SNEHAL A. PATIL D15A 39

Advance Devops:1

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

Theory:

Amazon EC2 (Elastic Compute Cloud)

Amazon Elastic Compute Cloud (EC2) is a cloud service that offers scalable compute power in the cloud. It enables developers to easily deploy and manage virtual servers, known as instances, providing flexibility in handling varying workloads.

Key Concepts of EC2:

1. **Instances:** Instances are virtual servers hosted on Amazon's EC2 platform, designed to run applications in the cloud. You can launch instances as needed and scale them up or down depending on your application's requirements.
2. **Amazon Machine Image (AMI):** An AMI is a blueprint used to create an instance. It includes a pre-configured operating system, application server, and application software, allowing for consistent and rapid deployment of instances.
3. **Instance Types:** EC2 offers a range of instance types, each optimized for specific use cases, such as compute-intensive tasks, memory-intensive applications, or storage-heavy operations. Each type varies in its combination of CPU, memory, storage, and network capacity.
4. **Elastic IP Addresses:** Elastic IPs are static IP addresses that can be associated with an EC2 instance. They are useful for maintaining a consistent IP address for your instance, even if the underlying instance changes.
5. **Security Groups:** Security groups function as virtual firewalls, controlling inbound and outbound traffic to and from your EC2 instances. You can define rules based on IP addresses, ports, and protocols to manage the traffic securely.
6. **Auto Scaling:** Auto Scaling enables automatic adjustments in the number of EC2 instances in response to current demand, ensuring that your application performs optimally while minimizing costs.

Amazon S3 (Simple Storage Service)

Amazon S3 is an object storage service offering high scalability, availability, security, and performance. It is ideal for storing and retrieving large amounts of data from anywhere on the web.

Key Concepts of S3:

1. **Buckets:** Buckets serve as containers in S3, where all objects are stored. Each bucket is unique across AWS and is used to organize and manage the stored data.
2. **Objects:** Objects are the basic units stored in S3. They consist of the data itself, metadata, and a unique key (identifier). Objects can range from documents to videos, images, and other file types.
3. **Keys:** A key is a unique identifier for an object within a bucket. Each object in S3 has a key that allows you to access and manage it.
4. **Versioning:** S3 supports versioning, which allows you to maintain multiple versions of an object within a bucket. This feature protects against accidental deletions or overwrites by preserving older versions.
5. **Access Control:** S3 provides various mechanisms for controlling access to data, such as bucket policies, access control lists (ACLs), and IAM policies. These tools ensure that only authorized users can access and manage your data.
6. **Lifecycle Management:** S3 lifecycle policies help automate the transition of objects between different storage classes or delete them after a specified period, optimizing storage costs over time.
7. **Storage Classes:** S3 offers multiple storage classes designed for different access patterns, such as S3 Standard for frequently accessed data, S3 Intelligent-Tiering for cost optimization, and S3 Glacier for long-term archival storage.

AWS Cloud9

AWS Cloud9 is a cloud-based integrated development environment (IDE) that allows you to write, run, and debug code directly from a web browser. It supports various programming languages and comes pre-configured with essential tools and libraries, streamlining the development process.

Key Features of AWS Cloud9:

1. **Cloud-Based IDE:** AWS Cloud9 provides a fully-featured development environment accessible through a browser, eliminating the need for local IDE installation and setup.
2. **Collaborative Development:** Cloud9 enables real-time collaboration, allowing multiple users to work on the same project simultaneously. Features like chat and simultaneous editing make it ideal for teamwork and pair programming.
3. **Pre-configured Environment:** Cloud9 comes with pre-configured tools and libraries for various programming languages and frameworks, enabling developers to start coding immediately without the hassle of setting up the environment.
4. **Seamless Integration with AWS Services:** Cloud9 integrates seamlessly with AWS services like EC2, S3, and Lambda, allowing developers to deploy and manage applications directly from the IDE.
5. **Terminal Access:** Cloud9 provides full terminal access to the underlying instance, enabling developers to run shell commands and manage their environment directly.

Implementation:

