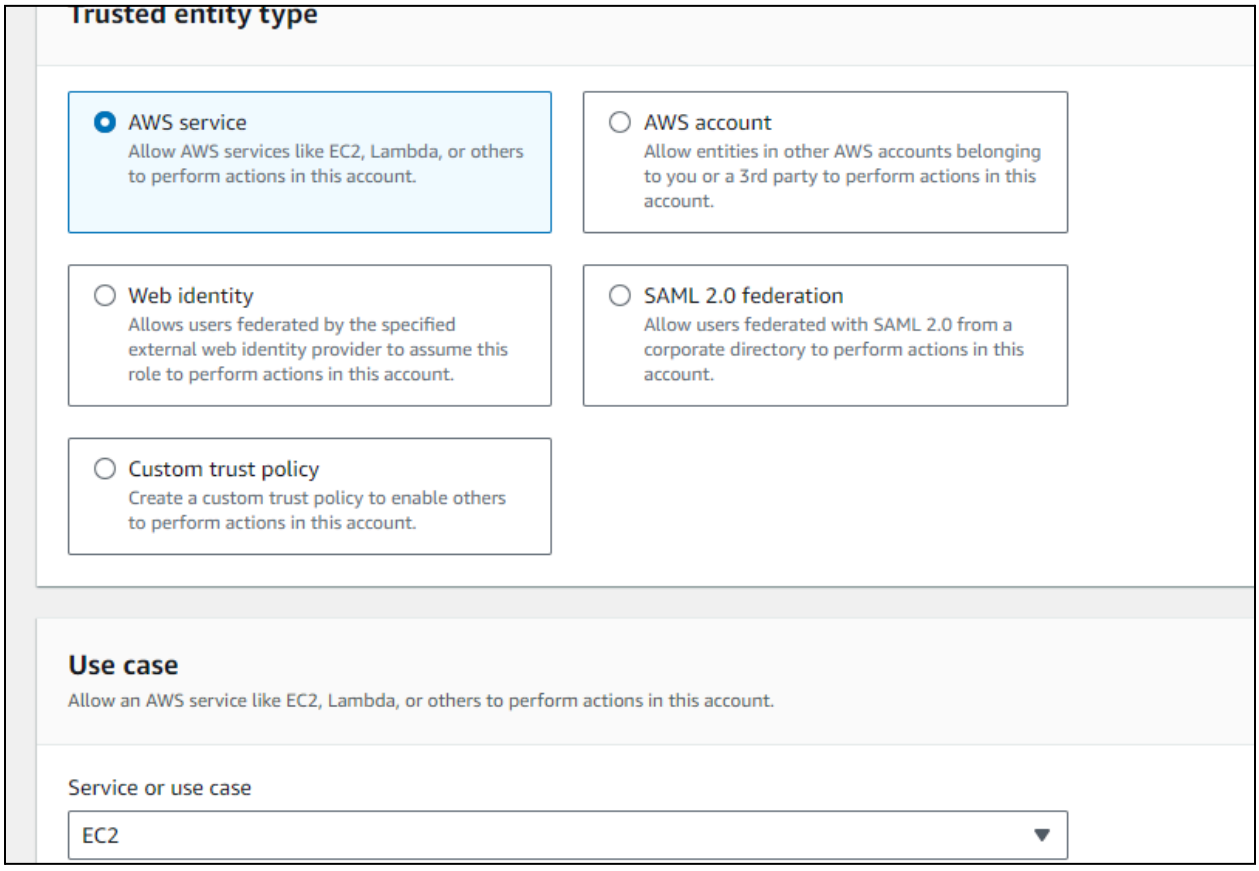
Updates

Managed updates	Deployment batch size	Deployment batch size type
Activated	100	Percentage

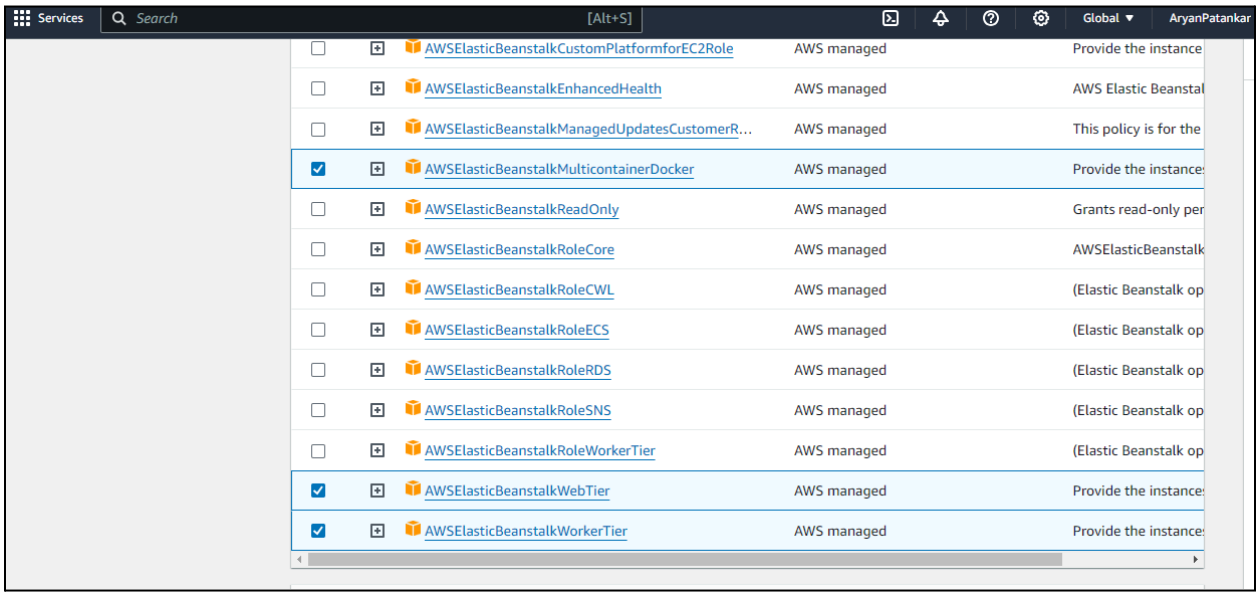

Platform software
Lifecycle
false
Log streaming
Deactivated
Allow URL fopen
On
Display errors
Off
Document root
–
Max execution time
60
Memory limit
256M
Zlib output compression
Off
Proxy server
nginx
Logs retention
7
Rotate logs
Deactivated
Update level
minor
X-Ray enabled
Memory limit
256M
Zlib output compression
Off
Proxy server
nginx
Logs retention
7
Rotate logs
Deactivated
Update level
minor
X-Ray enabled
Deactivated
Environment properties
Key
Value
No environment properties
There are no environment properties defined
Cancel
Previous
Submit

Step 5: After clicking on the submit button, you would notice that the Elastic Beanstalk environment is being created and it may take some time for the environment to load completely.

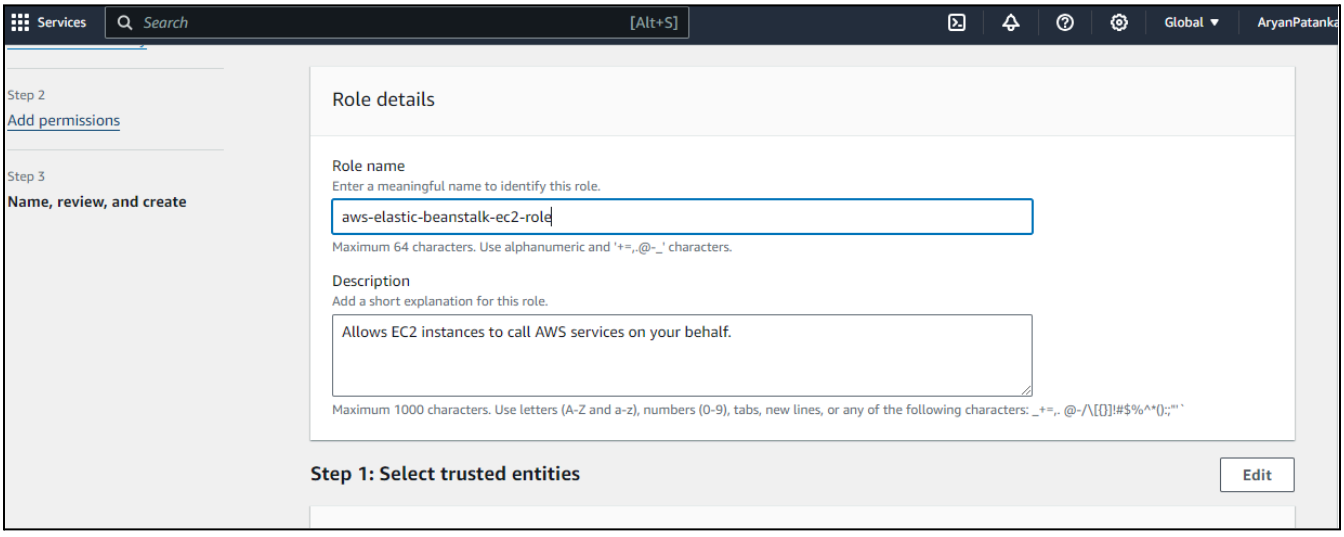


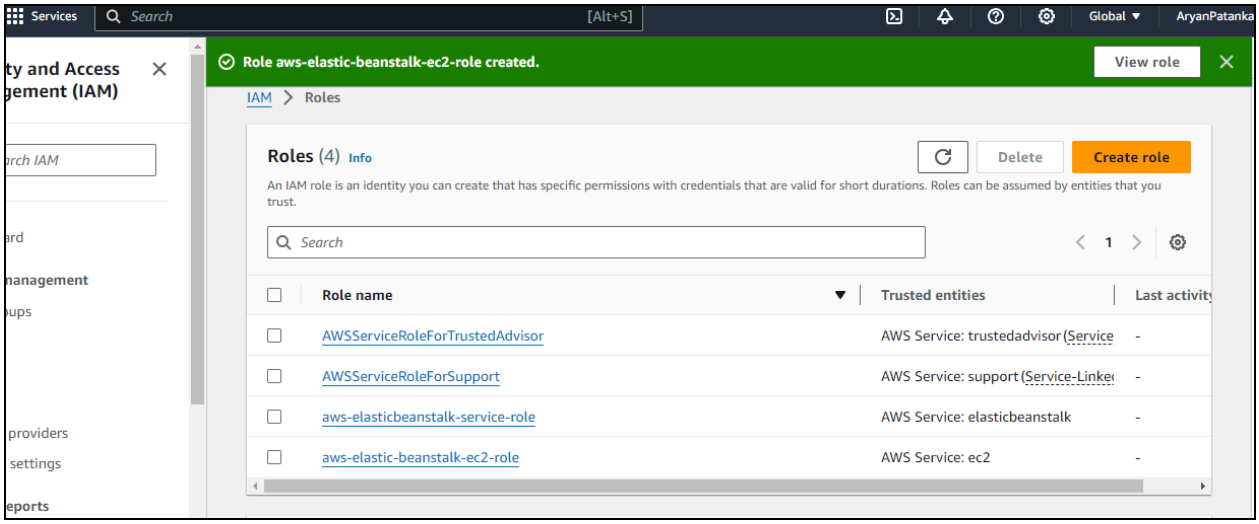


Step 6: Meanwhile,if a role is already not defined, then you need to create a new role for the elastic beanstalk and ensure there is a blue checkmark on the following three permissions given below.

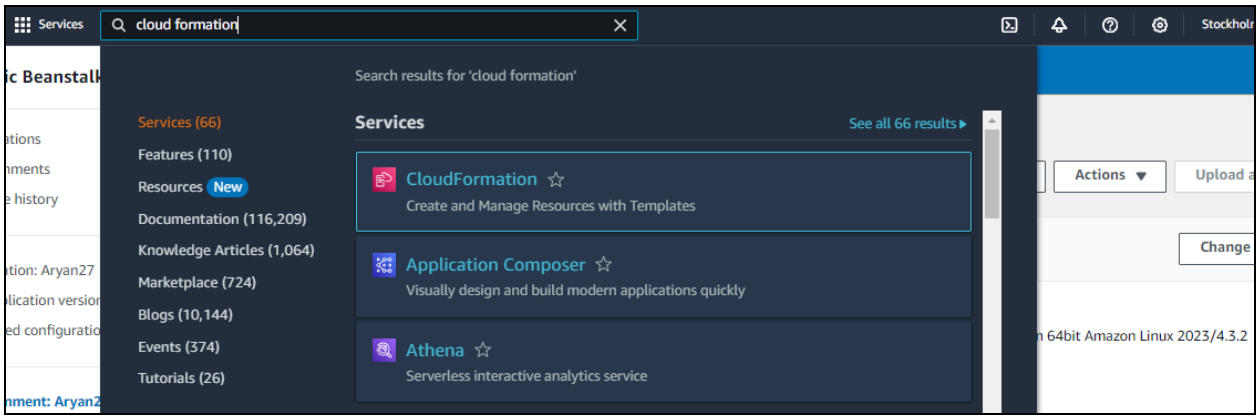


Step 7: Enter a role name and proceed.You would notice the role being successfully created after some time.

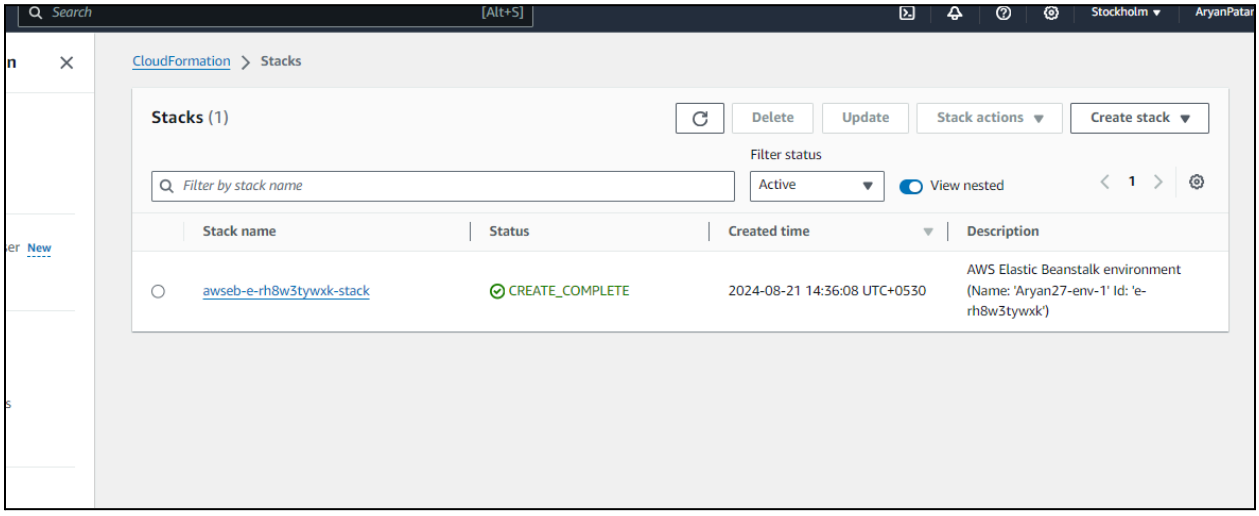


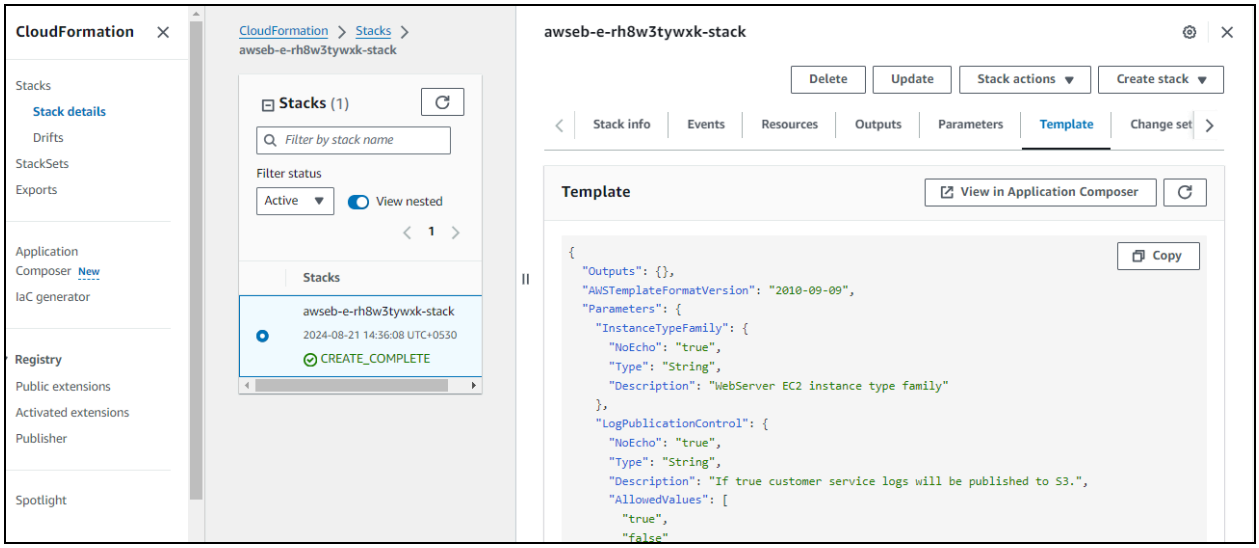


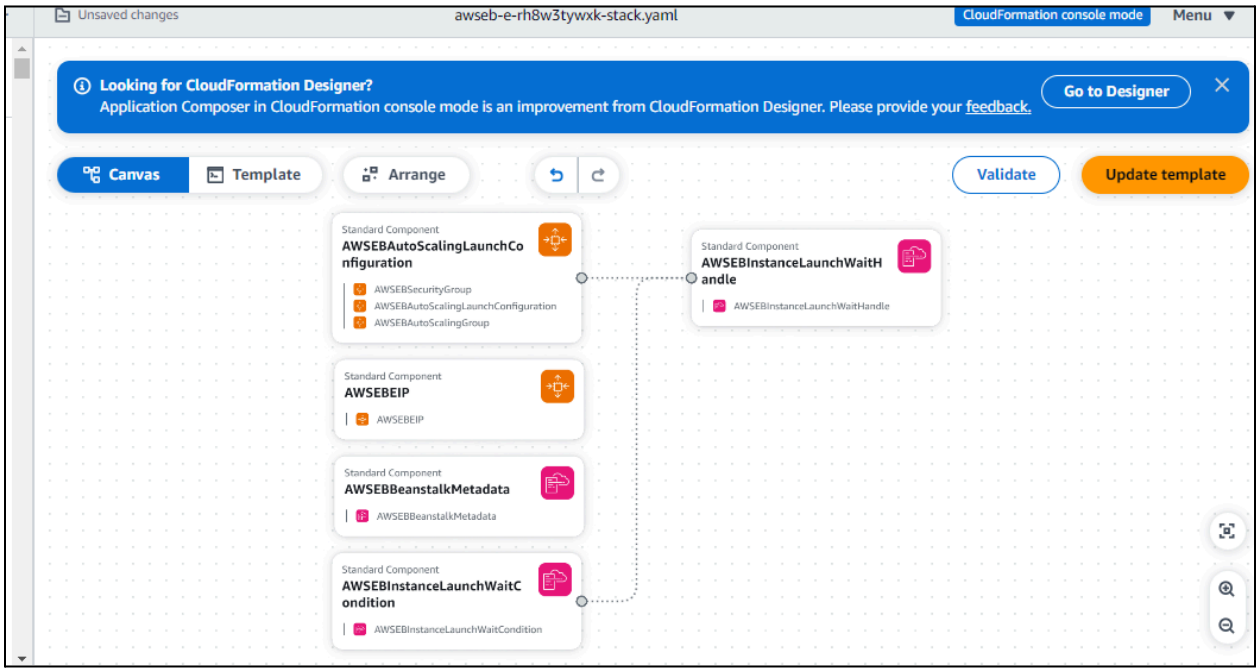
Step 8(optional):Search for CloudFormation as it helps you to manage AWS resources in a text file or a template.



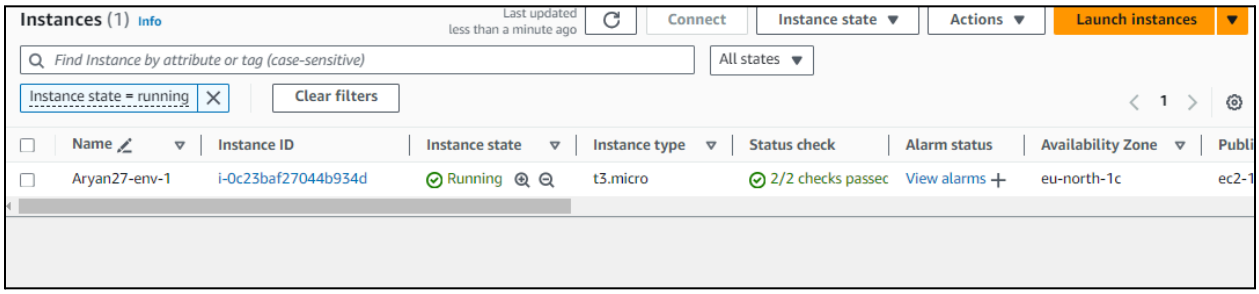
Here, the stacks option given below is a collection of AWS resources.



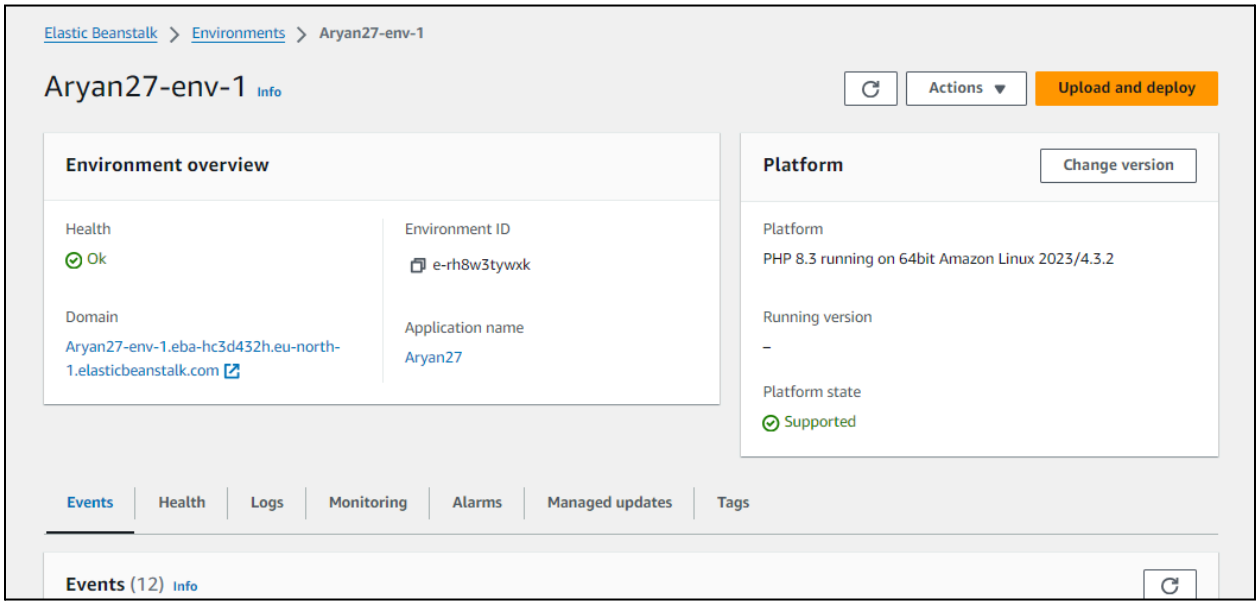


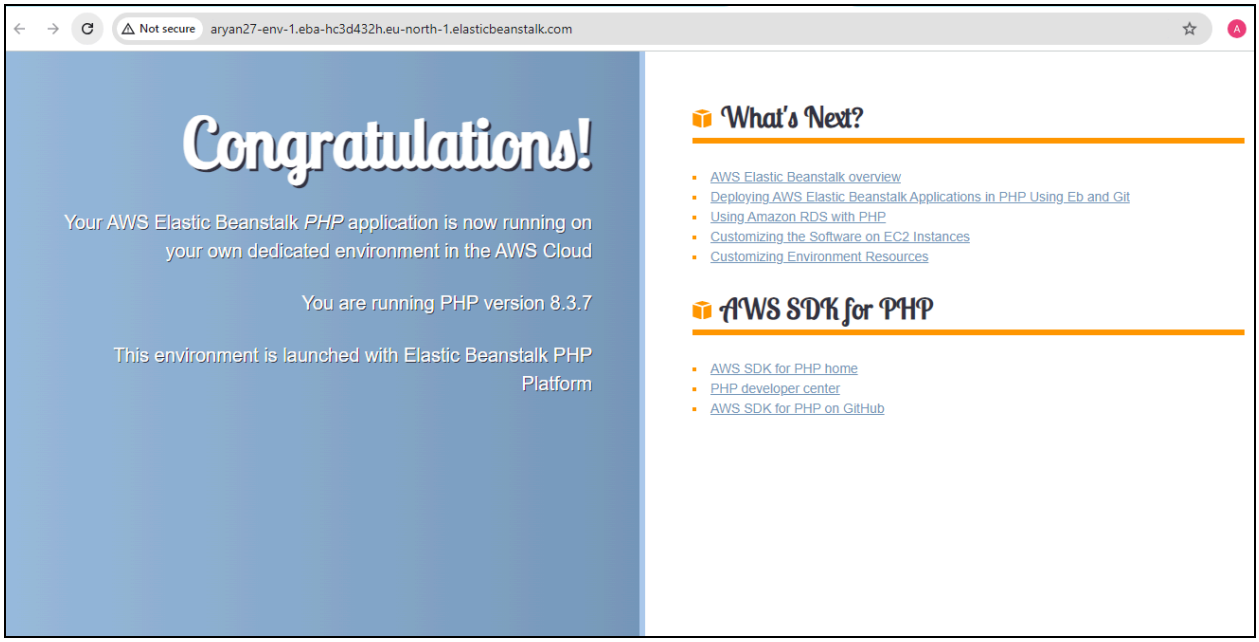


Step 9: Now,we search for EC2 in the services section and we notice that an instance of the Elastic Beanstalk app has already been created and it is running.

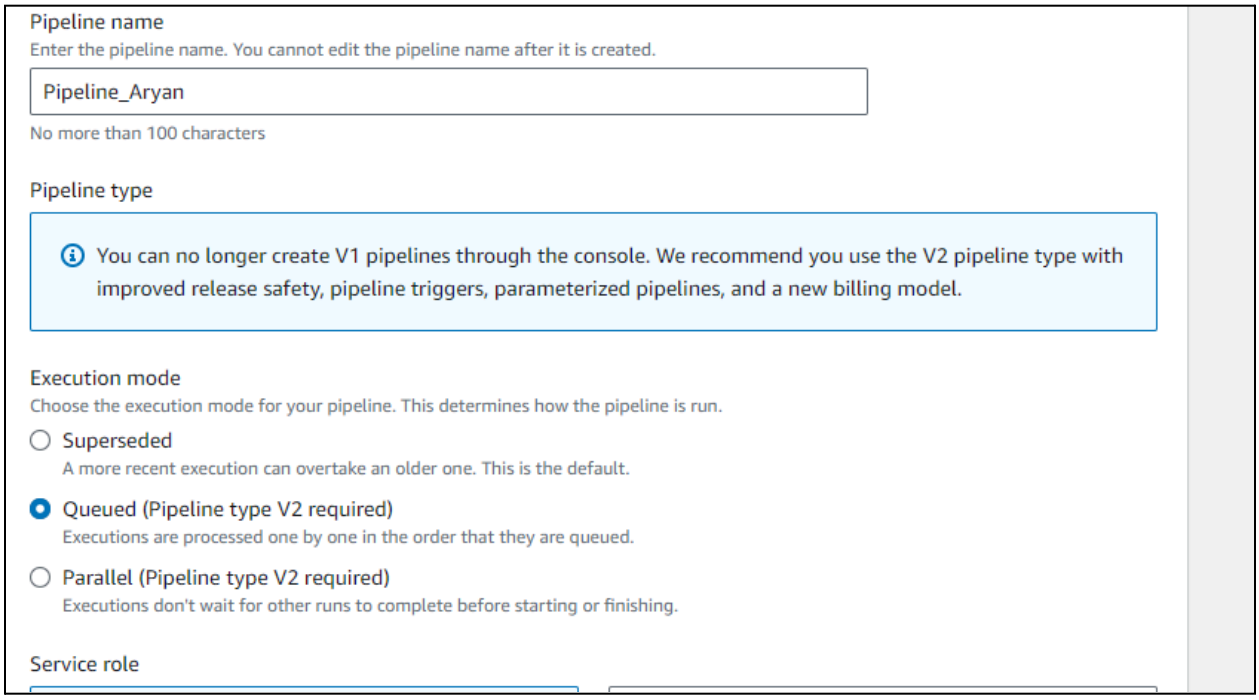


Step 10:Click on the domain link given below,after which we are redirected to a Congratulations page implying that our sample PHP application has been successfully hosted.

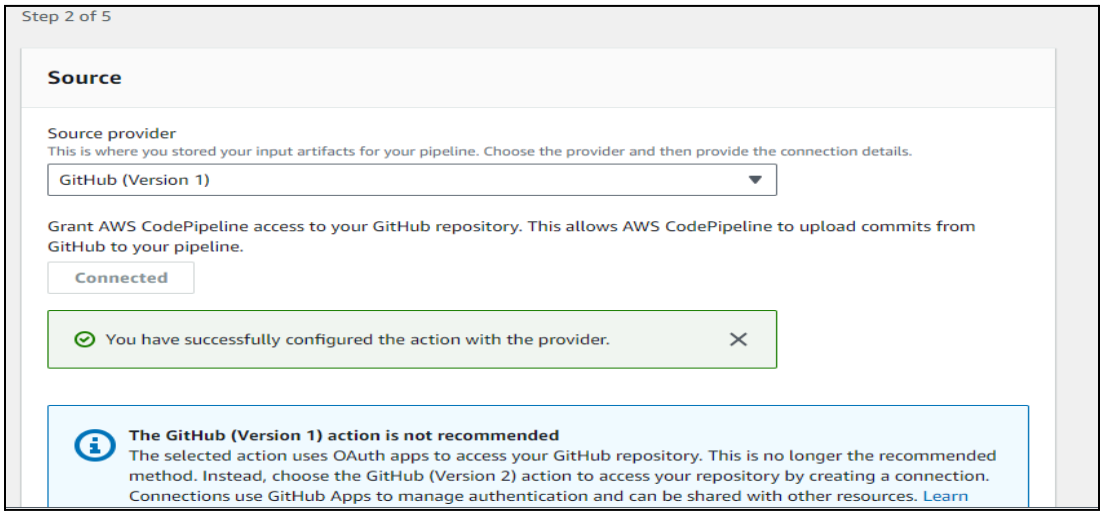


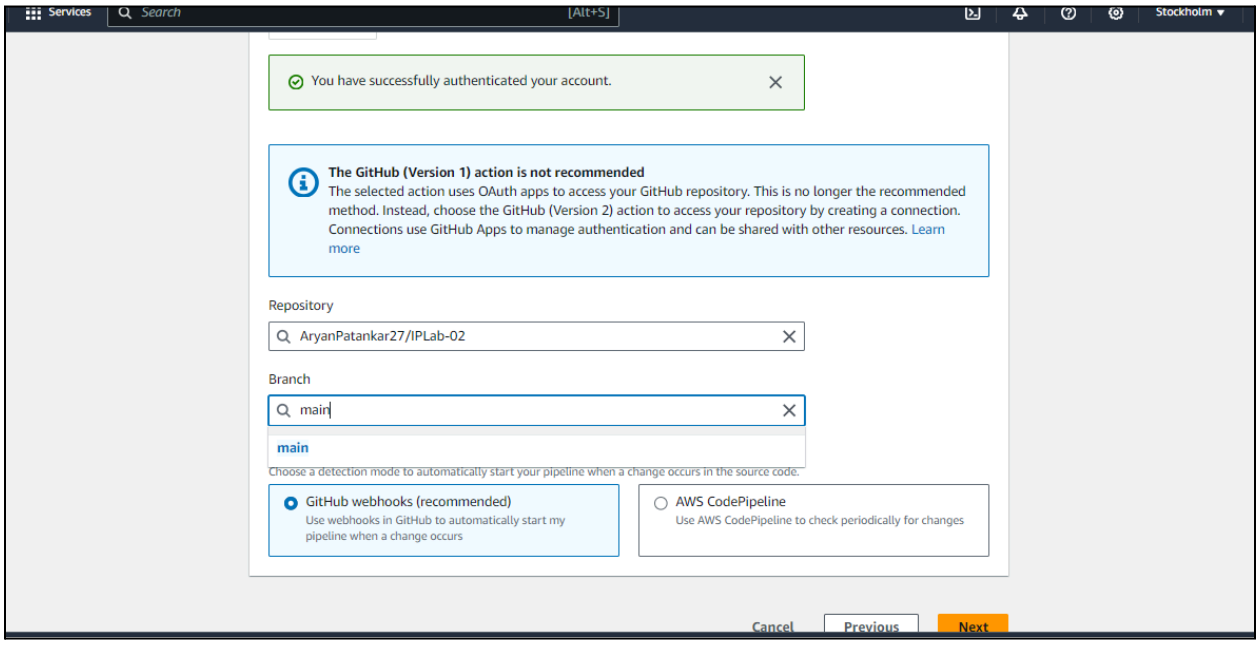


Step 11:Now,we will be deploying our website using CodePipeline,so follow all the steps given below and proceed.



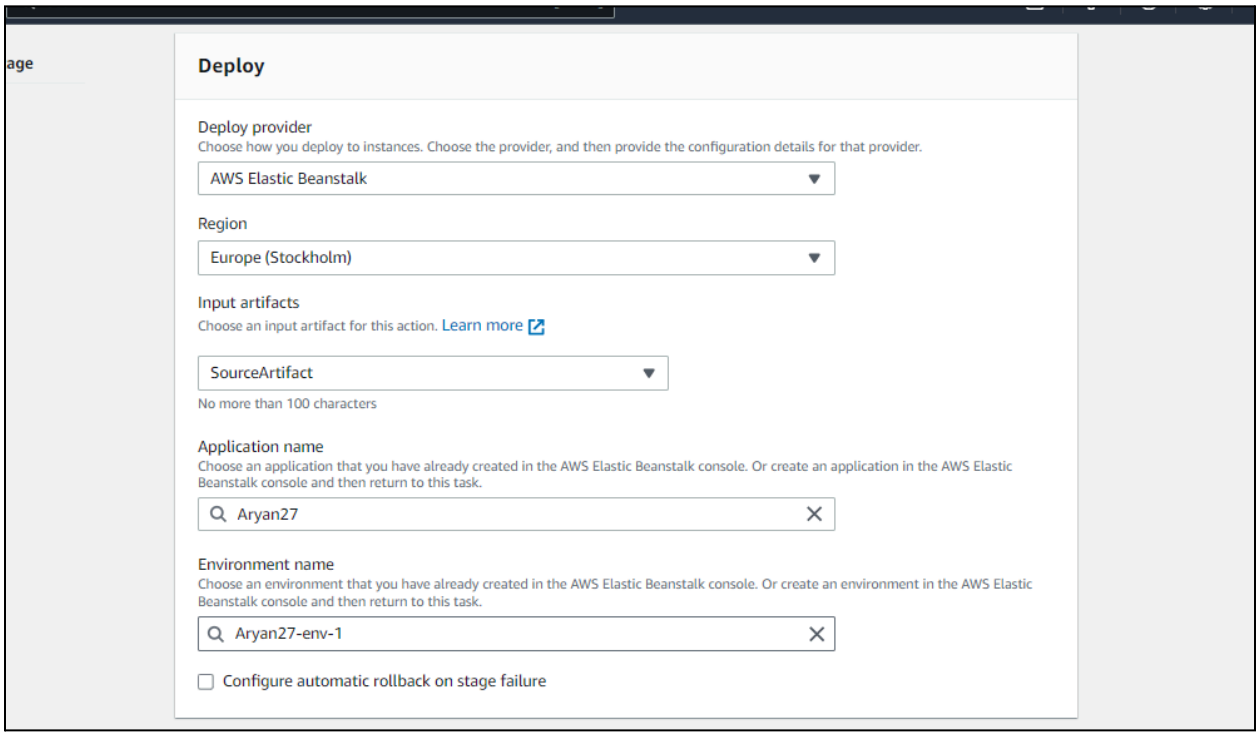
Step 12: In the source stage, choose GitHub v2 as the provider, then connect your GitHub account to
AWS by creating a connection. You'd need your GitHub credentials and then you'd need to
authorize and install AWS on the forked GitHub Repository.

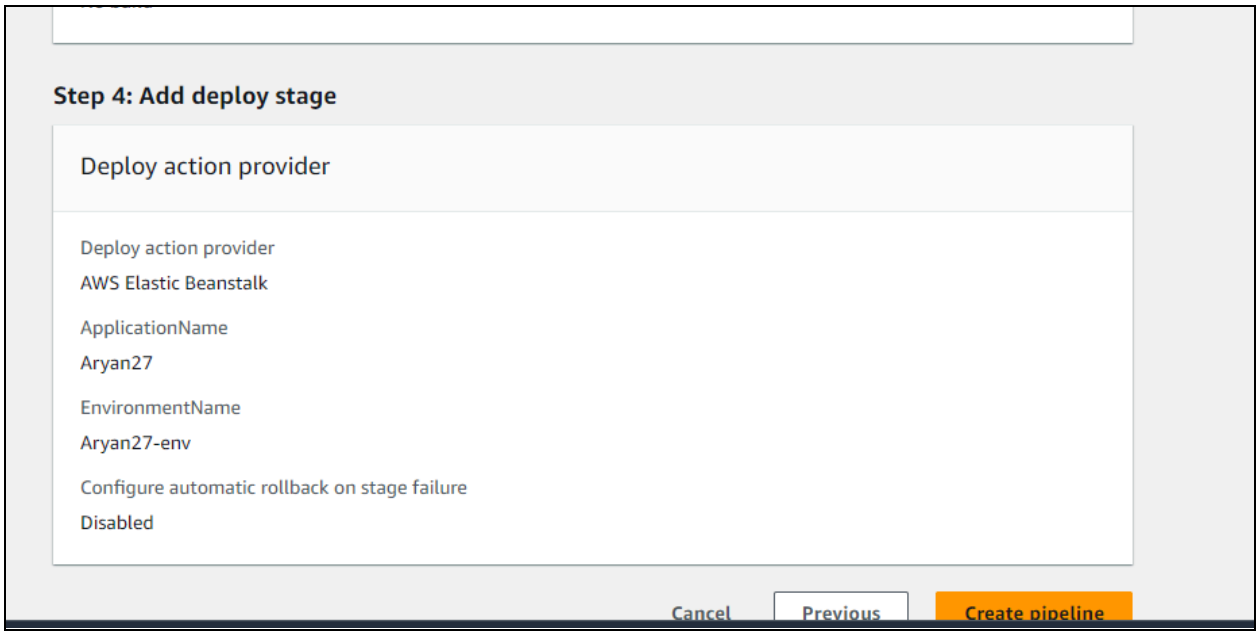




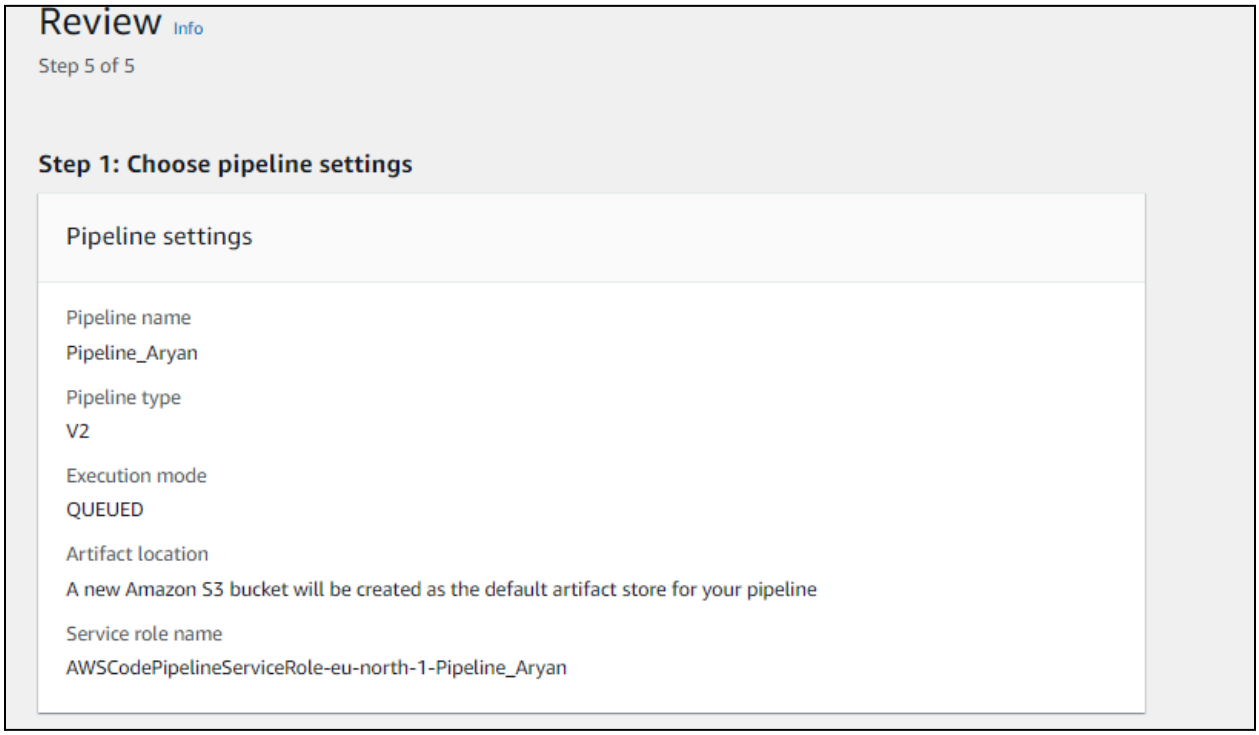
Then, simply choose this forked repository and the branch which you will be able to find in the search box. After that, click Continue and skip the build stage. Proceed to the Deployment stage.

Step 13: Choose Beanstalk as the Deploy Provider, same region as the Bucket and Beanstalk, name and
environment name. Click Next, Review and create the pipeline.





Step 14: Review all the selected steps once.



Step 2: Add source stage

Source action provider

Source action provider
GitHub (Version 1)

PollForSourceChanges
false

Repo
IPLab-02

Owner
AryanPatankar27

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

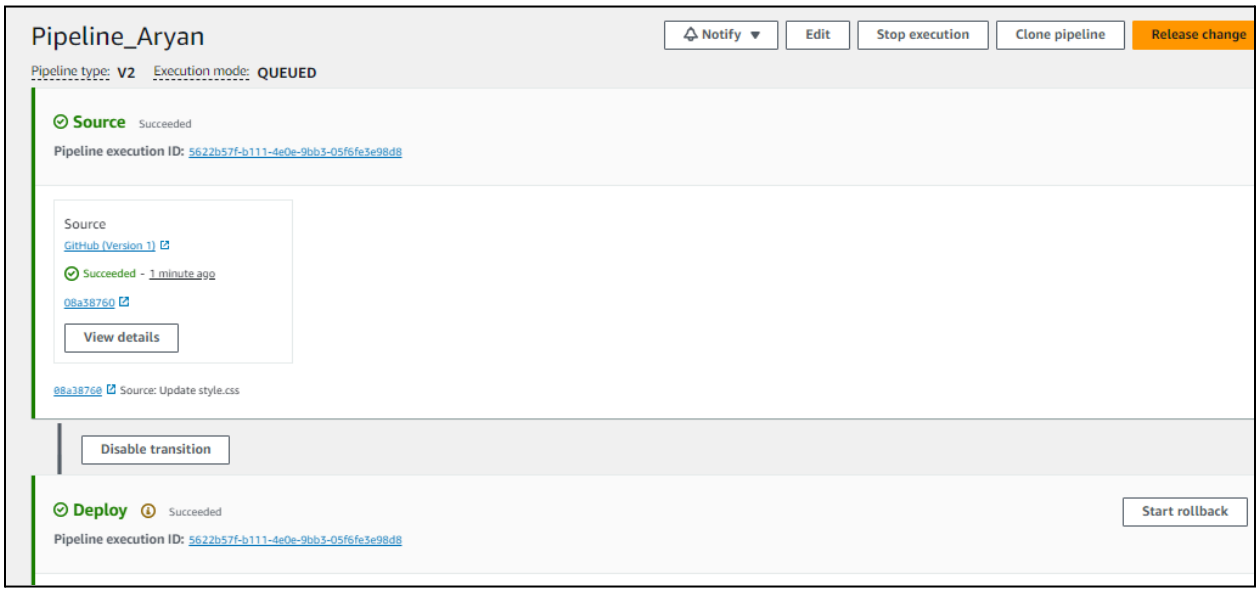
ApplicationName
Aryan27

EnvironmentName
Aryan27-env-1

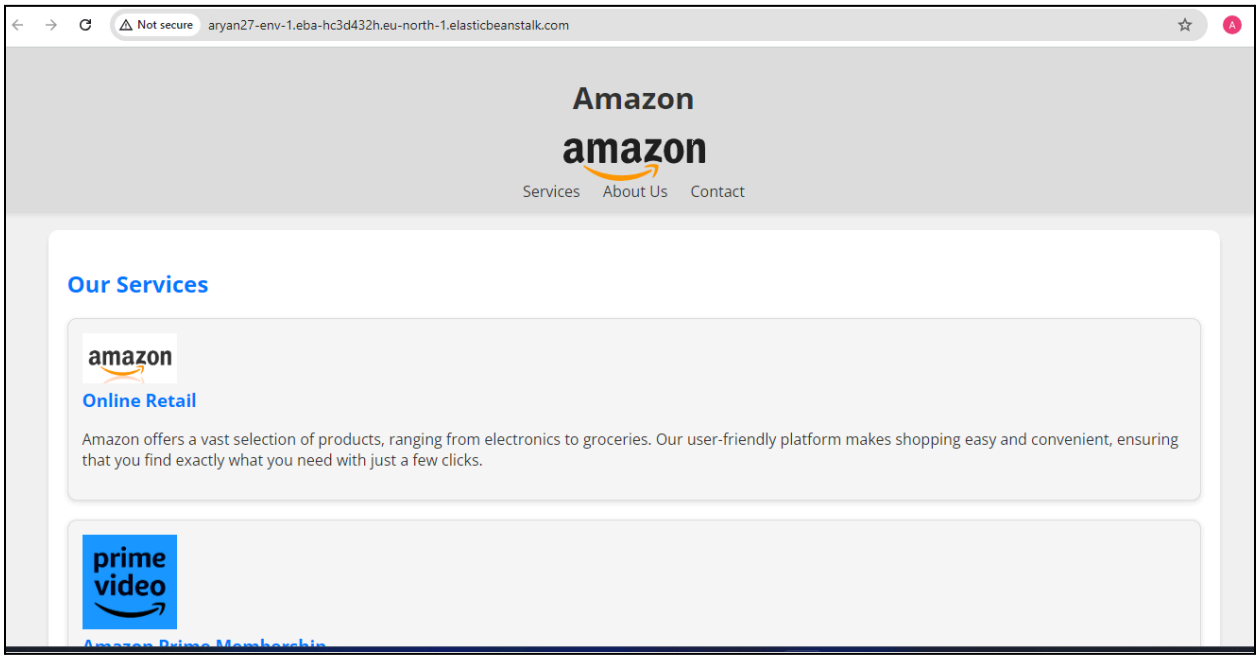
Configure automatic rollback on stage failure
Disabled

Cancel | Previous | Create pipeline

Step 15: In a few minutes, we will have our pipeline created. Once we have the success message on the
Deploy part, we can go ahead and check our URL provided in the EBS environment.