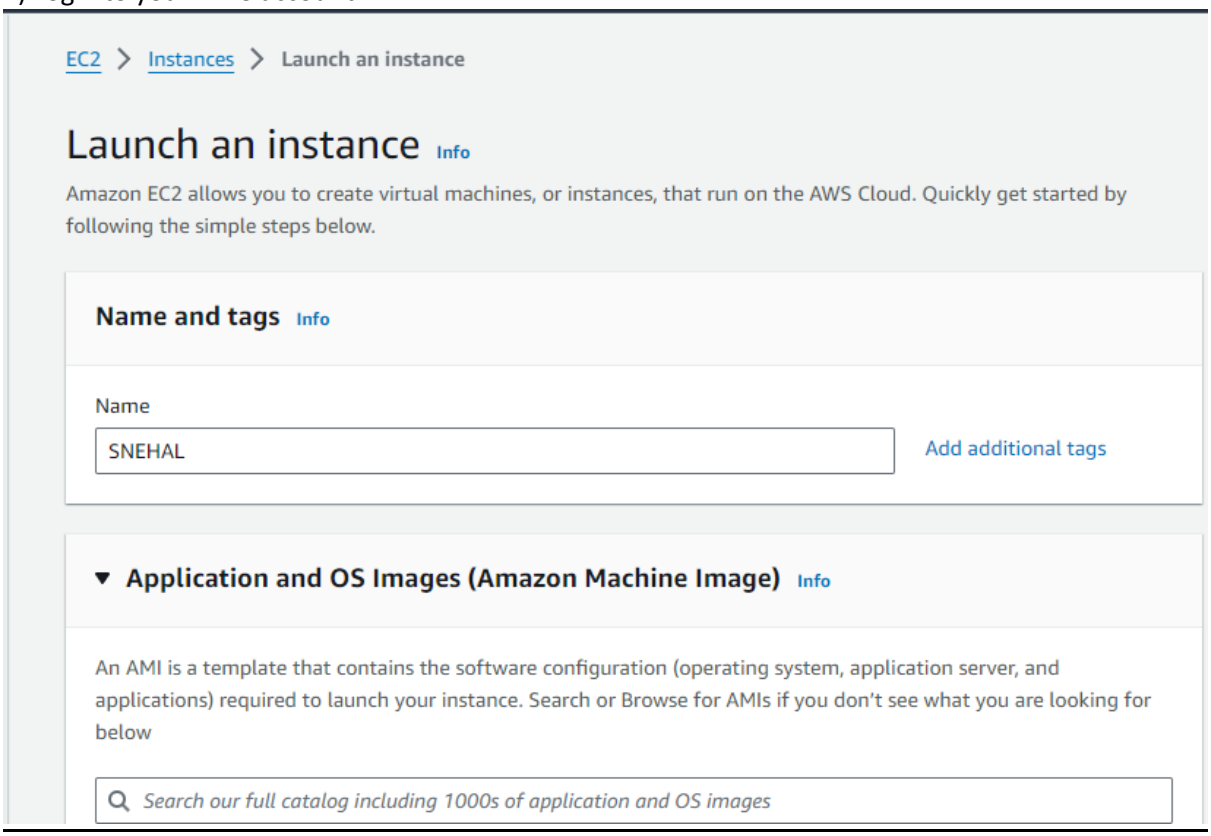
EC2 Instance Creation and Static Site Hosting

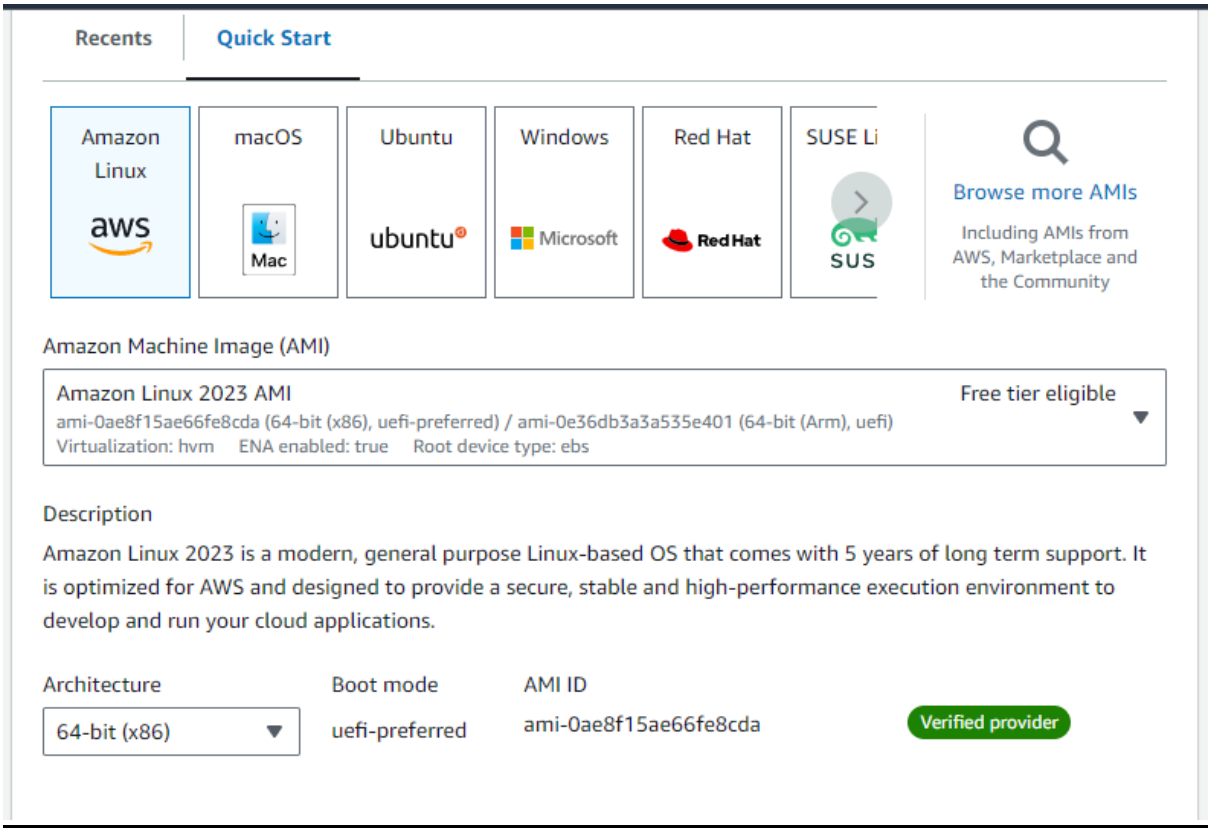
1. **Log in to your AWS account:** Begin by logging into your AWS account through the AWS Management Console.
2. **Navigate to EC2:** From the dashboard, click on the EC2 service to access the instance management console.
3. **Launch an Instance:** Click on the "Launch Instance" button to begin setting up a new EC2 instance. Choose an appropriate AMI, select an instance type, configure instance details, and add storage as needed.
4. **Configure Security Group:** Set up a security group to control inbound and outbound traffic to your instance, ensuring that your application is accessible but secure.
5. **Assign an Elastic IP (Optional):** To maintain a static IP address, you can allocate an Elastic IP and associate it with your instance.
6. **Connect and Deploy:** Once the instance is running, connect to it via SSH or the AWS Cloud9 terminal, and deploy your static website or application by uploading files to the web server directory.
7. **Access Your Site:** After deployment, your static site will be accessible via the instance's public IP address or domain name associated with the Elastic IP.

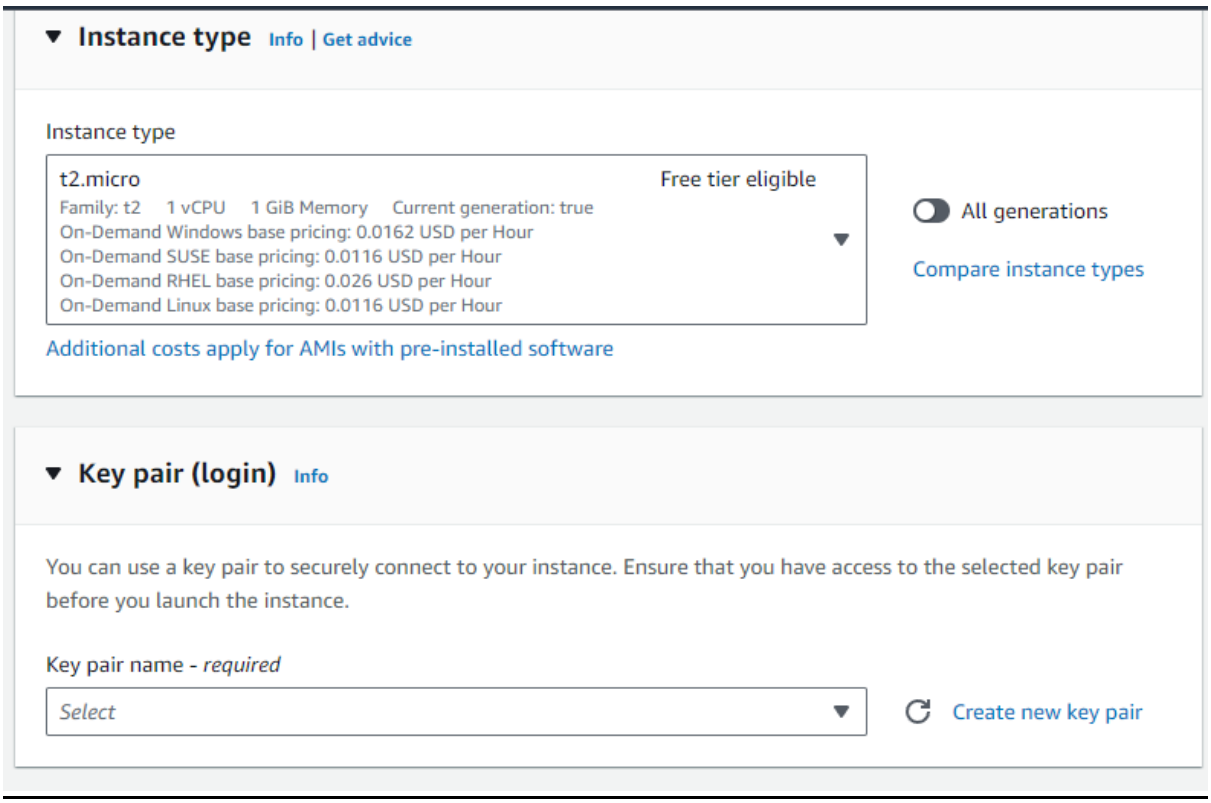
Implementation :

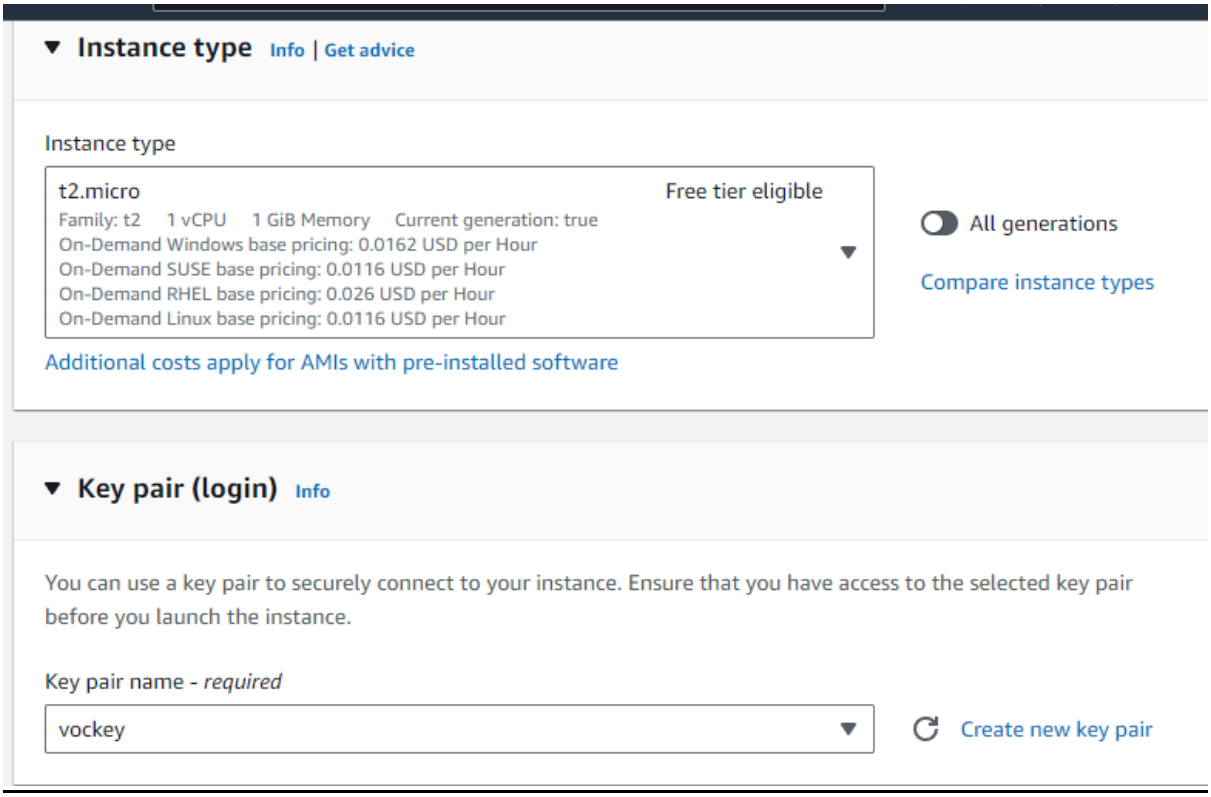
EC2 Instance Creation and static site hosting