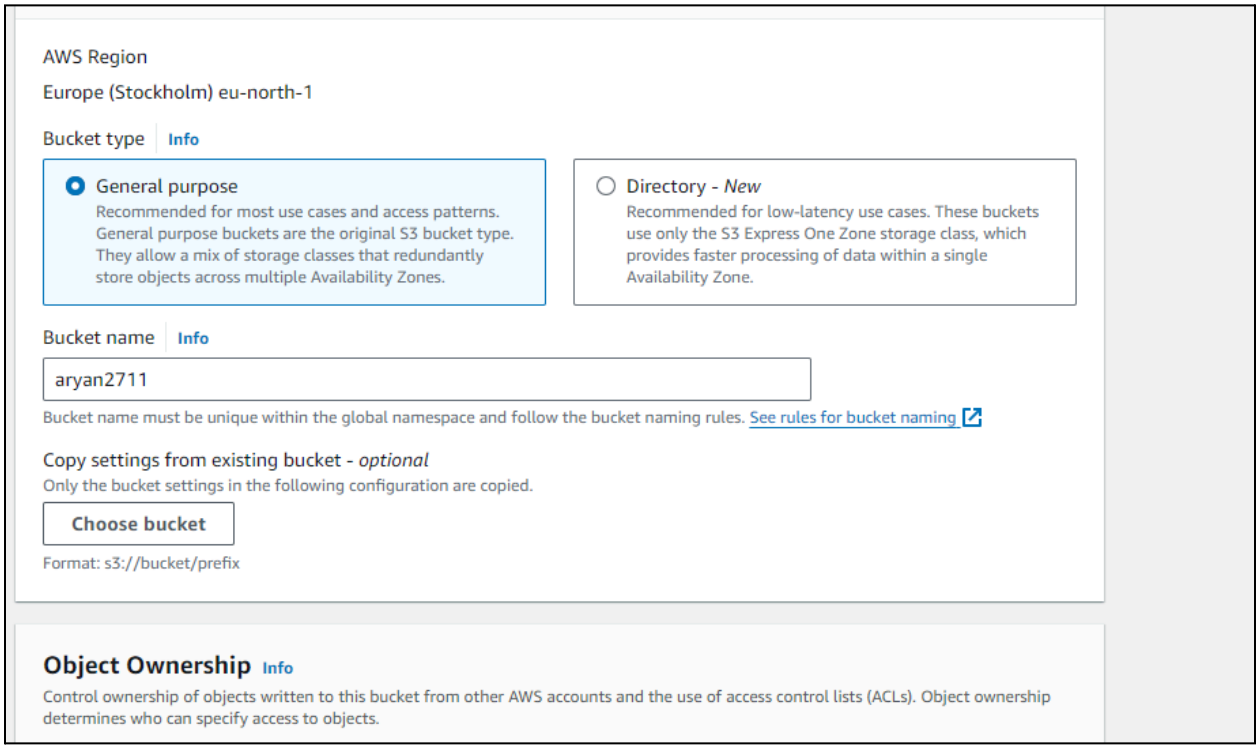
This is the sample website we just created.

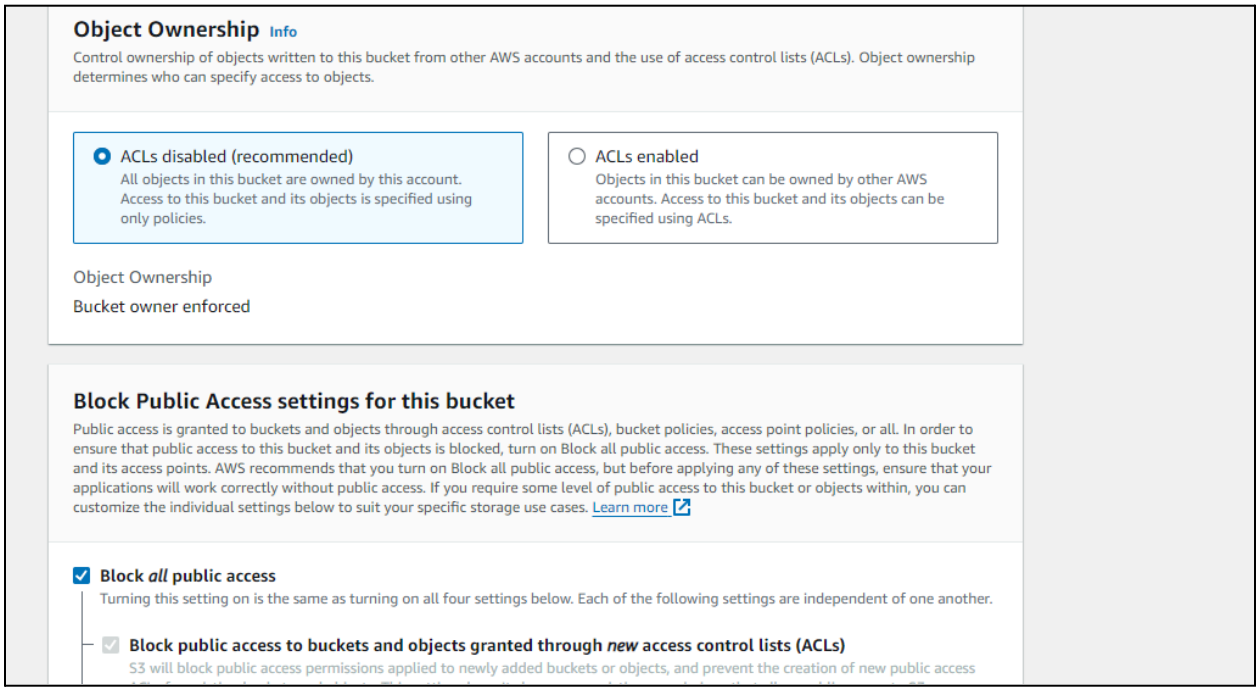


If you can see this, that means that you successfully created an automated software using CodePipeline.

Using S3 Bucket

Step 16: Now, we will be deploying our website using the S3 bucket.So proceed with the options as given below.





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

▶ **Advanced settings**

After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

Cancel | **Create bucket**

Successfully created bucket "aryan2711"
To upload files and folders, or to configure additional bucket settings, choose **View details**.

View details

Storage lens provides visibility into storage usage and activity trends. Learn more

General purpose buckets | Directory buckets

General purpose buckets (3) Info | All AWS Regions

Buckets are containers for data stored in S3.

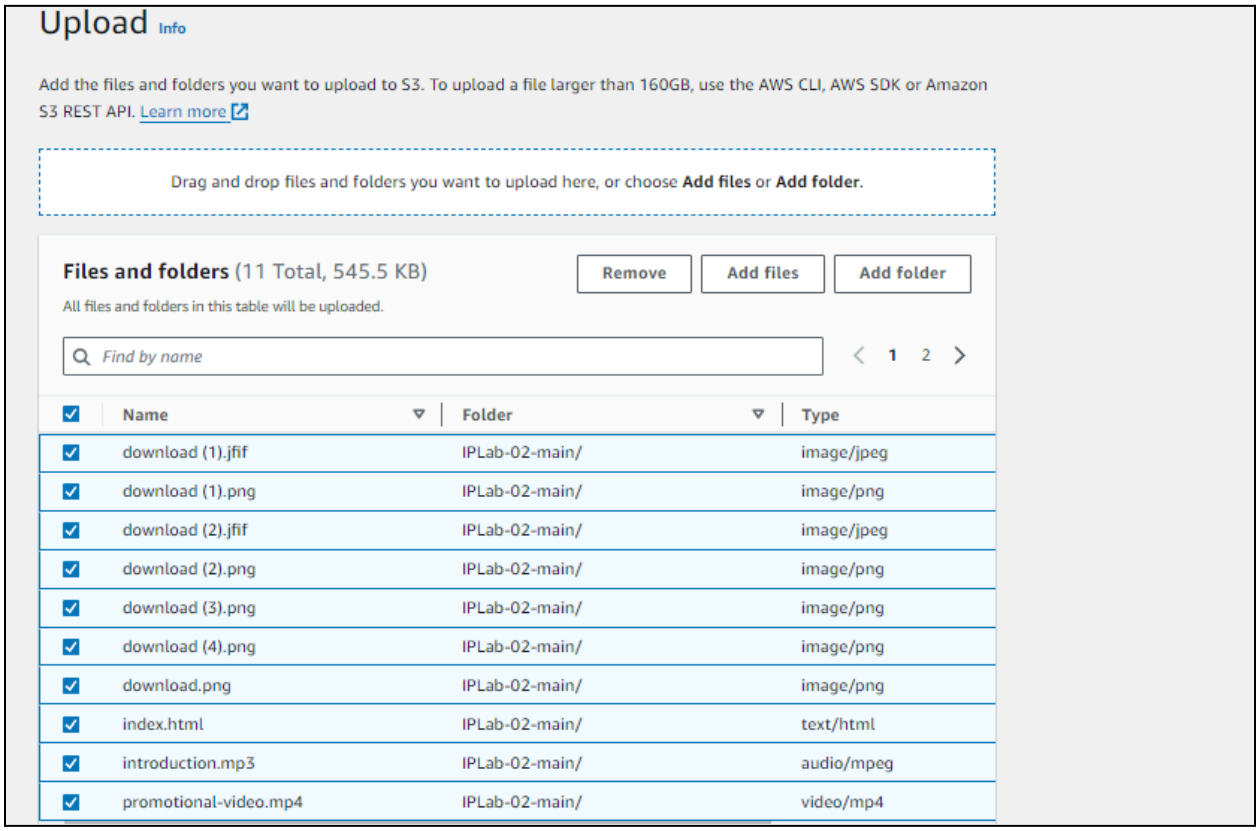
Copy ARN | Empty | Delete | **Create bucket**

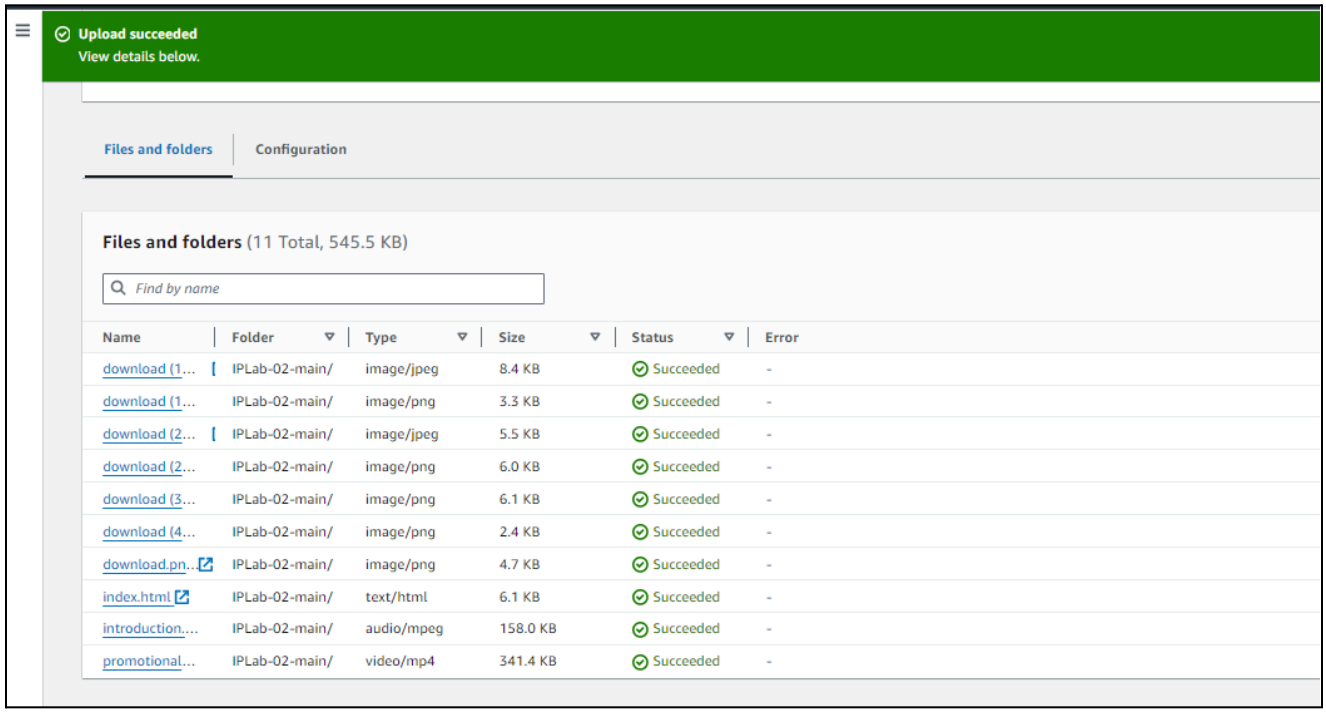
Find buckets by name

1

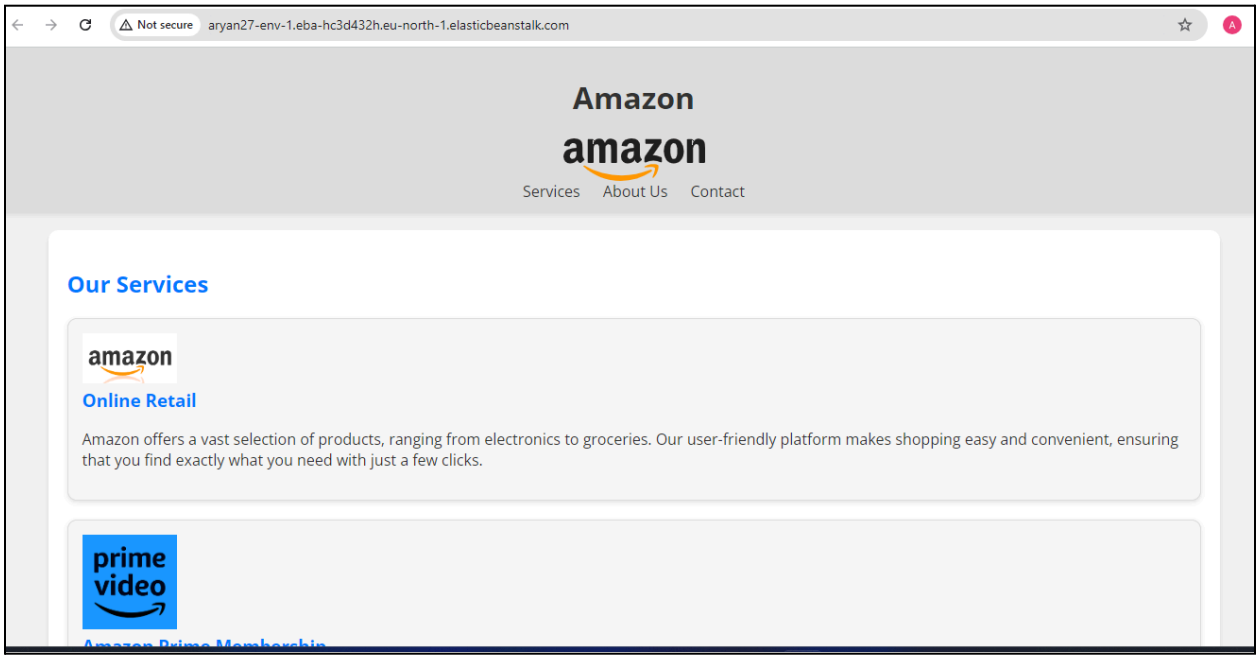
Name	AWS Region	IAM Access Analyzer	Creation date
aryan2711	Europe (Stockholm) eu-north-1	View analyzer for eu-north-1	August 21, 2024, 15:42:44 (UTC+05:30)
codepipeline-eu-north-1-365572256475	Europe (Stockholm) eu-north-1	View analyzer for eu-north-1	August 21, 2024, 15:29:16 (UTC+05:30)
elasticbeanstalk-eu-north-1-405894863107	Europe (Stockholm) eu-north-1	View analyzer for eu-north-1	August 21, 2024, 14:00:18 (UTC+05:30)

Step 17: Upload all the files that you want on your website that is to be hosted.





Step 18: Here,if the upload of files is successful you would get the following page,meaning your website has been successfully hosted using the S3 bucket.



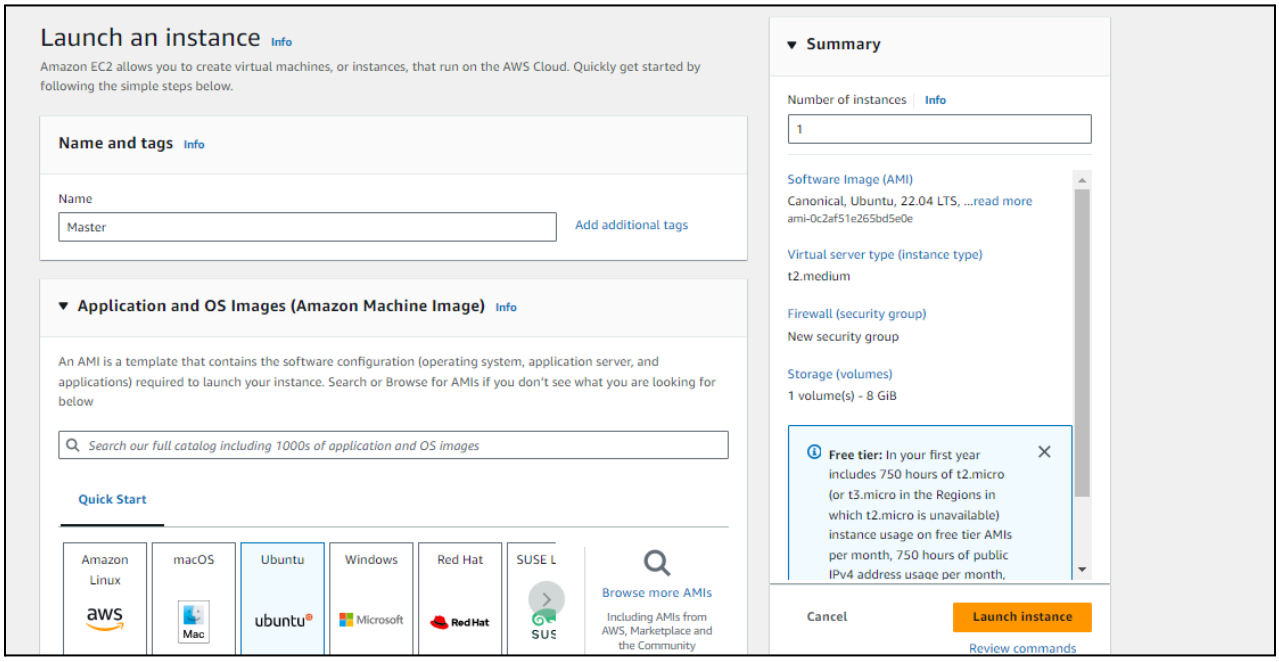
ADVANCE DEVOPS EXP 3

Name:Aryan Anil Patankar
Class:D15A
Roll No:34

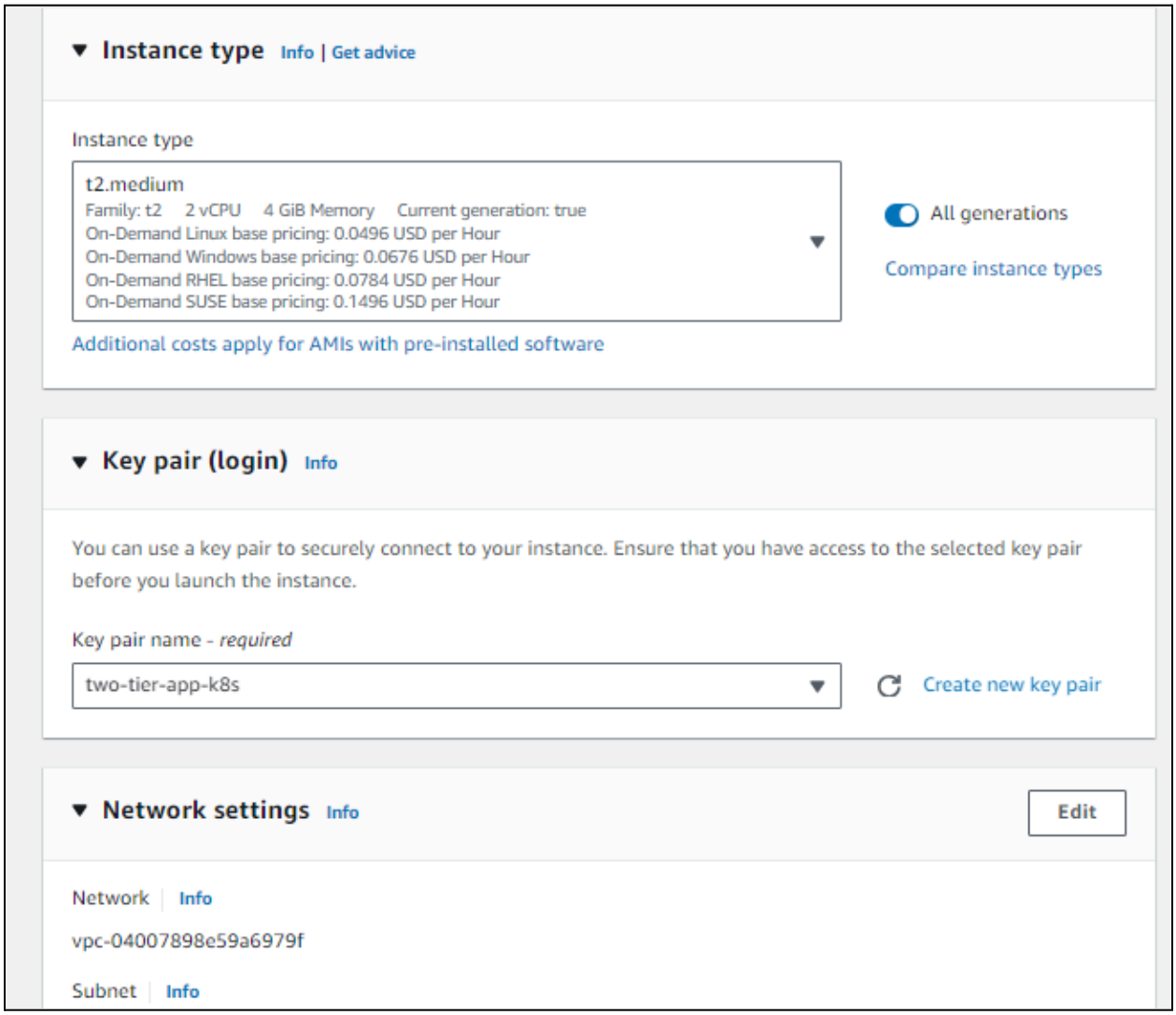
Aim:To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Step 1: Pre-requisites

1.1 Create 3 EC2 instances,one for the master node and two for the worker nodes.



1.2 Proceed with the following settings and create a new key pair as follows(use the same key pair for all the three nodes)



Create key pair

Key pair name

Key pairs allow you to connect to your instance securely.

two-tier-app-k8s

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

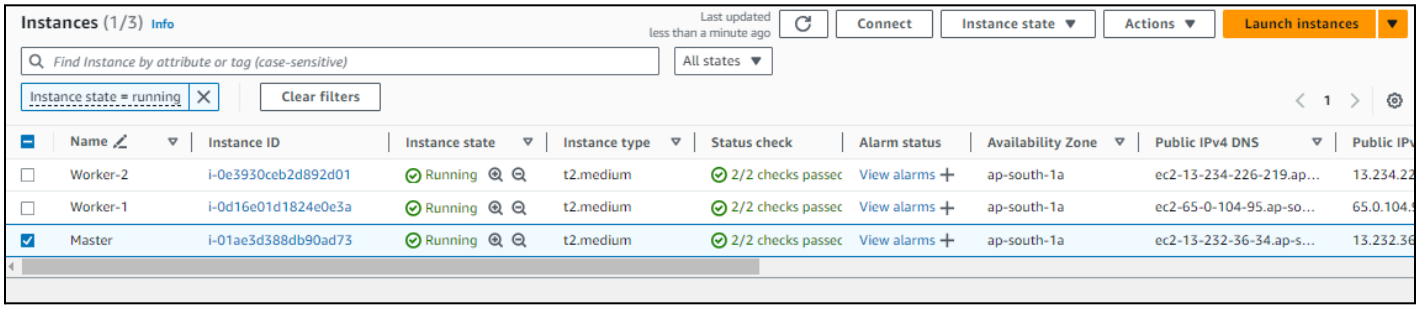
- RSA — RSA encrypted private and public key pair
- ED25519 — ED25519 encrypted private and public key pair

Private key file format

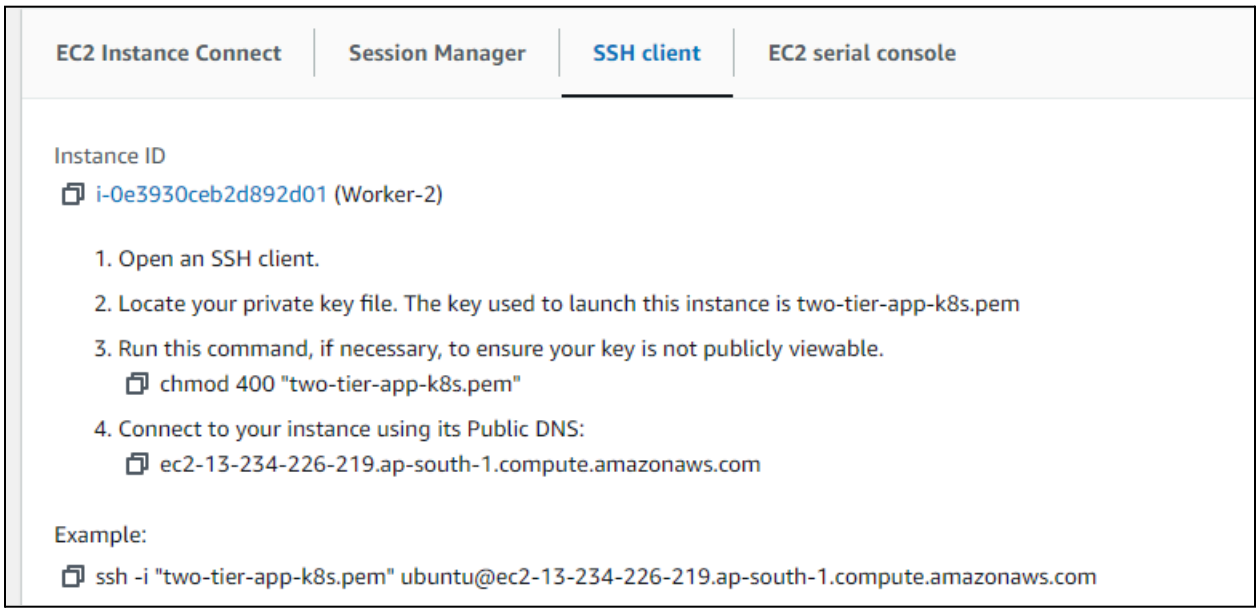
- .pem — For use with OpenSSH
- .ppk — For use with PuTTY

When prompted, store the private key in a secure and accessible location on

Cancel | Create key pair



1.3 After the instances have been created,copy the text given in the example part of each of the three instances into git bash.

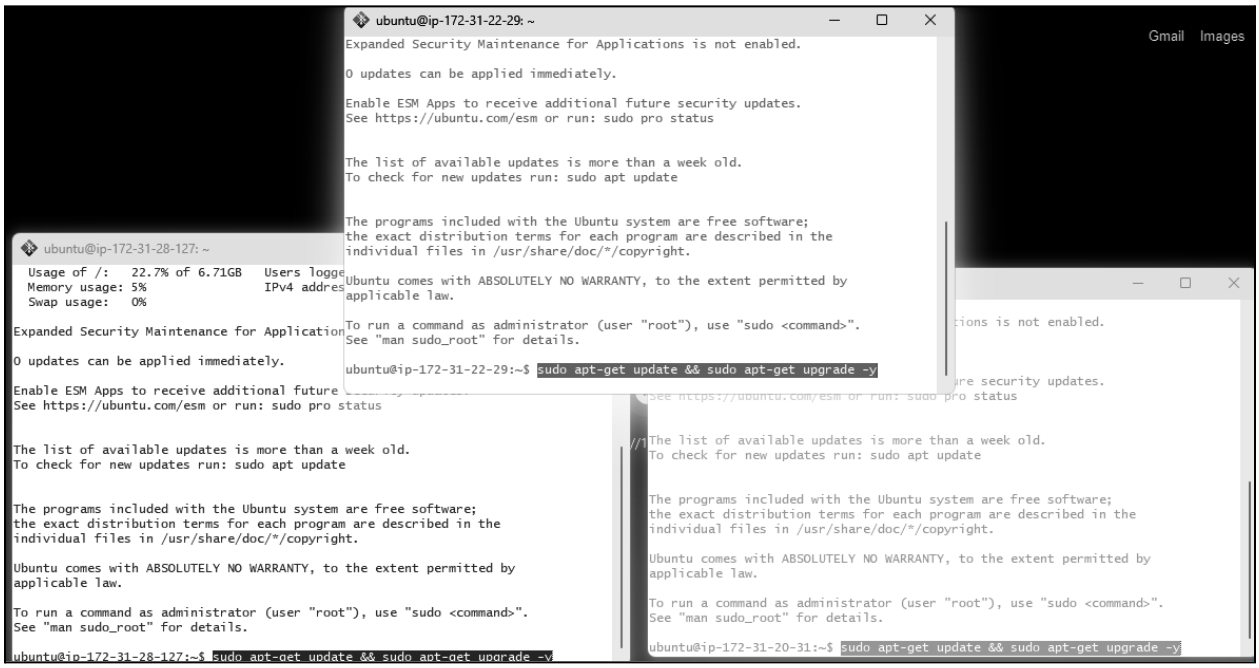


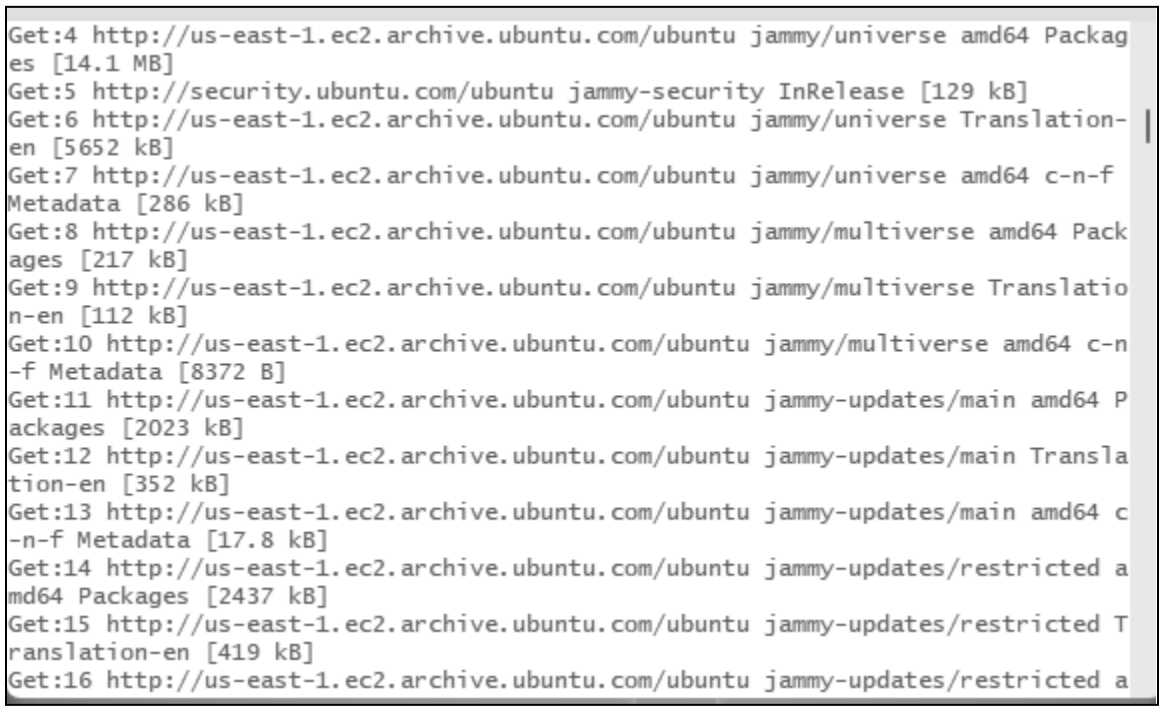

```
acer@TMP214-53 MINGW64 ~/Downloads
$ ssh -i "two-tier-app-k8s.pem" ubuntu@ec2-13-232-36-34.ap-south-1.compute.amazo
naws.com
The authenticity of host 'ec2-13-232-36-34.ap-south-1.compute.amazonaws.com (13.
232.36.34)' can't be established.
ED25519 key fingerprint is SHA256:uVGEO+FWYefj60j0ft70Sralv8NrzEi/IwxAtBY+EPE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-13-232-36-34.ap-south-1.compute.amazonaws.com' (
ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1022-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Wed Sep 11 14:07:10 UTC 2024

  System load:  0.0               Processes:             106
  Usage of /:   20.7% of 7.57GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for eth0: 172.31.45.227
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
```

Step 2:Prepare Nodes

2.1. Update the package manager on all nodes:

sudo apt-get update && sudo apt-get upgrade -y





2.2. Disable Swap (Kubernetes requires swap to be off):
sudo swapoff -a
sudo sed -i '/ swap / s/^/#/' /etc/fstab

```
ubuntu@ip-172-31-22-29:~$ sudo swapoff -a
sudo sed -i '/ swap / s/^/#/' /etc/fstab
```

2.3. Load necessary kernel modules for networking and iptables:
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe br_netfilter

```
ubuntu@ip-172-31-22-29:~$ cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe br_netfilter
overlay
br_netfilter
```

2.4. Configure sysctl settings for Kubernetes networking:
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system

```
ubuntu@ip-172-31-22-29:~$ cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe br_netfilter
overlay
br_netfilter
ubuntu@ip-172-31-22-29:~$ cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
* Applying /etc/sysctl.d/10-console-messages.conf ...
kernel.printk = 4 4 1 7
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...
kernel.kptr_restrict = 1
```

Step 3: Install Docker

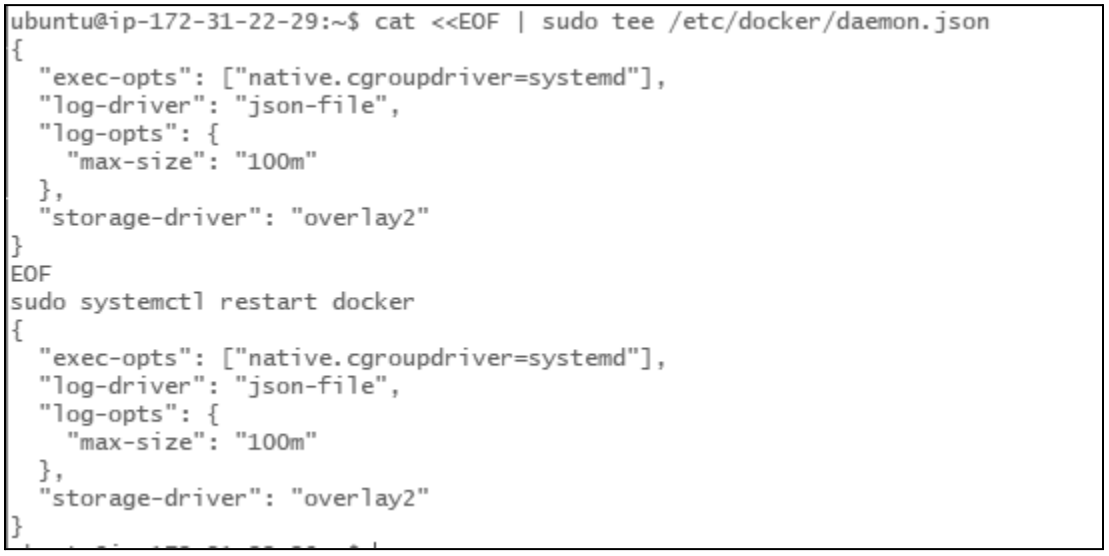
Kubernetes uses container runtimes like Docker. Install Docker on all nodes.

sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io

```
ubuntu@ip-172-31-22-29:~$ sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl software-proper
ties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubun
tu $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Fetched 129 kB in 1s (241 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20230311ubuntu0.22.04.1).
ca-certificates set to manually installed.
curl is already the newest version (7.81.0-1ubuntu1.17).
curl set to manually installed.
software-properties-common is already the newest version (0.99.22.9).
software-properties-common set to manually installed.
```

Configure Docker for Kubernetes:

```
cat <<EOF | sudo tee /etc/docker/daemon.json
{
 "exec-opts": ["native.cgroupdriver=systemd"],
 "log-driver": "json-file",
 "log-opts": {
  "max-size": "100m"
 },
 "storage-driver": "overlay2"
}
EOF
sudo systemctl restart docker
```



Step 4: Install kubeadm, kubelet, kubectl

Install Kubernetes tools on all nodes.

4.1. Add Kubernetes APT repository:

```
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list
```

```
ubuntu@ip-172-31-22-29:~$ sudo curl -fsSLo /usr/share/keyrings/kubernetes-archiv
e-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https:/
/apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/k
ubernetes.list
deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.k
ubernetes.io/ kubernetes-xenial main
```

4.2. Install kubeadm, kubelet, and kubectl:

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

```
ubuntu@ip-172-31-22-29:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu jammy InRelease
```

Step 5: Initialize the Kubernetes Cluster on Master Node

On the master node:

sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
ubuntu@ip-172-31-22-29:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --v
=5
Found multiple CRI endpoints on the host. Please define which one do you wish to
 use by setting the 'criSocket' field in the kubeadm configuration file: unix://
/var/run/containerd/containerd.sock, unix:///var/run/crio/crio.sock
k8s.io/kubernetes/cmd/kubeadm/app/util/runtime.detectCRISocketImpl
        cmd/kubeadm/app/util/runtime/runtime.go:167
k8s.io/kubernetes/cmd/kubeadm/app/util/runtime.DetectCRISocket
        cmd/kubeadm/app/util/runtime/runtime.go:175
k8s.io/kubernetes/cmd/kubeadm/app/util/config.SetNodeRegistrationDynamicDefaults
        cmd/kubeadm/app/util/config/initconfiguration.go:118
k8s.io/kubernetes/cmd/kubeadm/app/util/config.SetInitDynamicDefaults
        cmd/kubeadm/app/util/config/initconfiguration.go:64
k8s.io/kubernetes/cmd/kubeadm/app/util/config.DefaultedInitConfiguration
        cmd/kubeadm/app/util/config/initconfiguration.go:248
k8s.io/kubernetes/cmd/kubeadm/app/util/config.LoadOrDefaultInitConfiguration
        cmd/kubeadm/app/util/config/initconfiguration.go:282
k8s.io/kubernetes/cmd/kubeadm/app/cmd.newInitData
        cmd/kubeadm/app/cmd/init.go:319
k8s.io/kubernetes/cmd/kubeadm/app/cmd.newCmdInit.func3
        cmd/kubeadm/app/cmd/init.go:170
k8s.io/kubernetes/cmd/kubeadm/app/cmd/phases/workflow.(*Runner).InitData
        cmd/kubeadm/app/cmd/phases/workflow/runner.go:183
k8s.io/kubernetes/cmd/kubeadm/app/cmd.newCmdInit.func1
```

5.1. Set up kubectl on the master node:
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
ubuntu@ip-172-31-22-29:~$ sudo kubeadm config images pull

sudo kubeadm init

mkdir -p "$HOME"/.kube
sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config
sudo chown "$(id -u)":"$(id -g)" "$HOME"/.kube/config


# Network Plugin = calico
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml

kubeadm token create --print-join-command --v=5
Found multiple CRI endpoints on the host. Please define which one do you wish to use by setting the 'criSocket' field in the kubeadm configuration file: unix:///var/run/containerd/containerd
ock, unix:///var/run/crio/crio.sock
To see the stack trace of this error execute with --v=5 or higher
Found multiple CRI endpoints on the host. Please define which one do you wish to use by setting the 'criSocket' field in the kubeadm configuration file: unix:///var/run/containerd/containerd
ock, unix:///var/run/crio/crio.sock
```

Step 6: Install a Pod Network Add-on

To enable communication between pods, install a pod network plugin like Flannel or Calico.

Install Flannel:

kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

```
ubuntu@ip-172-31-22-29:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml --validate=false

E0913 15:35:04.261458   19259 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:808
E0913 15:35:04.261902   19259 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:808
E0913 15:35:04.263424   19259 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:808
E0913 15:35:04.263795   19259 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:808
E0913 15:35:04.265840   19259 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:808
E0913 15:35:04.266524   19259 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:808
unable to recognize "https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml": Get "http://localhost:8080/api?timeout=32s": dia
```

Step 7: Join Worker Nodes to the Cluster

On the **worker nodes**, run the command provided by the master node during initialization . It looks something like this:

sudo kubeadm join <master-ip>:6443 --token <token> --discovery-token-ca-cert-hash sha256:<hash>

```
clusterrolebinding.rbac.authorization.k8s.io/calico-cni-plugin created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
kubeadm join 172.31.62.216:6443 --token br7fe5.hq28adbmn1mu17ky --discovery-token-ca-cert-hash sha256:2bc469a8d14fbeb0f879328d2b416fad
32b29a8505d3f448b98703fff3b014d9
```

Step 8: Verify the Cluster

Once the worker node joins, check the status on the **master node**

```
ubuntu@ip-172-31-45-227:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-43-211   Ready    <none>          50s     v1.29.0
ip-172-31-45-13    Ready    <none>          34s     v1.29.0
ip-172-31-45-227   Ready    control-plane   5m17s   v1.29.0
ubuntu@ip-172-31-45-227:~$
```

ADVANCE DEVOPS EXP 4

Name:Aryan Anil Patankar
Class:D15A
Roll No:34

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

Step 1: Install Kubectl on Ubuntu

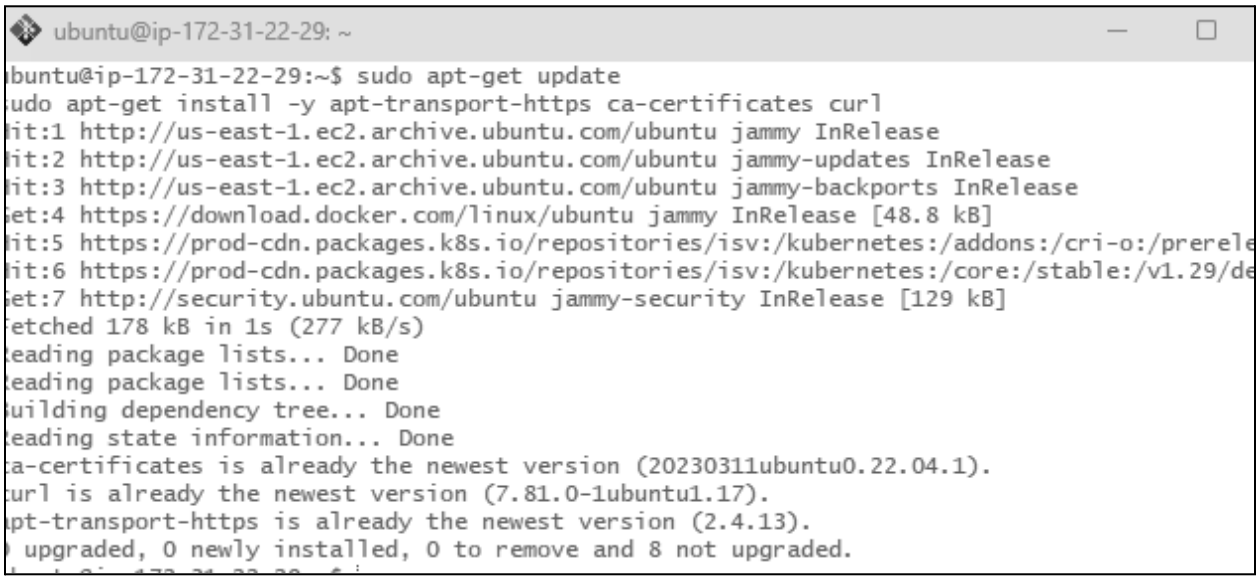
1.1 Add Kubernetes APT repository

First, add the Kubernetes repository to your system.