1) Login to your AWS account

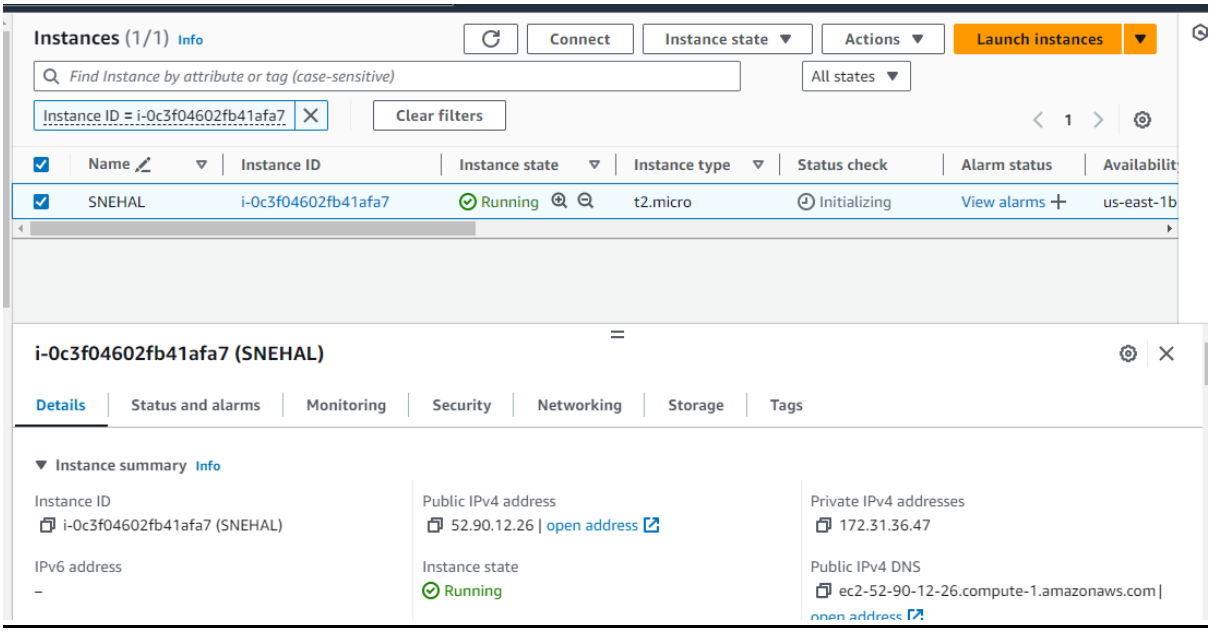



Recents
Quick Start
Amazon Linux
macOS
Mac
Ubuntu
ubuntu
Windows
Microsoft
Red Hat
Red Hat
SUSE Li
SUS
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)
Amazon Linux 2023 AMI
Free tier eligible
ami-0ae8f15ae66fe8cda (64-bit (x86), uefi-preferred) / ami-0e36db3a3a535e401 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs
Description
Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.
Architecture
Boot mode
AMI ID
64-bit (x86)
uefi-preferred
ami-0ae8f15ae66fe8cda
Verified provider


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



3) After an instance is created wait for it to come to Running state



Static Website hosting using EC2:

Follow the steps and then run the commands

```
See "man sudo_root" for details.

ubuntu@ip-172-31-41-61:~$ sudo su
root@ip-172-31-41-61:/home/ubuntu# sudo apt install
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@ip-172-31-41-61:/home/ubuntu# sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [294 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [68.1 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [3768 B]
Get:10 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [250 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [108 kB]
Get:12 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [8632 B]
Get:13 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [9412 B]
```

i-0dda4db17f16307ec (Snehal)

PublicIPs: 54.162.220.58 PrivateIPs: 172.31.41.61

```
Reading package lists... Done
root@ip-172-31-41-61:/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-sqlite3 libaprutil1-ldap libaprutil1t64 liblua5.4-0 ssl-cert
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1-dbd-sqlite3 libaprutil1-ldap libaprutil1t64 liblua5.4-0 ssl-cert
0 upgraded, 10 newly installed, 0 to remove and 53 not upgraded.
Need to get 2083 kB of archives.
After this operation, 8094 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libapr1t64 amd64 1.7.2-3.1build2 [107 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1t64 amd64 1.6.3-1.1ubuntu7 [91.9 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1-dbd-sqlite3 amd64 1.6.3-1.1ubuntu7 [11.2 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1-ldap amd64 1.6.3-1.1ubuntu7 [9116 B]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 liblua5.4-0 amd64 5.4.6-3build2 [166 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-bin amd64 2.4.58-1ubuntu8.4 [1329 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-data all 2.4.58-1ubuntu8.4 [163 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-utils amd64 2.4.58-1ubuntu8.4 [97.1 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2 amd64 2.4.58-1ubuntu8.4 [90.2 kB]
```

```
No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-41-61:/home/ubuntu# 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 12:30:09 UTC; 30s ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 2442 (apache2)
      Tasks: 55 (limit: 1130)
     Memory: 5.4M (peak: 5.7M)
        CPU: 40ms
     CGroup: /system.slice/apache2.service
             ├─2442 /usr/sbin/apache2 -k start
             ├─2445 /usr/sbin/apache2 -k start
             └─2446 /usr/sbin/apache2 -k start

Aug 18 12:30:09 ip-172-31-41-61 systemd[1]: Starting apache2.service - The Apache HTTP Server...
Aug 18 12:30:09 ip-172-31-41-61 systemd[1]: Started apache2.service - The Apache HTTP Server.
root@ip-172-31-41-61:/home/ubuntu# cd /var/www/html
root@ip-172-31-41-61:/var/www/html#
```

i-0dda4db17f16307ec (Snehal)

PublicIPs: 54.162.220.58 PrivateIPs: 172.31.41.61

EC2 > Security Groups > sg-0e7811c687e701e30 - launch-wizard-7

sg-0e7811c687e701e30 - launch-wizard-7

Actions

Details

Security group name	Security group ID	Description	VPC ID
launch-wizard-7	sg-0e7811c687e701e30	launch-wizard-7 created 2024-08-18T11:25:33.225Z	vpc-08963bc0f8afcd789

Owner	Inbound rules count	Outbound rules count
608111999703	1 Permission entry	1 Permission entry