1. **Install prerequisites:**

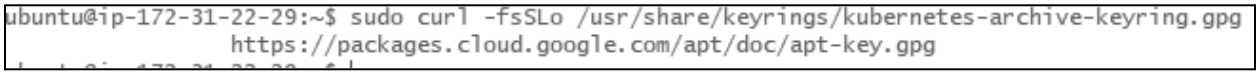
   sudo apt-get update

   sudo apt-get install -y apt-transport-https ca-certificates curl



2. **Add the GPG key for Kubernetes:**

   sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg



3. **Add the Kubernetes repository:**

echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-focal main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-22-29:~$  echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring
.gpg]  https://apt.kubernetes.io/ kubernetes-focal main" | sudo tee /etc/apt/sources.list.d/ku
bernetes.list
deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]  https://apt.kubernetes.io/
 kubernetes-focal main
```

1.2 Install kubectl

Now install kubectl:

sudo apt-get update

sudo apt-get install -y kubectl

```
ubuntu@ip-172-31-22-29:~$ sudo apt-get update
sudo apt-get install -y kubectl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri-o:/prerelease:/main/deb  InRelease
Ign:7 https://packages.cloud.google.com/apt kubernetes-focal InRelease
Err:8 https://packages.cloud.google.com/apt kubernetes-focal Release
  404  Not Found [IP: 172.253.62.138 443]
Reading package lists... Done
E: The repository 'https://apt.kubernetes.io kubernetes-focal Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
kubectl is already the newest version (1.29.0-1.1).
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
```

Verify the installation(extra):

kubectl version --client

```
ubuntu@ip-172-31-22-29:~$ kubectl version --client
Client Version: v1.29.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
```

Step 2: Deploying Your Application on Kubernetes

2.1 Set up Kubernetes Cluster

1. If you haven't already set up a Kubernetes cluster (e.g., with kubeadm), use minikube or any managed Kubernetes service (like EKS, GKE, etc.) to get a cluster running.
2. Once your cluster is ready, verify the nodes:

   kubectl get nodes

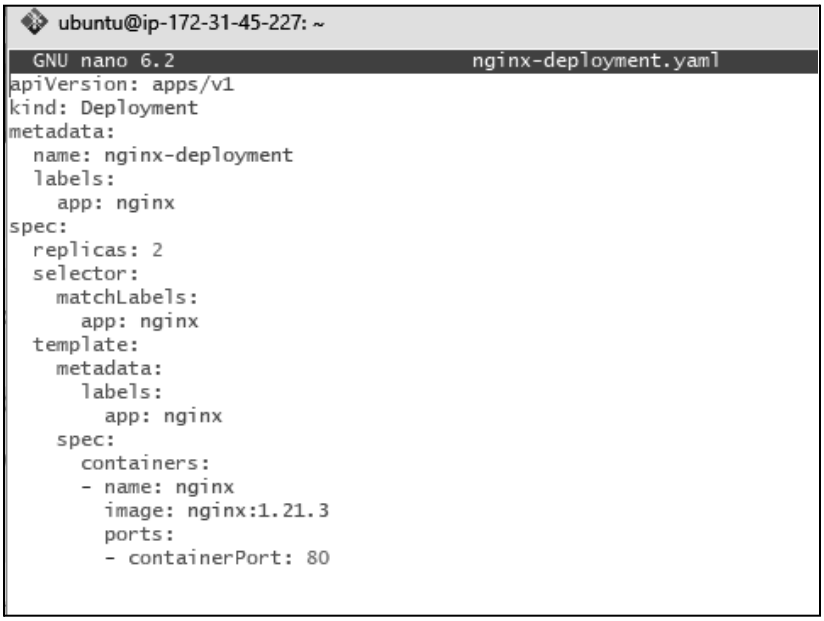
```
ubuntu@ip-172-31-45-227:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
ip-172-31-43-211    Ready    <none>          50s     v1.29.0
ip-172-31-45-13     Ready    <none>          34s     v1.29.0
ip-172-31-45-227    Ready    control-plane   5m17s   v1.29.0
ubuntu@ip-172-31-45-227:~$
```

Step 3:**Create the Deployment YAML file**

a)Create the YAML file: Use a text editor to create a file named nginx-deployment.yaml

```
ubuntu@ip-172-31-45-227:~$ nano nginx-deployment.yaml
```

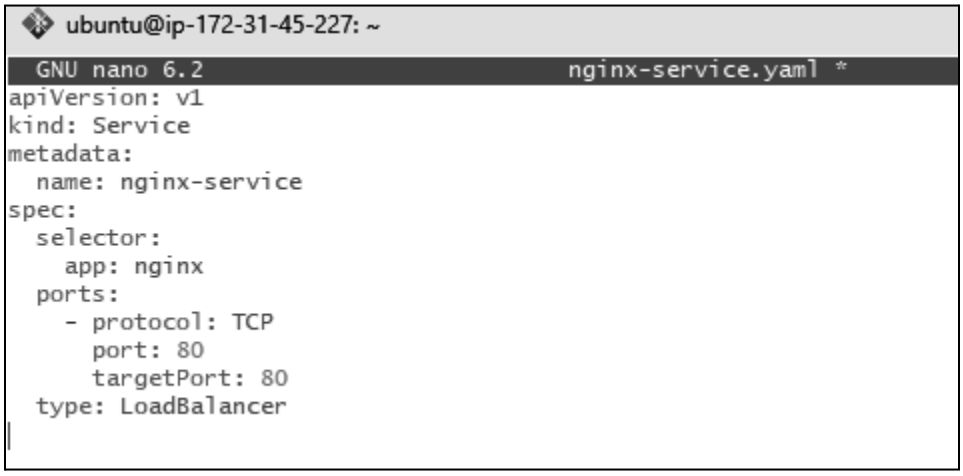
b)Add the Deployment Configuration: Copy and paste the following YAML content into the file. Save and exit the editor (Press Ctrl+X, then Y, and Enter).



Step 4:Create the Service YAML File

a)Create the YAML File: Create another file named nginx-service.yaml

b)Add the Service Configuration: Copy and paste the following YAML content into the file given below.



Step 5:Apply the YAML Files

a)Deploy the Application: Use kubectl to create the Deployment and Service from the YAML files.

```
ubuntu@ip-172-31-45-227:~$ kubectl apply -f nginx-deployment.yaml
kubectl apply -f nginx-service.yaml
deployment.apps/nginx-deployment created
service/nginx-service created
```

b)Verify the Deployment: Check the status of your Deployment,Pods and Services.

```
ubuntu@ip-172-31-45-227:~$ kubectl get deployments
kubectl get pods
kubectl get services
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2            2           40s
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6b4d6fdbf-6k84m   1/1     Running   0          40s
nginx-deployment-6b4d6fdbf-9d8j6   1/1     Running   0          40s
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP      10.96.0.1        <none>        443/TCP        40m
nginx-service   LoadBalancer   10.106.182.152   <pending>     80:32317/TCP   40s
```

Describe the deployment(Extra)

```
ubuntu@ip-172-31-45-227:~$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   1/1     1            1           14h
ubuntu@ip-172-31-45-227:~$ kubectl describe deployment
Name:                   nginx-deployment
Namespace:              default
CreationTimestamp:      Wed, 11 Sep 2024 17:16:17 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 2
Selector:               app=nginx
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx:latest
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:
      /usr/share/nginx/html from website-volume (rw)
  Volumes:
   website-volume:
    Type:      ConfigMap (a volume populated by a ConfigMap)
    Name:      nginx-website
    Optional:  false
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  nginx-deployment-6b4d6fdbf (0/0 replicas created)
NewReplicaSet:   nginx-deployment-776b8fd845 (1/1 replicas created)
Events:          <none>
```

Step 6:Ensure Service is Running

6.1 **Verify Service**: Run the following command to check the services running in your cluster:

kubectl get service

```
ubuntu@ip-172-31-45-227:~$ kubectl get service
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.96.0.1        <none>        443/TCP        16h
nginx           NodePort    10.106.0.176     <none>        80:32618/TCP   76m
nginx-service   NodePort    10.106.182.152   <none>        80:30007/TCP   15h
nginx2          NodePort    10.99.32.156     <none>        80:31421/TCP   8s
```

Step 7:Forward the Service Port to Your Local Machine

kubectl port-forward allows you to forward a port from your local machine to a port on a service running in the Kubernetes cluster.

1. **Forward the Service Port**: Use the following command to forward a local port to the service's target port.

   kubectl port-forward service/<service-name> <local-port>:<service-port>

```
ubuntu@ip-172-31-45-227:~$ kubectl port-forward service/nginx-service 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

This command will forward local port 8080 on your machine to port 80 of the service nginx-service running inside the cluster.

2. This means port forwarding is now active, and any traffic to localhost:8080 will be routed to the nginx-service on port 80.

```
ubuntu@ip-172-31-45-227:~$ kubectl port-forward service/nginx-service 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
^Cubuntu@ip-172-31-45-227:~kubectl port-forward service/nginx-service 8081:8080
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
^Cubuntu@ip-172-31-45-227:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-776b8fd845-k9cx4   1/1     Running   0          113m
ubuntu@ip-172-31-45-227:~$ kubectl logs nginx-deployment-776b8fd845-k9cx4
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/09/12 06:35:51 [notice] 1#1: using the "epoll" event method
2024/09/12 06:35:51 [notice] 1#1: nginx/1.27.1
2024/09/12 06:35:51 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/09/12 06:35:51 [notice] 1#1: OS: Linux 6.5.0-1022-aws
2024/09/12 06:35:51 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/09/12 06:35:51 [notice] 1#1: start worker processes
2024/09/12 06:35:51 [notice] 1#1: start worker process 24
2024/09/12 06:35:51 [notice] 1#1: start worker process 25
```

Step 8: Access the Application Locally

1. **Open a Web Browser**: Now open your web browser and go to the following URL:

   http://localhost:8080

You should see the application (in this case, Nginx) that you have deployed running in the Kubernetes cluster, served locally via port 8080.

In case the port 8080 is unavailable, try using a different port like 8081

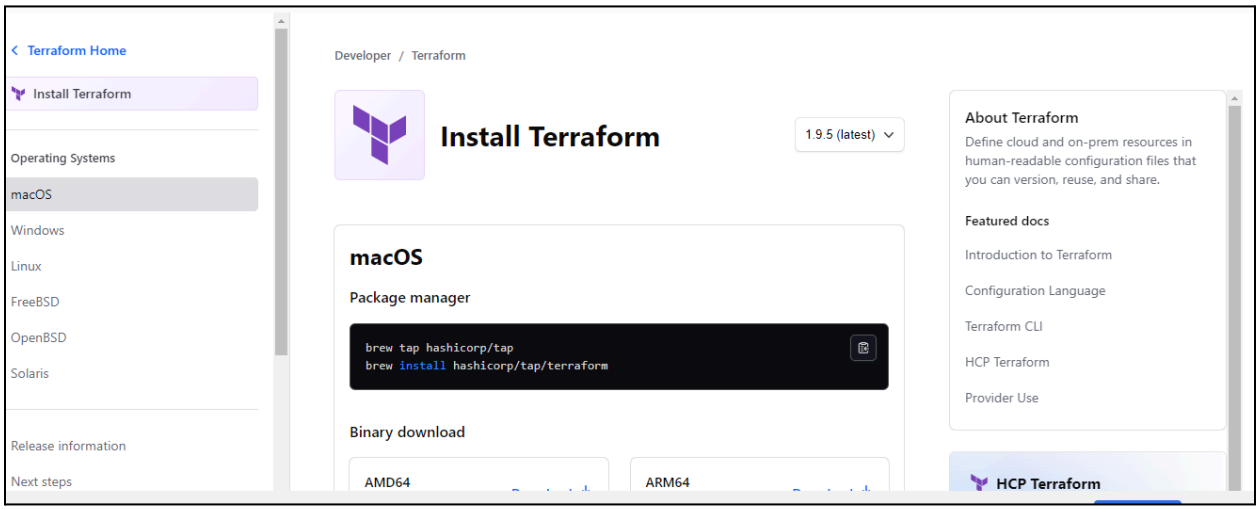
ADVANCE DEVOPS EXP 5

Name:Aryan Anil Patankar
Class:D15A
Roll No:34

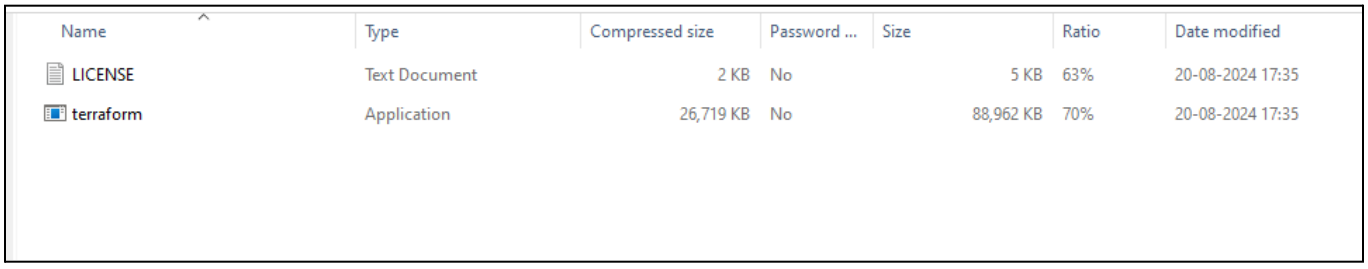
Aim:To understand terraform lifecycle, core concepts/terminologies and install it on a Linux Machine and Windows.

Installation for Windows:

Step 1: Go to the website terraform.io and install Terraform from there..Select the AMD64 option for Windows and download Terraform.

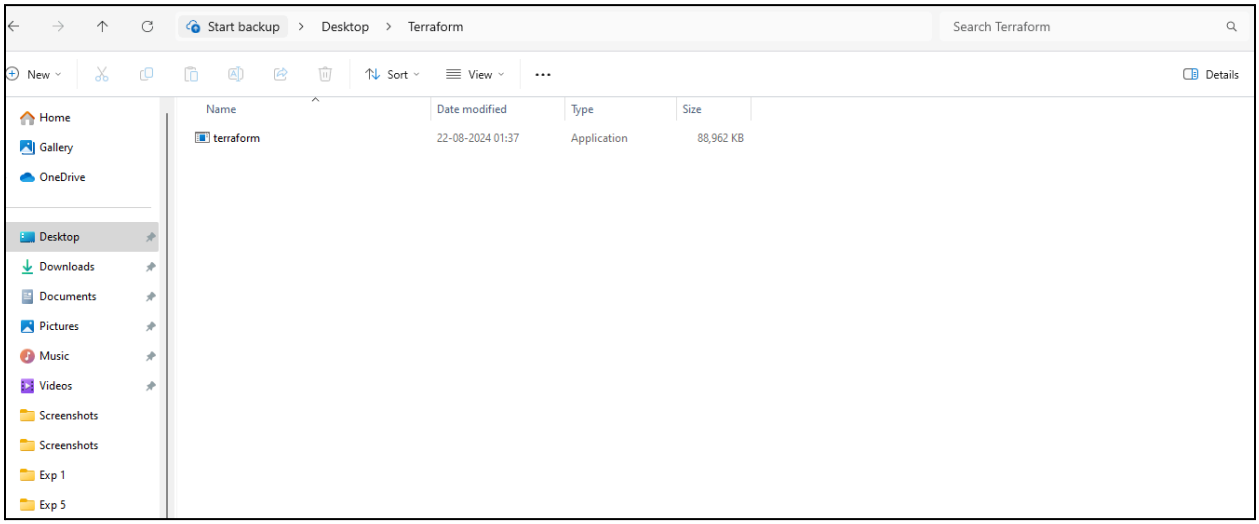


Step 2: Go to the zip file where Terraform is installed.

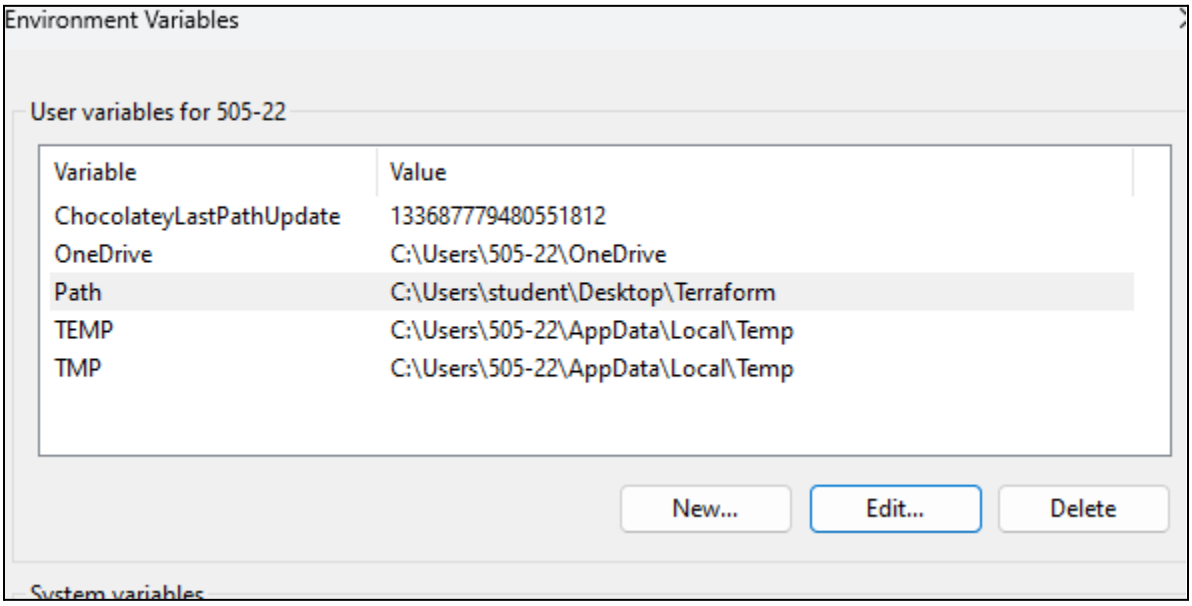


Name	Type	Compressed size	Password ...	Size	Ratio	Date modified
LICENSE	Text Document	2 KB	No	5 KB	63%	20-08-2024 17:35
terraform	Application	26,719 KB	No	88,962 KB	70%	20-08-2024 17:35

Step 3: Since the installed file is a zip file,create a new folder on desktop and copy the installed terraform application there.



Step 4: Now go to search bar,select edit environment variables option,then go to the path option.Now add the file path of the directory wherein we have installed the terraform application.



Step 5: Now go to the folder where we have installed terraform and open Powershell inside it.After this type ‘terraform’ to make sure that terraform has been installed on the system.
The command ‘terraform –version simply checks the current version of terraform that has been installed.

```
PS C:\Users\student\Desktop\Terraform> terraform
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure

All other commands:
  console       Try Terraform expressions at an interactive command prompt
  fmt           Reformat your configuration in the standard style
  force-unlock  Release a stuck lock on the current workspace
  get           Install or upgrade remote Terraform modules
  graph         Generate a Graphviz graph of the steps in an operation
  import        Associate existing infrastructure with a Terraform resource
  login         Obtain and save credentials for a remote host
  logout        Remove locally-stored credentials for a remote host
  metadata      Metadata related commands
  output        Show output values from your root module
  providers     Show the providers required for this configuration
  refresh       Update the state to match remote systems
  show          Show the current state or a saved plan
  state         Advanced state management
  taint         Mark a resource instance as not fully functional
  test          Execute integration tests for Terraform modules
  untaint       Remove the 'tainted' state from a resource instance
  version       Show the current Terraform version
  workspace     Workspace management
```

```
Global options (use these before the subcommand, if any):
  -chdir=DIR    Switch to a different working directory before executing the
                given subcommand.
  -help         Show this help output, or the help for a specified subcommand.
  -version      An alias for the "version" subcommand.
PS C:\Users\student\Desktop\Terraform> terraform --version
Terraform v1.9.4
on windows_amd64

Your version of Terraform is out of date! The latest version
is 1.9.5. You can update by downloading from https://www.terraform.io/downloads.html
PS C:\Users\student\Desktop\Terraform>
```

ADVANCE DEVOPS EXP 6

Name:Aryan Anil Patankar
Class:D15A
Roll No:34

Aim:To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform.
(S3 bucket or Docker) fdp.

Part A:Creating docker image using terraform

Prerequisite:
1) Download and Install Docker Desktop from https://www.docker.com/

Step 1: Check Docker functionality

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

C:\Users\student>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
  checkpoint  Manage checkpoints
  compose*    Docker Compose
  container   Manage containers
  context     Manage contexts
  debug*      Get a shell into any image or container
  desktop*    Docker Desktop commands (Alpha)
  dev*        Docker Dev Environments
  extension*  Manages Docker extensions
  feedback*   Provide feedback, right in your terminal!
```

Check for the docker version with the following command.

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

C:\Users\student>
```

Now, create a folder named 'Terraform Scripts' in which we save our different types of scripts which will be further used in this experiment.

Step 2: Firstly create a new folder named 'Docker' in the 'TerraformScripts' folder. Then create a new docker.tf file using Atom editor and write the followingcontents into it to create a Ubuntu Linux container.

Script:

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "2.21.0"
    }
  }
}

provider "docker" {
  host = "npipe:////./pipe/docker_engine"
}

# Pull the image
resource "docker_image" "ubuntu" {
  name = "ubuntu:latest"
}

# Create a container
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name  = "foo"
  command = ["sleep", "3600"]
```

}

docker.tf

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "2.21.0"
    }
  }
}

provider "docker" {
  host = "npipe:////./pipe/docker_engine"
}

# Pull the image
resource "docker_image" "ubuntu" {
  name = "ubuntu:latest"
}

# Create a container
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name  = "foo"
  command = ["sleep", "3600"]
}
```

Step 3: Execute Terraform Init command to initialize the resources

```
PS C:\Users\Admin\TerraformScripts> cd Docker
PS C:\Users\Admin\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.
```

Step 4: Execute Terraform plan to see the available resources

```
PS C:\Users\Admin\TerraformScripts\Docker> terraform plan

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

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = [
          + "sleep",
          + "3600",
        ]
      + 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.
```

Step 5: Execute Terraform apply to apply the configuration, which will automatically create and run the Ubuntu Linux container based on our configuration. Using command : **“terraform apply”**

```
PS C:\Users\Admin\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          = [
          + "sleep",
          + "3600",
        ]
      + 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.

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: Creation complete after 9s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Creating...
docker_container.foo: Creation complete after 2s [id=01adf07e5918931fee9b90073726a03671037923dd92032ce0e15bbb764a6f24]

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

Docker images, Before Executing Apply step:

```
PS C:\Users\Admin\TerraformScripts\Docker> docker images
REPOSITORY                          TAG       IMAGE ID       CREATED         SIZE
```

Docker images, After Executing Apply step:

```
PS C:\Users\Admin\TerraformScripts\Docker> docker images
REPOSITORY                          TAG       IMAGE ID       CREATED         SIZE
ubuntu                              latest    edbfe74c41f8   3 weeks ago     78.1MB
```

Step 6: Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container.

```
PS C:\Users\Admin\TerraformScripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=01adf07e5918931fee9b90073726a03671037923dd92032ce0e15bbb764a6f24]

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          = [
          - "sleep",
          - "3600",
        ] -> 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         = "01adf07e5918" -> null
      - id               = "01adf07e5918931fee9b90073726a03671037923dd92032ce0e15bbb764a6f24" -> 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
      - 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      = "default" -> 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=01adf07e5918931fee9b90073726a03671037923dd92032ce0e15bbb764a6f24]
docker_container.foo: Destruction complete after 0s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 1s

Destroy complete! Resources: 2 destroyed.
```

Docker images After Executing Destroy step

```
PS C:\Users\Admin\TerraformScripts\Docker> docker images
REPOSITORY                          TAG       IMAGE ID       CREATED         SIZE
```

ADVANCE DEVOPS EXP 7

Name:Aryan Anil Patankar
Class:D15A
Roll No:34

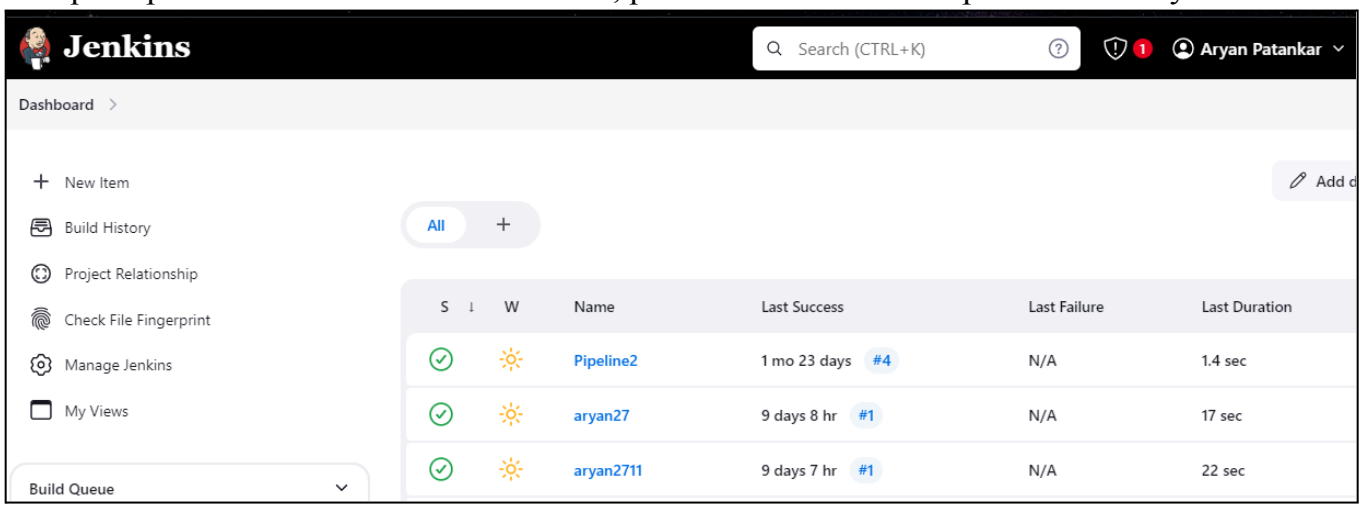
Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Integrating Jenkins with SonarQube:

- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

Steps to integrate Jenkins with SonarQube

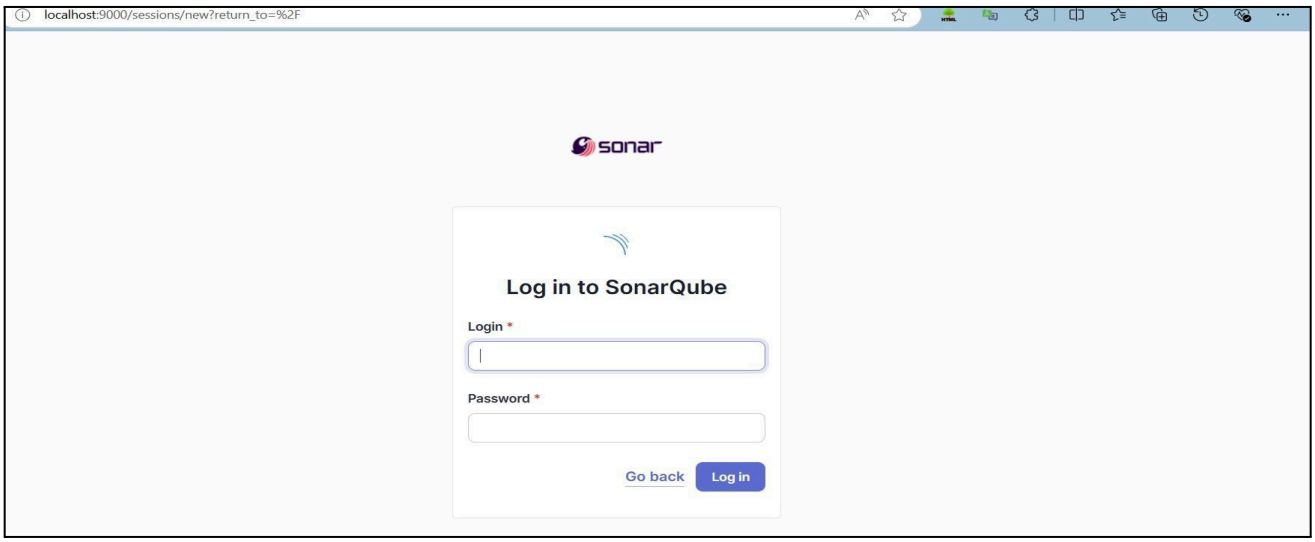
1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.



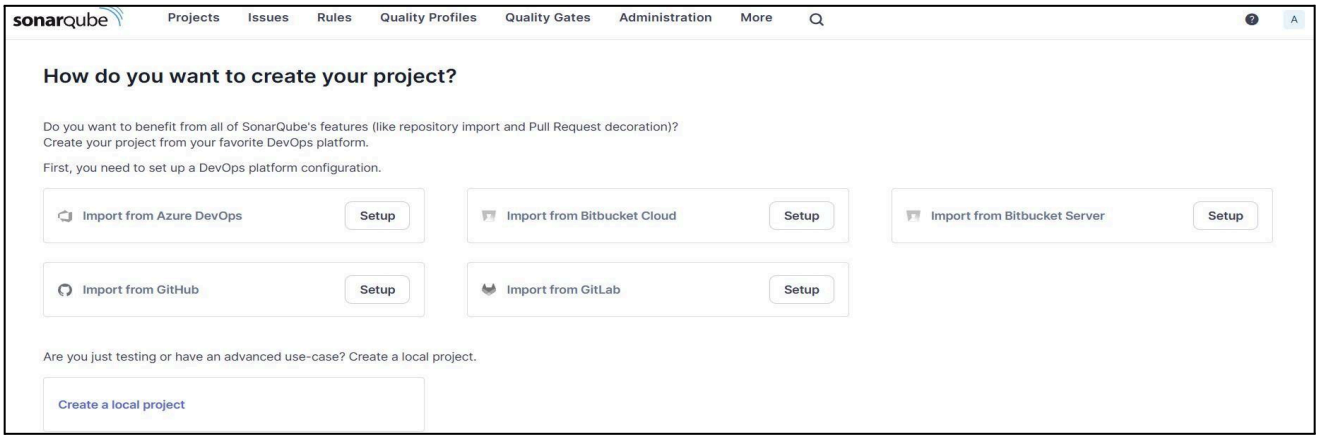
2. Run SonarQube in a Docker container using this command -
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