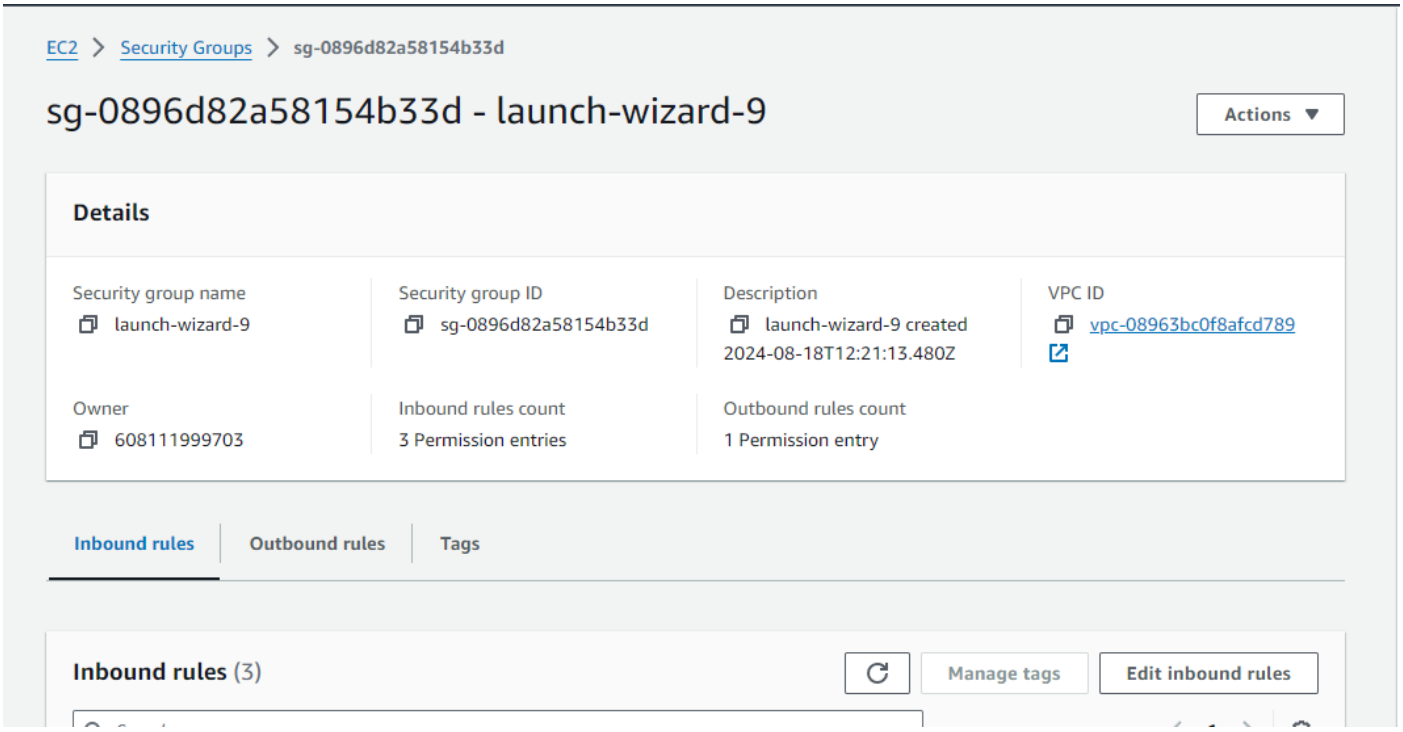
Inbound rules | Outbound rules | Tags

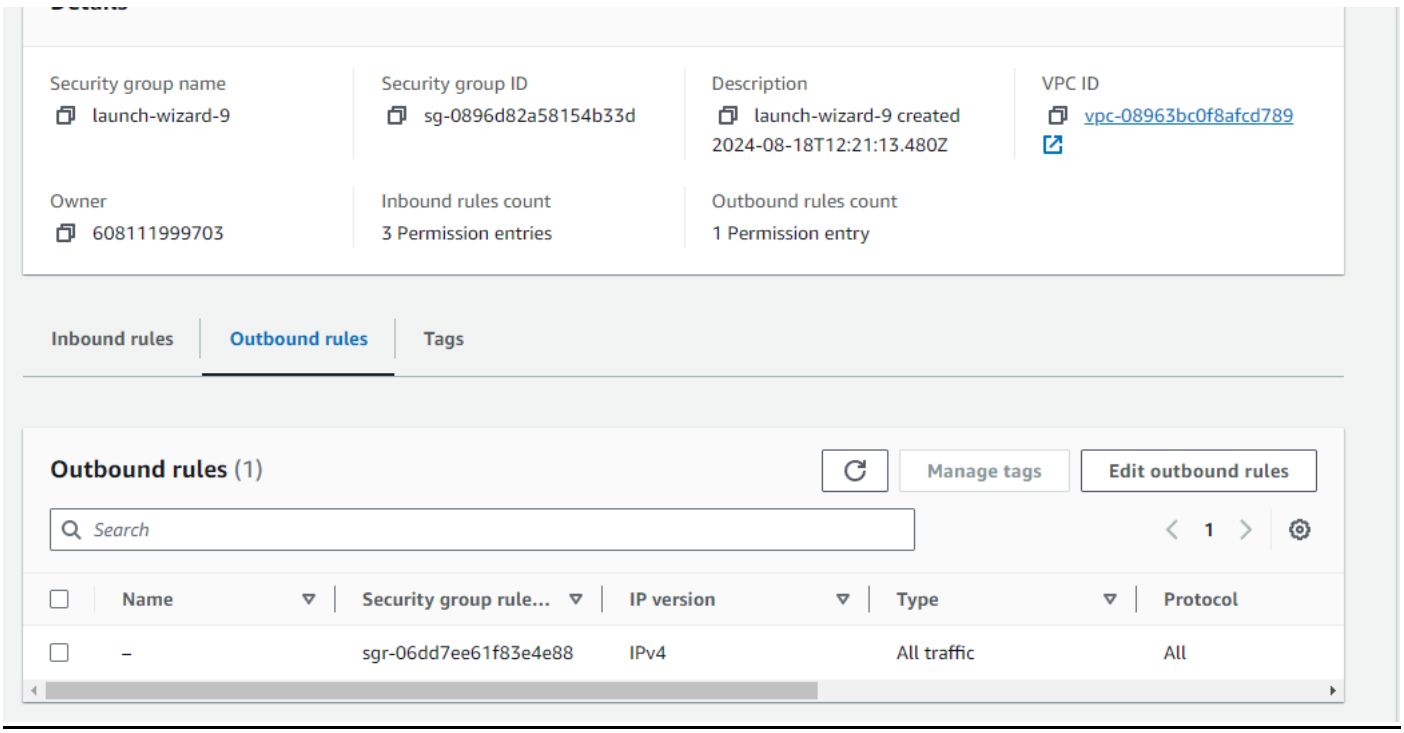
Inbound rules (1)