```
PS C:\Windows\system32> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
5ab3928e5e27607e3661d129731e4e600a9019574c7dc2767aa9b3bfdaa941be
```

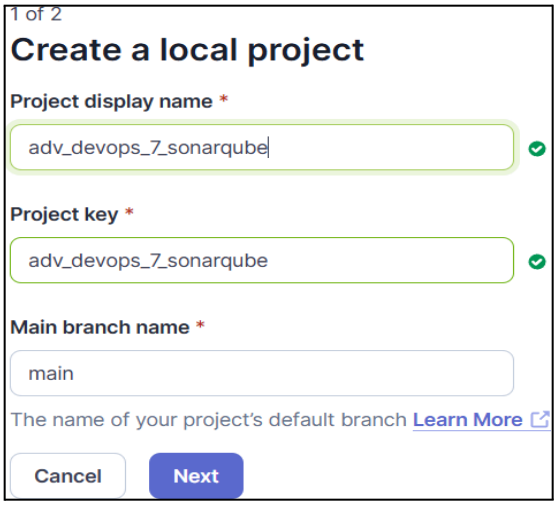
3. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.

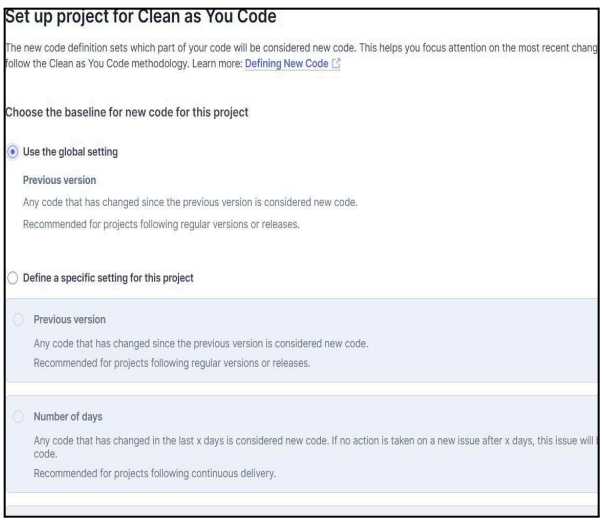


4. Login to SonarQube using username admin and password admin.

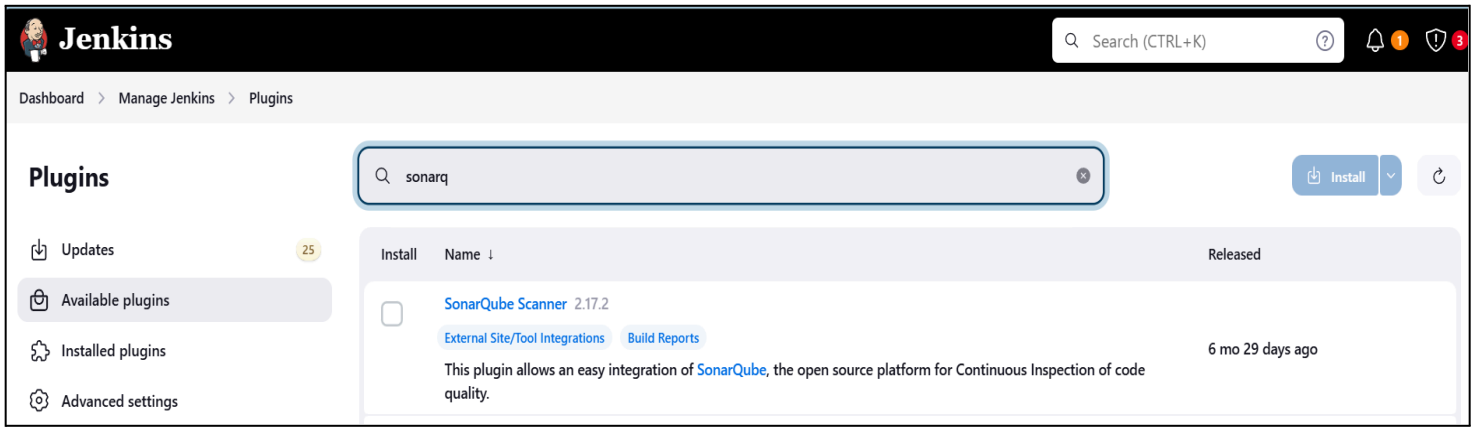


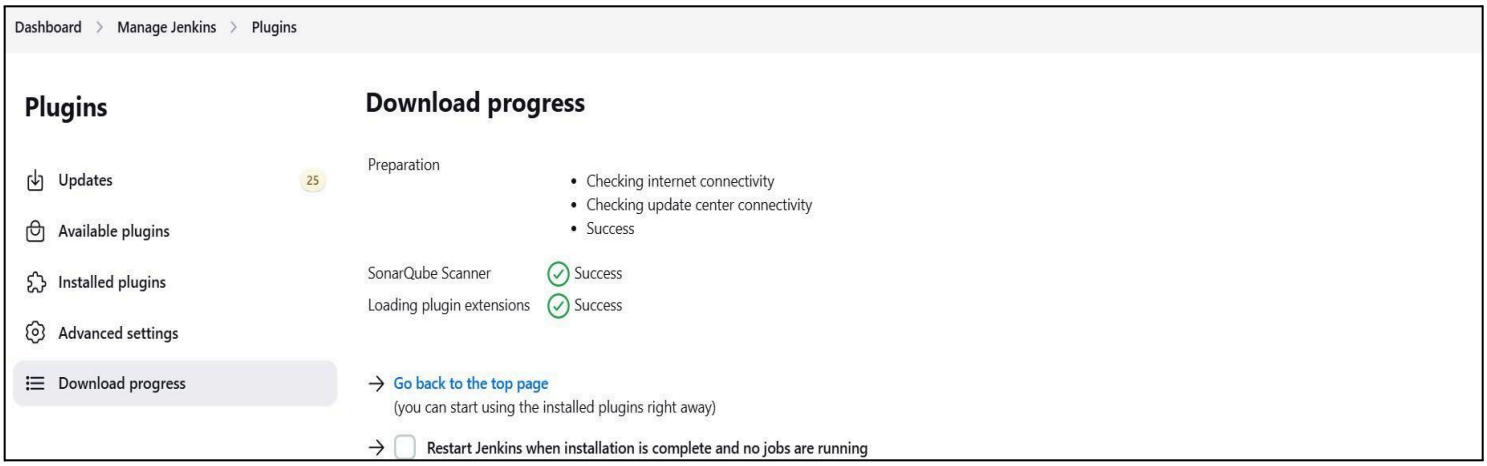
5. Create a manual project in SonarQube with the name sonarqube





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





6. Under Jenkins ‘Manage Jenkins’ then go to ‘system’, scroll and look for **SonarQube Servers**
and enter the details.
Enter the Server Authentication token if needed.

In SonarQube installations: Under **Name** add <project name of sonarqube>,here we have named it as **adv_devops_7_sonarqube**
In **Server URL** Default is **http://localhost:9000**

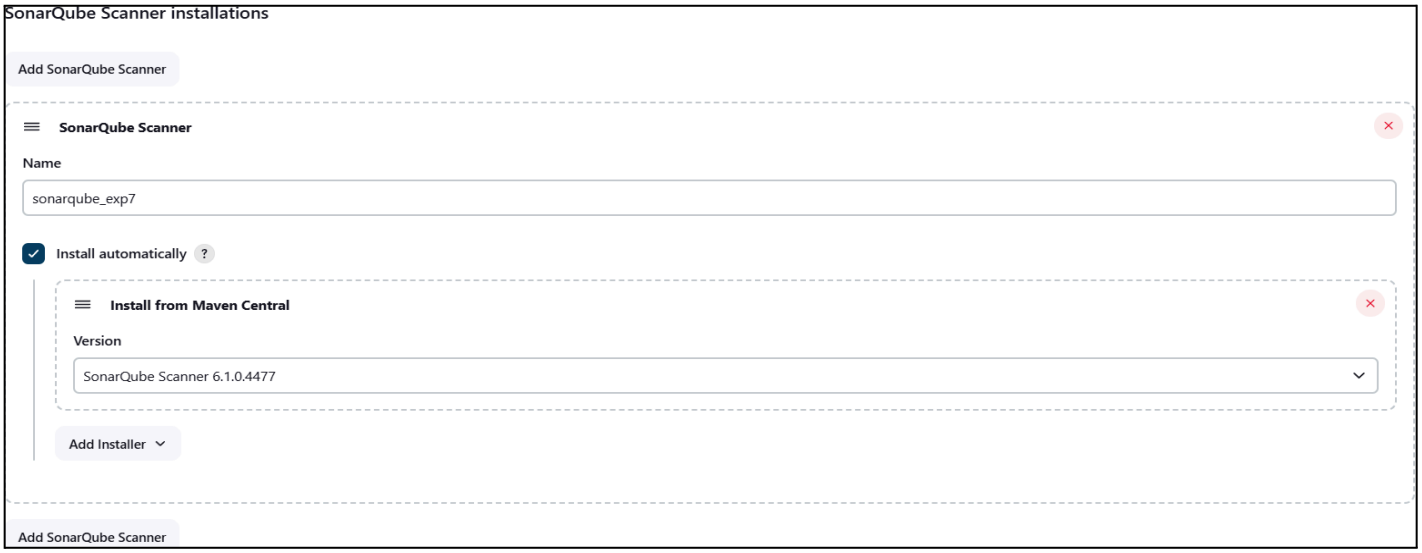

SonarQube servers
If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.
Environment variables
SonarQube installations
List of SonarQube installations
Name
adv_devops_7_sonarqube
Server URL
Default is http://localhost:9000
https://localhost:9000
Server authentication token
SonarQube authentication token. Mandatory when anonymous access is disabled.
- none -
Add
Advanced

7. Search for SonarQube Scanner under Global Tool Configuration.

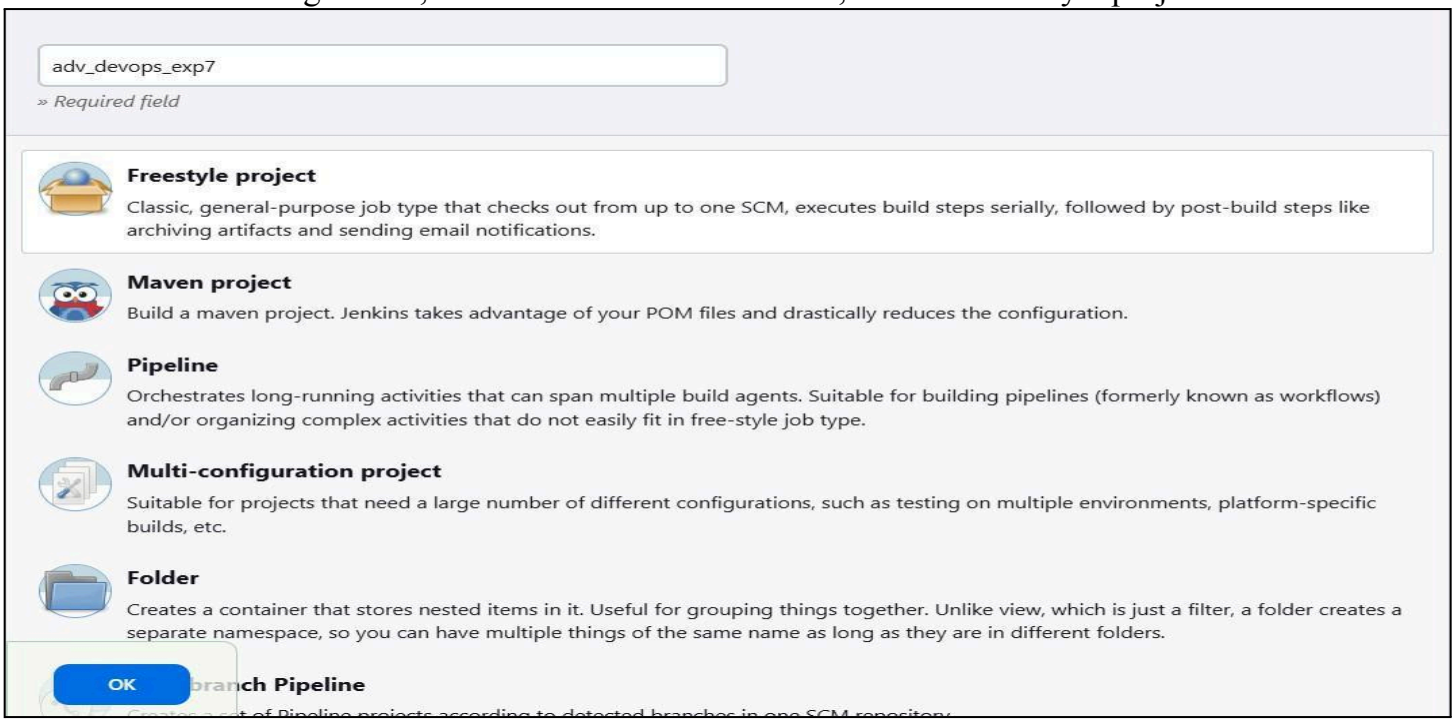
Choose the latest configuration and choose Install automatically.

Dashboard > Manage Jenkins > Tools

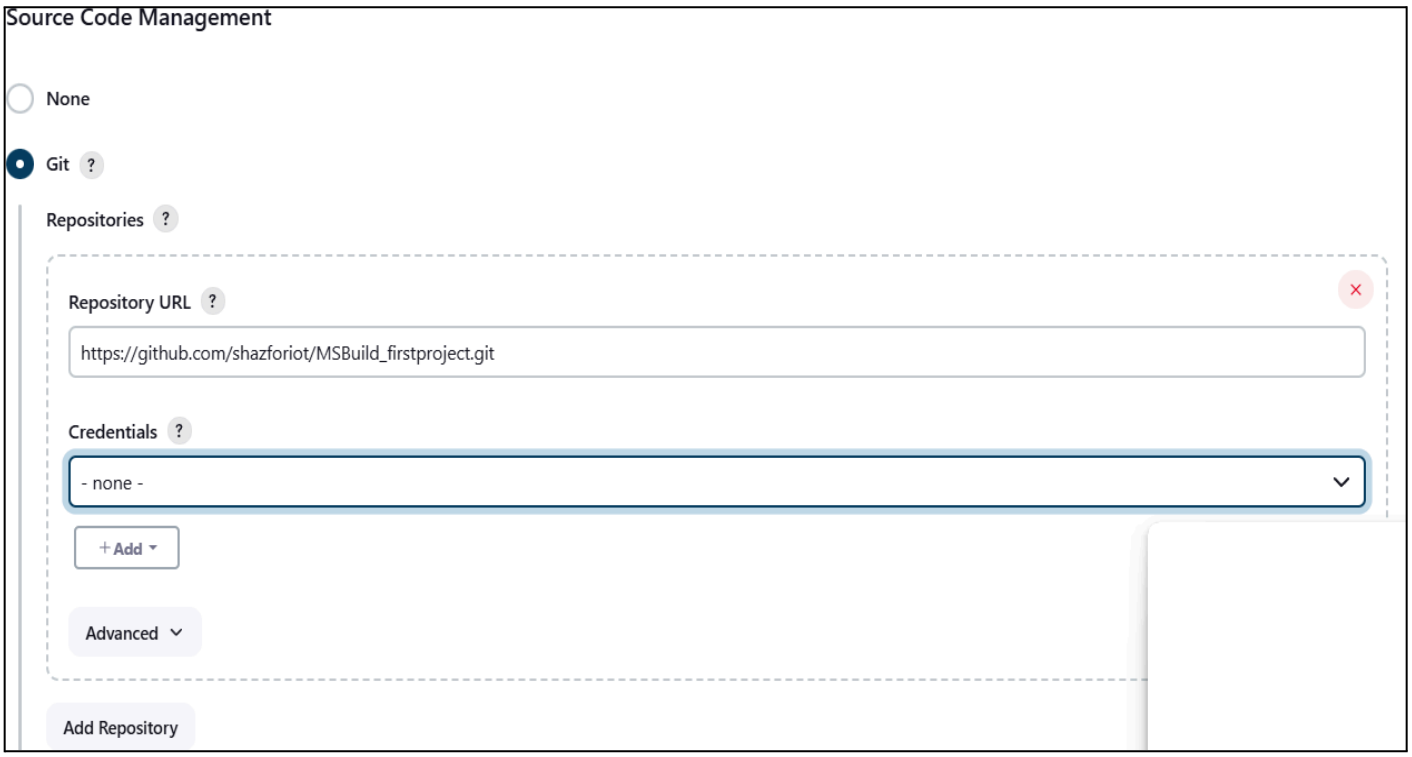
Check the “Install automatically” option. → Under name any name as identifier → Check the “Install automatically” option.



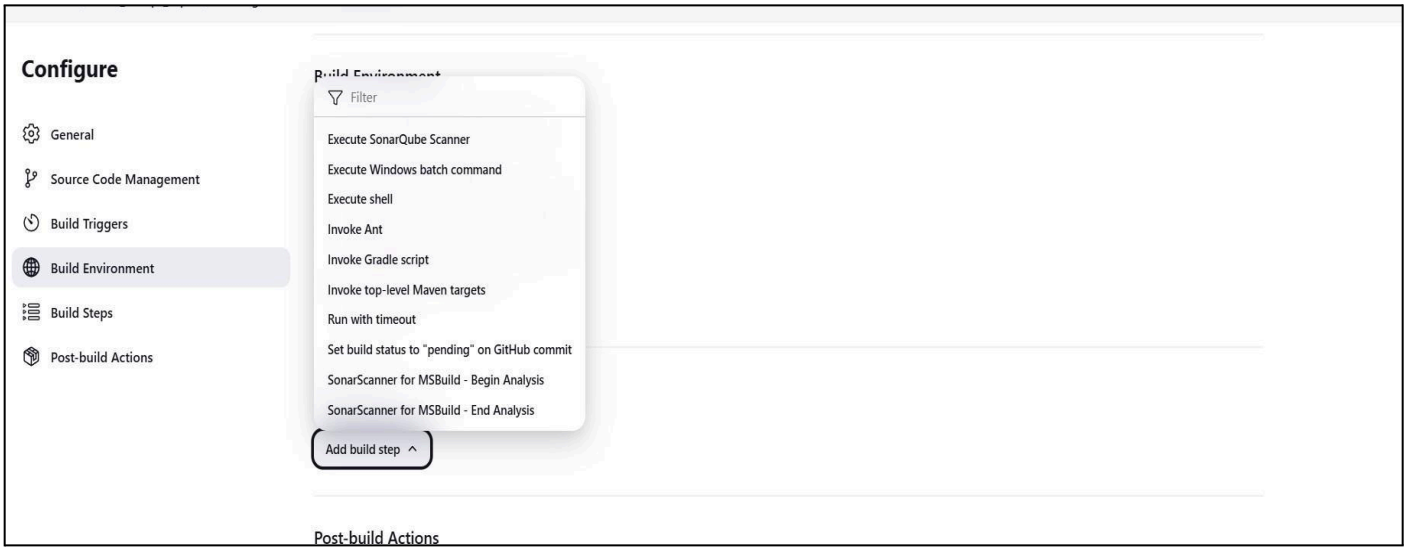
8. After the configuration, create a New Item in Jenkins, choose a freestyle project.



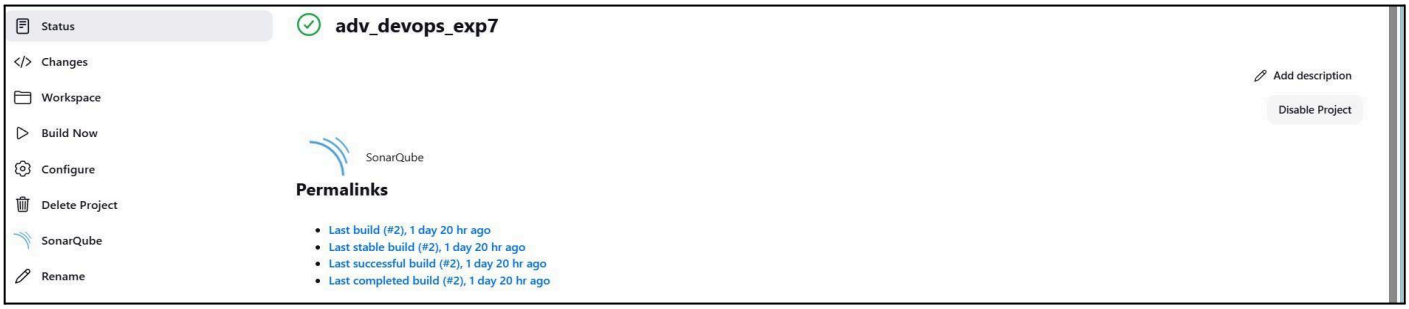
9. Choose this GitHub repository in Source Code Management.
https://github.com/shazforiot/MSBuild_firstproject.git
It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.



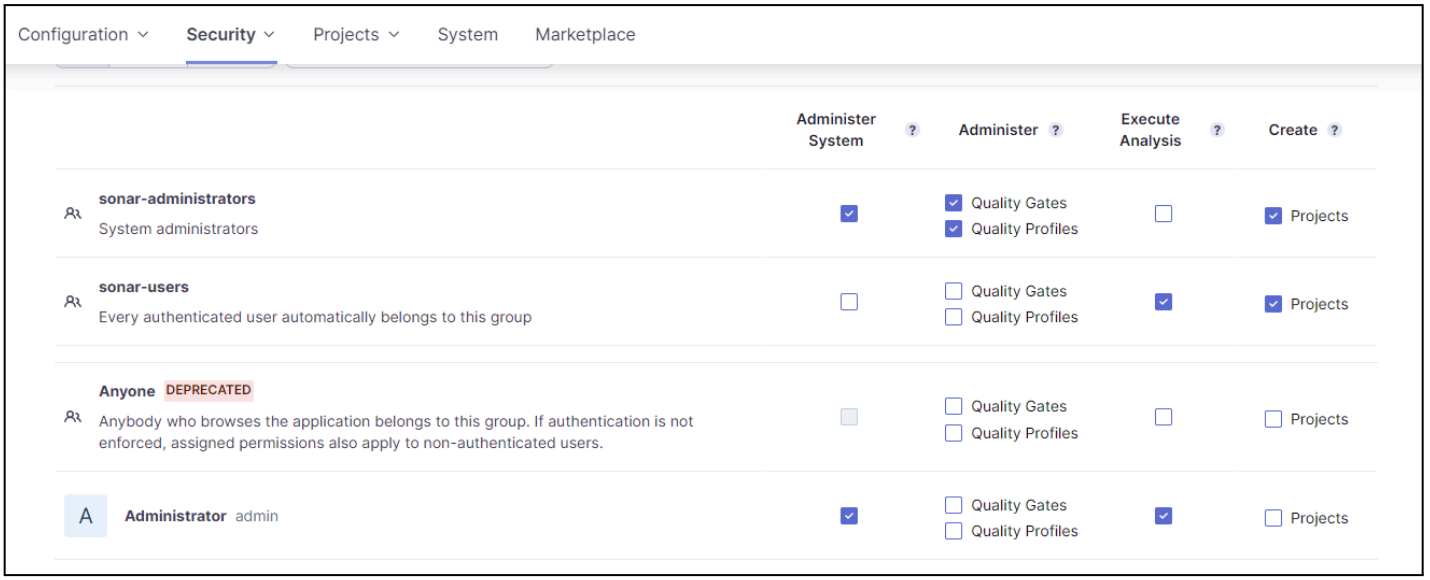
10. Under **Select project → Configuration → Build steps → Execute SonarQube Scanner**, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.



Then save



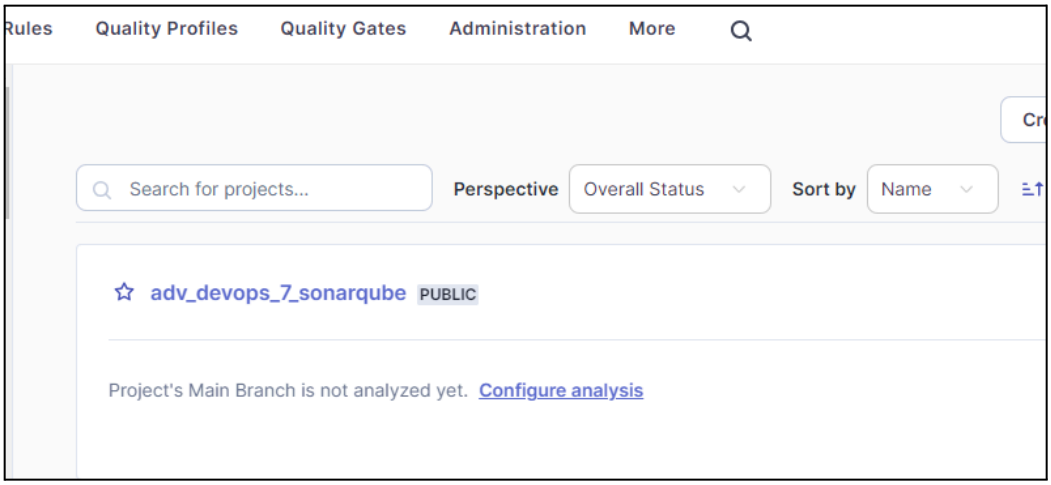
11. Go to http://localhost:9000/<user_name>/permissions and allow Execute Permissions to the Admin user



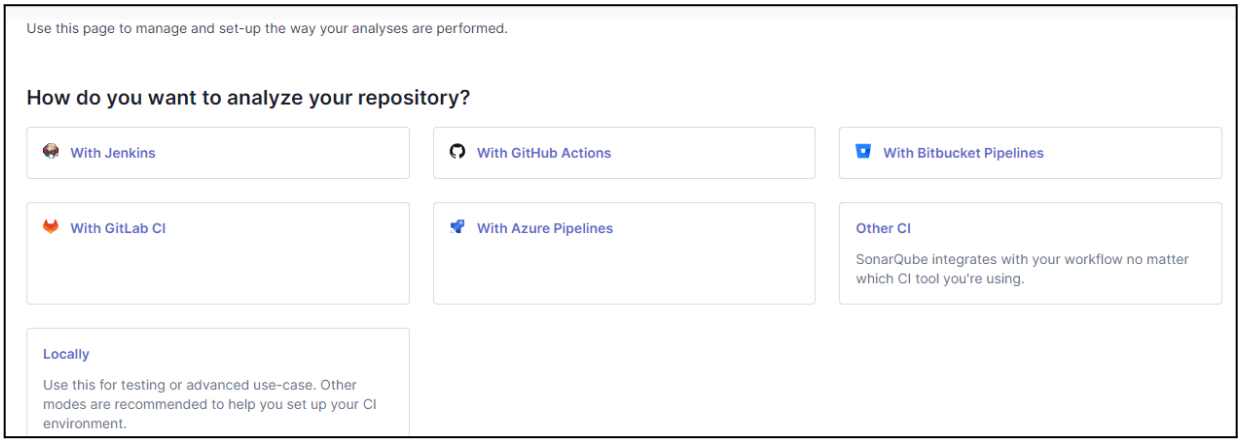
IF CONSOLE OUTPUT FAILED:

Step 1: Generate a New Authentication Token in SonarQube

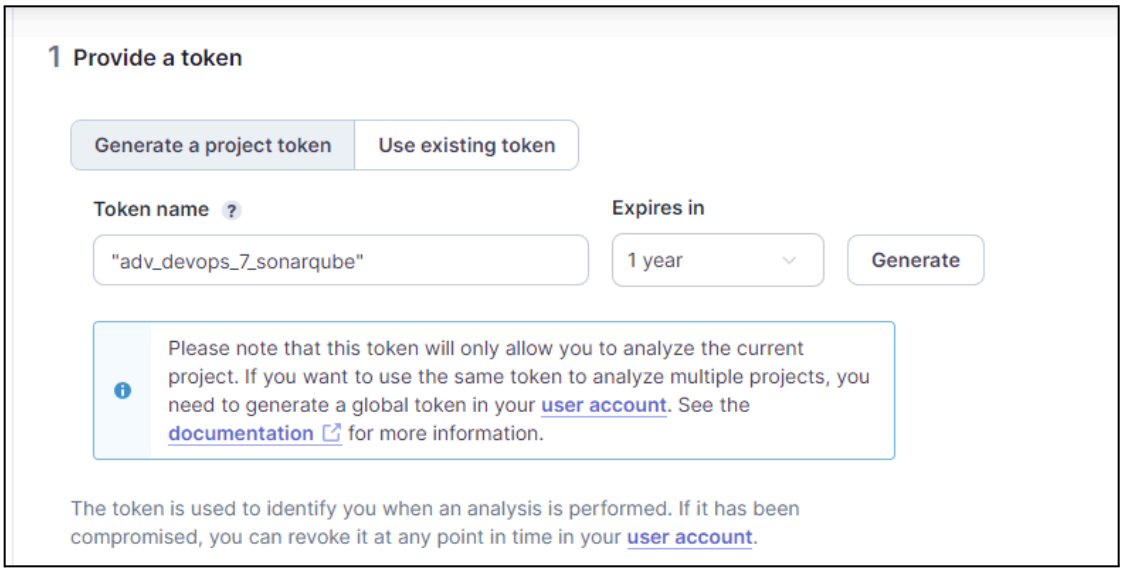
1. **Login to SonarQube:**
   - Open your browser and go to **http://localhost:9000**.
   - Log in with your admin credentials (default username is **admin**, and the password is either **admin** or your custom password if it was changed).
2. **Generate a New Token:**
   - Go to the project that you have created on SonarQube.



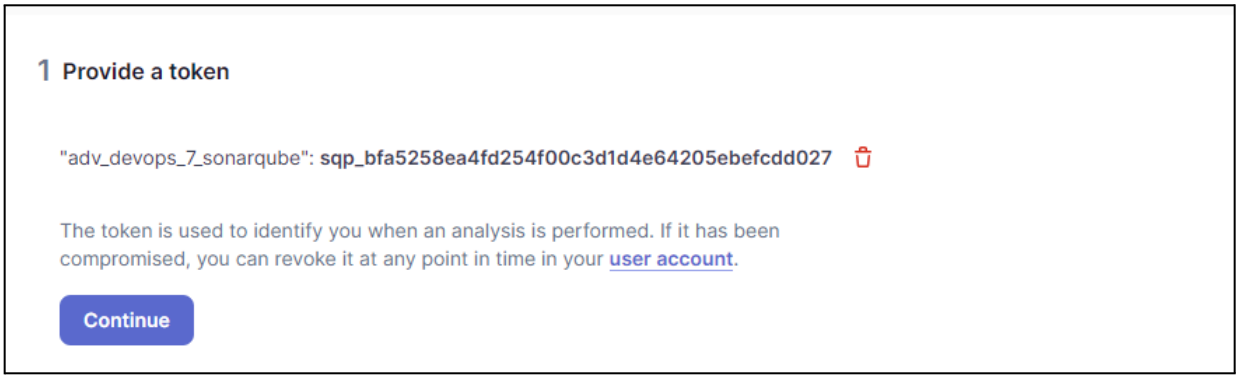
   - Click on **Locally**



   - Further,Generate a Project token with the following details and click on generate.

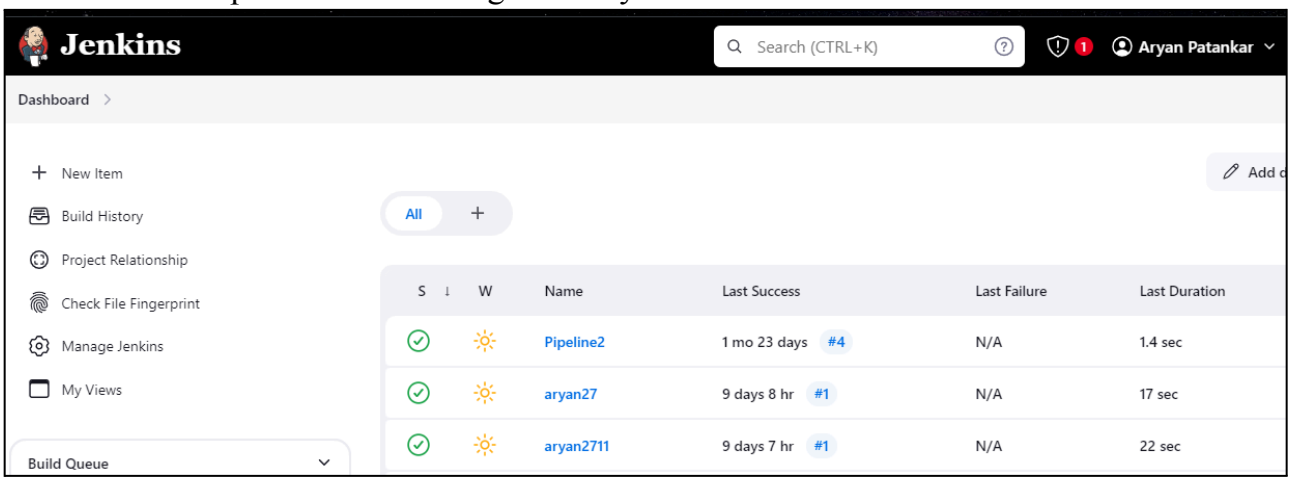


- Copy the token you get here and save it securely as we would need it in Jenkins.



Step 2: Update the Token in Jenkins

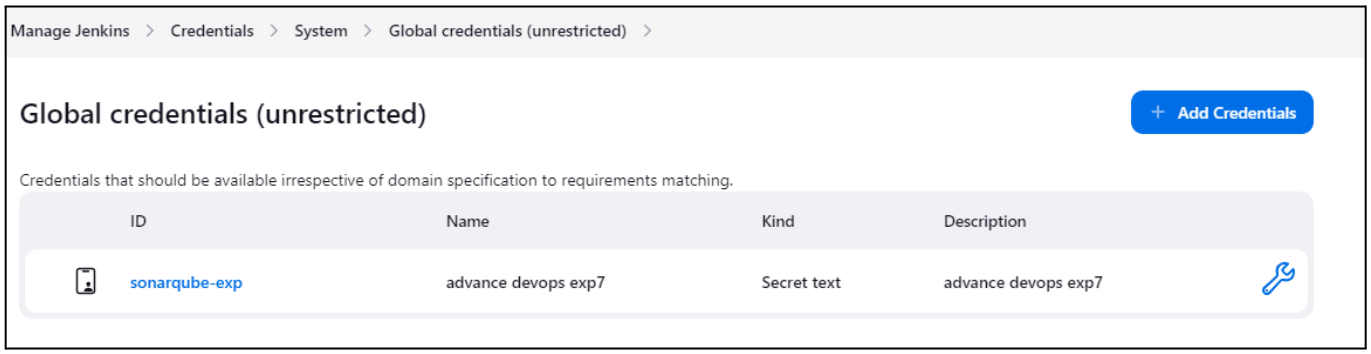
1. **Go to Jenkins Dashboard:**
   - Open Jenkins and log in with your credentials.



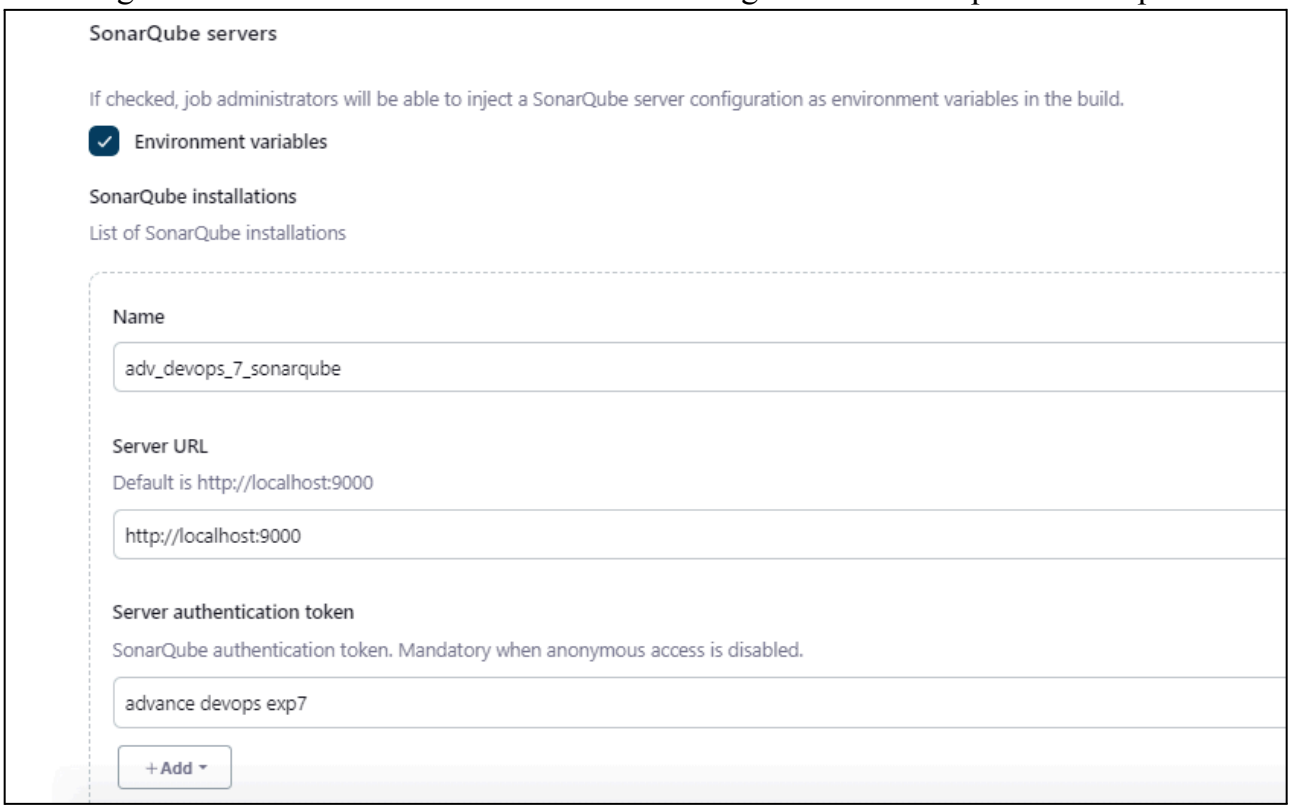
2. Go to Dashboard—>Manage Jenkins—>Credentials

3. Click on **global** under the domains part of Stores scoped to Jenkins section.Further click on add credentials.Proceed with the following details.Make sure to copy the token generated earlier in sonarqube and give any suitable name as the ID.

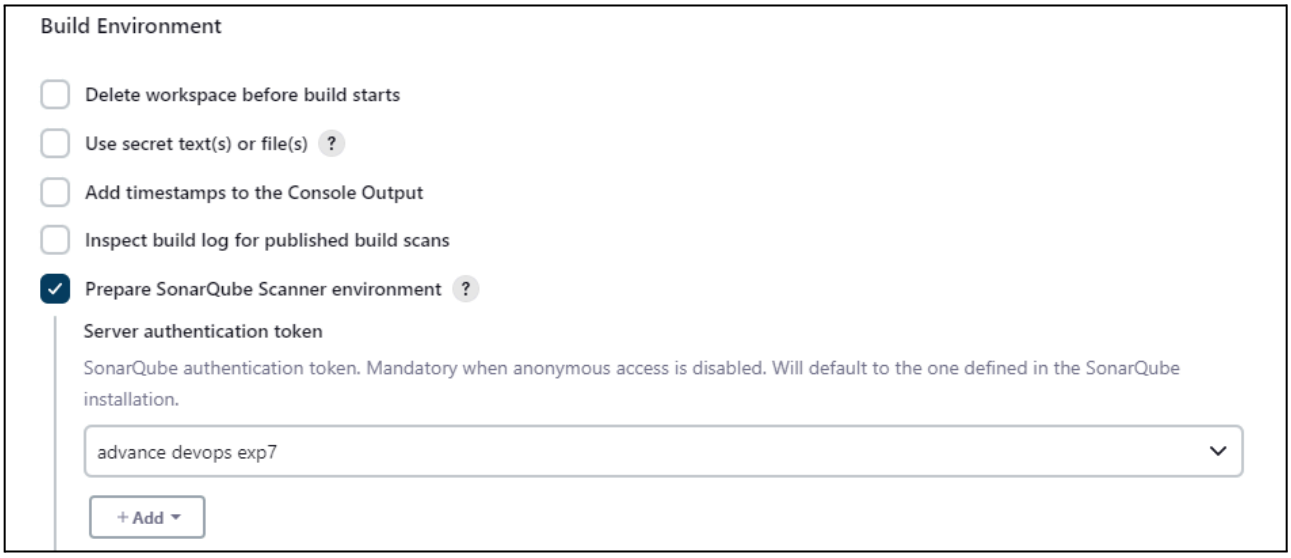
4.After clicking on create we see that the given token has been added in Jenkins credentials.

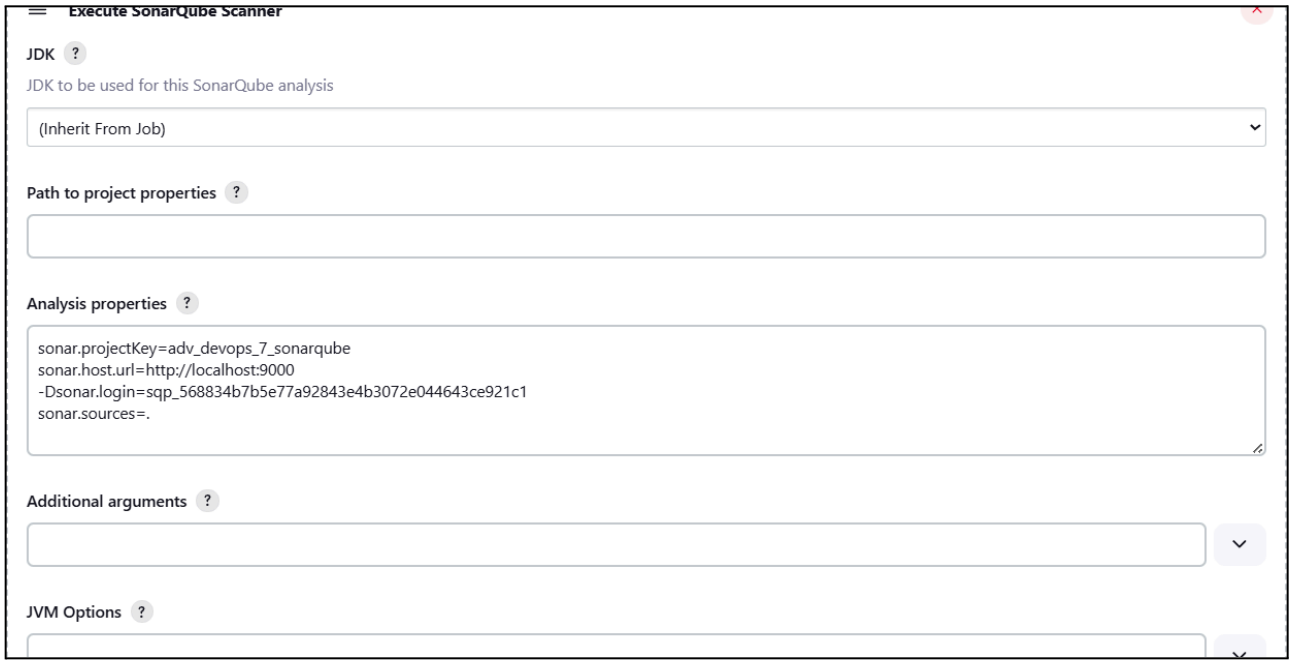


5.Now go to **Manage Jenkins—>System—>SonarQube servers** and proceed with the following details. Reference the authentication token generated in the previous step.

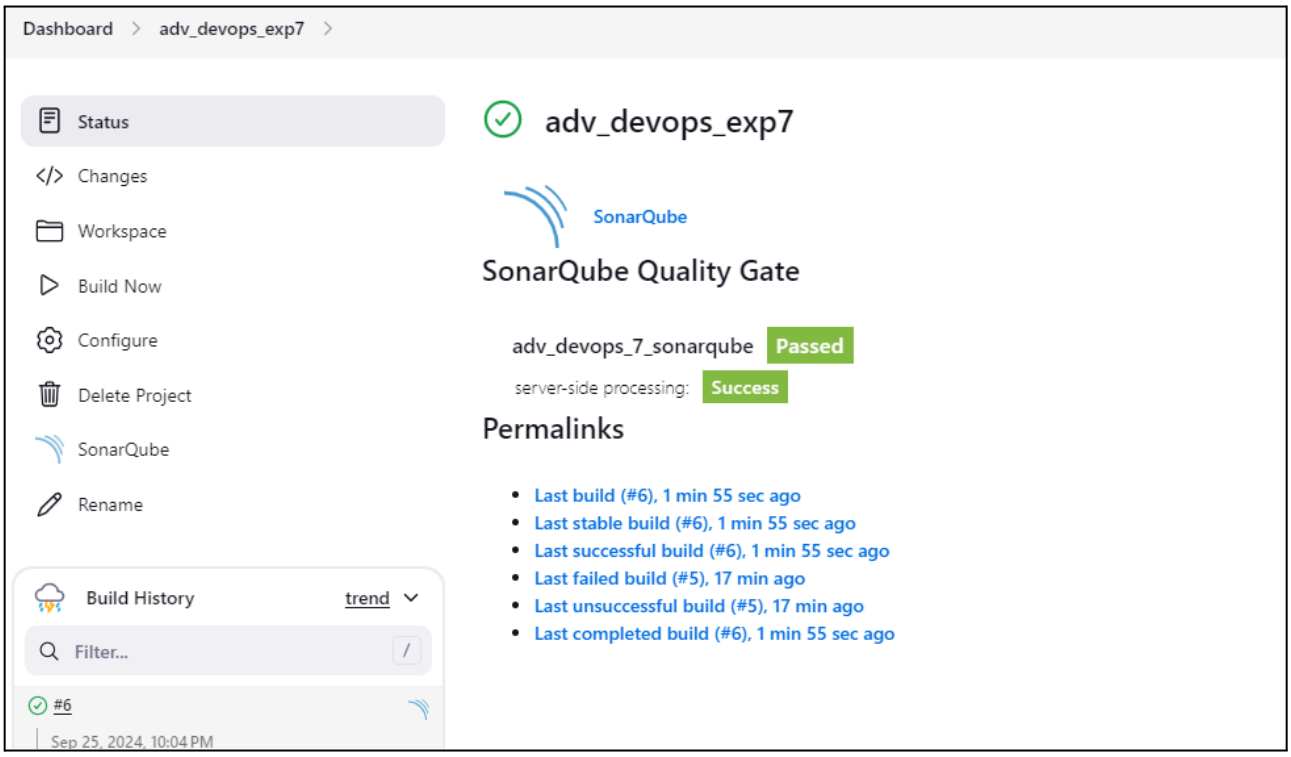


6.Check the SonarQube Scanner Environment and add the server authentication token

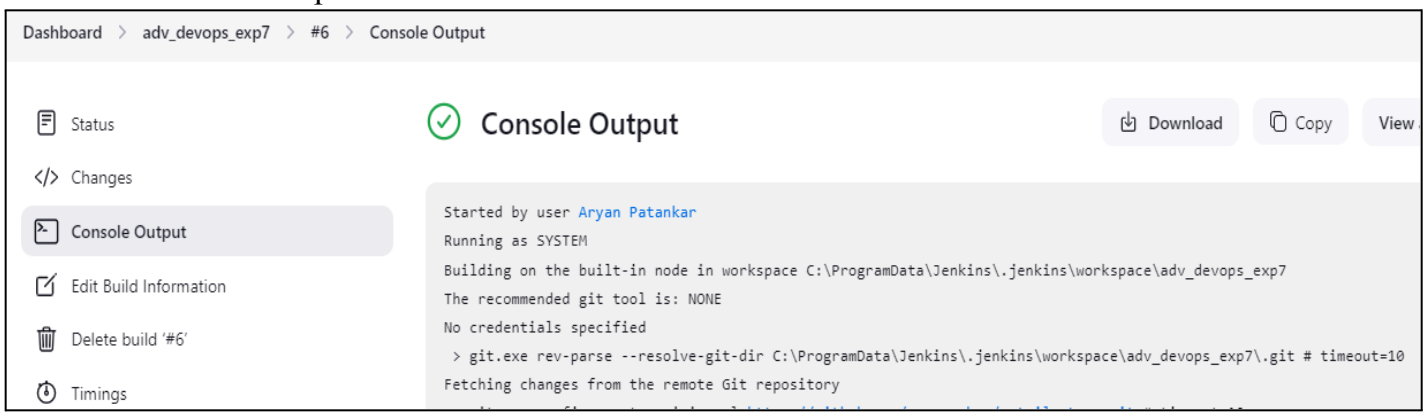




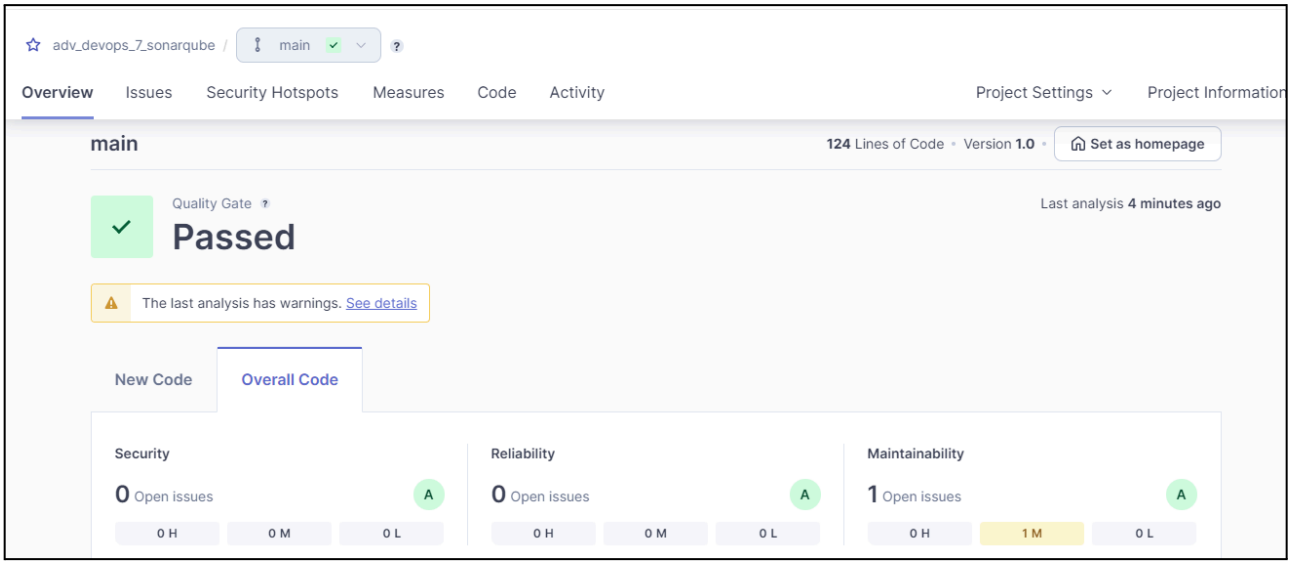
12.Run the Jenkins build.



Check the console Output



13. Once the build is complete, check project on SonarQube



In this way, we have integrated Jenkins with SonarQube for SAST.

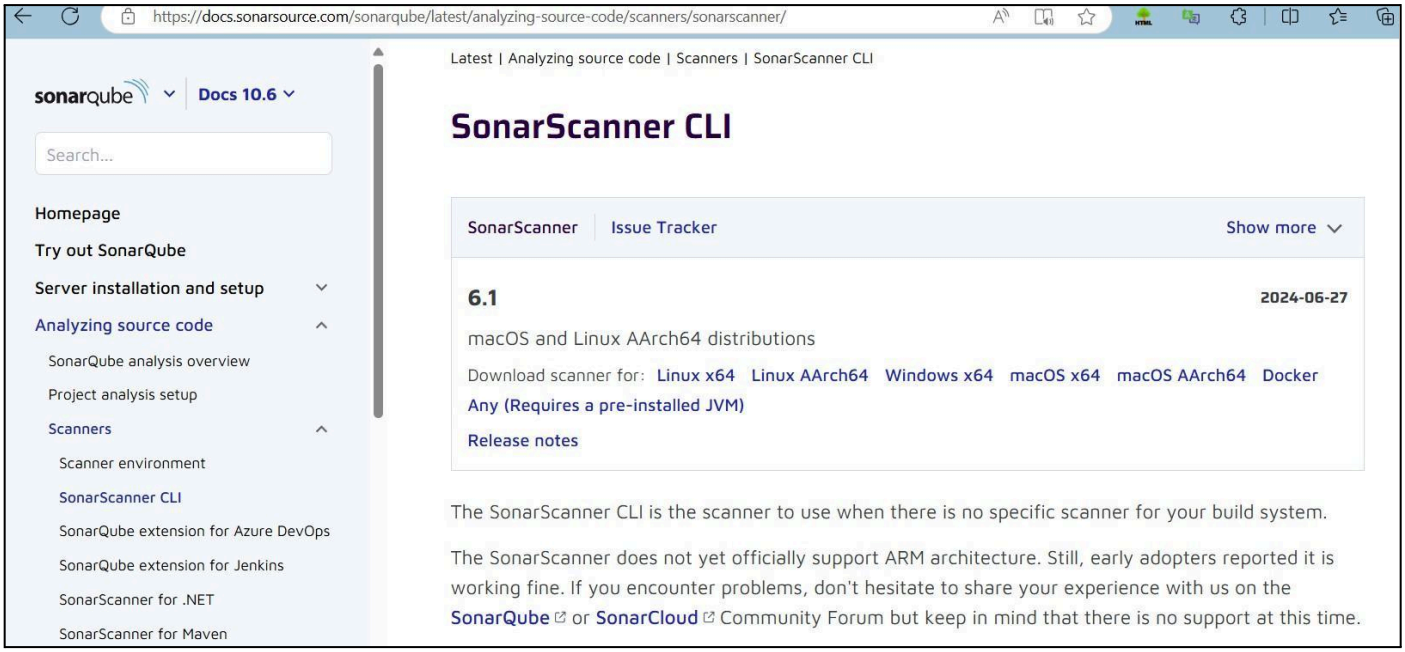
ADVANCE DEVOPS EXP 8

Name:Aryan Anil Patankar
Class:D15A
Roll No:34

Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

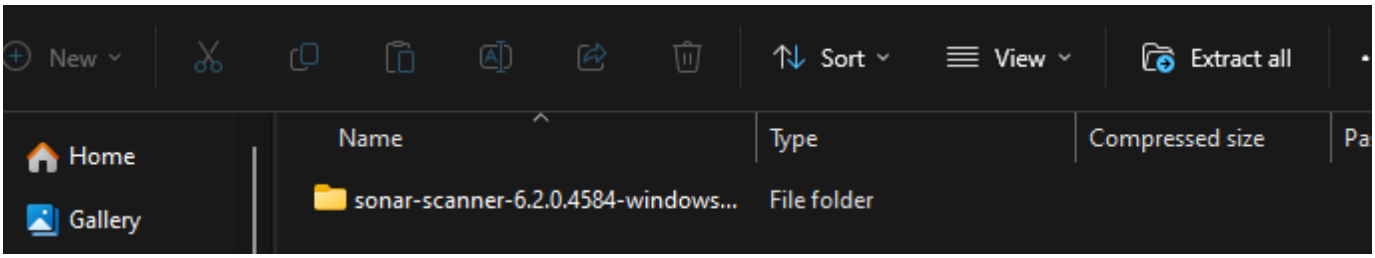
Step 1: Download sonar scanner

https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner/



ner/ Visit this link and download the sonarqube scanner CLI.

Extract the downloaded zip file in a folder.

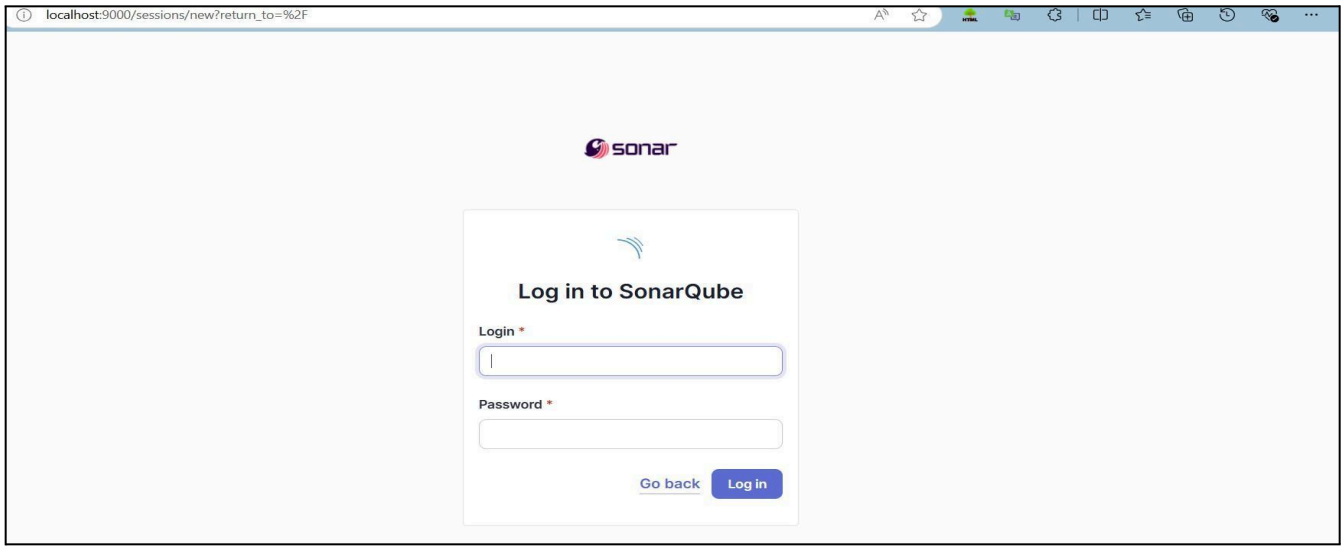


1. Install sonarqube image

Command: **docker pull sonarqube**

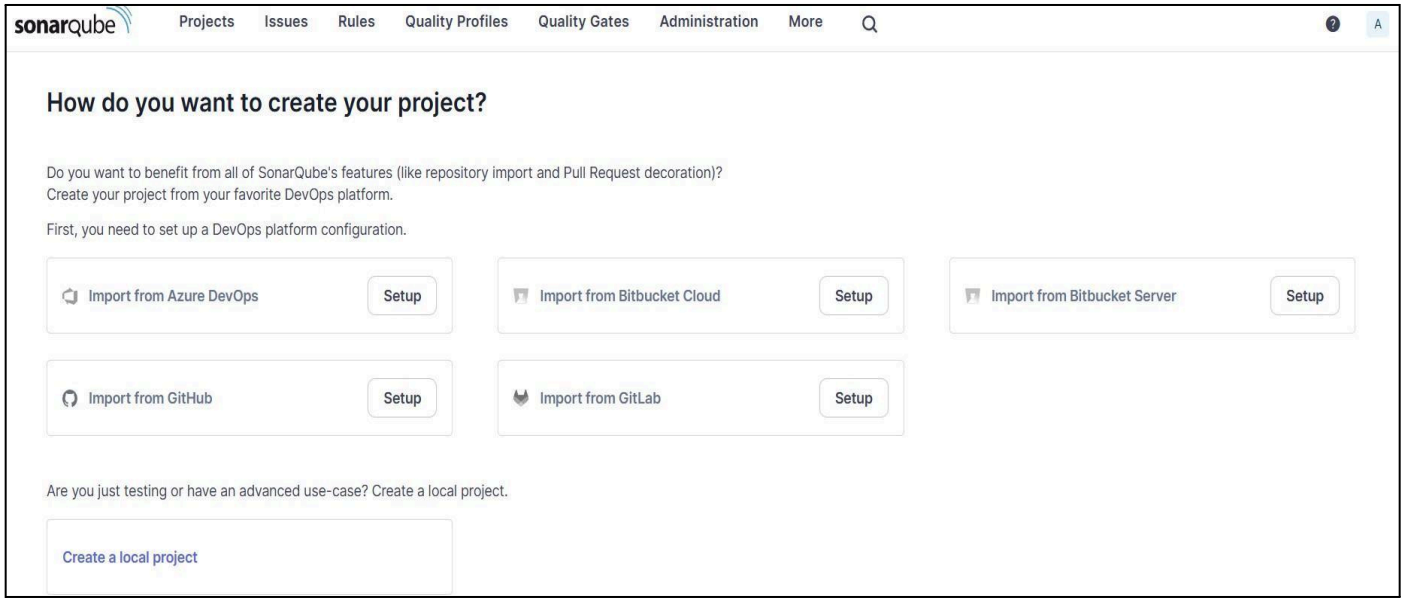
```
C:\Windows\System32>docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Image is up to date for sonarqube:latest
docker.io/library/sonarqube:latest
```

2. Once the container is up and running, you can check the status of

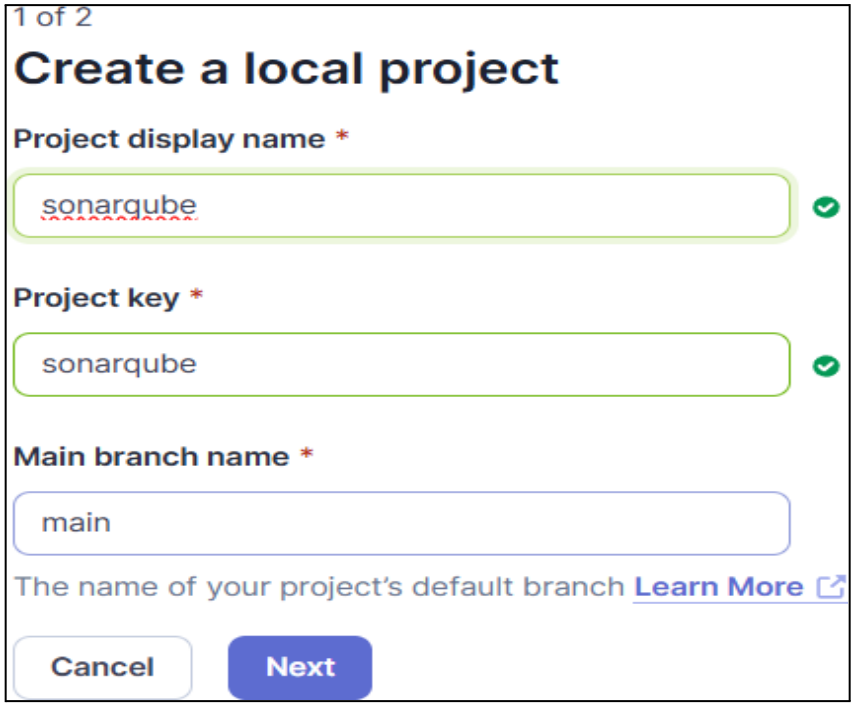


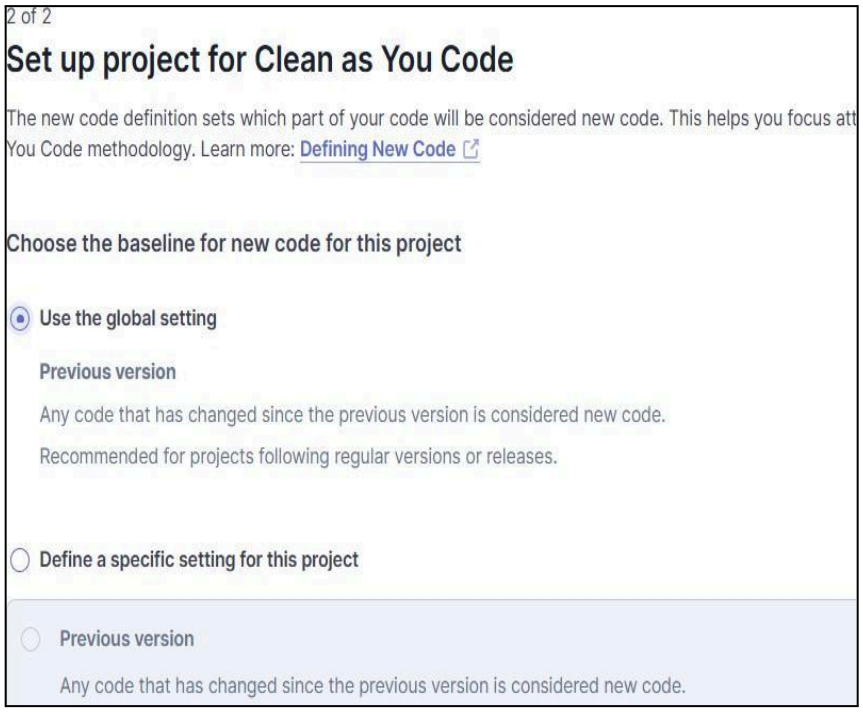
SonarQube at localhost port 9000.

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

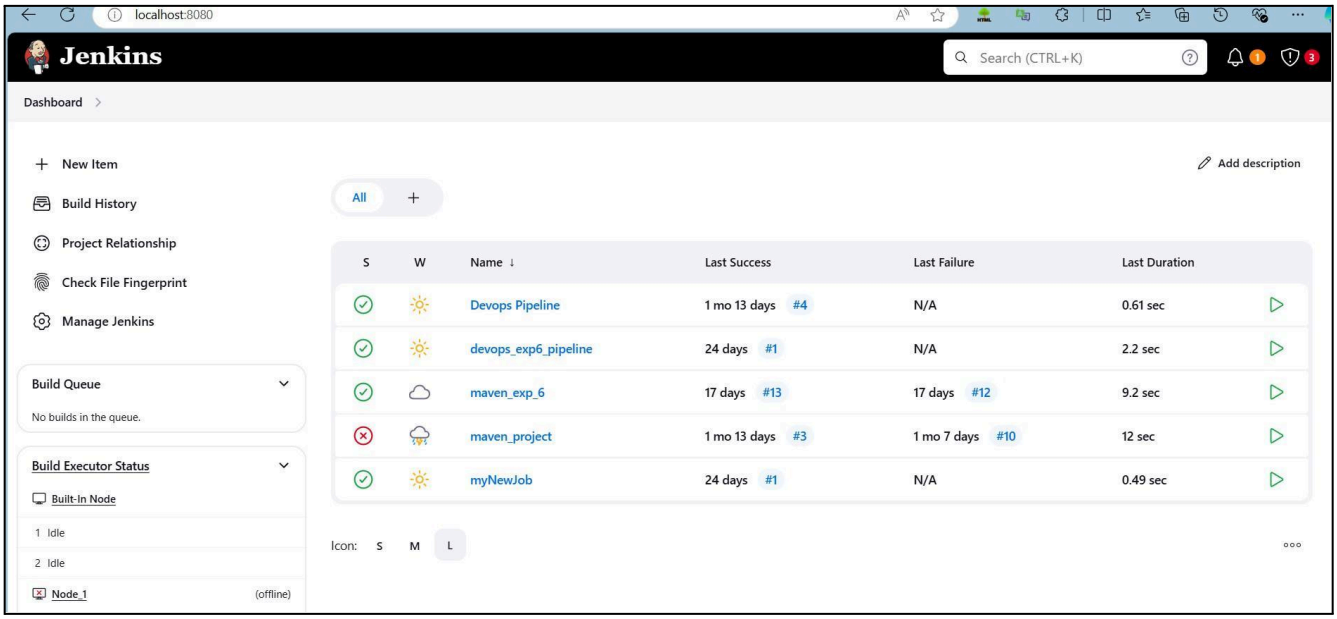


4. Create a manual project in SonarQube with the name sonarqube

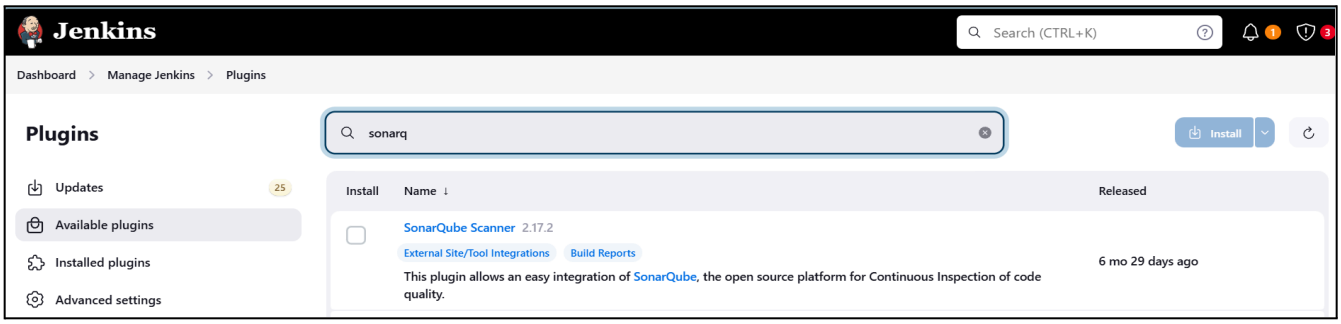


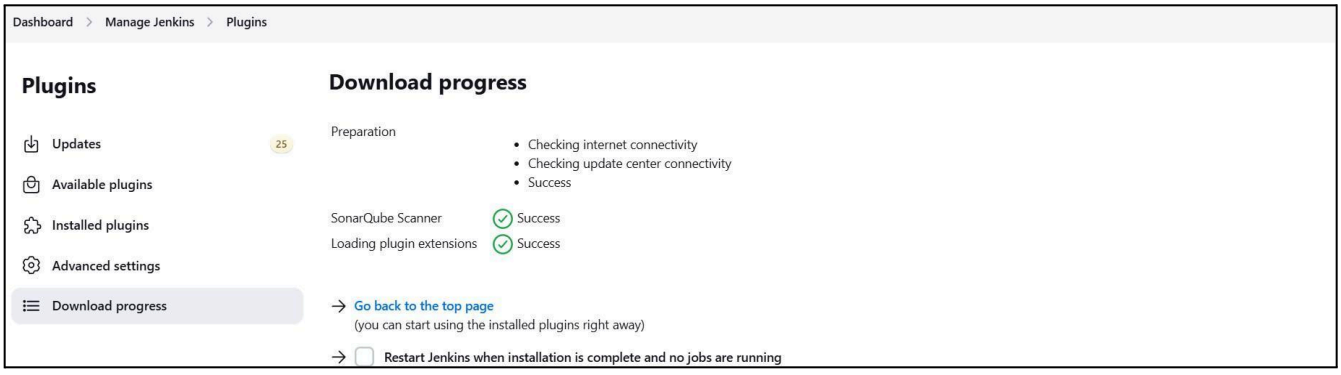


5. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.



6. Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.





7. Under Jenkins ‘Manage Jenkins’ then go to ‘system’, scroll and look for **SonarQube Servers**

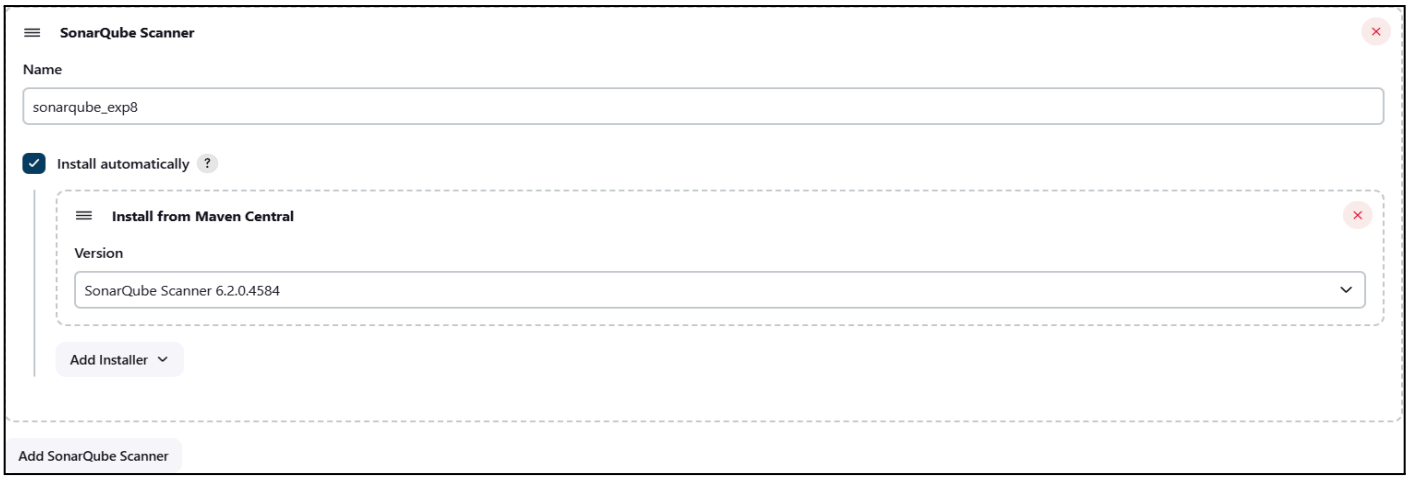
and enter the details.
Enter the Server Authentication token if needed.

In SonarQube installations: Under **Name** add <project name of sonarqube> for me **adv_devops_7_sonarqube**
In **Server URL** Default is **http://localhost:9000**

8. Search for SonarQube Scanner under Global Tool Configuration.
Choose the latest configuration and choose Install automatically.

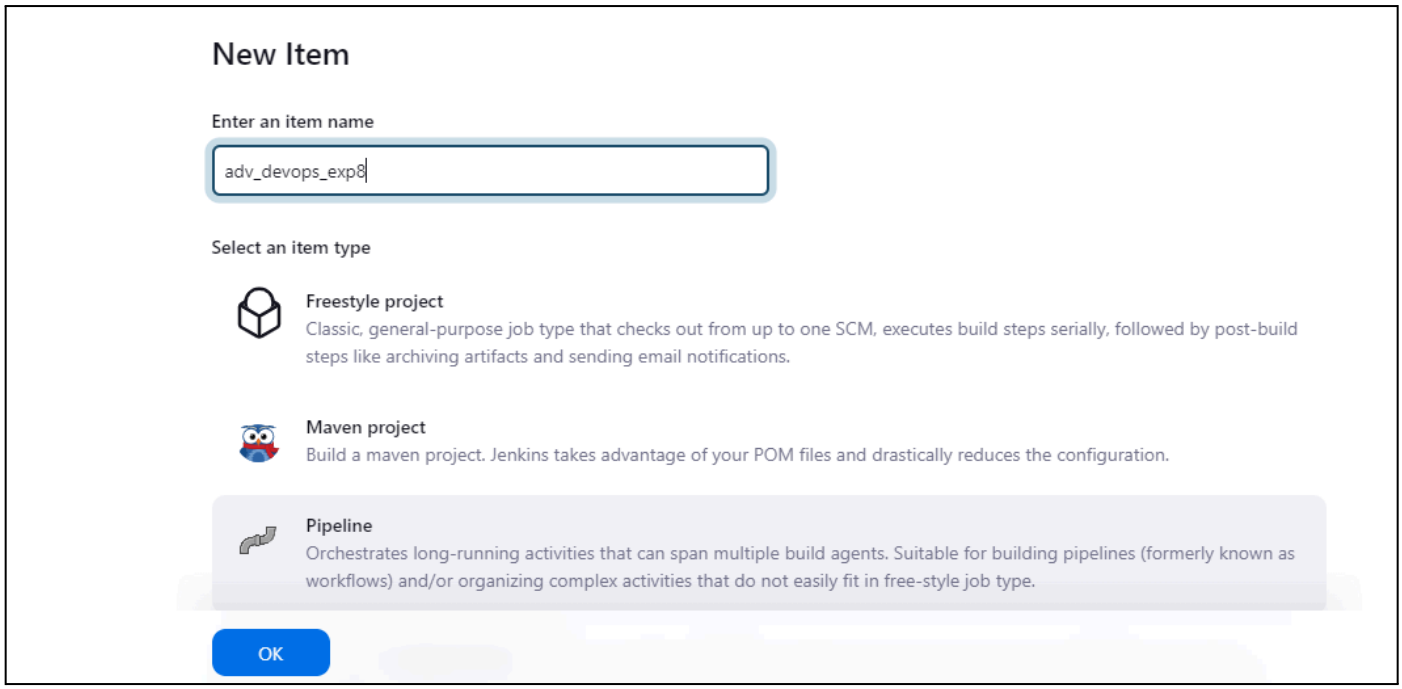
Dashboard > Manage Jenkins > Tools

Check the “Install automatically” option. → Under name any name as identifier → Check



the “Install automatically” option.

9. After configuration, create a New Item → choose a pipeline project.

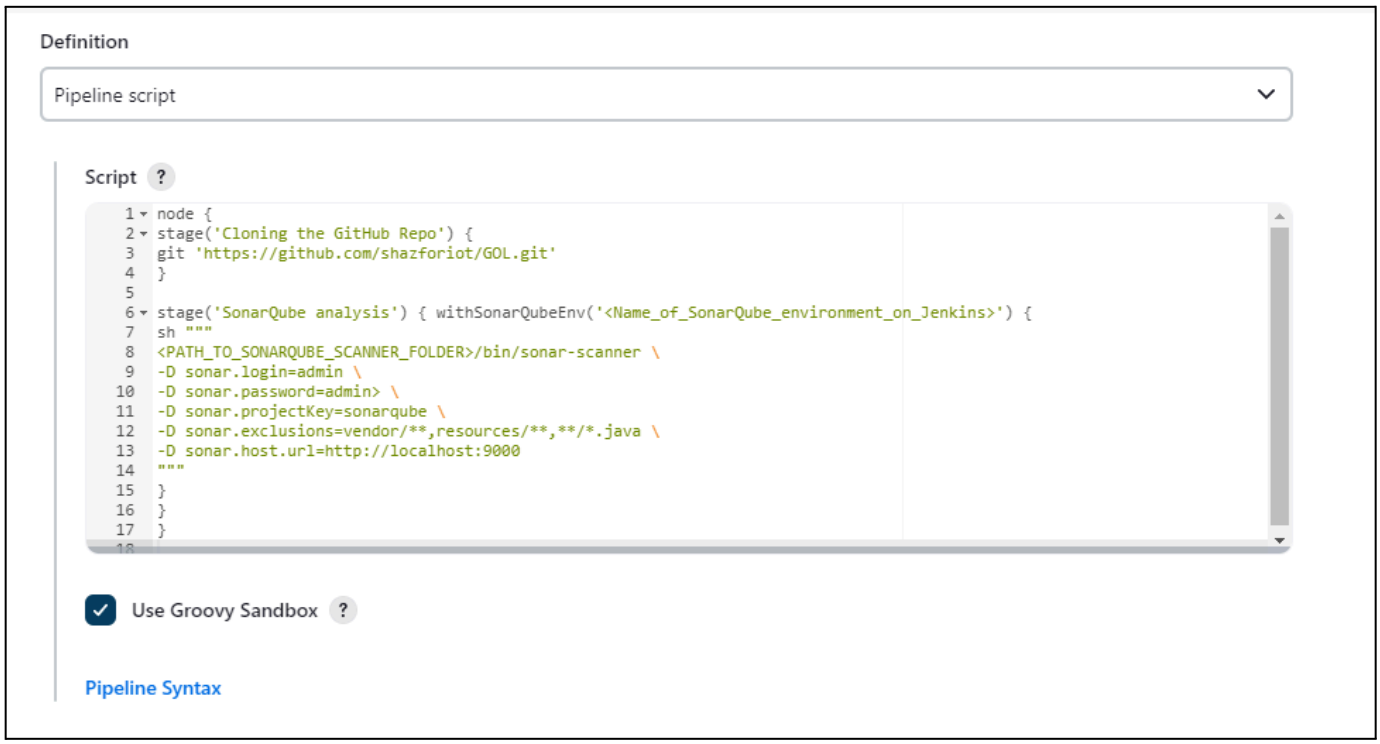


10. Under Pipeline script, enter the following:

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

stage('SonarQube analysis') {
  withSonarQubeEnv('<Name_of_SonarQube_environment_on_Jenki
  ns>') {
    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/)
    """
  }
}
```

}

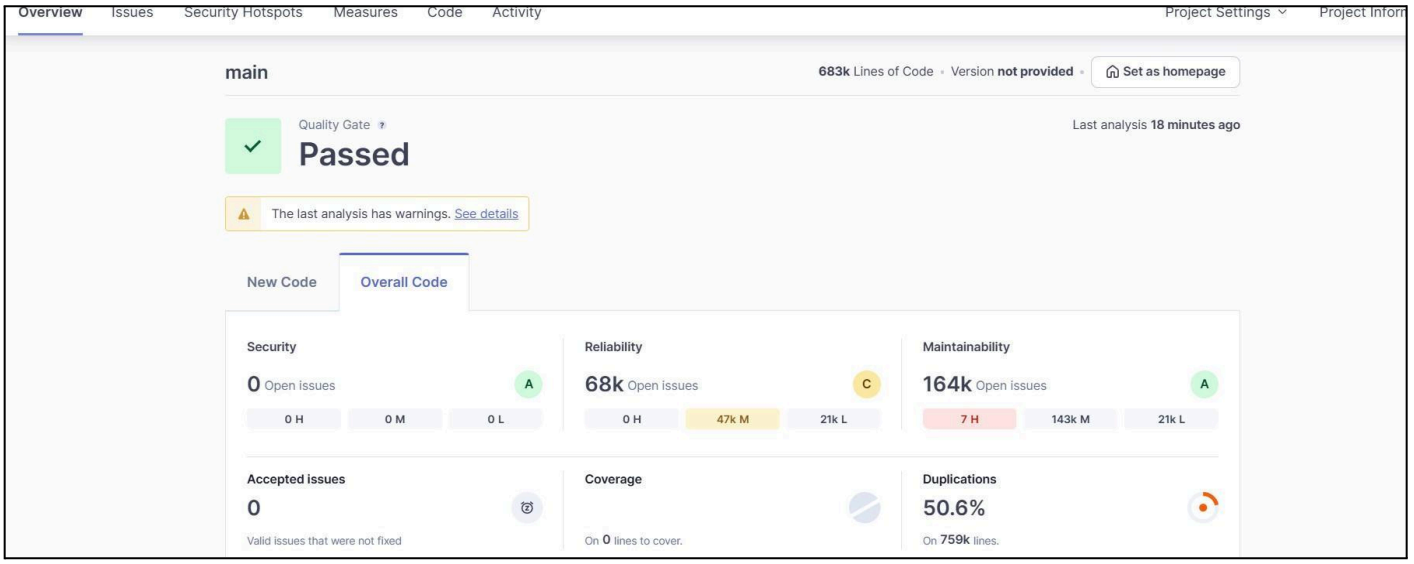
It is a java sample project which has a lot of repetitions and issues that will be detected by SonarQube.



11. Build project

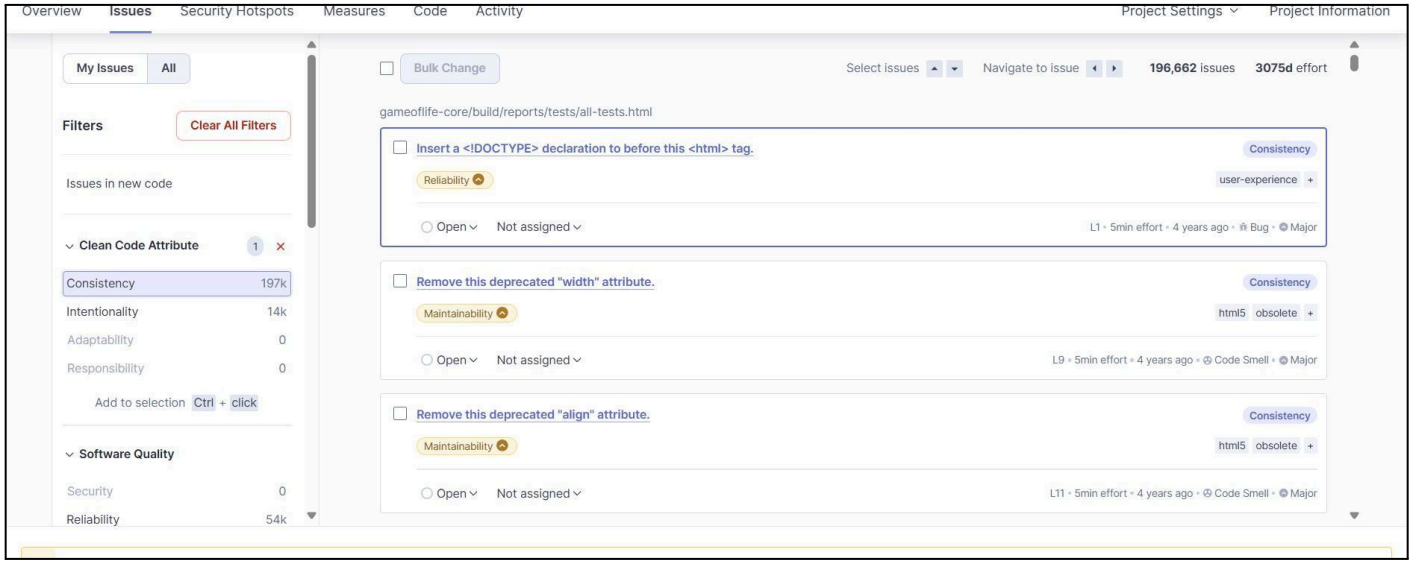
12. Check console

13. Now, check the project in SonarQube

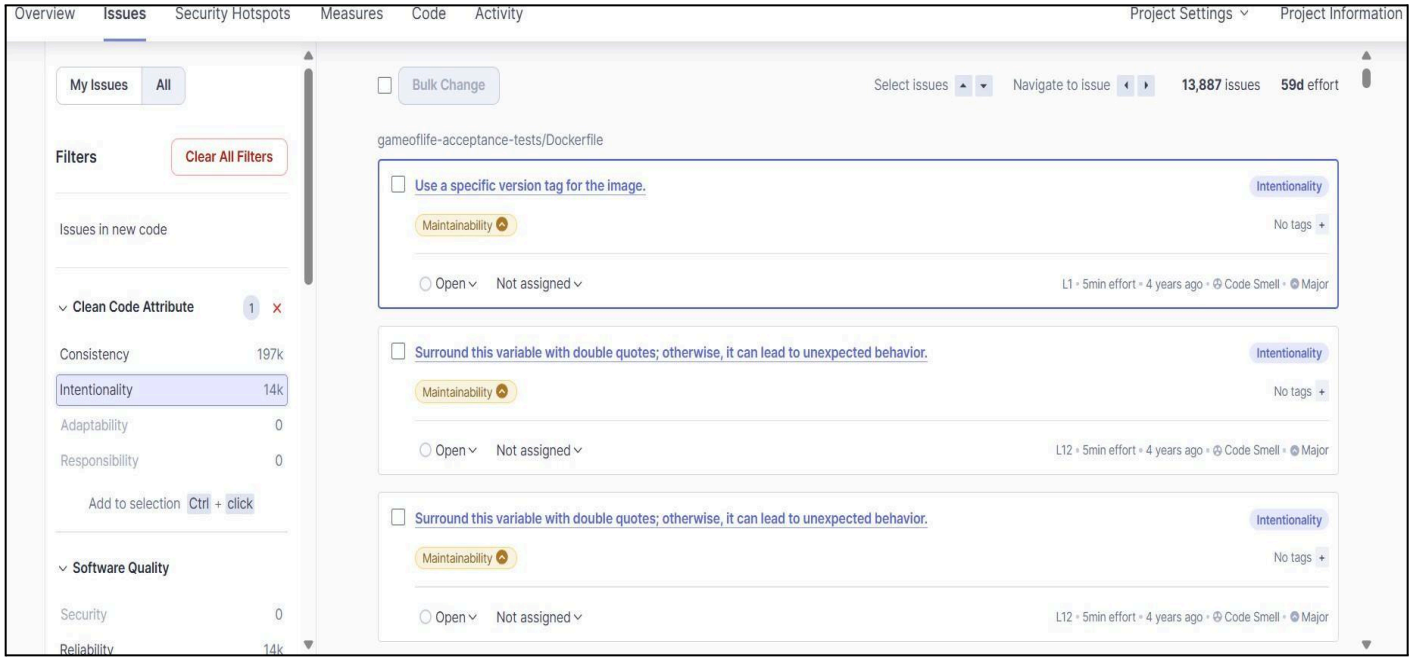


14. Code Problems

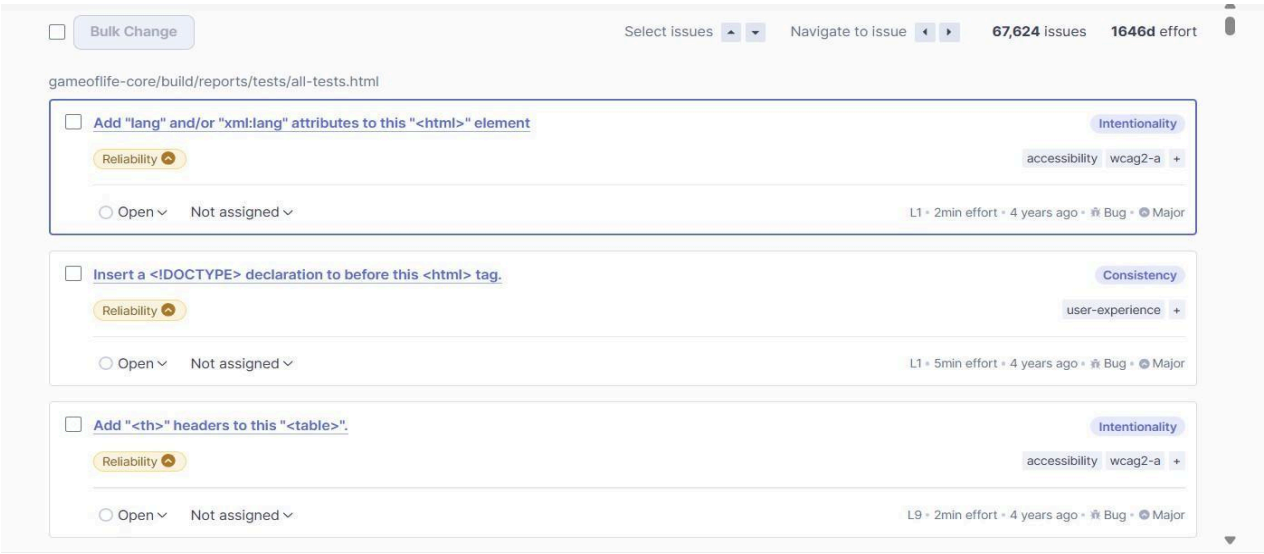
- Consistency



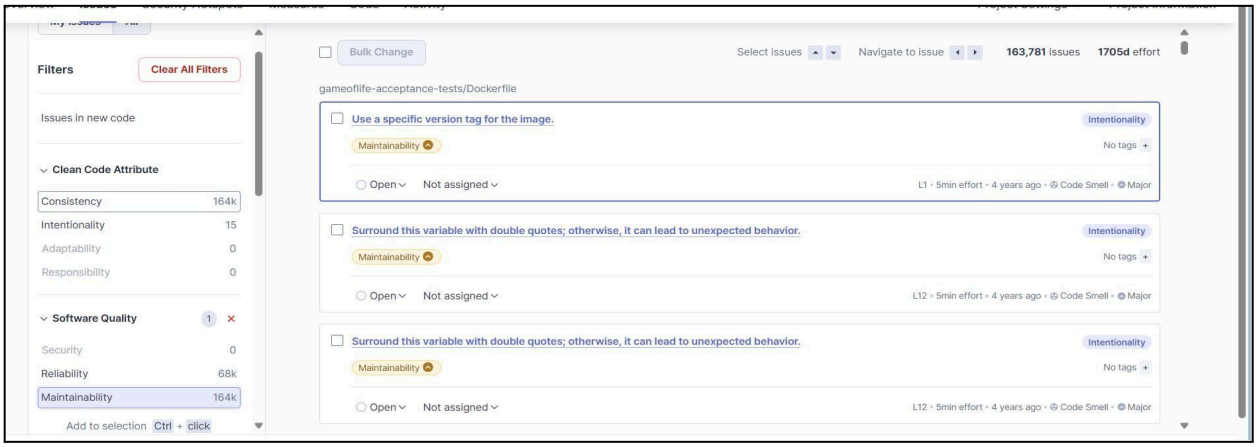
- Intentionality



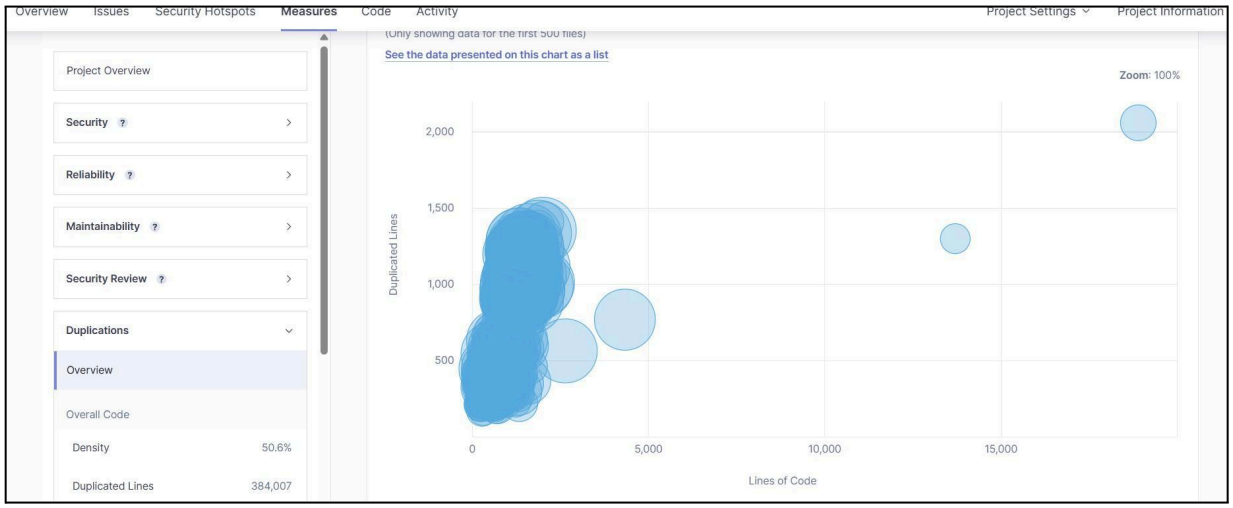
- Bugs



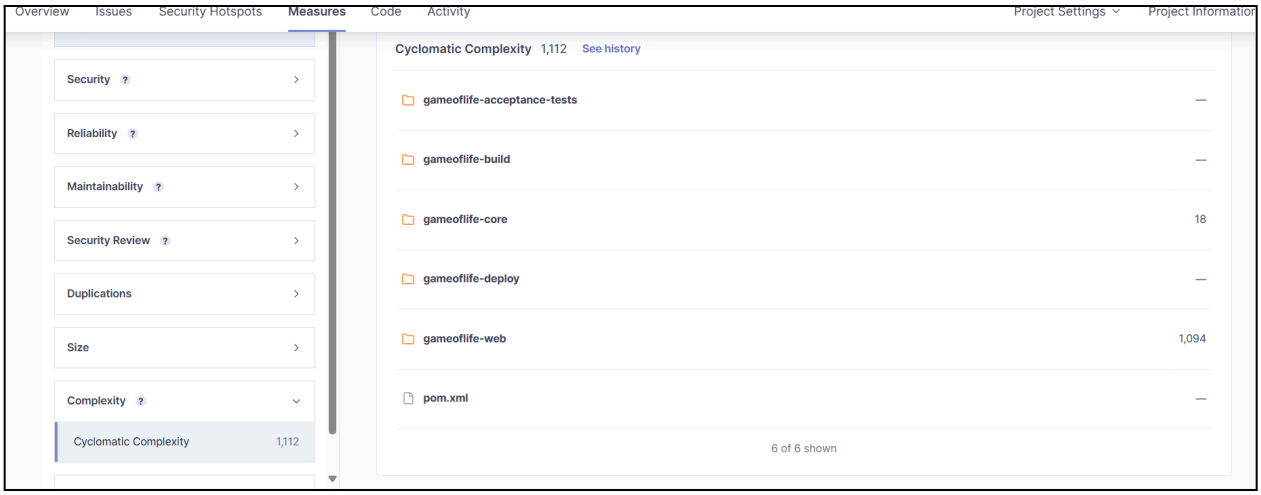
- Code Smells



- Duplications



- Cyclomatic Complexities



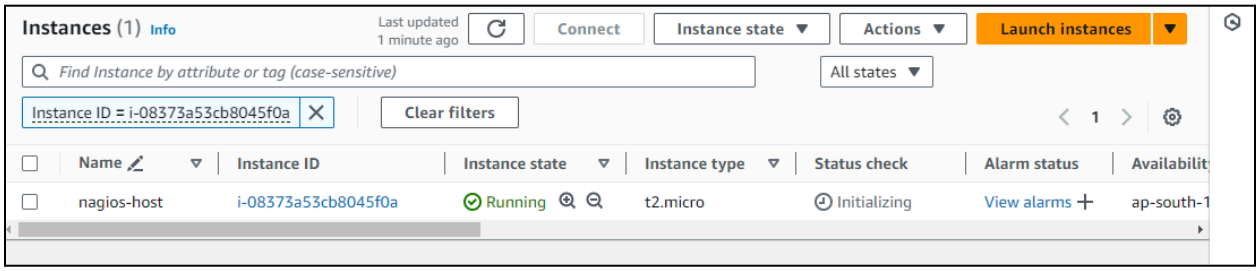
In this way, we have integrated Jenkins with SonarQube for SAST.

ADVANCE DEVOPS EXPERIMENT 9

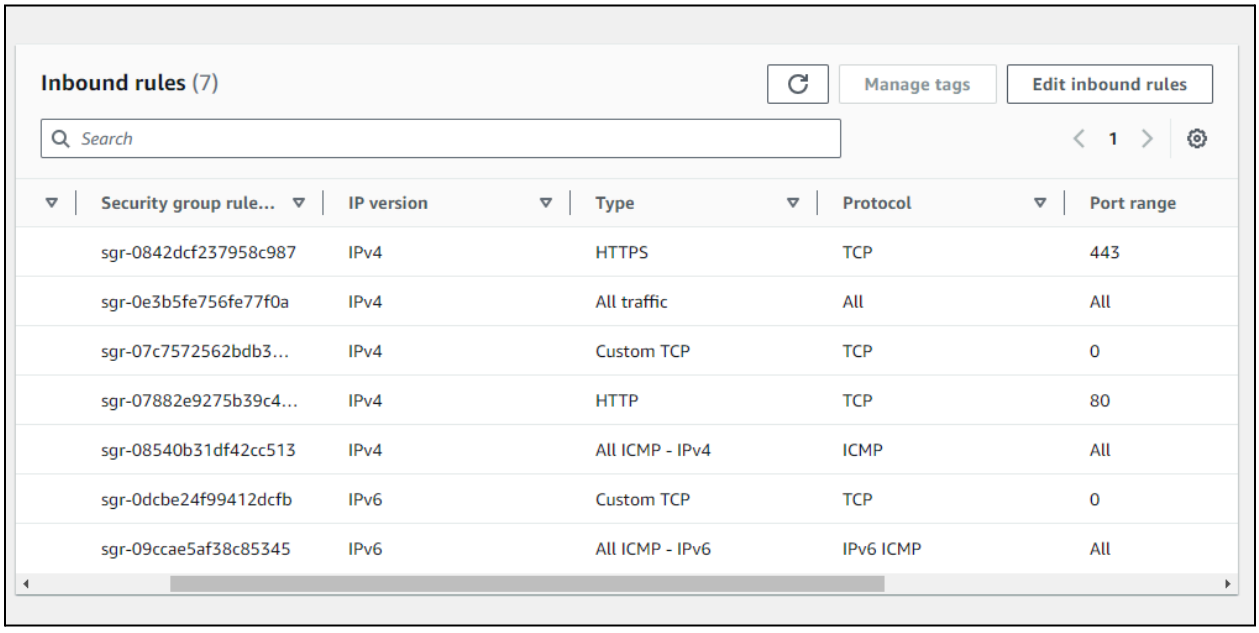
Name:Aryan Anil Patankar
Class:D15A
Roll No:34

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

Step 1: Create an Amazon Linux EC2 instance and name it as nagios-host



Step 2: Edit the following inbound rules of the specified security groups and ensure HTTP.HTTPS,SSH,ICMP are accessible from anywhere



Step 3:Connect to your EC2 instance via the connect option available in EC2 instances menu

```
[ec2-user@ip-172-31-33-14 ~]$ sudo yum install httpd php
Last metadata expiration check: 0:19:23 ago on Thu Sep 26 08:42:17 2024.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
httpd	x86_64	2.4.62-1.amzn2023	amazonlinux	48 k
php8.3	x86_64	8.3.10-1.amzn2023.0.1	amazonlinux	10 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
libsodium	x86_64	1.0.19-4.amzn2023	amazonlinux	176 k
libxslt	x86_64	1.1.34-5.amzn2023.0.2	amazonlinux	241 k
mailcap	noarch	2.1.49-3.amzn2023.0.3	amazonlinux	33 k

Step 4: Update and install the required packages

Use the following commands:

sudo yum update

sudo yum install httpd php

sudo yum install gcc glibc glibc-common

sudo yum install gd gd-devel

```
[ec2-user@ip-172-31-33-14 ~]$ sudo yum install gcc glibc glibc-common
Last metadata expiration check: 0:20:32 ago on Thu Sep 26 08:42:17 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

```
[ec2-user@ip-172-31-33-14 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:21:27 ago on Thu Sep 26 08:42:17 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

Step 5:Create a new nagios user by writing the following commands
sudo adduser -m nagios
sudo passwd nagios

```
Complete!
[ec2-user@ip-172-31-33-14 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-33-14 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-33-14 ~]$
```

Step 6: Create a new user group using **sudo groupadd nagcmd** and
Add users to the group using the following commands:
sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache

```
Complete!
[ec2-user@ip-172-31-33-14 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-33-14 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-33-14 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-33-14 ~]$ sudo usermod -a -G nagcmd nagios
[ec2-user@ip-172-31-33-14 ~]$ sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-33-14 ~]$ mkdir downloads
[ec2-user@ip-172-31-33-14 ~]$ cd downloads
[ec2-user@ip-172-31-33-14 downloads]$ wget https://sourceforge.net/projects/nagios/files/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz/download?use
_mirror=excellmedia
--2024-09-26 09:15:54--  https://sourceforge.net/projects/nagios/files/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz/download?use_mirror=excellmedi
a
Resolving sourceforge.net (sourceforge.net)... 172.64.150.145, 104.18.37.111, 2606:4700:4400::6812:256f, ...
Connecting to sourceforge.net (sourceforge.net)|172.64.150.145|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://downloads.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz?ts=gAAAAABm9SZKLFW7LwD1QAJ2jNzqmSJwAPA1mQ-
eAJYK8z5NmrV_ifVkhbsV-qOfPsLUyICC6yvdHu6UeeIyvNZsVGUtir9BeQ%3D%3D&use_mirror=excellmedia&r= [following]
```

Step 7: Create a directory for Nagios downloads using the following commands-
Commands -
mkdir ~/downloads
cd ~/downloads

Also download Nagios and plugin source files
Commands -
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

```
Connecting to prdownloads.sourceforge.net (prdownloads.sourceforge.net)|204.68.111.105|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://downloads.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz [following]
--2024-09-26 09:38:43--  https://downloads.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz
Resolving downloads.sourceforge.net (downloads.sourceforge.net)... 204.68.111.105
Connecting to downloads.sourceforge.net (downloads.sourceforge.net)|204.68.111.105|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://excellmedia.dl.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz?viasf=1 [following]
--2024-09-26 09:38:45--  https://excellmedia.dl.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz?viasf=1
Resolving excellmedia.dl.sourceforge.net (excellmedia.dl.sourceforge.net)... 202.153.32.19, 2401:fb00:0:1fe:8000::5
Connecting to excellmedia.dl.sourceforge.net (excellmedia.dl.sourceforge.net)|202.153.32.19|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1805059 (1.7M) [application/x-gzip]
Saving to: 'nagios-4.0.8.tar.gz'

nagios-4.0.8.tar.gz                 100%[=====================================================================>]   1.72M  8.14MB/s    in 0.2s

2024-09-26 09:38:45 (8.14 MB/s) - 'nagios-4.0.8.tar.gz' saved [1805059/1805059]

[ec2-user@ip-172-31-33-14 downloads]$ ls
'download?use_mirror=excellmedia'   nagios-4.0.8.tar.gz
[ec2-user@ip-172-31-33-14 downloads]$ tar -xzf nagios-4.0.8.tar.gz
[ec2-user@ip-172-31-33-14 downloads]$
```

Step 8-Extract the nagios source file with the following commands
tar zxvf nagios-4.4.6.tar.gz
cd nagios-4.4.6

Then run the configuration script with the following command
/configure --with-command-group=nagcmd

```
        Nagios user/group:  nagios,nagios
       Command user/group:  nagios,nagcmd
             Event Broker:  yes
        Install ${prefix}:  /usr/local/nagios
    Install ${includedir}:  /usr/local/nagios/include/nagios
                Lock file:  ${prefix}/var/nagios.lock
   Check result directory:  ${prefix}/var/spool/checkresults
           Init directory:  /etc/rc.d/init.d
  Apache conf.d directory:  /etc/httpd/conf.d
             Mail program:  /bin/mail
                  Host OS:  linux-gnu
          IOBroker Method:  epoll

 Web Interface Options:
 ------------------------
                 HTML URL:  http://localhost/nagios/
                  CGI URL:  http://localhost/nagios/cgi-bin/
 Traceroute (used by WAP):  /usr/bin/traceroute


Review the options above for accuracy.  If they look okay,
type 'make all' to compile the main program and CGIs.

[ec2-user@ip-172-31-33-14 nagios-4.0.8]$
```

Step 9-Compile the source code with the following commands
make all

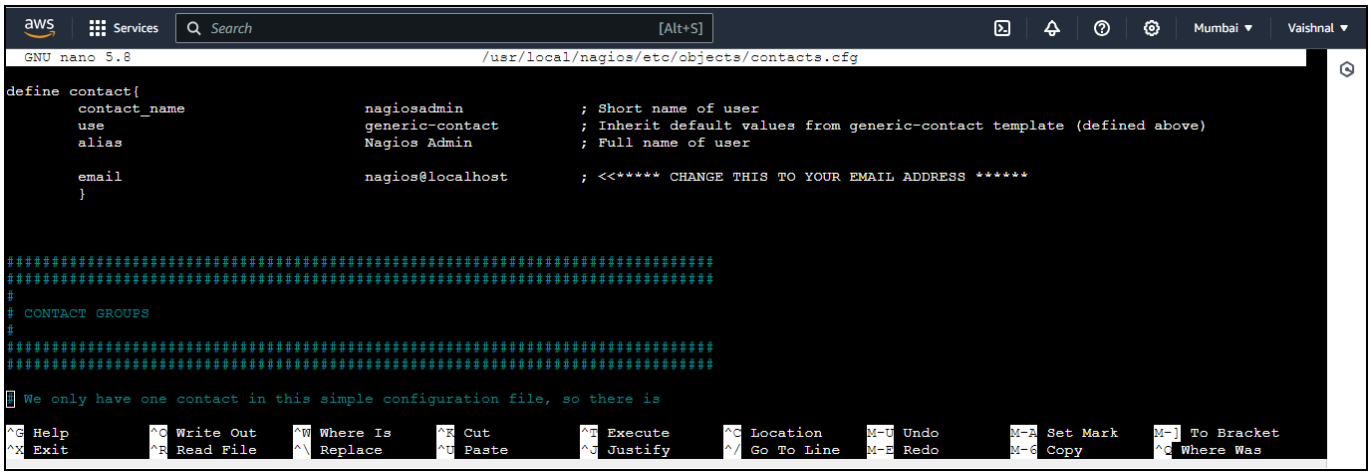
```
[ec2-user@ip-172-31-33-14 nagios-4.0.8]$ make all
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.0.8/base'
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o nagios.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o broker.o broker.c
gcc -Wall -I.. -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 nerd.o nerd.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:224:12:
workers.c:209:17: warning: ‘%s’ directive argument is null [-Wformat-overflow=]
  209 |                 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
commands.c: In function ‘process_passive_service_check’:
commands.c:2247:19: warning: assignment discards ‘const’ qualifier from pointer target type [-Wdiscarded-qualifiers]
```

Step 10-Install binaries,init script and sample config files
Commands -
./sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode

```
*** Config files installed ***

Remember, these are *SAMPLE* config files.  You'll need to read
the documentation for more information on how to actually define
services, hosts, etc. to fit your particular needs.

/usr/bin/install -c -m 775 -o nagios -g nagcmd -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***

[ec2-user@ip-172-31-33-14 nagios-4.0.8]$ 
```

Step 11-Edit the Config File to Change the Email Address
Commands -
sudo nano /usr/local/nagios/etc/objects/contacts.cfg
● Change the email address in the contacts.cfg file to your preferred email

Step 12-Configure the Web Interface
Commands -
sudo make install-webconf

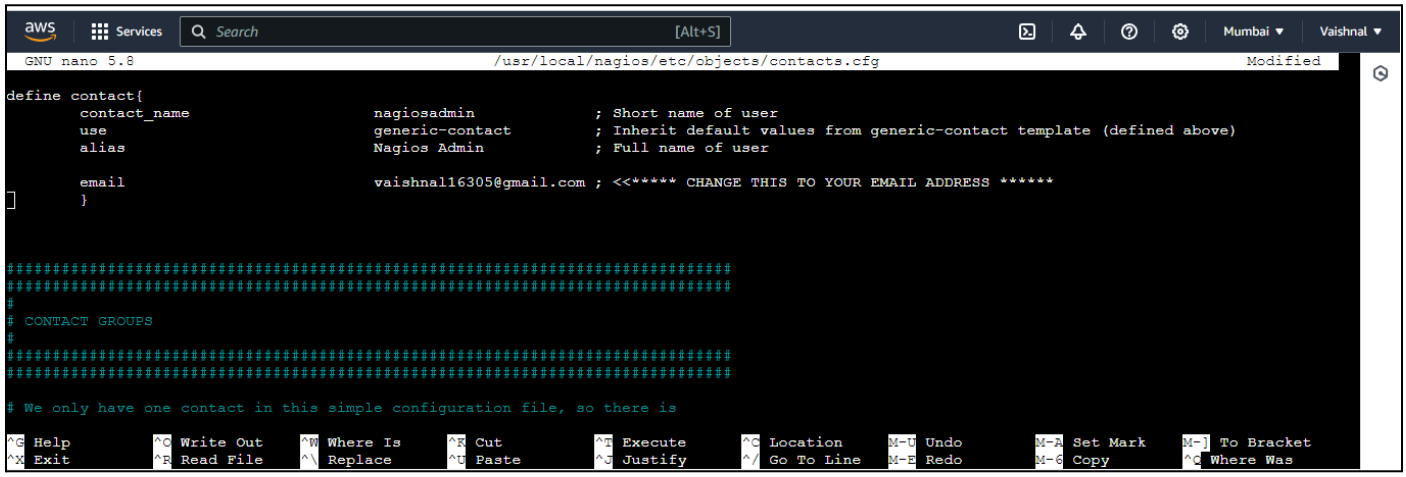


Step 13-Create a Nagios Admin Account

Commands -

sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

● You will be prompted to enter and confirm the password for the nagiosadmin user



Step 14-. Extract the Plugins Source File

Commands -

cd ~/downloads

tar zxvf nagios-plugins-2.3.3.tar.gz

cd nagios-plugins-2.3.3

```
*** External command directory configured ***

[ec2-user@ip-172-31-33-14 nagios-4.0.8]$ sudo nano /usr/local/nagios/etc/objects/contacts.cfg
[ec2-user@ip-172-31-33-14 nagios-4.0.8]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf

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

Step 15-19. Compile and Install Plugins

Commands -

./configure --with-nagios-user=nagios --with-nagios-group=nagios

make

sudo make install

```
[ec2-user@ip-172-31-33-14 nagios-4.0.8]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
```

Step 16-Start Nagios

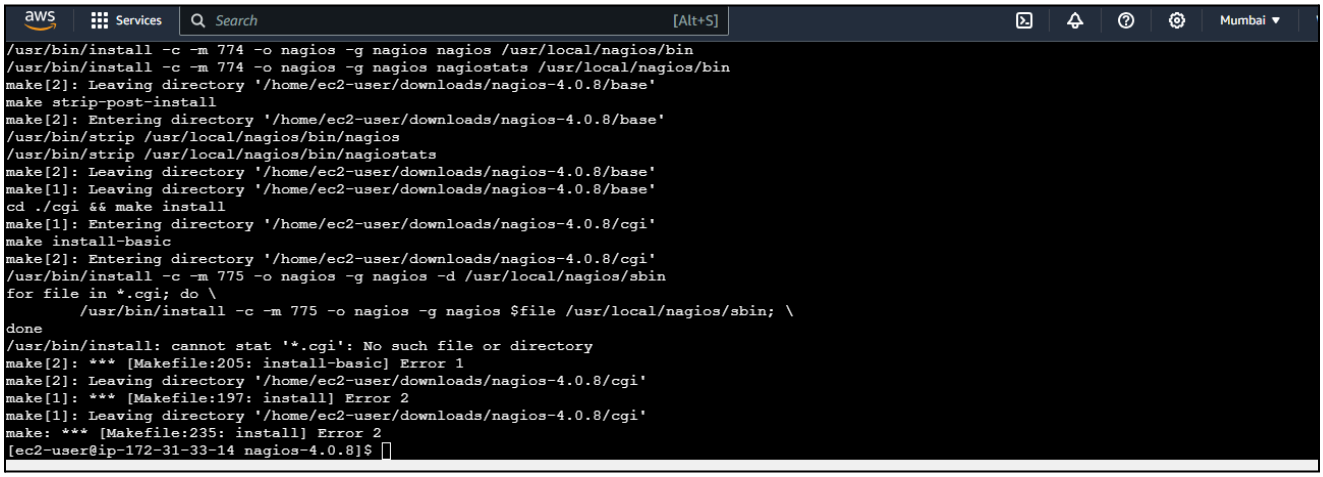
Commands -

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



Step 17-Access Nagios Web Interface

- Copy the Public IP address of your EC2 instance.
- Open your browser and navigate to http:///nagios.
- Enter the username nagiosadmin and the password you set in Step 16.