Manage tags | Edit inbound rules

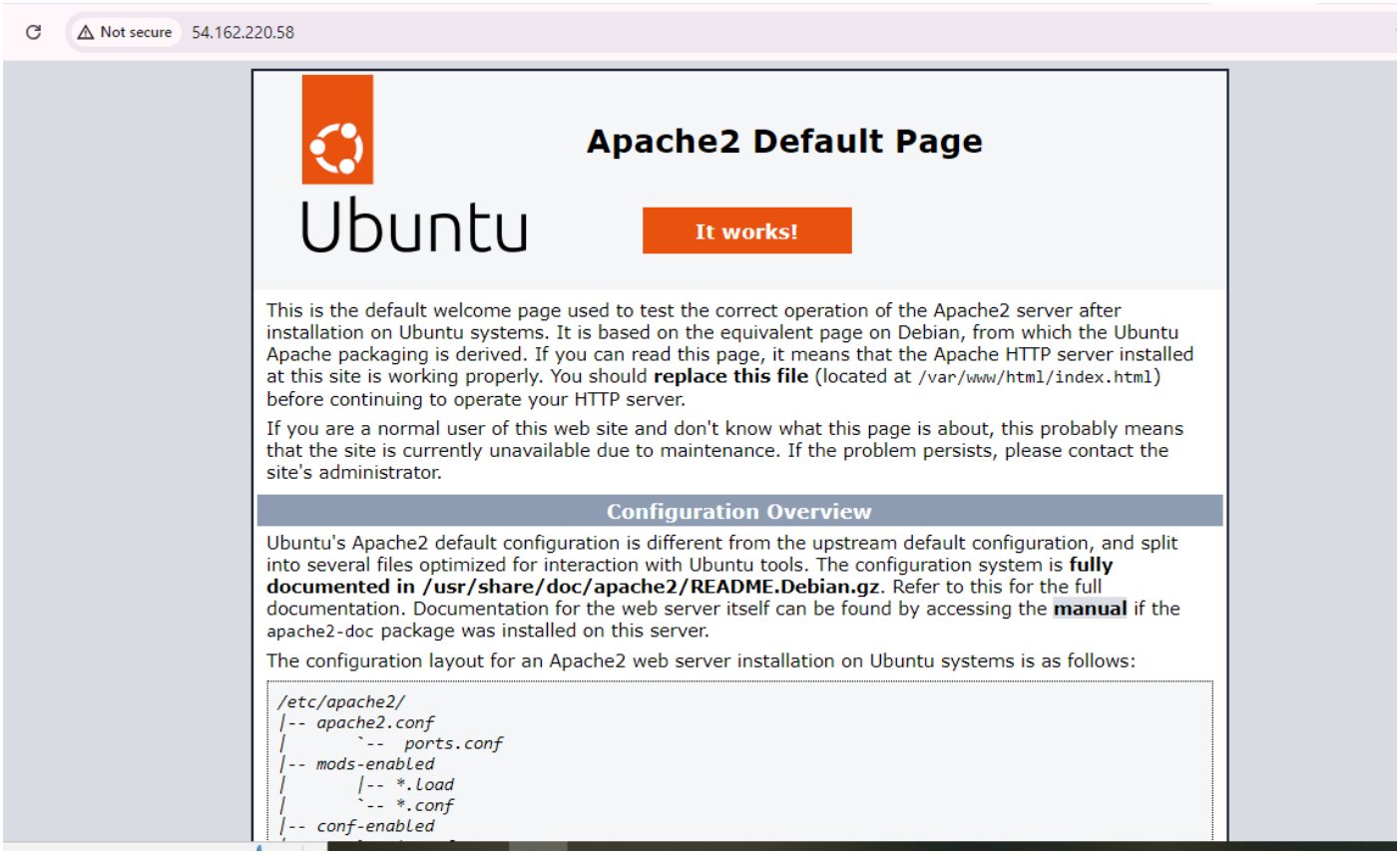
Search

1

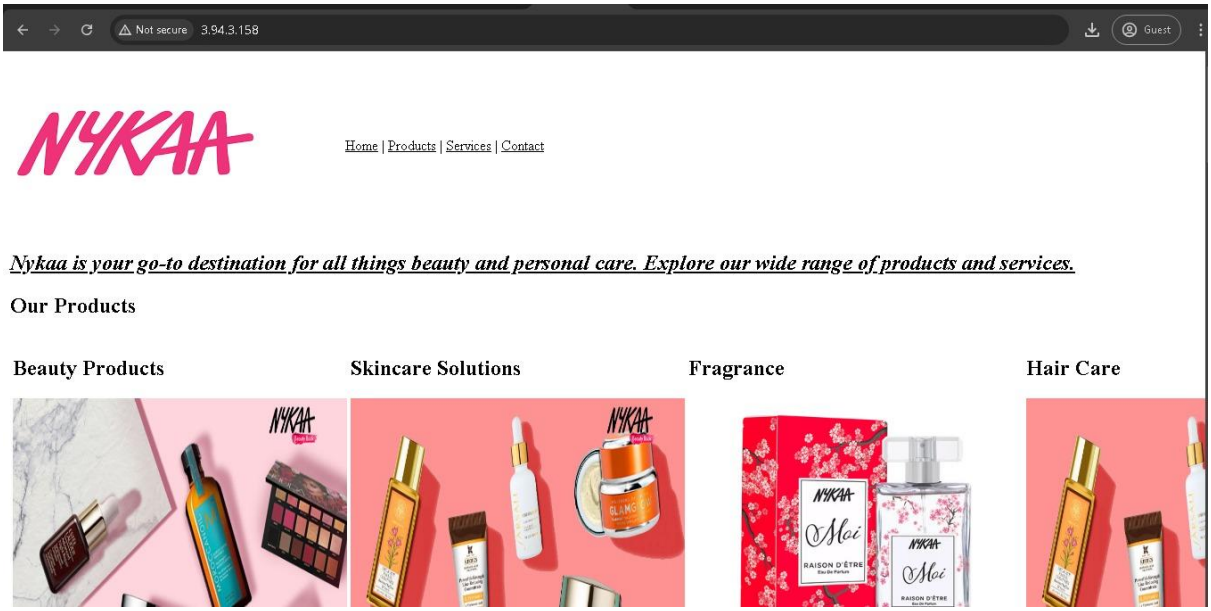

EC2 > Security Groups > sg-0896d82a58154b33d
sg-0896d82a58154b33d - launch-wizard-9
Actions
Details
Security group name
launch-wizard-9
Security group ID
sg-0896d82a58154b33d
Description
launch-wizard-9 created 2024-08-18T12:21:13.480Z
VPC ID
vpc-08963bc0f8afcd789
Owner
608111999703
Inbound rules count
3 Permission entries
Outbound rules count
1 Permission entry
Inbound rules
Outbound rules
Tags
Inbound rules (3)
Manage tags
Edit inbound rules


Security group name
launch-wizard-9
Security group ID
sg-0896d82a58154b33d
Description
launch-wizard-9 created 2024-08-18T12:21:13.480Z
VPC ID
vpc-08963bc0f8afcd789
Owner
608111999703
Inbound rules count
3 Permission entries
Outbound rules count
1 Permission entry
Inbound rules
Outbound rules
Tags
Outbound rules (1)
Manage tags
Edit outbound rules
Search
1
Name
Security group rule...
IP version
Type
Protocol
–
sgr-06dd7ee61f83e4e88
IPv4
All traffic
All

After that the ip-address which was given while running the instance, copythat and paste that on chrome, make sure that it is http and not https