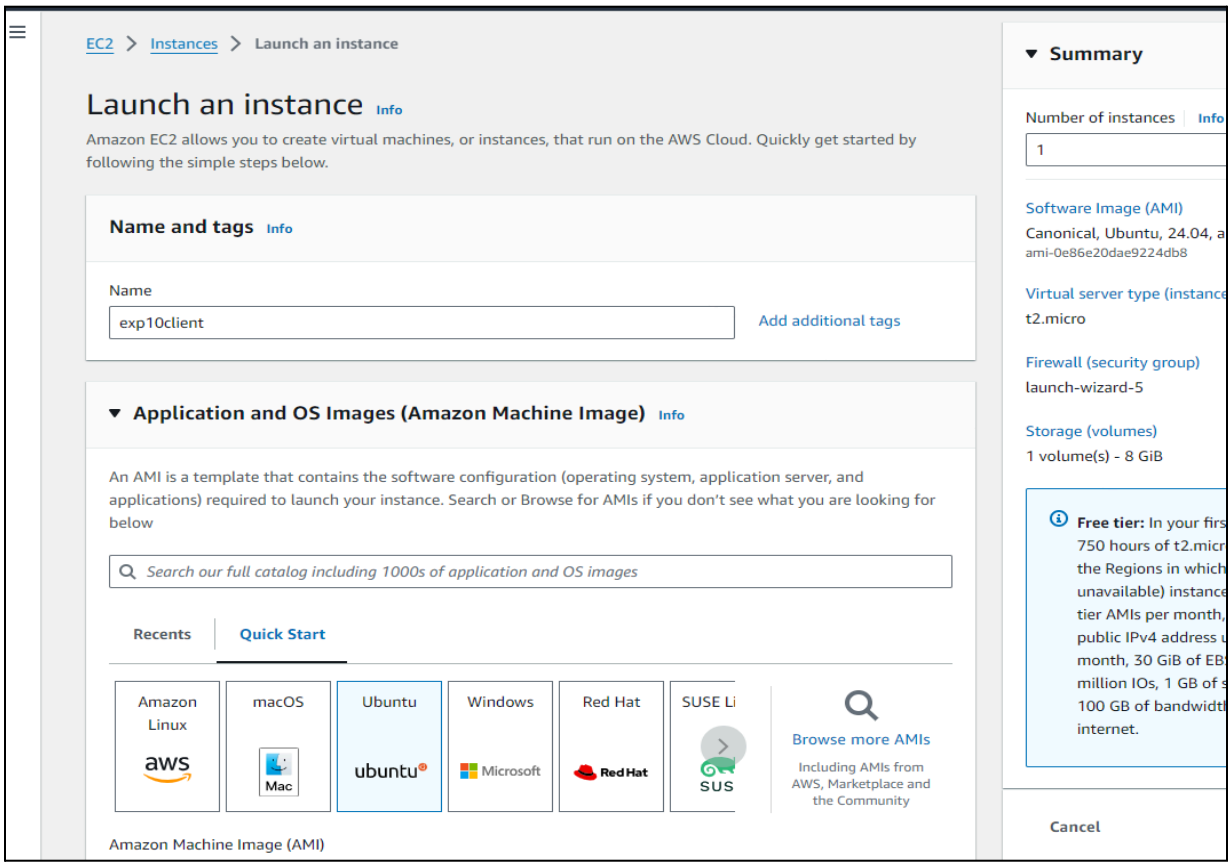

Nagios
General
Home
Documentation
Current Status
Tactical Overview
Map (Legacy)
Hosts
Services
Host Groups
Summary
Grid
Service Groups
Summary
Grid
Problems
Services (Unhandled)
Hosts (Unhandled)
Network Outages
Quick Search:
Reports
Availability
Trends (Legacy)
Alerts
History
Summary
Histogram (Legacy)
Nagios Core
Unable to get process status
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...

ADVANCE DEVOPS EXPERIMENT 10

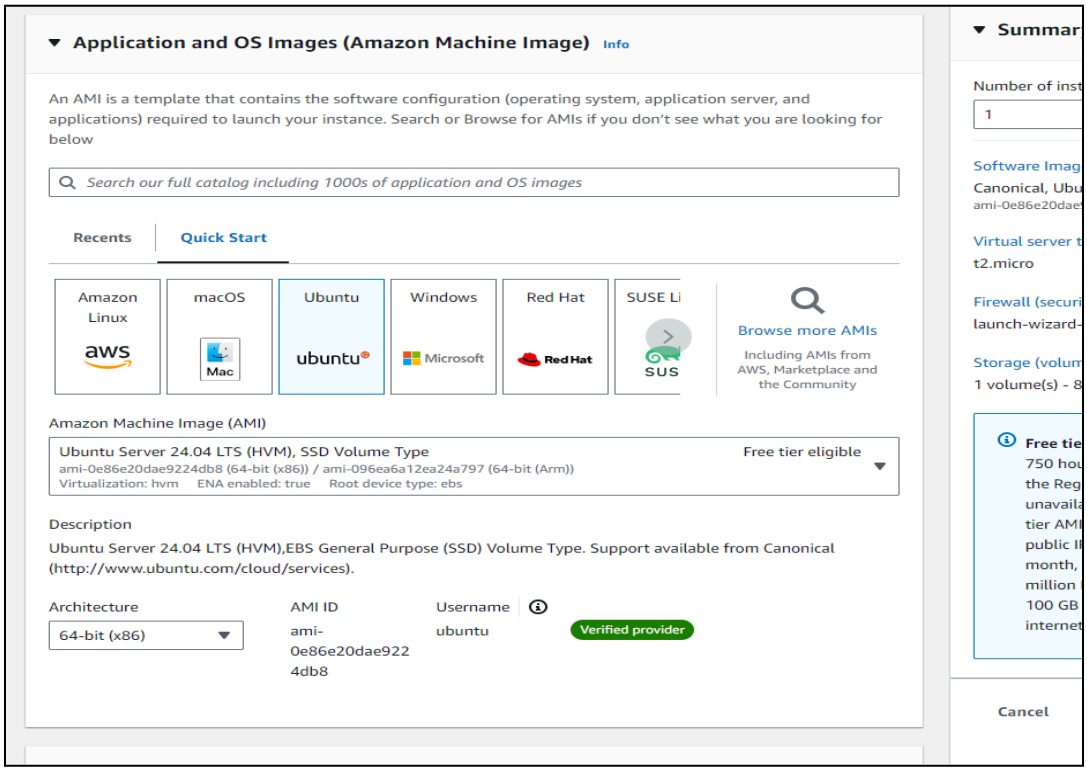
Name:Aryan Anil Patankar
Class;D15A
Roll No:34

1) Launch an instance

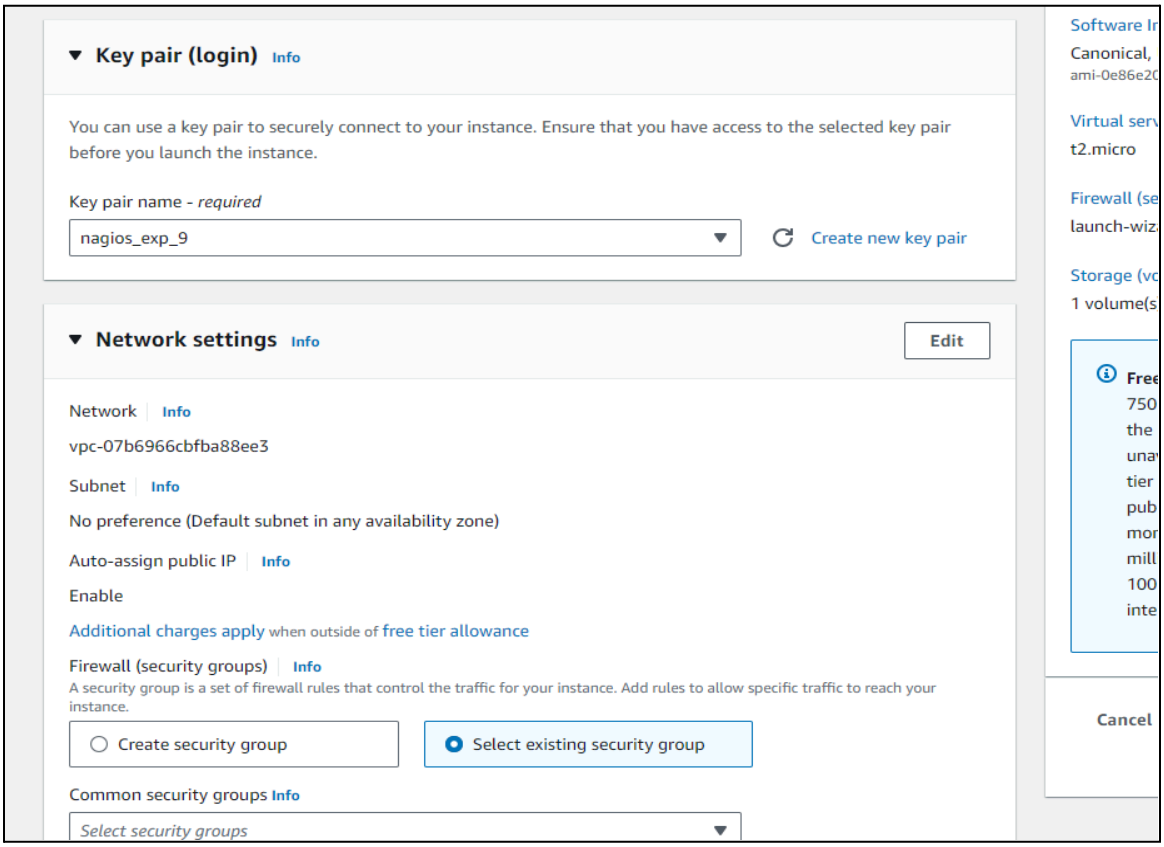
Launch an ec2 instance.
Select Ubuntu as the os give a meaningful name of the instance.



Select the same security group as given in exp9.

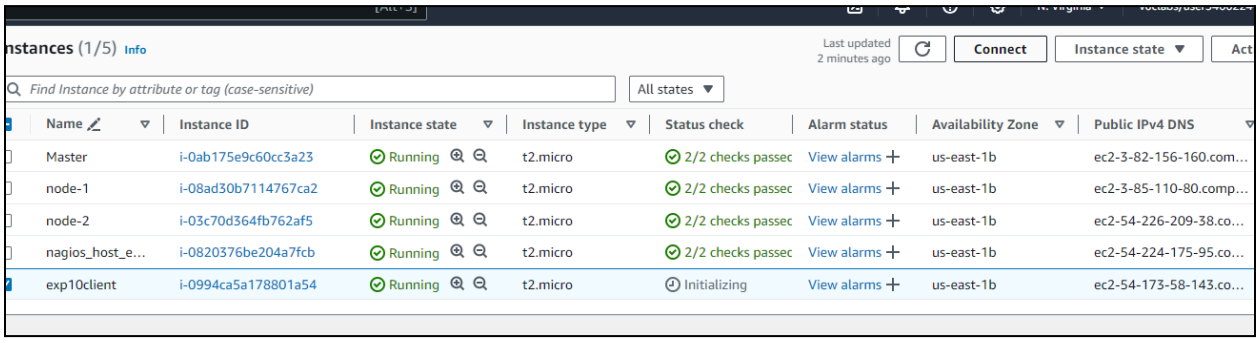


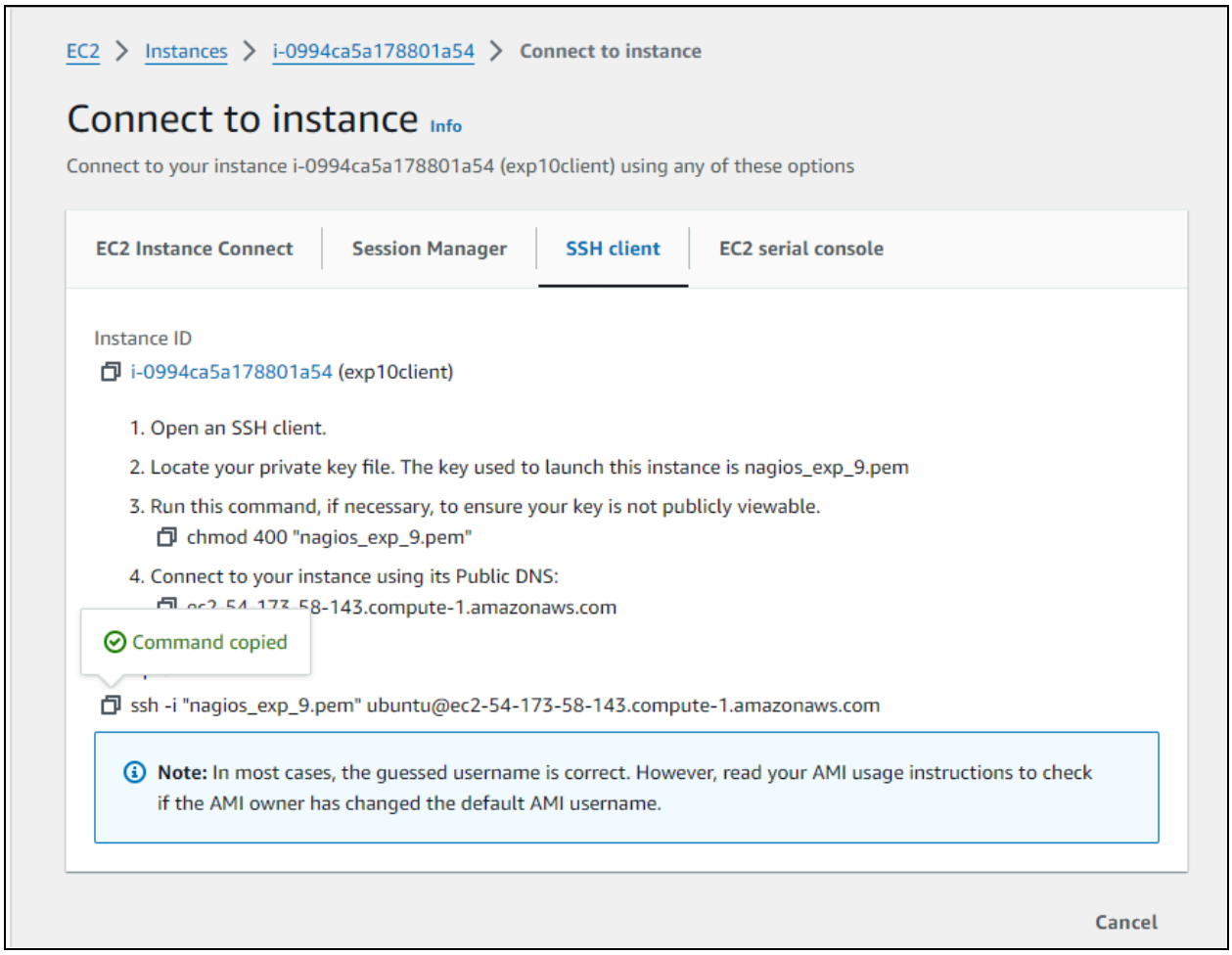
Make sure to select the same key-pair login used in the exp9 machine.



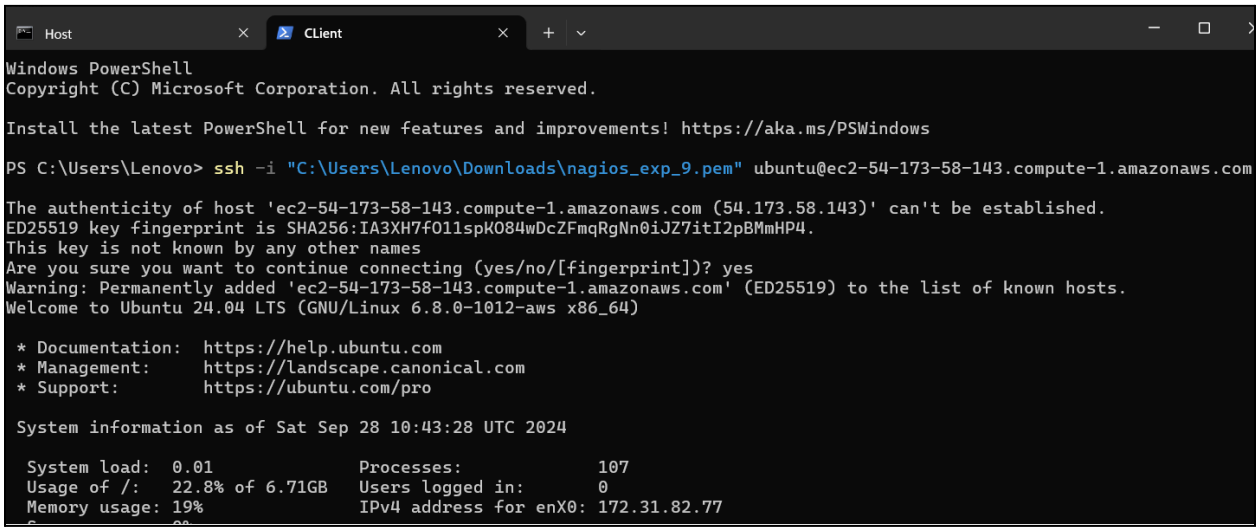
click on launch instance.

Now connect with this client machine using the ssh through your terminal(open a new terminal in your local machine and we will need both of the terminals open)





Note to change the path of the .pem file.



2) Go to nagios host machine (Host machine)

Perform the following commands

ps -ef | grep nagios

```
[ec2-user@ip-172-31-80-137 ~]$ ps -ef | grep nagios
nagios      3152       1  0 08:36 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
nagios      3153    3152  0 08:36 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      3154    3152  0 08:36 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      3155    3152  0 08:36 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      3156    3152  0 08:36 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      3160    3152  0 08:36 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
ec2-user   11528    2972  0 10:44 pts/0    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-80-137 ~]$
```

sudo su

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

```
[root@ip-172-31-80-137 ec2-user]# mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-80-137 ec2-user]# ls
```

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

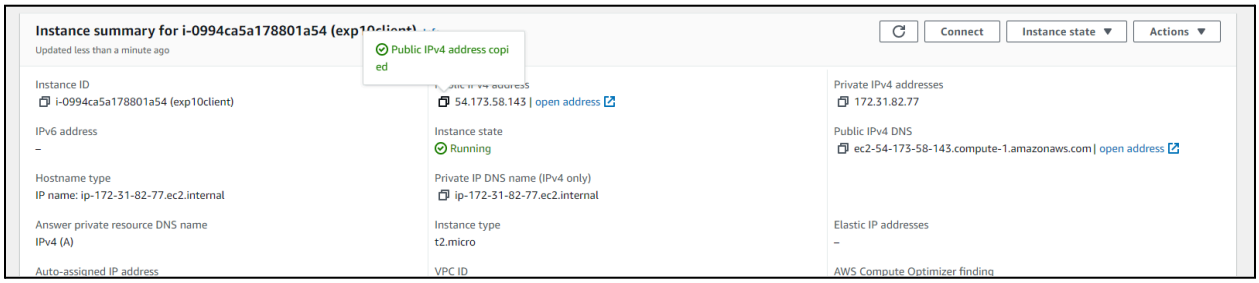
```
[root@ip-172-31-80-137 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxho
sts/linuxserver.cfg
```

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

```
[root@ip-172-31-80-137 ec2-user]# nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
```

Change hostname and alias to linuxserver

Change address to public ip address of client instance (Ubuntu instance) you can get the ip address by clicking on the instance id on the instances section there you will get the public ipv4 address



```
#
# HOST DEFINITION
#
###############################################################################

# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will in>
                                                    ; in (or inherited by) the lin>
    host_name               linuxserver
    alias                   linuxserver
    address                 54.173.58.143
}
```

Change hostgroup_name to linux-servers1

```
# Define an optional hostgroup for Linux machines

define hostgroup {

    hostgroup_name          linux-servers1          ; The name of the hostgroup
    alias                   Linux Servers           ; Long name of the group
    members                 localhost               ; Comma separated list of host>
}
```

Change the occurrences of hostname further in the document from localhost to linuxserver
example like:

```
    host_name                       localhost
```

changed to

```
define service {

    use                     local-service           ; Name of service templat
    host_name               linuxserver
    service_description     PING
    check_command           check_ping!100.0,20%!500.0,60%
}
```

This is the last one

```
define service {

    use                     local-service           ; Name of service template to >
    host_name               linuxserver
    service_description     HTTP
    check_command           check_http
    notifications_enabled   0
```

now ctrl+O and enter to save and then ctrl+X for exiting.
Open nagios configuration file and add the line shown below
nano /usr/local/nagios/etc/nagios.cfg

```
[root@ip-172-31-80-137 ec2-user]# nano /usr/local/nagios/etc/nagios.cfg
```

##Add this line below the opened nano interface where similar lines are commented.
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/

```
 GNU nano 5.8                                                    /usr/local/nagios/etc/nagios.cfg
# These are the object configuration files in which you define hosts,
# host groups, contacts, contact groups, services, etc.
# You can split your object definitions across several config files
# if you wish (as shown below), or keep them all in a single config file.

# You can specify individual object config files as shown below:
cfg_file=/usr/local/nagios/etc/objects/commands.cfg
cfg_file=/usr/local/nagios/etc/objects/contacts.cfg
cfg_file=/usr/local/nagios/etc/objects/timeperiods.cfg
cfg_file=/usr/local/nagios/etc/objects/templates.cfg

# Definitions for monitoring the local (Linux) host
cfg_file=/usr/local/nagios/etc/objects/localhost.cfg

# Definitions for monitoring a Windows machine
#cfg_file=/usr/local/nagios/etc/objects/windows.cfg

# 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/



# OBJECT CACHE FILE
# This option determines where object definitions are cached when
# Nagios starts/restarts.  The CGIs read object definitions from
```

ctrl+o and enter for saving and ctrl+x to exit nano editor.

Verify configuration files
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

```
[root@ip-172-31-80-137 ec2-user]# /usr/local/nagios/bin/nagios -v /usr/local/nagios
/etc/nagios.cfg

Nagios Core 4.5.5
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2024-09-17
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 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-172-31-80-137 ec2-user]#
```

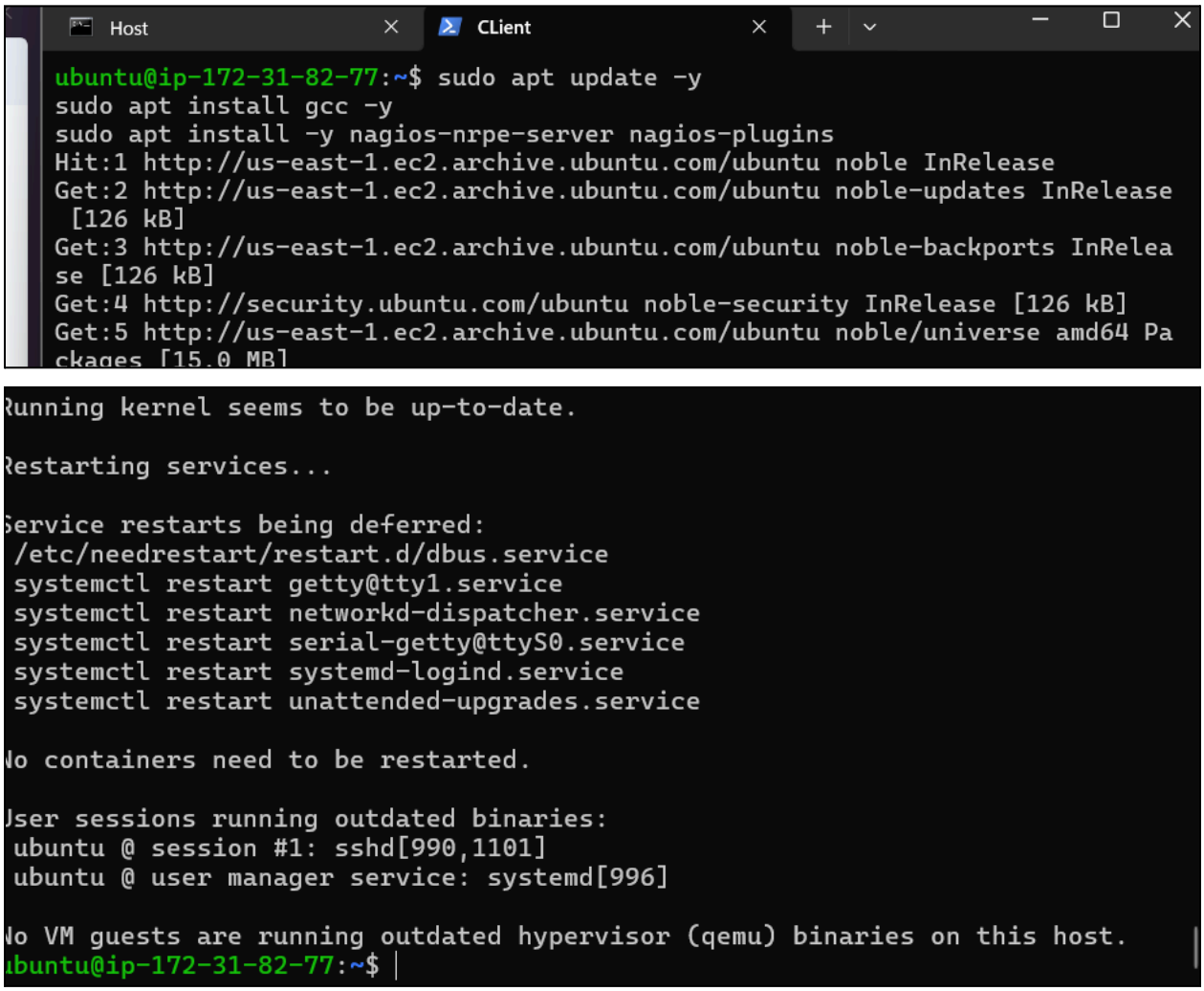
Restart nagios service.
service nagios restart

```
Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-80-137 ec2-user]# service nagios restart
Redirecting to /bin/systemctl restart nagios.service
[root@ip-172-31-80-137 ec2-user]#
```

3) Go to client machine (ubuntu machine)

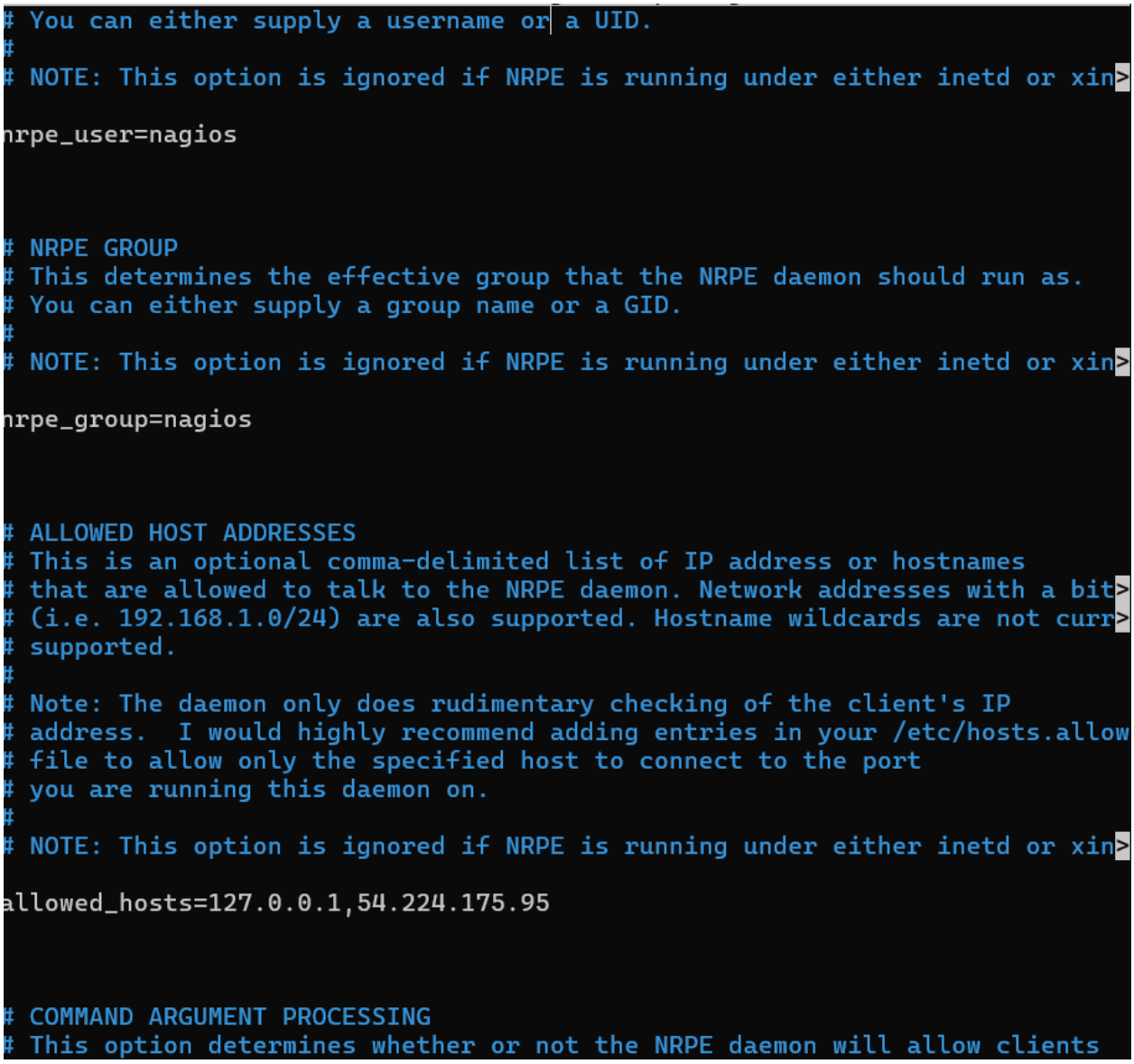
Perform the following commands
sudo apt update -y
sudo apt install gcc -y

sudo apt install -y nagios-nrpe-server nagios-plugins



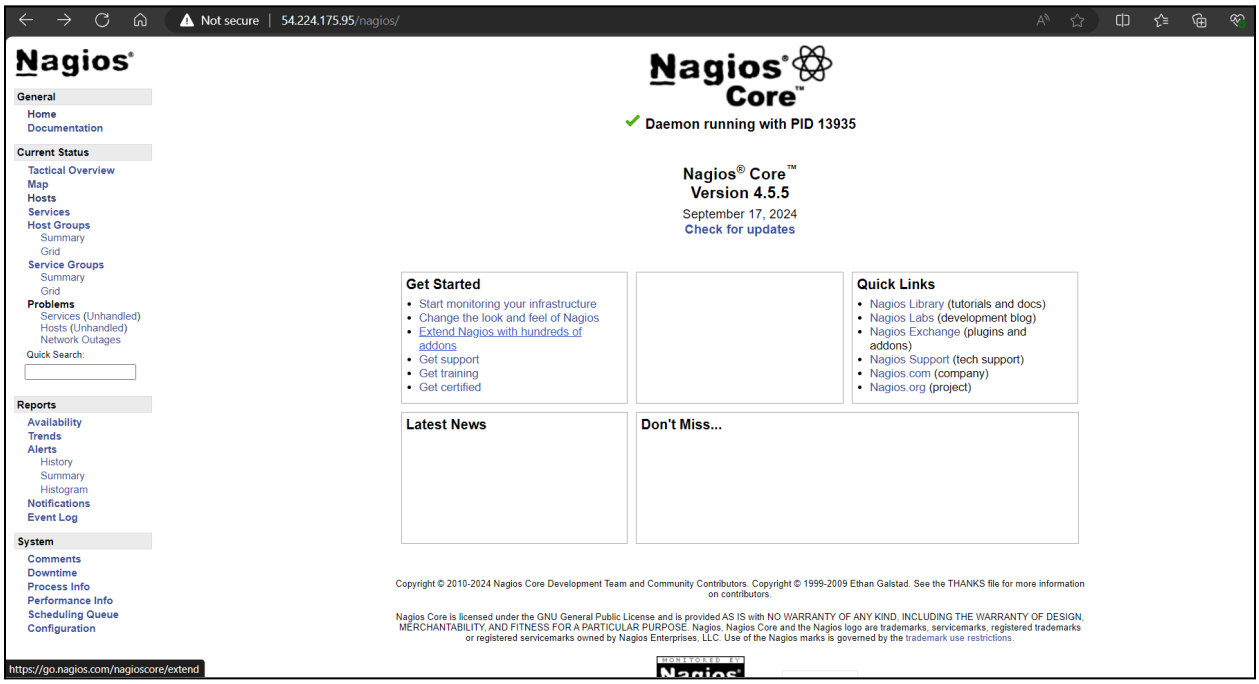
Open the nrpe.cfg file in nano editor
sudo nano /etc/nagios/nrpe.cfg

Under allowed_hosts, add the nagios host ip address (public)

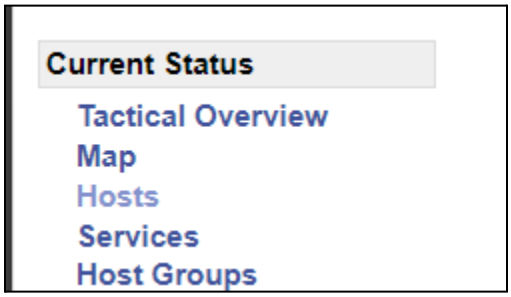


again save and exit the nano editor.

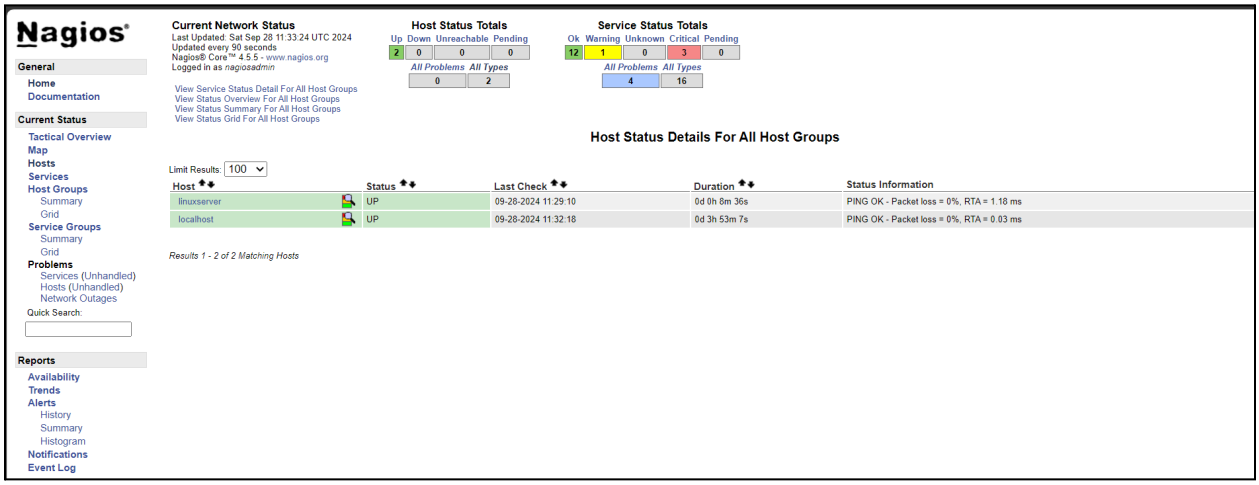
4) Go to nagios dashboard and click on hosts

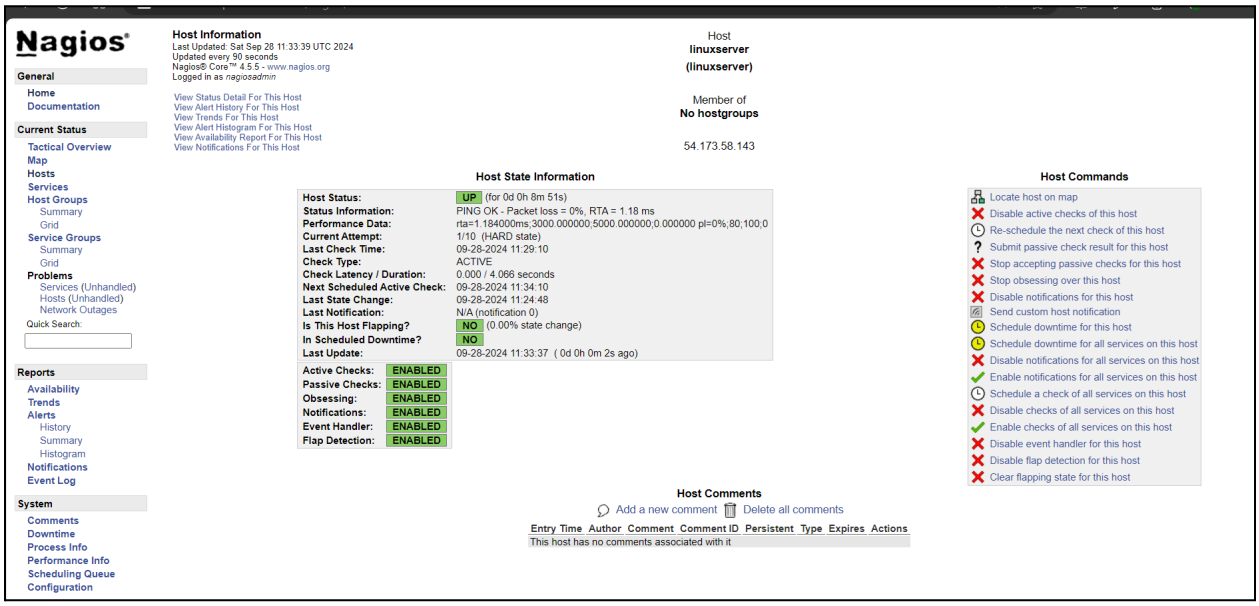


Click on hosts

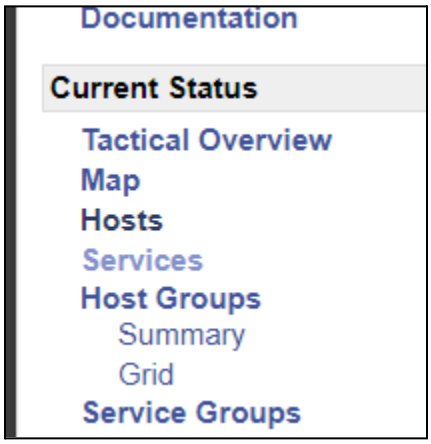


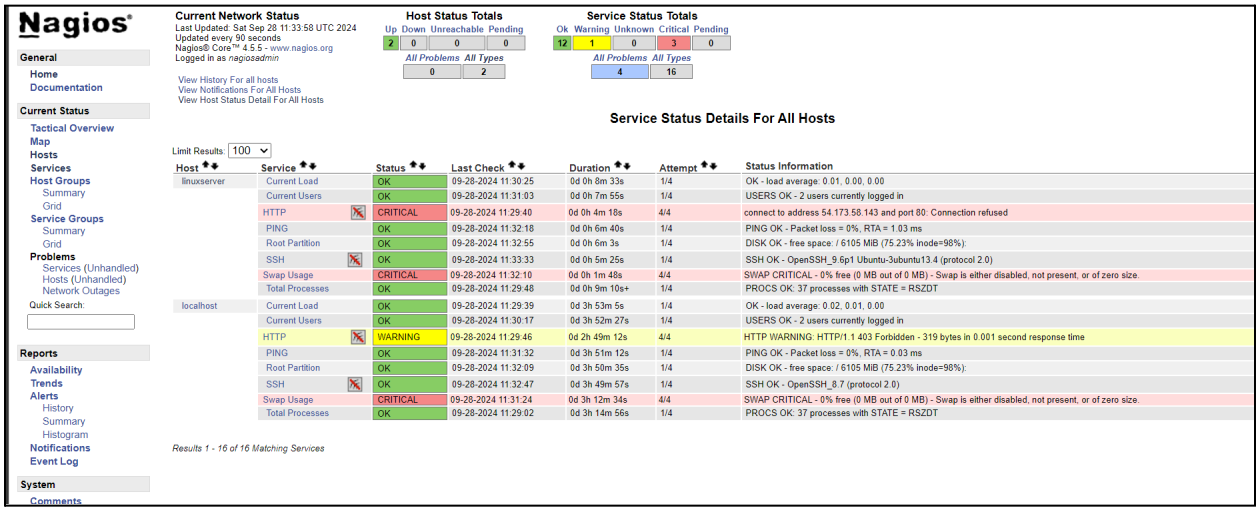
5) Click on linux server





6) Click on nagios services





Conclusion:

In this lab, we successfully configured a monitoring setup between a Nagios host machine (referred to as "exp9 machine") and a client machine (created specifically for this experiment). The goal was to set up Nagios to monitor a remote Linux server, which involved configuring both the Nagios host and client machine (Ubuntu instance) in an EC2 environment.

ADVANCE DEVOPS EXP 11

Name:Aryan Anil Patankar

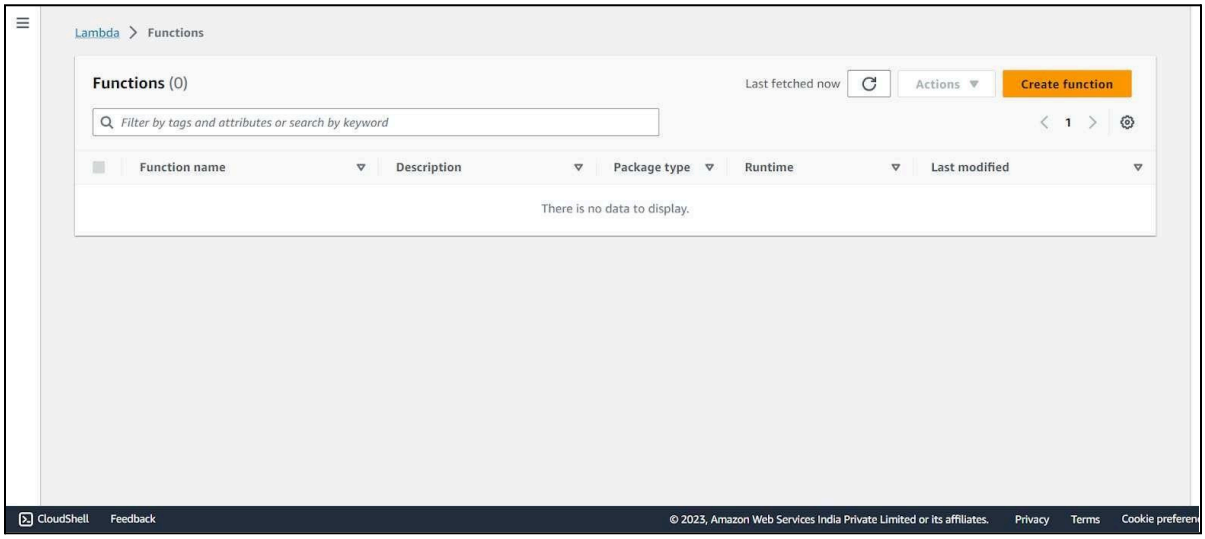
Class: D15A

Roll No:34

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

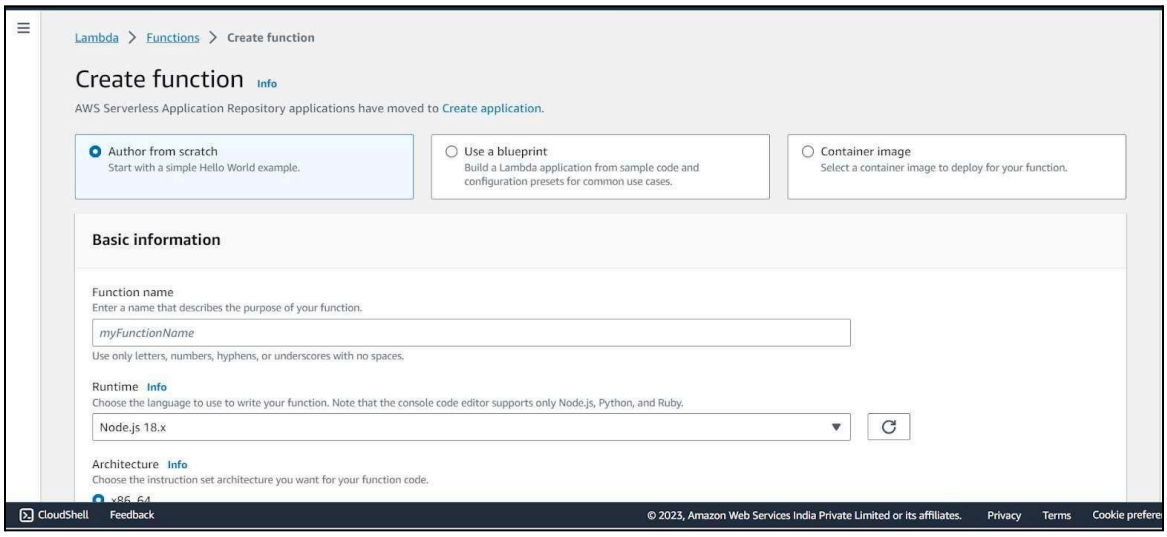
Steps to create an AWS Lambda function

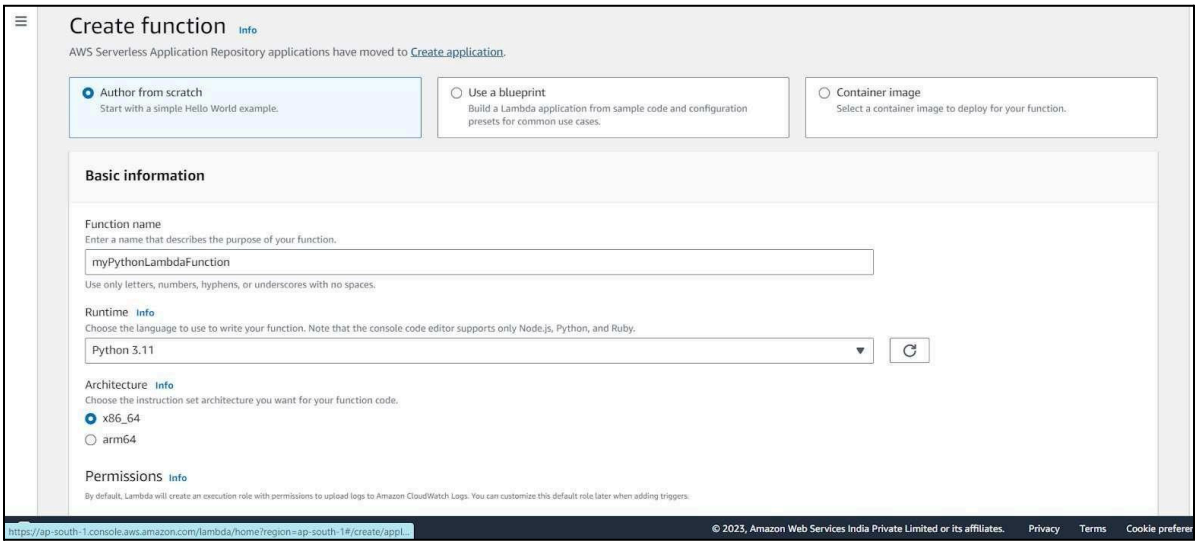
Step 1:Open up the Lambda Console and click on the Create button.
Be mindful of where you create your functions since Lambda is region-dependent.

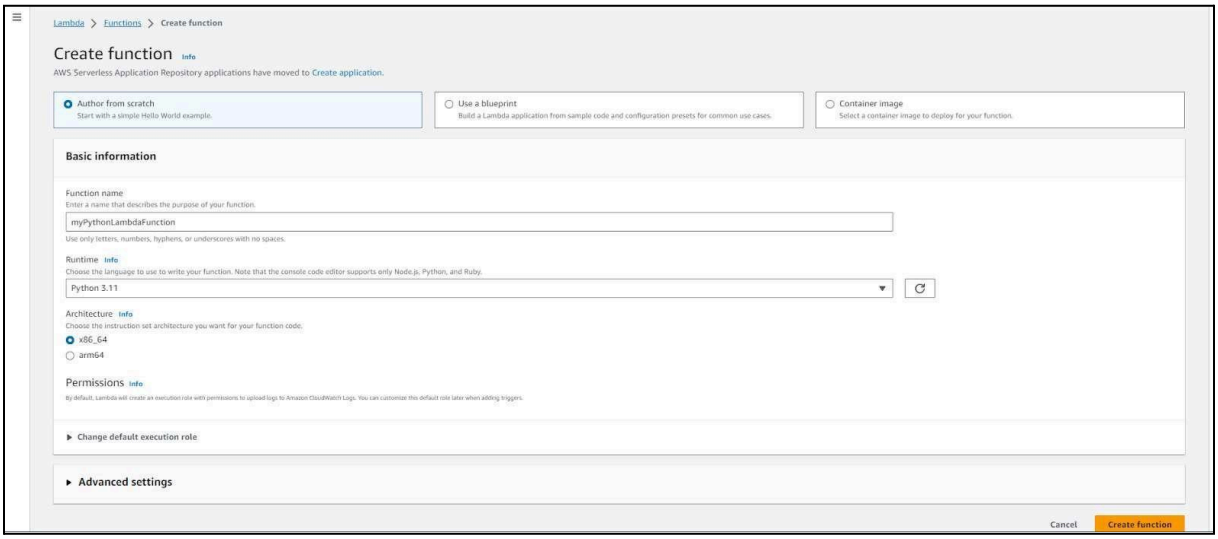


2. Choose to create a function from scratch or use a blueprint, i.e templates defined by AWS for you with all configuration presets required for the most common use cases.

Then, choose a runtime env for your function, under the dropdown, you can see all the options AWS supports, Python, Nodejs, .NET and Java being the most popular ones. After that, choose to create a new role with basic Lambda permissions if you don't have an existing one.

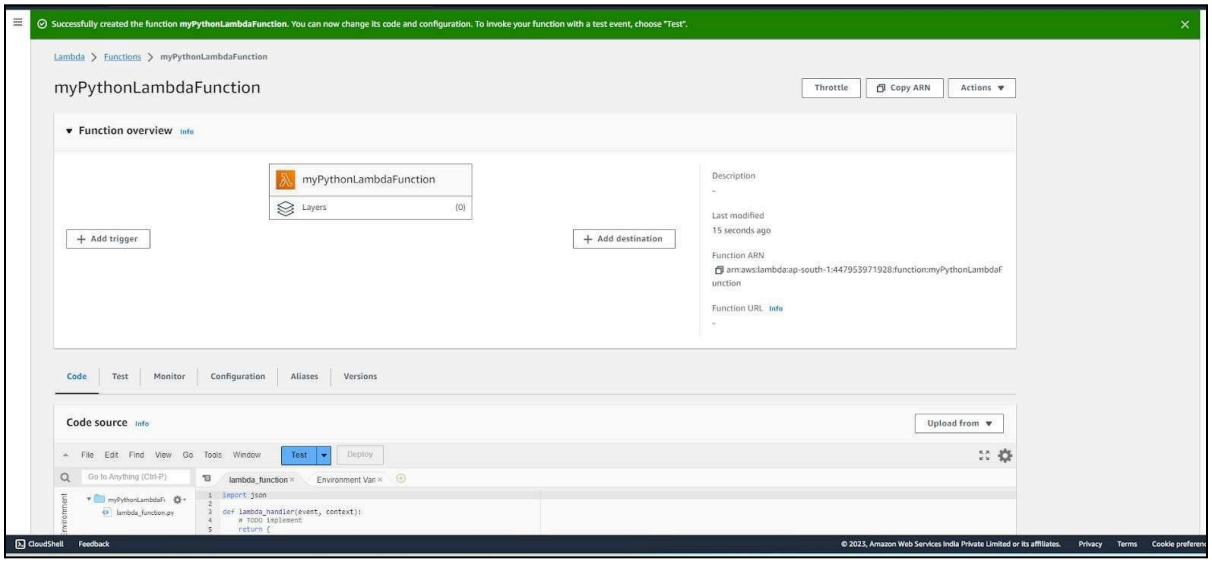


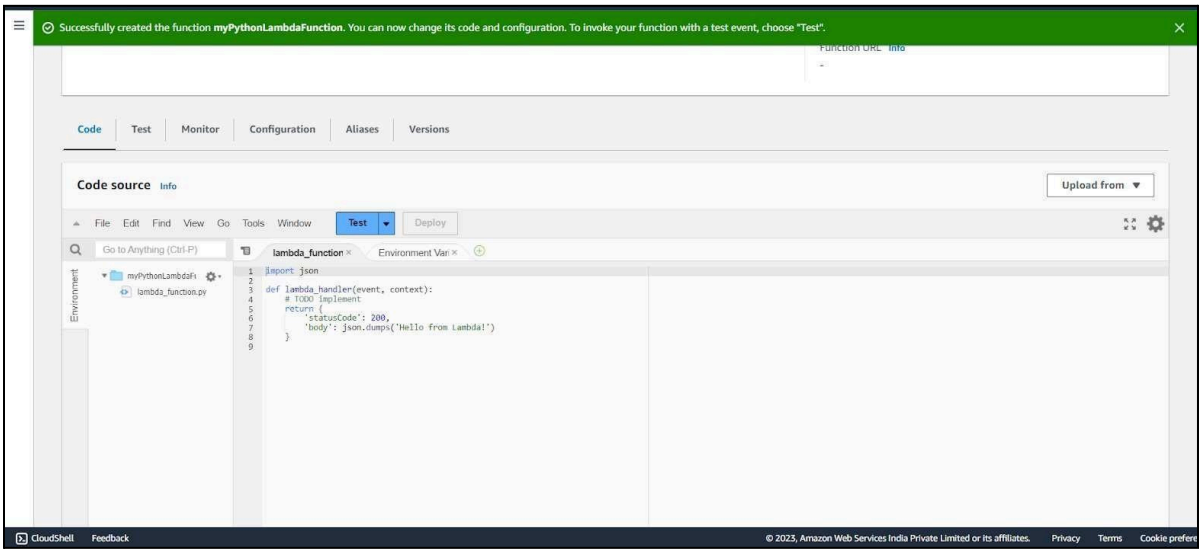




Click on the Create button.

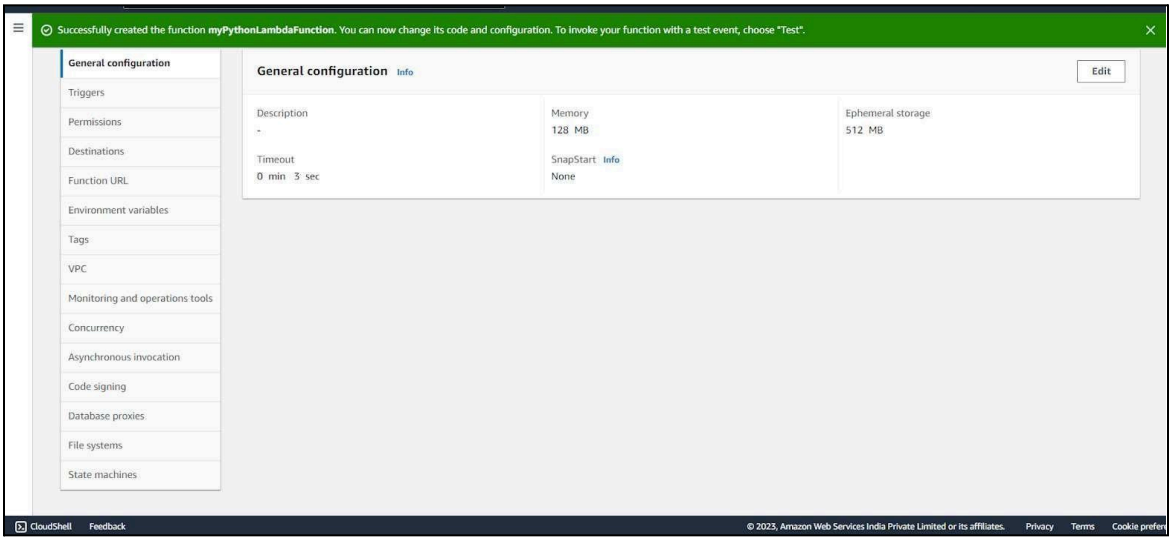
3. This process will take a while to finish and after that, you'll get a message that your function was successfully created.

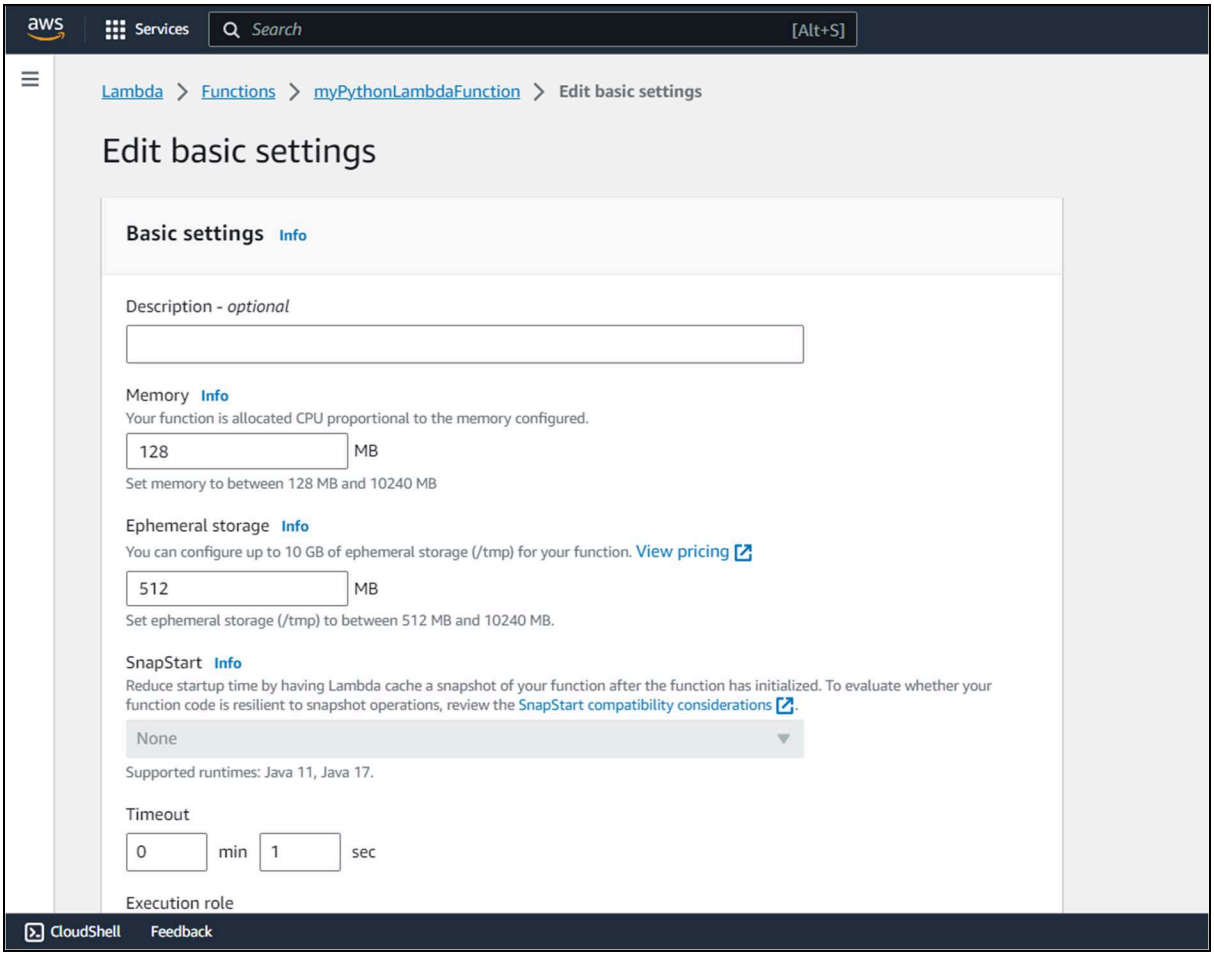




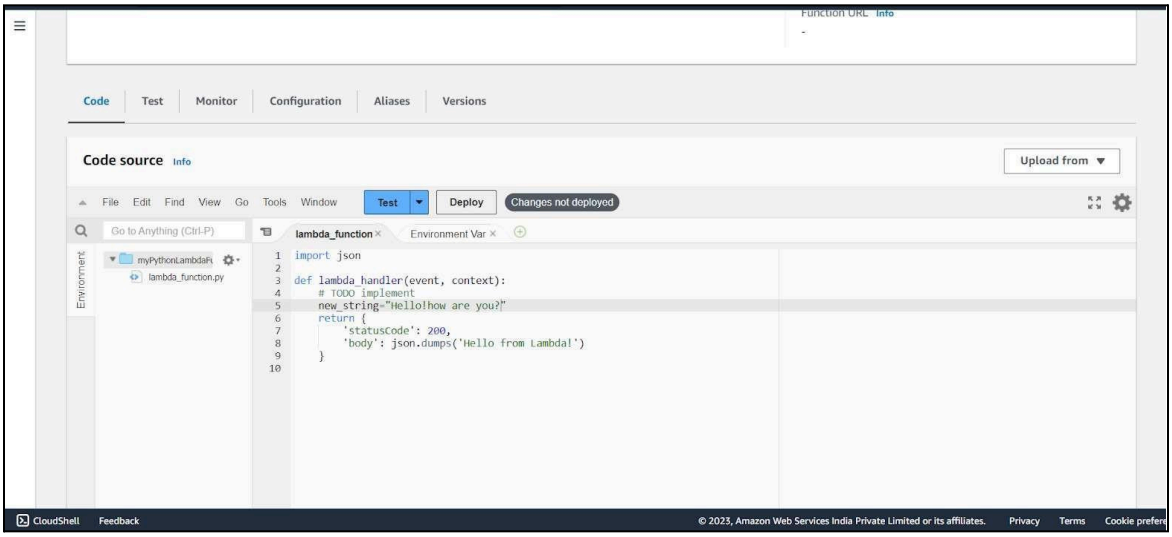
4. To change the configuration, open up the Configuration tab and under General Configuration, choose Edit.

Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.

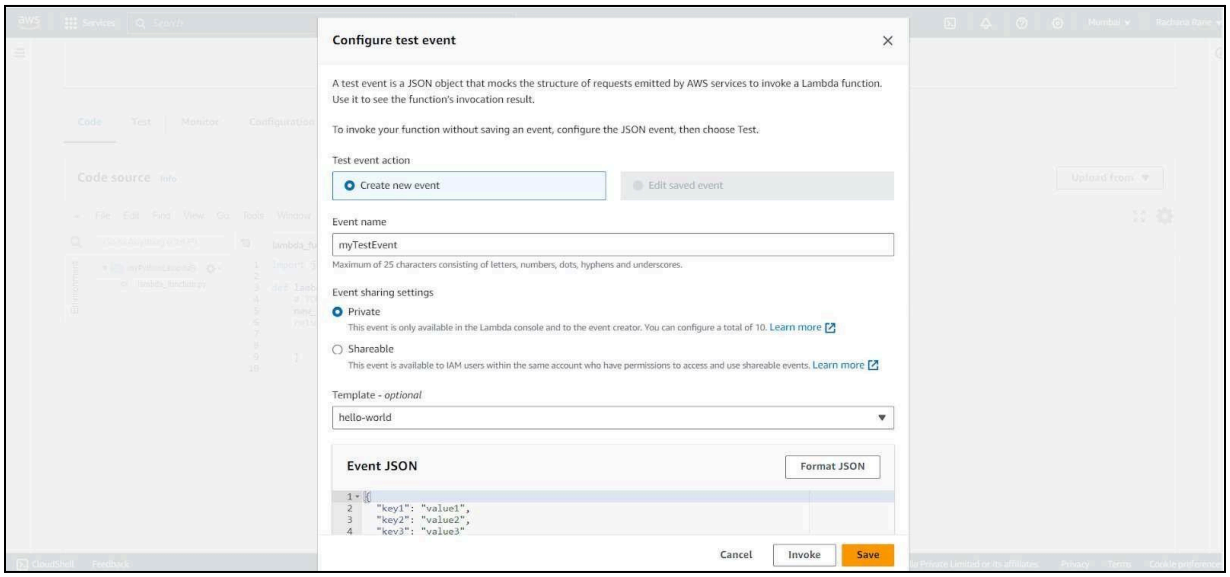




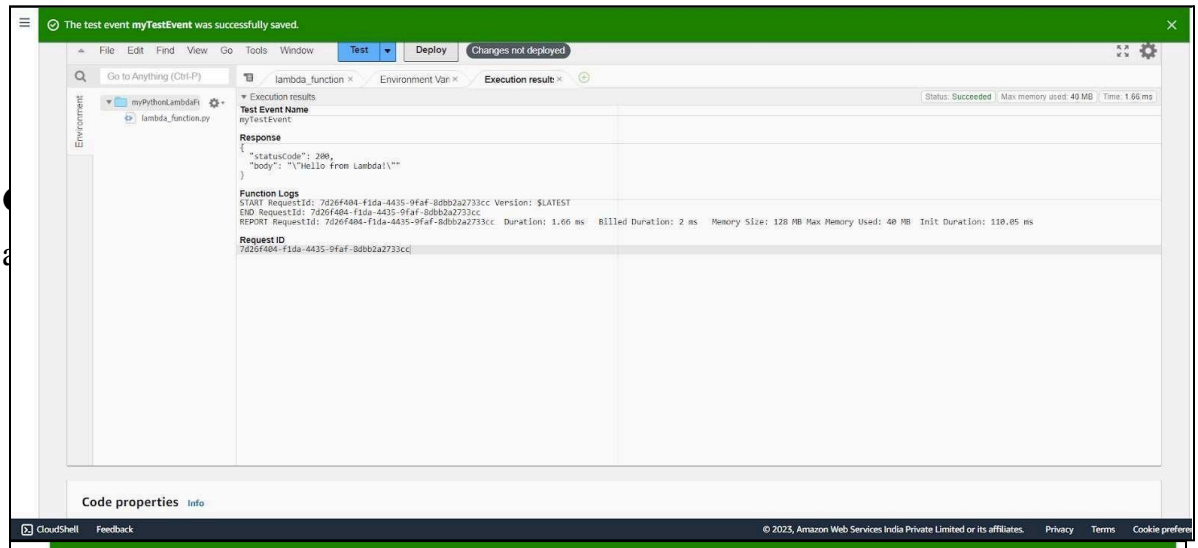
5. You can make changes to your function inside the code editor. You can also upload a zip file of your function or upload one from an S3 bucket if needed. Press Ctrl + S to save the file and click Deploy to deploy the changes.

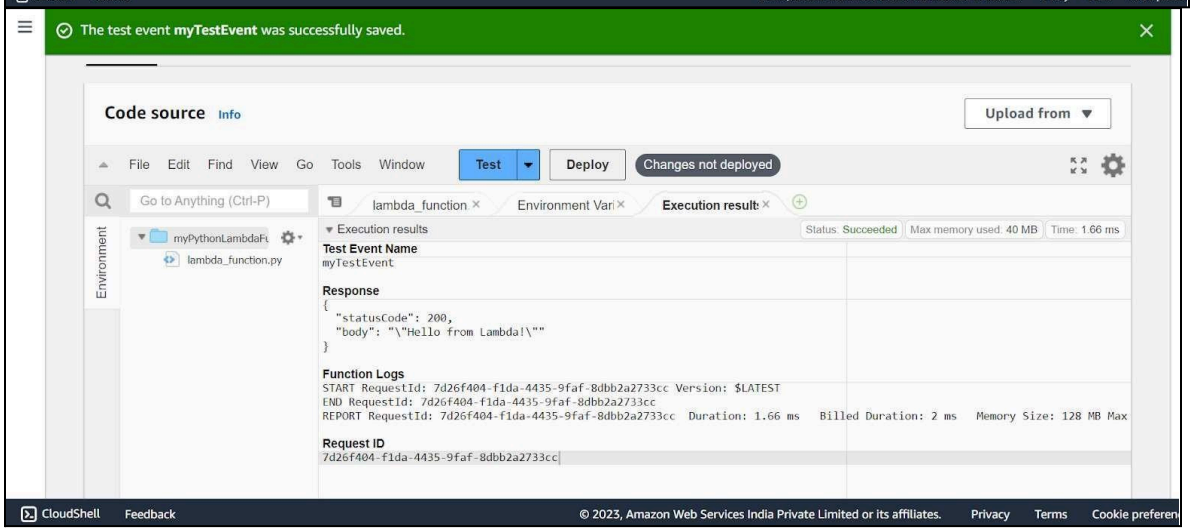


6. Click on Test and you can change the configuration, like so. If you do not have anything in the request body, it is important to specify two curly braces as valid JSON, so make sure they are there.



7. Now click on Test and you should be able to see the results.

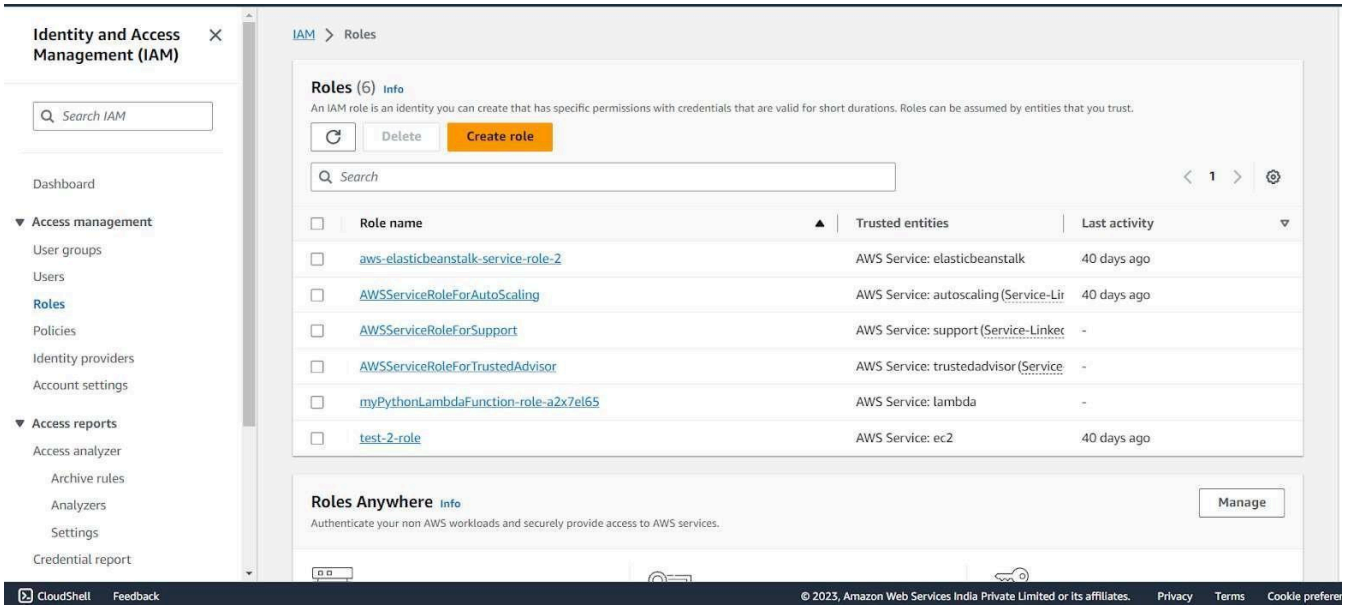

The test event myTestEvent was successfully saved.
File Edit Find View Go Tools Window
Test
Deploy
Changes not deployed
Go to Anything (Ctrl-P)
lambda_function
Environment Var
Execution result
Environment
myPythonLambdaFi
lambda_function.py
Execution results
Status: Succeeded
Max memory used: 40 MB
Time: 1.66 ms
Test Event Name
myTestEvent
Response
{
"statusCode": 200,
"body": "\"Hello from Lambda!\""
}
Function Logs
START RequestId: 7d26f404-f1da-4435-9faf-8dbb2a2733cc Version: $LATEST
END RequestId: 7d26f404-f1da-4435-9faf-8dbb2a2733cc
REPORT RequestId: 7d26f404-f1da-4435-9faf-8dbb2a2733cc Duration: 1.66 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 40 MB Init Duration: 110.05 ms
Request ID
7d26f404-f1da-4435-9faf-8dbb2a2733cc
Code properties Info
CloudShell
Feedback
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie prefere


The test event myTestEvent was successfully saved.
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Changes not deployed
Go to Anything (Ctrl-P)
lambda_function
Environment Vari
Execution result
Environment
myPythonLambdaFi
lambda_function.py
Execution results
Status: Succeeded
Max memory used: 40 MB
Time: 1.66 ms
Test Event Name
myTestEvent
Response
{
"statusCode": 200,
"body": "\"Hello from Lambda!\""
}
Function Logs
START RequestId: 7d26f404-f1da-4435-9faf-8dbb2a2733cc Version: $LATEST
END RequestId: 7d26f404-f1da-4435-9faf-8dbb2a2733cc
REPORT RequestId: 7d26f404-f1da-4435-9faf-8dbb2a2733cc Duration: 1.66 ms Billed Duration: 2 ms Memory Size: 128 MB Max
Request ID
7d26f404-f1da-4435-9faf-8dbb2a2733cc
CloudShell
Feedback
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferen

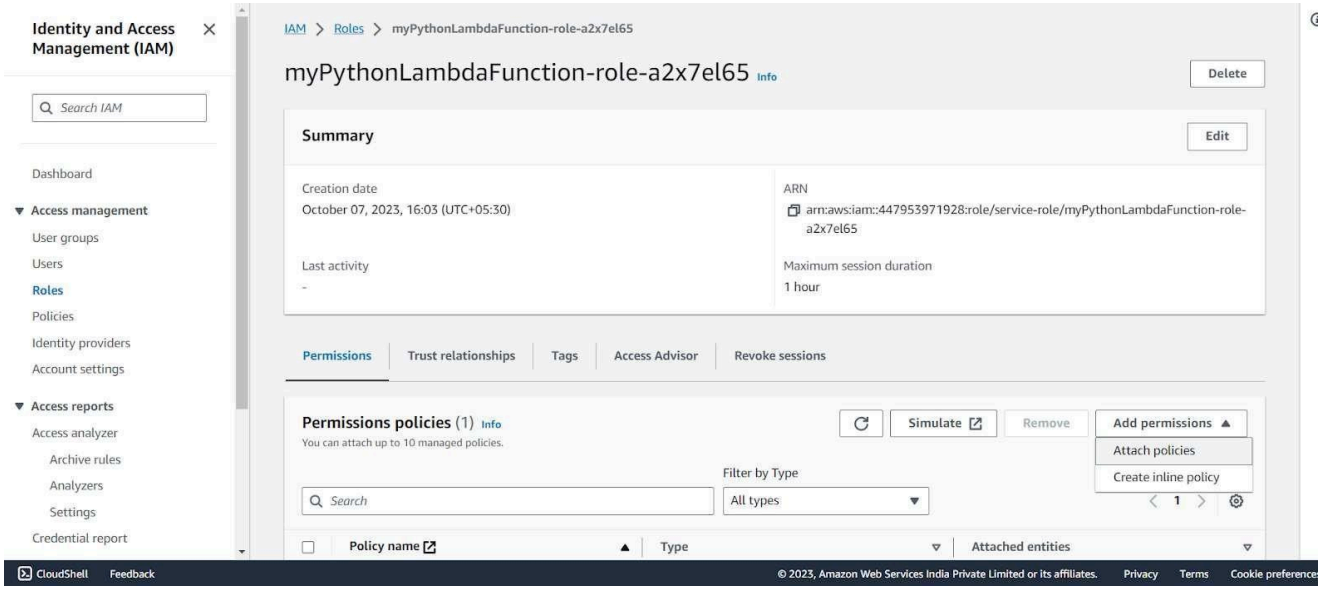
Adv. DevOps Exp. 12

Name-Aryan Anil Patankar
Class-D15A
Roll No-34

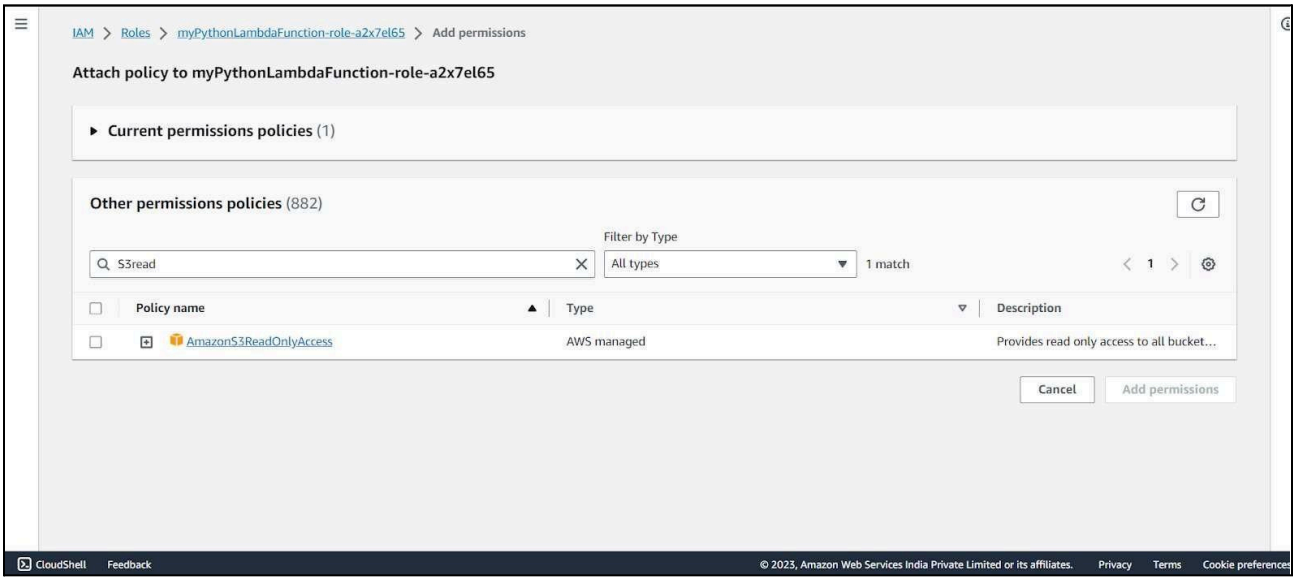
Step 1: Open up the IAM Console and under Roles, choose the Role we previously created for the Python Lambda Function (You can find your role name configuration of your Lambda function).



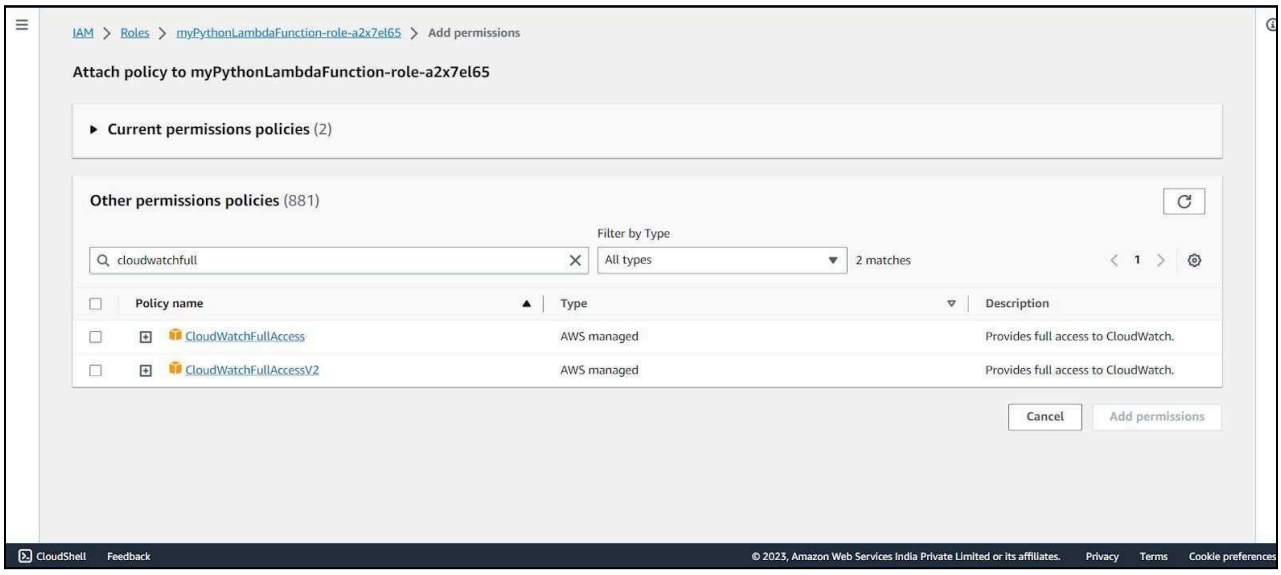
Step 2: Under Attach Policies, add S3-ReadOnly and CloudWatchFull permissions to this role.



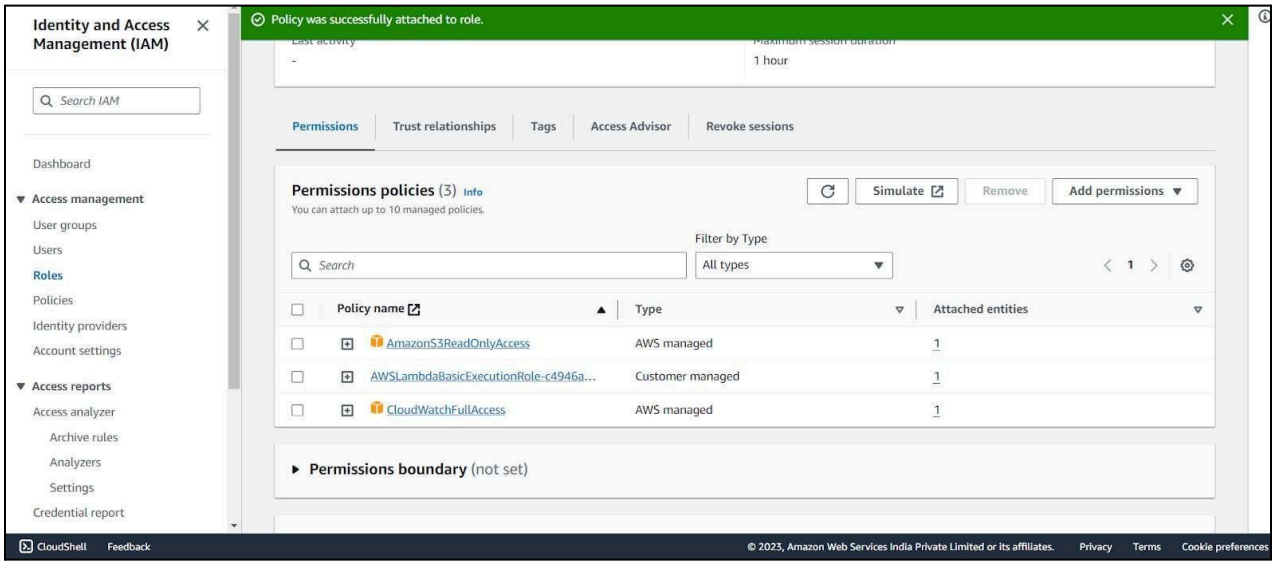
S3-ReadOnly

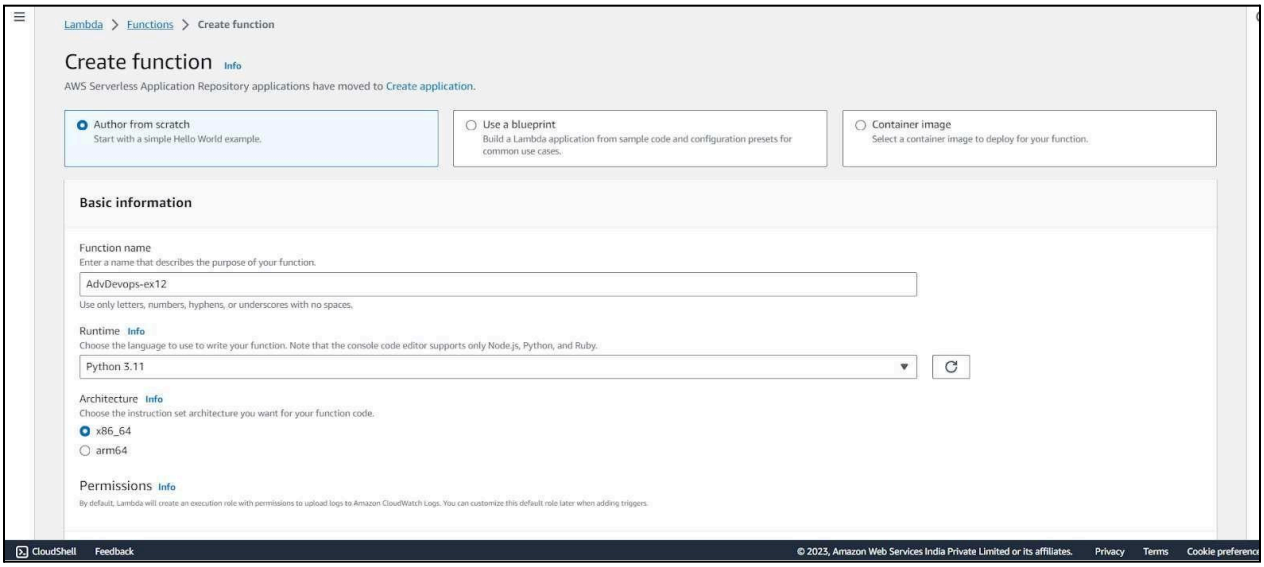


CloudWatchFull



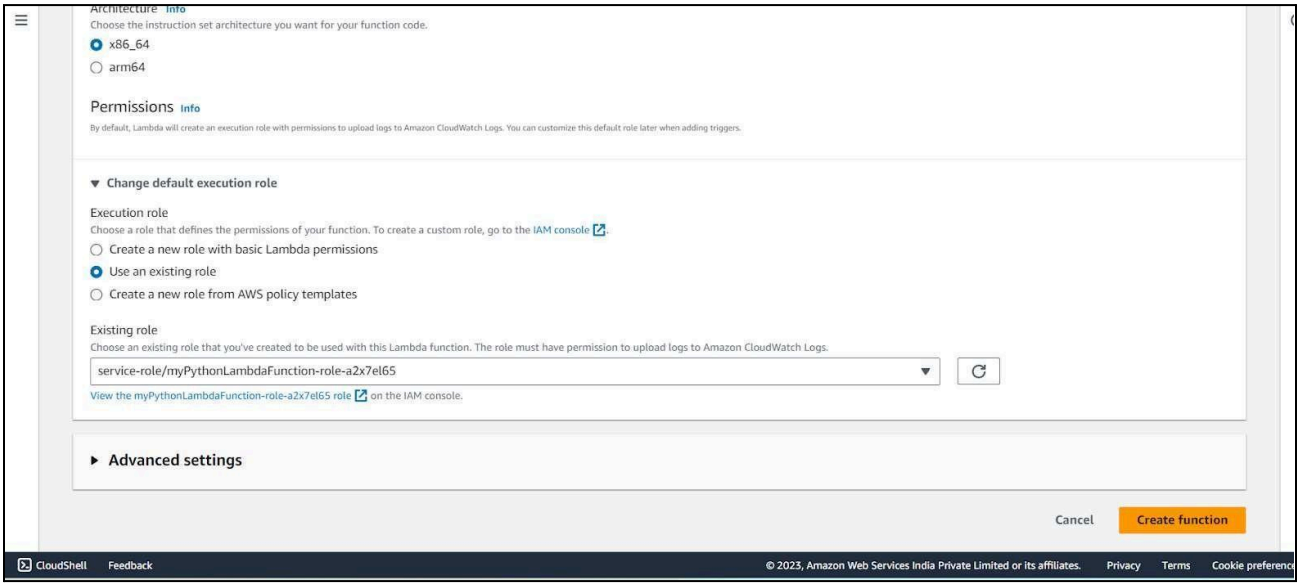
After successful attachment of policy you will see something like this you will be able to see the updated policies.



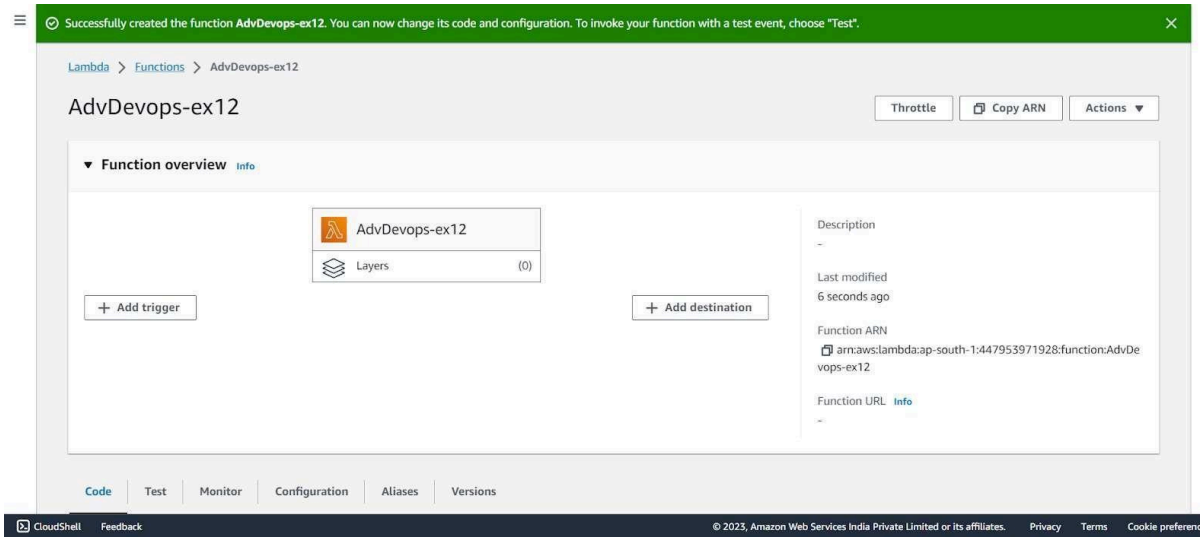


Step 3: Open up AWS Lambda and create a new Python function.

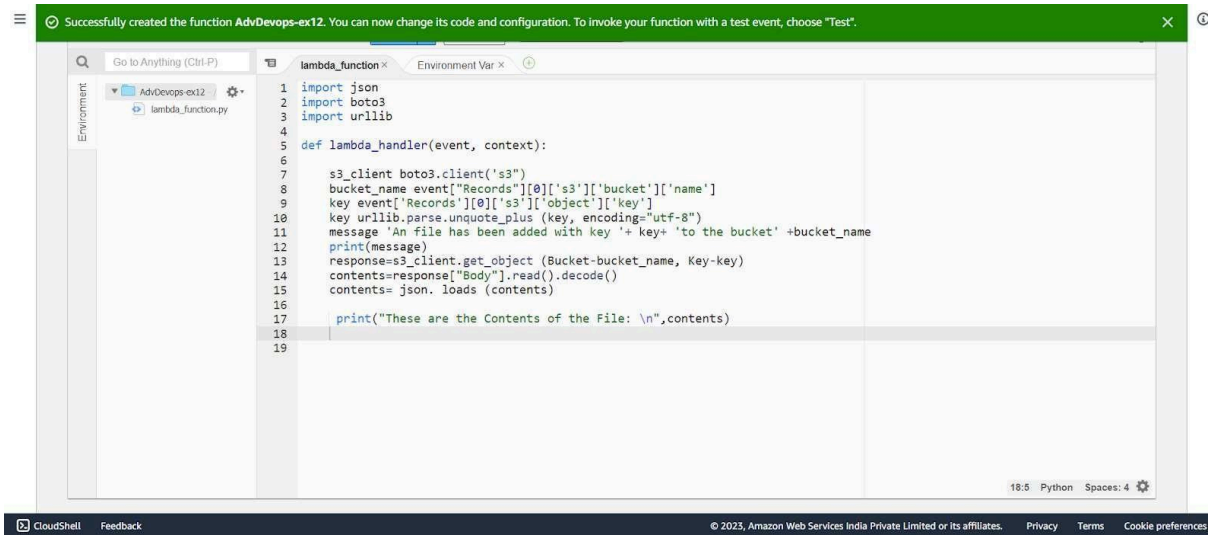
Under Execution Role, choose the existing role, then select the one which was previously created and to which we just added permissions.



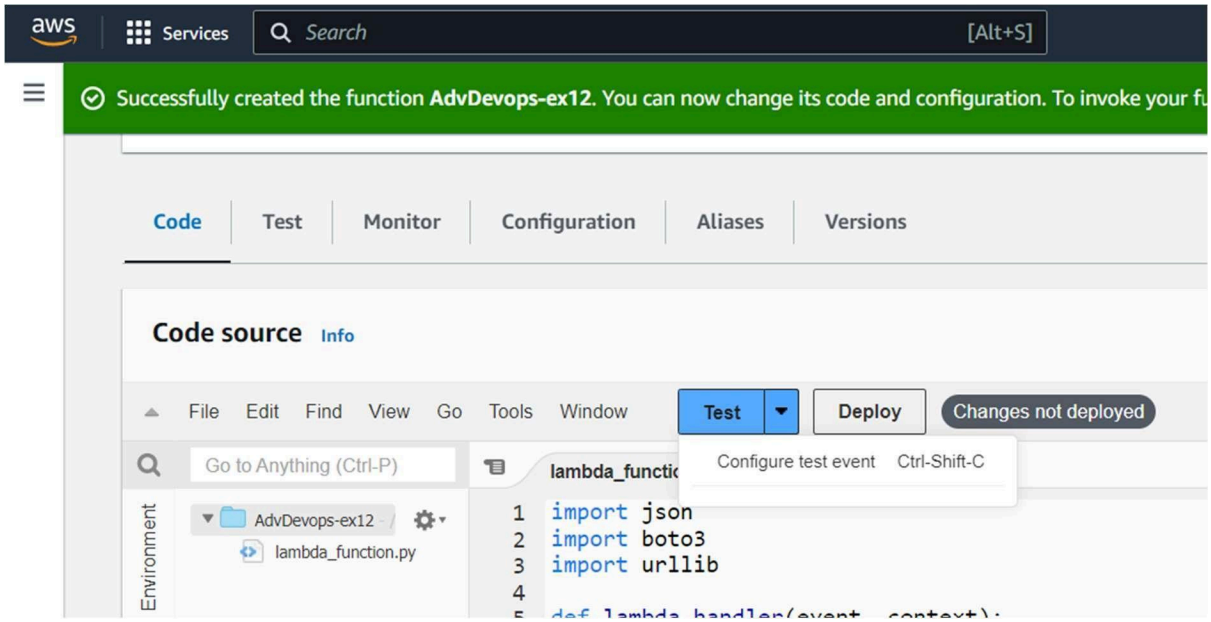
Step 4: The function is up and running.

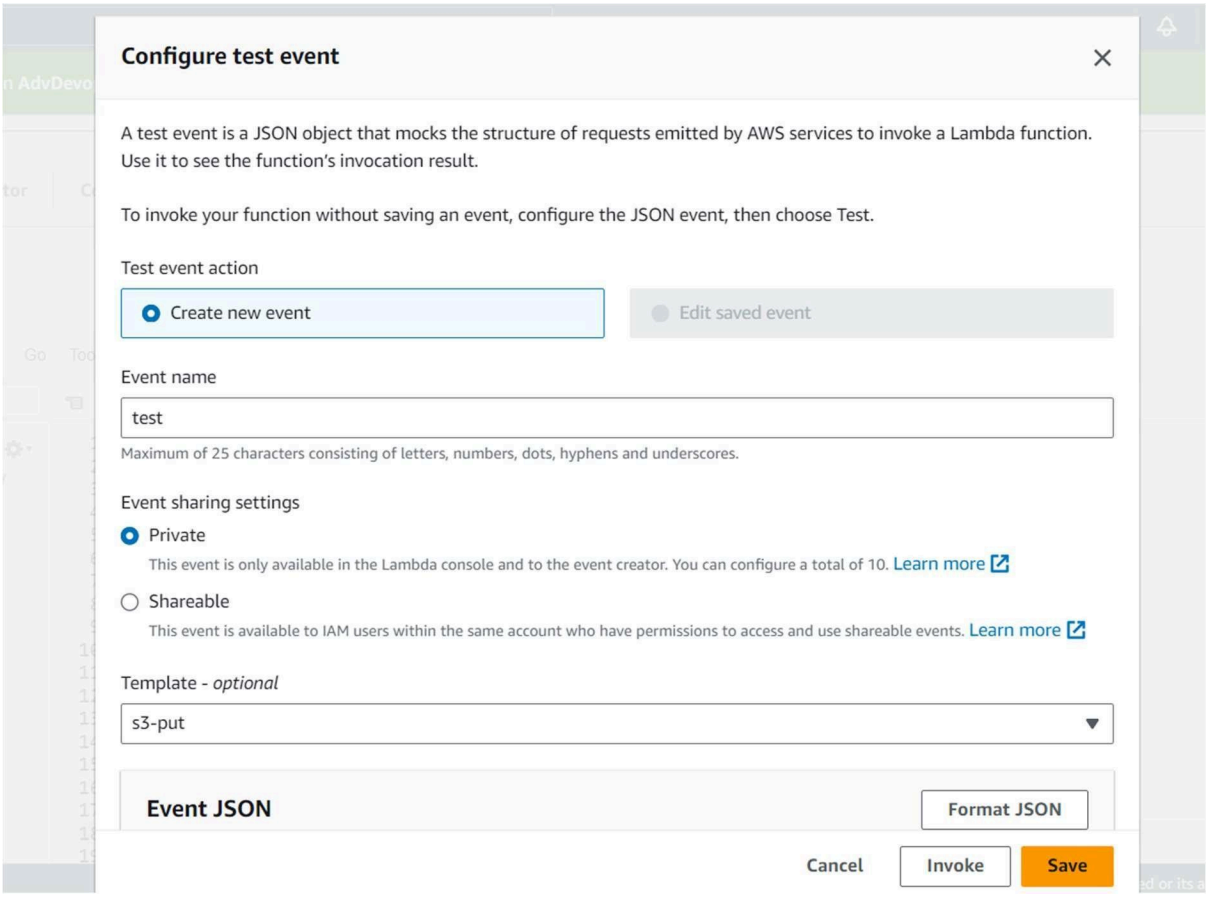


Step 5: Make the following changes to the function and click on the deploy button. This code basically logs a message and logs the contents of a JSON file which is uploaded to an S3 Bucket and then deploy the code.



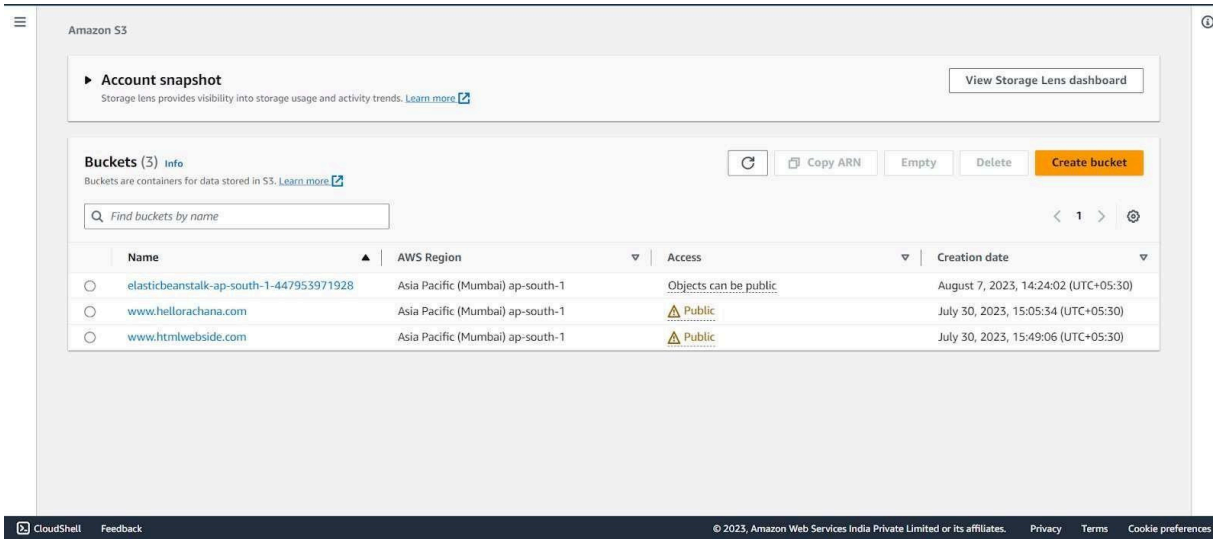
Step 6: Click on Test and choose the ‘S3 Put’ Template.



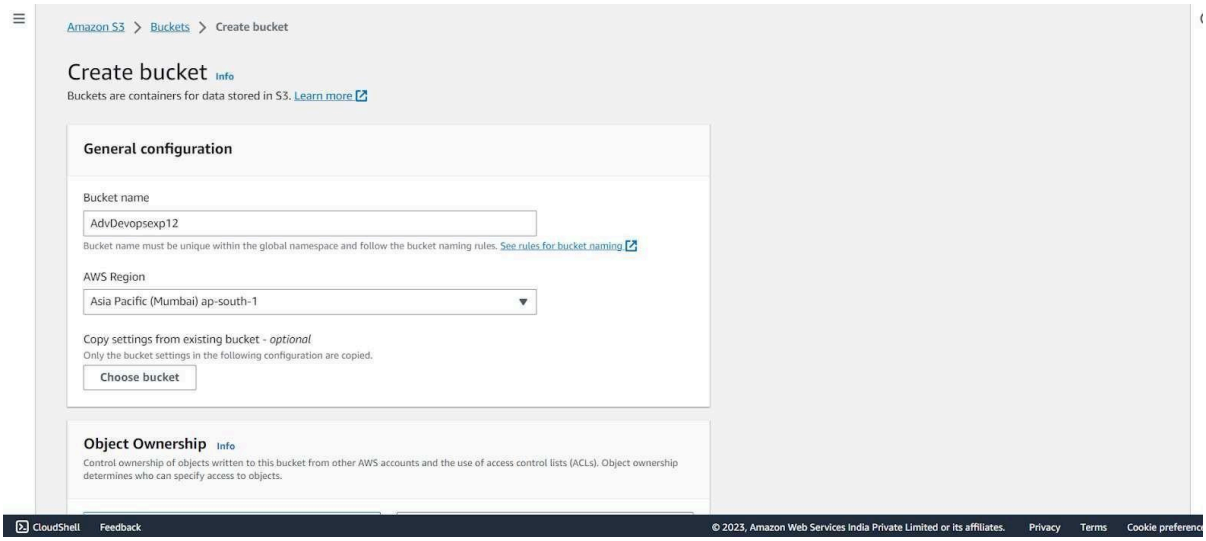


And Save it.

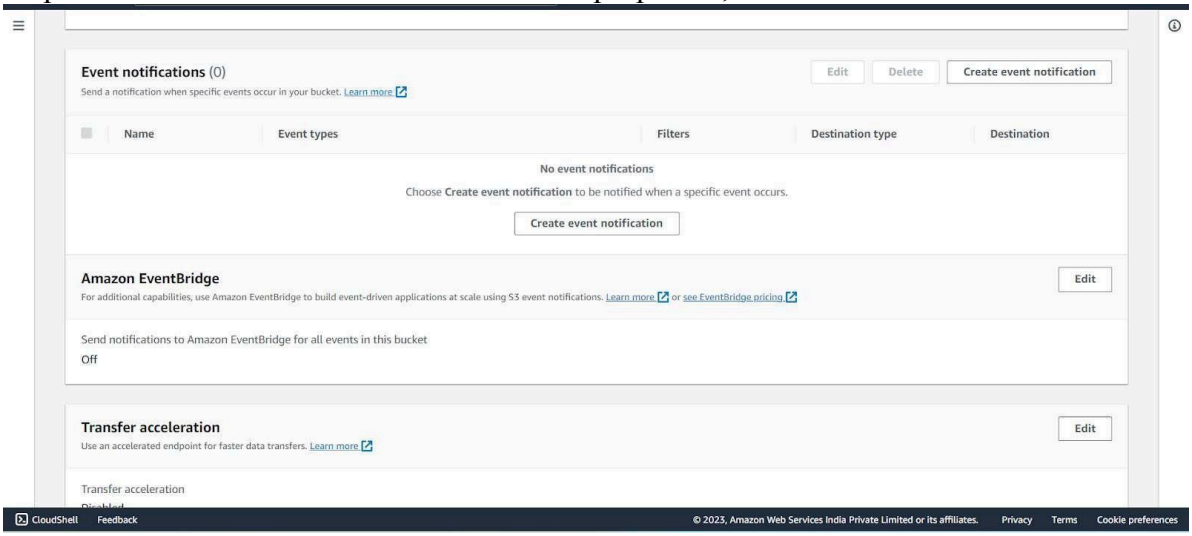
Step 7: Open up the S3 Console and create a new bucket.



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

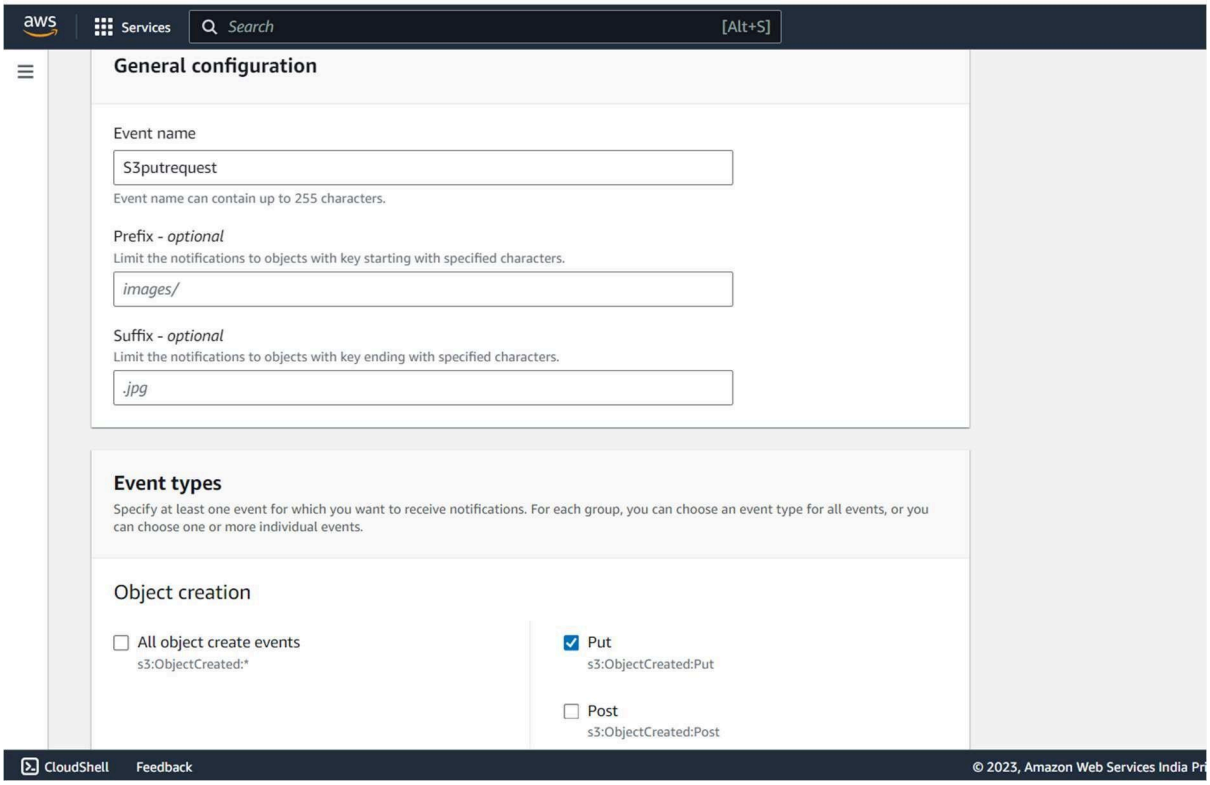


Step 9: Click on the created bucket and under properties, look for events.

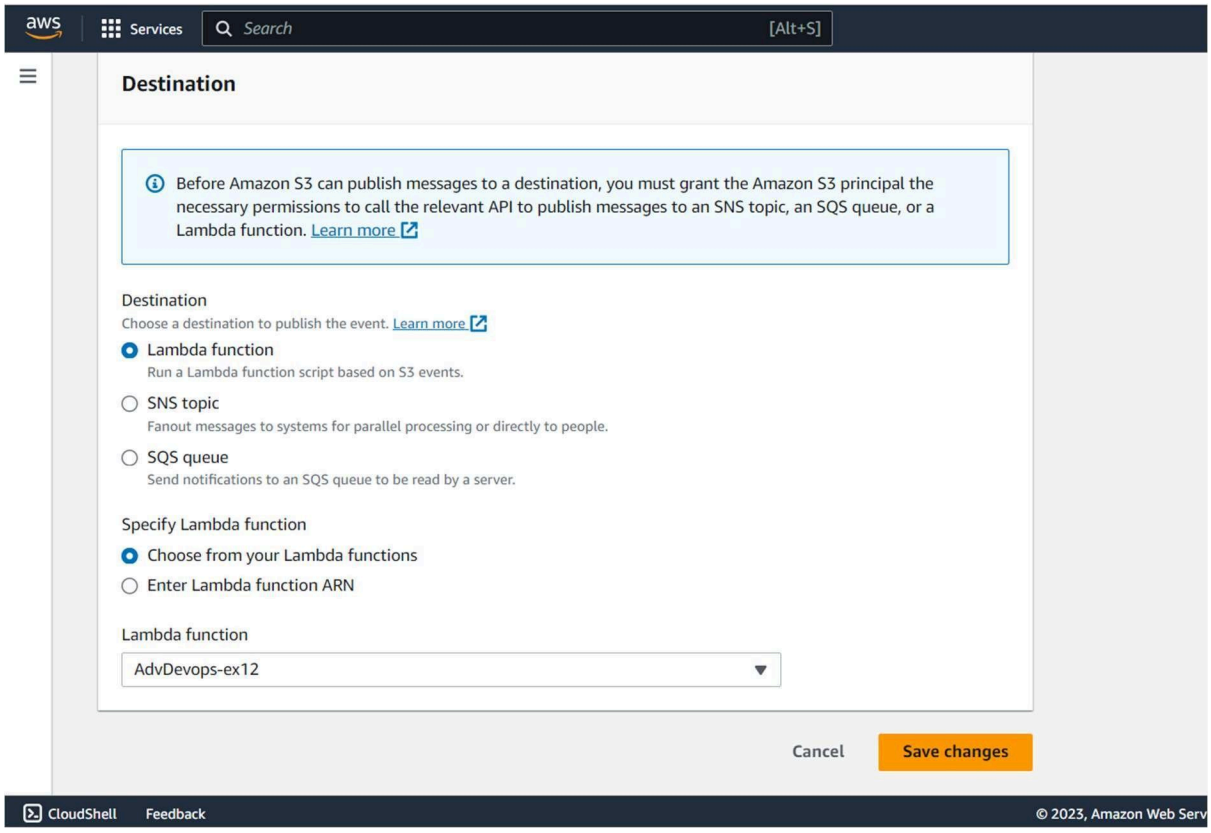


Click on Create Event Notification.

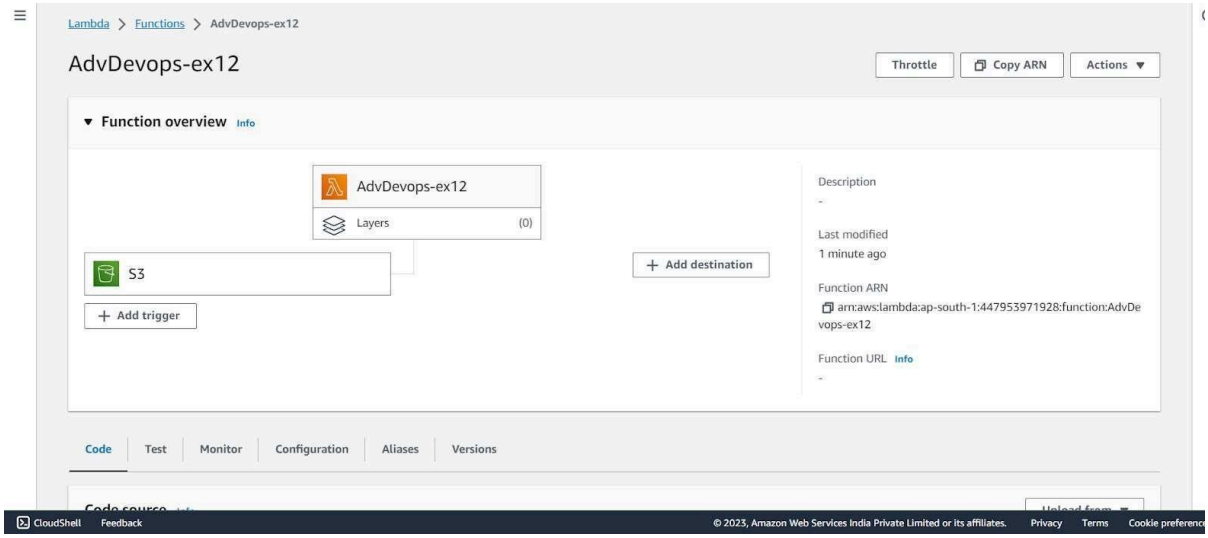
Step 10: Mention an event name and check Put under event types.



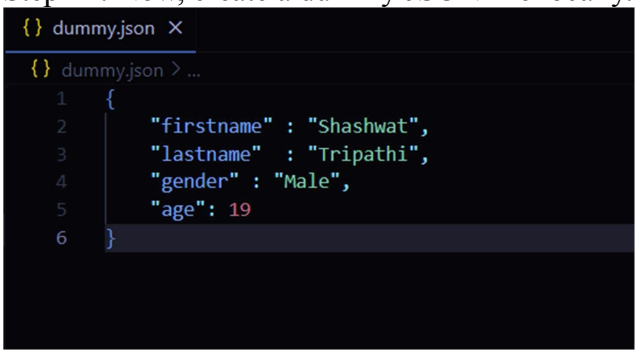
Choose Lambda function as destination and choose your lambda function and save the changes.



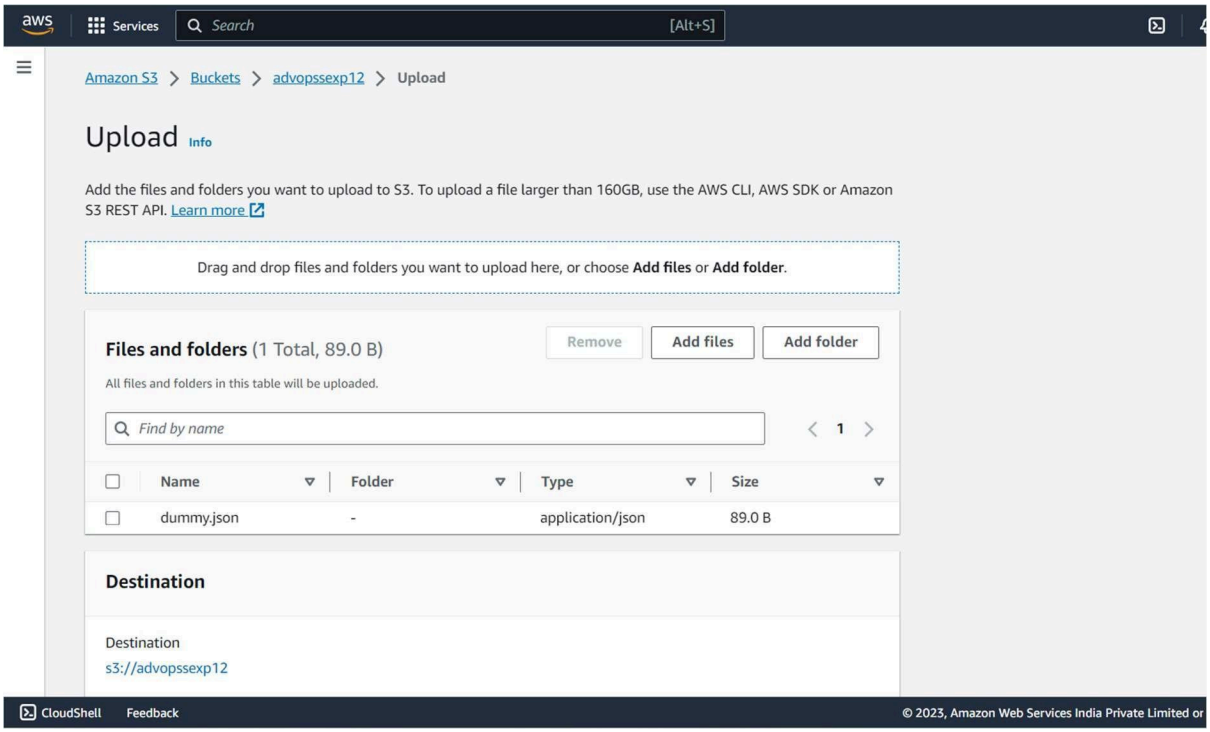
Step 11: Refresh the Lambda function console and you should be able to see an S3 Trigger in the overview.



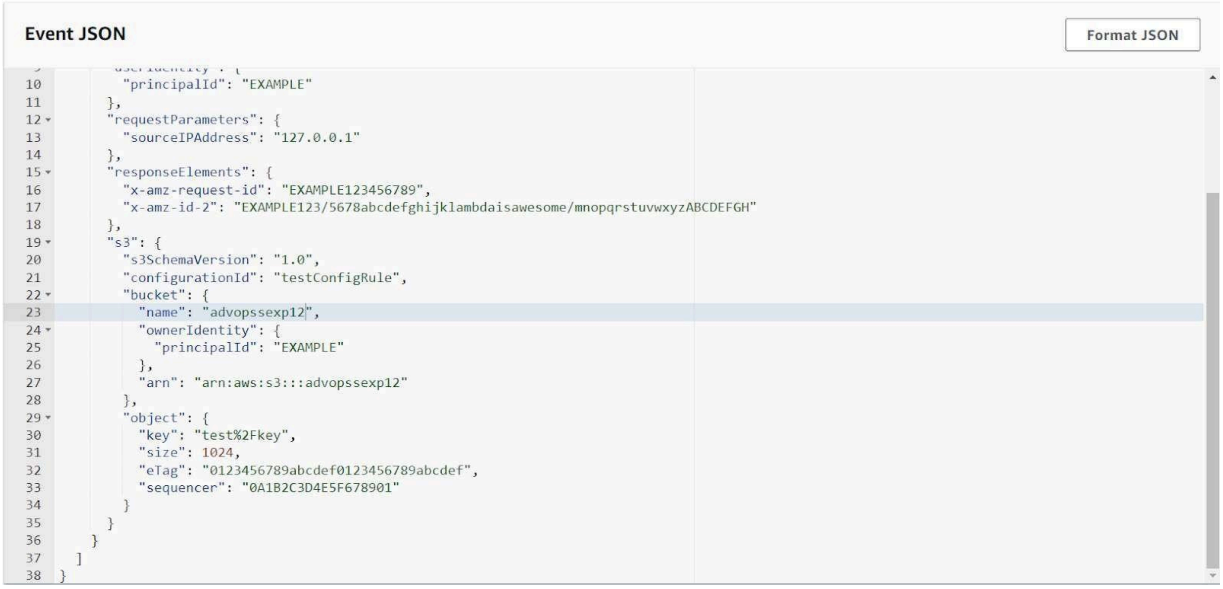
Step 12: Now, create a dummy JSON file locally.



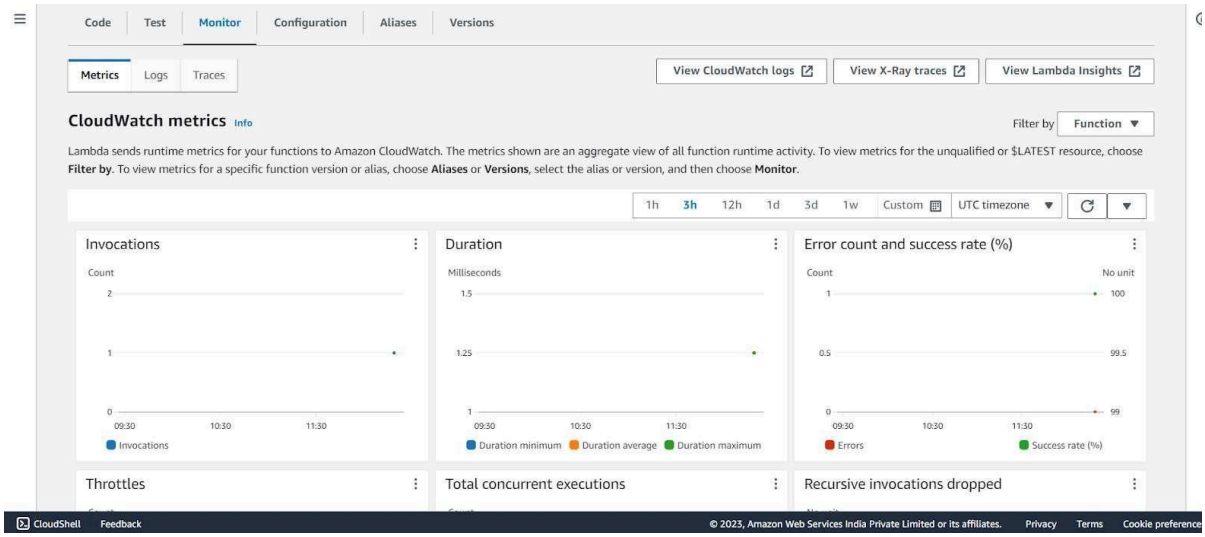
Step 13: Go back to your S3 Bucket and click on Add Files to upload a new file.
Step 14: Select the dummy data file from your computer and click Upload.



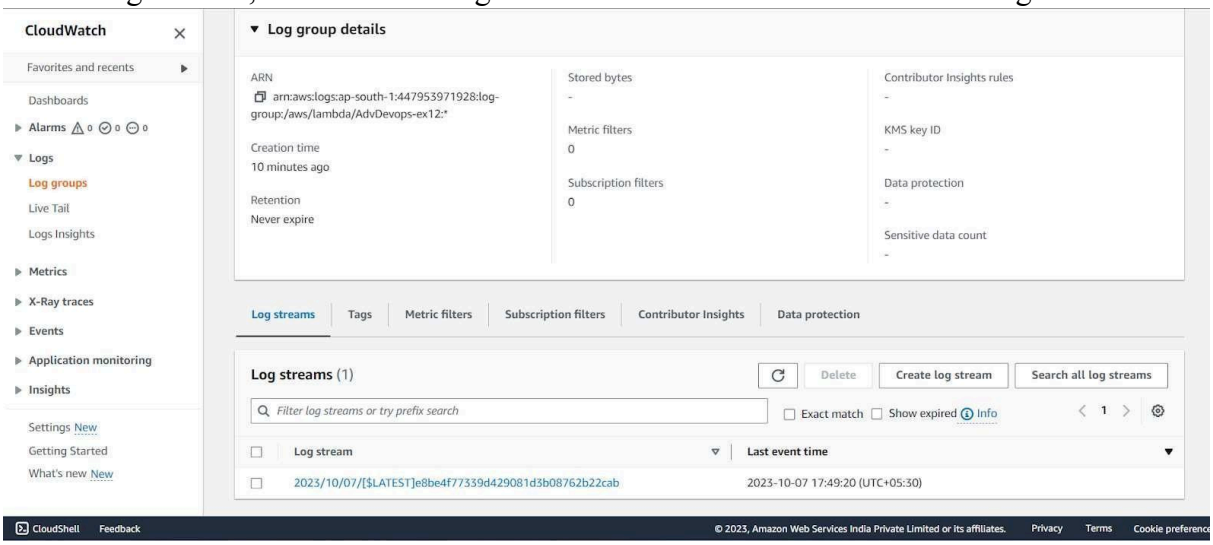
Step 15: After this make the necessary changes in the Test configuration file which we created it previously by replacing the Bucket Name and the ARN of Bucket.



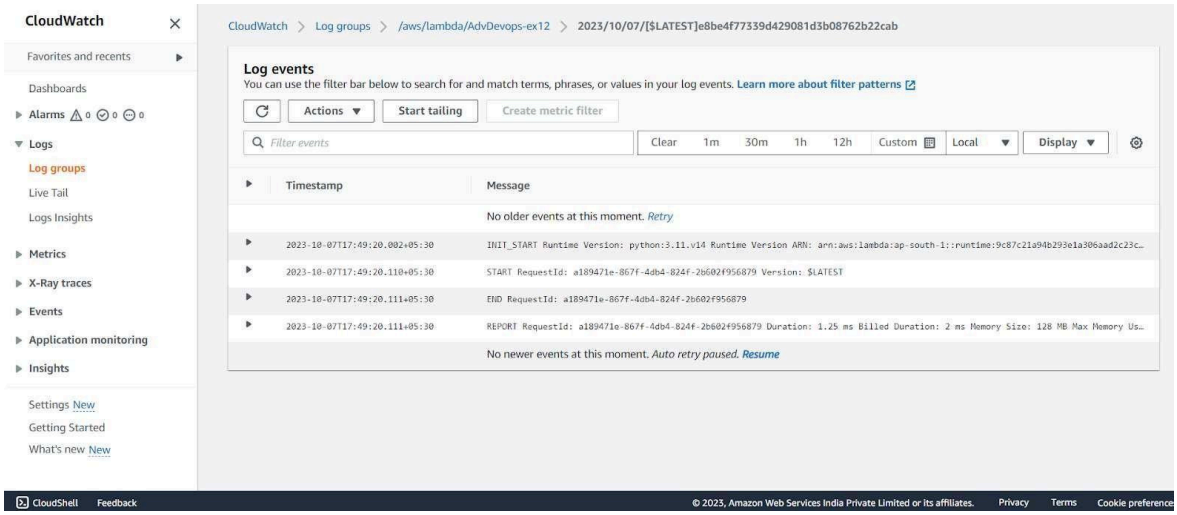
Step 16: Go back to your Lambda function , Refresh it and check the Monitor tab.



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.

Aryan Patankar
DISA
34

05/05

Advance Devops Assignment 1

Q.1 Use S3 bucket and host video streaming

Ans.

1. Create an S3 bucket
   i) Login to the AWS console
   ii) Navigate to Amazon S3, click Create Bucket and set a unique name
   iii) Configure permissions for public access if necessary
2. Upload Video Files
   i) Go to your bucket and upload video files (eg. MP4)
   ii) Set public access permissions for the uploaded files
3. Generate Pre-signed URLs
   i) If you want to restrict access, generate pre-signed URLs using the AWS CLI or SDK for limited-time access
4. Set up a CloudFront for streaming
   • Create a CloudFront Distribution with your S3 bucket as the origin
   • Use the CloudFront domain to improve video distribution and delivery performance
5. Configure Video Player
   • Embed the video using an HTML5 video player or Javascript player:

```
<video width = "600" controls>
  <source src = "https://your-bucket-name.s3.amazonaws.com/your-video.mp4"
   type = "video/mp4">
</video>
```

6. Monitor and Optimize
   • Use CloudWatch to monitor S3 and CloudFront performance.
   • Compress video files or convert them to HLS for better streaming
7. Security
   • Use CloudFront Signed URLs or Signed Cookies for access control.
   • Enable HTTPS and encryption for secure delivery.

8- Manage Costs
- Track storage and delivery costs in the AWS Billing Dashboard

Q.2) Discuss BMW and Hotstar case studies using AWS

Ans. BMW case study

1. Data collection and management: BMW collects vast amount of data from millions of connected cars, leveraging AWS IoT to manage and analyze the data effectively.
2. Real-Time Data Processing: BMW uses AWS services like Amazon Kinesis to process real-time telemetry data, providing instant insights for services like navigation, traffic alerts and remote diagnostics
3. Scalability: AWS scalable cloud infrastructure allows BMW to handle the growing data volumes from its global fleet of connected vehicles without compromising performance
4. Machine Learning and AI: BMW utilizes AWS's machine learning services, including Amazon sagemaker for predictive analytics and personalized driving experience, optimizing vehicle performance based on user behaviour.
5. Security Compliance: AWS offers end-to-end encryption, protecting sensitive vehicle and driver data, ensuring compliance with global data protection regulations.
6. Innovation in Autonomous Driving: BMW leverages AWS's high performance computing (HPC) and data analytics capabilities to drive innovations in autonomous driving technologies.

- Disney+ Hotstar case study

1. High Scalability During Peak Events: AWS allows Disney+ Hotstar to scale on demand, handling upto 25 million concurrent viewers during live events like the Indian Premier League (IPL).
2. Content Delivery Network (CDN): AWS CloudFront ensures that Hotstar's video content is delivered with low latency and minimal buffering,

enhancing user experience globally.
3. Cost Efficiency: AWS pay-as-you-go pricing model helps Disney+ Hotstar reduce infrastructure costs, scaling resources only when needed during peak viewing times
4. Global Reach with Edge Locations: With AWS's network of edge locations, Hotstar ensures fast and reliable content delivery to viewers, regardless of their geographic locations.
5. Live Streaming and VOD: AWS enables seamless streaming of both live events and video-on-demand (VOD) content, delivery of high-quality video to millions of users on various data
6. Real Time Data Analytics: Hotstar uses AWS analytics such as Amazon Kinesis and Amazon Cloudwatch to monitor viewer engagement and platforms performance in real-time, enabling quick optimizations.

Q-3 Why Kubernetes and advantages of Kubernetes and its disadvantages. Explain how Adidas uses Kubernetes.

Ans. Kubernetes is open source platform designed to automate deployment, scaling and management of containerized applications.
Advantages of Kubernetes:
1. Portability: Applications can be moved easily between different environments (development, testing, production) without major changes
2. Scalability: Kubernetes can automatically scale applications up or down based on traffic and demand.
3. Reliability: It features self-healing capabilities, meaning it can restart failed containers and balance workloads to ensure high availability.
4. Self-healing: kubernetes health of containers and automatically restarts, replace containers if they fail
5. Efficiency: It optimizes resource usage by running multiple containers on

on a single host, improving overall efficiency.

Disadvantages of Kubernetes

1. Complexity: It can be calculated and complicated to set up and manage, especially for those new to container technology
2. Steep Learning Curve: Requires time and knowledge to fully understand and utilize its features
3. Resource Intensive: It may require more computing resources than simpler solutions, which can increase costs
4. Management Overhead: Requires ongoing management and maintenance, which can add to operational workload.

- How Adidas Uses Kubernetes

Adidas has adopted Kubernetes to enhance its IT infrastructure and improve its ability to respond to market needs. Here's how they benefit from Kubernetes

1. Faster Application Development: Kubernetes streamlines the deployment process, allowing Adidas to bring new products and features to market quickly.
2. Operational Efficiency: It automates many manual tasks, reducing the amount of time and effort required to manage applications and increasing reliability.
3. Scalability for Demand: During peak sales periods, Kubernetes helps Adidas scale its applications to handle increased customer traffic smoothly.
4. Encouraging Innovation: With a flexible platform, Adidas can experiment with new technologies and business ideas without significant risk.

Specific Use case at Adidas:

- Microservices Architecture: Adidas breaks down application into smaller, independent services that can be deployed and managed individually.
- Continuous Delivery: Kubernetes supports a continuous delivery pipeline, making it easier to build, test and deploy applications quickly.

Q.4. What are Nagios and explain how Nagios are used in E-services

Ans. Nagios is an open-source monitoring platform/ designed to oversee system, networks and infrastructure. It helps organization identify and resolve IT infrastructure problems before they impact critical business processes.

Nagios used in E-services:
Publicly available services such as HTTP, FTP, SMTP, etc. These services are network accessible services like web services, email servers while private services need intermediary agents for monitoring
Nagios used plugins to monitors E-services many of which came pre-installed and additional plugins can be found online or developed by users. To monitor a service, a host must first be defined in Nagios configuration files. Once host is defined services like HTTP, FTP or SSH can be monitored by associating them with specific plugins. Nagios provide alert if services failed to respond within defined time frames or if errors are detected.
When issues arise, Nagios integrates with incident management tools to streamline the process of resolving problems. This helps team quickly address and fix issues, minimize downtime.
Nagios can monitor the performance of applications from the user's perspective ensuring that response times are fast and services are running smoothly.

By implementing Nagios, e-services can maintain high availability, enhance reliability and ensure a positive experience for users. This monitoring capability is crucial in today's digital landscape, where any downtime can lead to lost revenue and customer dissatisfaction.

Aryan Patankar
DISA
34

Advance Devops Assignment 2

Q.1 create a REST API with serverless framework

Ans. Creating REST API with serverless framework is an efficient way to deploy serverless applications that can scale automatically without managing servers.

(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 and business logic

(iii) REST API: Representational state transfer is architecture style for designing network applications.

Steps for creating REST API for serverless framework:

1) Install serverless framework:
You start by installing serverless framework CLI globally using node package manager (npm). This allow you to manage serverless applications directly from your terminal.

2) Creating a Node.js serverless project:
A directory is created for your project, where you will initialize a serverless service (project). This service will house all your lamba functions, configurations and cloud resources. Using the command serverless create you set up a template for AWS Node.js microservices that will eventually deploy to AWS Lambda.

3) Project Structure:
The project scaffold creates essential files like handler.js (which contains code for Lambda functions) and serverless.yml.

4) Create a REST API Resource:

In the serverless.yml file you define function that handles post requests of HTTP

5) Deploy the service:
With the 'sls deploy' commands serverless framework packages your applications, uploads necessary resources to AWS and set up the infrastructure

6) Testing the API: Once deployed you can test REST API using tools like curl or Postman by making post requests to generated API.

7) Storing data in DynamoDB: To store submitted candidate data you integrate AWS DynamoDB as a database

8) ~~Add~~ Adding more functionalities: Adding functionalities like 'list all candidates, get candidates by ID'

9) AWS IAM Permissions
You need to ensure that serverless framework is given right permissions to interact with AWS resources like DynamoDB

10) Monitoring and Maintenance
After deployment serverless framework provides service information like deployed endpoints, API key, log streams.

Q.2 Case study for SonarQube
Creating your own profile in sonarqube for testing project quality. Use sonarqube to analyze your github code. Install sonarlint in your Java intelly Ide and analyze java code. Analyze python project with sonarqube.

→ SonarQube is an open source platform used for continuous inspection of co

quality. It detects bugs, code smells and security vulnerabilities in project across various programming languages.

1) Profile creation in SonarQube:
Quality profile in SonarQube are essential configurations that define rules applied during code analysis. Each project has a quality profile for every supported language with default being 'sonar way' profile comes built in for all languages. Custom profiles can be created by copying or extending existing ones. Copying creates an independent profile, while extending inherit rules from parent profile and reflects future changes automatically. You can activate or deactivate rules, prioritize certain rules and configure parameter to tailor profile to specific projects. Permissions to manage quality profile are restricted to users with administrative privileges. SonarQube allows for the comparison of two profiles to check for differences in activated rules and users can track changes via event log. Quality profiles can also be imported from other instances via backup and restore. To ensure profiles include new rules its important to check against updated built in profiles or use SonarQube rules page.

2) Using SonarCloud to analyze GitHub code:
SonarCloud is cloud-based counterpart of SonarQube that integrates directly with GitHub, BitBucket, Azure and Gitlab repositories. To get started with SonarCloud via GitHub signup via SonarCloud product page and connect your GitHub organization or personal account. Once connect, SonarCloud mirrors your GitHub setup with each project corresponding to GitHub repository. After setting up the organization choose subscription plan (free for public repo). Next, impo repositories into your SonarCloud organization where each GitHub

repo becomes a sonarcloud project. Define 'new code' to focus on recent changes and choose between automatic analysis or CI-based analysis. Automatic analysis happens directly in sonarcloud, while CI based analysis integrates with your build process once the analysis is complete result can be viewed in both sonar-cloud and github including security import issue.

3) Sonarlint in Java IDE:
Sonarlint is an IDE that performs on-the-fly code analysis as you write code. It helps developers detect bugs, security vulnerabilities and code smells directly in the development environment such as Intellij Idea or Eclipse. To set it up, install the SonarLint plugin, configure the connection with sonarqube or sonarcloud and select the project profile to analyze Java code. This approach ensures immediate feedback on code quality, promoting clean and maintainable code from beginning.

4) Analyzing Python Projects with SonarQube:
SonarQube supports Python test coverage, reporting but it requires third party tool like coverage.py to generate the coverage port. To enable coverage adjust your blind process so that coverage tool runs before sonar scanner and ensures report file is saved in different path.
For setup you can use Tox, PyTest and coverage.py to configure and run test. In your tox.ini include configurations for pytest and coverage to generate coverage report in xml format. The build process can also be automated using Github Actions, which install dependencies, runs, tests and invoke SonarQube scan. Ensure report in. coberwrata XML format and path where scanner can access it.

5) Analyzing Node.js projects with SonarQube.
For Node.js project SonarQube can analyze Javascript and Typescript code. Similar to the python setup, you can configure SonarQube to analyze node.js projects by installing the appropriate plugins and using SonarScanner to scan the projects. SonarQube will check the code against Industry standard rules and best practices, flagging issues related to security vulnerabilities bugs and performance optimization.

3. At a large organization, your centralized operations team may get many repetitive infrastructure requests, you can use Terraform to build a "self-serve" infrastructure model that lets product teams manage their own infrastructure independently. You can create and use Terraform modules that codify the standards for deploying and managing services in your organization, allowing teams to efficiently deploy services in compliance with your organization's practices. Terraform Cloud can also integrate with ticketing system like ServiceNow to automatically generate new infrastructure requests.

Ans. Implementing a "self-serve" 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.

- The Need for self-service infrastructure:
In large organizations, centralized operations teams often face an overwhelming number of repetitive requests. This can lead to delays in service delivery and frustration among product teams who need to move quickly. A self-service model allows teams to provision and manage their infrastructure without relying on the operations team for every request

- Benefits of using Terraform

1. Modularity and Reusability:
- Terraform modules encapsulate standard configurations for various infrastructure components (eg. networks, databases, compute resources).
- Teams can reuse these modules across different projects, reducing redundancy and minimizing the risk of errors.

2. Standardizations
- By defining best practices within modules, organizations can ensure that all deployments comply with internal policies and standards.
- This consistency helps maintain security and operational integrity across the organization

3. Increased Efficiency
- Product teams can deploy services quickly by using pre-defined modules, significantly reducing the time spent on infrastructure setup.
- This allows team to focus on developing features rather than managing infrastructure

4. Integration with Ticketing Systems
- Terraform Cloud can integrate with ticketing systems like ServiceNow to automate the generation of infrastructure requests.
- This integration streamlines workflows by allowing teams to initiate requests directly from their ticketing platform, reducing manual intervention

→ Implementation steps

1. Identify Infrastructure components
- Begin by identifying which components of your infrastructure can be modularized (eg. VPCs, security groups, load balancers)

2. Develop Terraform modules.
- Create reusable modules that define the desired configurations and

resources.
- Ensure each module includes input variables for customization and outputs for integration with other modules.

3. Establish Governance and Best Practices:
- Define guidelines for module usage, versioning, and documentation to ensure clarity and maintainability.
- Encourage teams to contribute to module development and share improvements.

4. Testing and validation
- Implement a testing framework to validate module functionality before deployment.
- Use tools like terraform plan to preview changes and catch potential issues early.

- Best Practices for module management
- Utilize the terraform registry:
- Leverage existing community modules from the Terraform Registry to avoid reinventing solutions and ensure adherence to best practices.
- Version control: Implement versioning for your modules to track changes over time. This helps manage dependencies effectively and minimize disruptions during updates.
- Documentation: Maintain comprehensive documentation for each module including usage examples, input/output descriptions and any dependencies.
- Encourage collaboration: Foster a culture of collaboration by sharing modules across teams. This promotes consistency in deployments and facilitates knowledge within the organization.

By adopting a self-service infrastructure model with Terraform, organizations can empower product teams to efficiently manage their own infrastructure while ensuring compliance with established standards

This approach not only streamlines processes but also enhances agility in responding to changing business needs. Ultimately, it leads to a more responsive IT environment that supports innovation and growth within the organization.