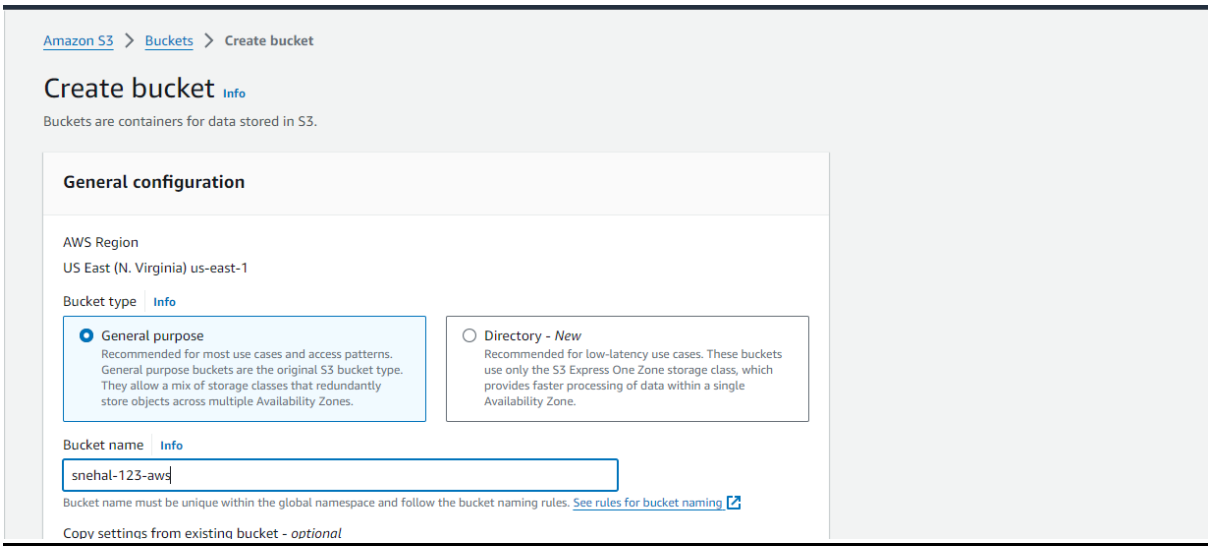



```
ubuntu@ip-172-31-40-177:~$ pwd
/home/ubuntu
ubuntu@ip-172-31-40-177:~$ mkdir Snehal
ubuntu@ip-172-31-40-177:~$ cd Snehal
ubuntu@ip-172-31-40-177:~/Snehal$ git clone https://github.com/Snehal490102/dynamic-web-hosting.git
Cloning into 'dynamic-web-hosting'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 15 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (15/15), 15.04 KiB | 3.76 MiB/s, done.
Resolving deltas: 100% (3/3), done.
ubuntu@ip-172-31-40-177:~/Snehal$ ls
dynamic-web-hosting
ubuntu@ip-172-31-40-177:~/Snehal$ cd dynamic-web-hosting/
ubuntu@ip-172-31-40-177:~/Snehal/dynamic-web-hosting$ ls
README.md  index.js  package-lock.json  package.json
ubuntu@ip-172-31-40-177:~/Snehal/dynamic-web-hosting$ npm i
Command 'npm' not found, but can be installed with:
sudo apt install npm
ubuntu@ip-172-31-40-177:~/Snehal/dynamic-web-hosting$ sudo apt install npm
Reading package lists... Done
```



STATIC WEBSITE HOSTING USING S3 BUCKET:

Step1: Create bucket



Encryption type Info

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

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

Bucket Key

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

- Disable
- Enable

▶ **Advanced settings**

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

Cancel | Create bucket

Amazon S3 > Buckets

▶ **Account snapshot** - *updated every 24 hours* All AWS Regions

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

View Storage Lens dashboard

General purpose buckets | Directory buckets

General purpose buckets (2) 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
elasticbeanstalk-us-east-1-608111999703	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 8, 2024, 15:22:13 (UTC+05:30)
snehal-123-aws	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 17, 2024, 22:59:32 (UTC+05:30)

Edit static website hosting Info

Static website hosting

Static website hosting

Use this bucket to host a website or redirect requests. Learn more

Edit

Static website hosting

Enabled

Hosting type

Bucket hosting

Bucket website endpoint copied

When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. Learn more

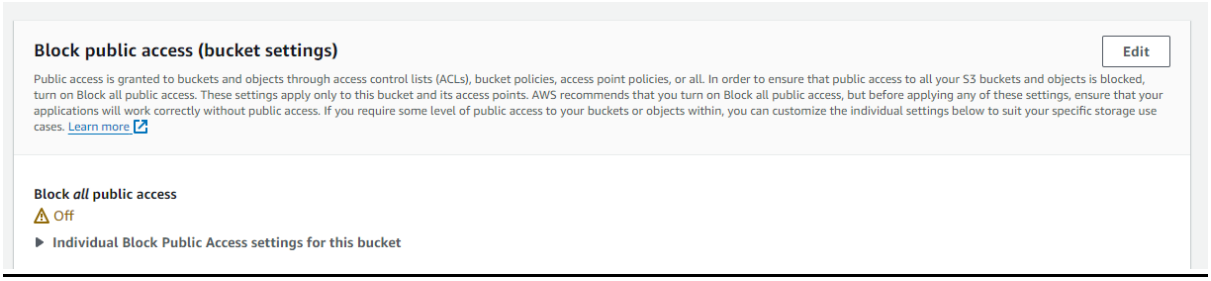
http://snehal-123-aws.s3-website-us-east-1.amazonaws.com

For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see Using Amazon S3 Block Public Access

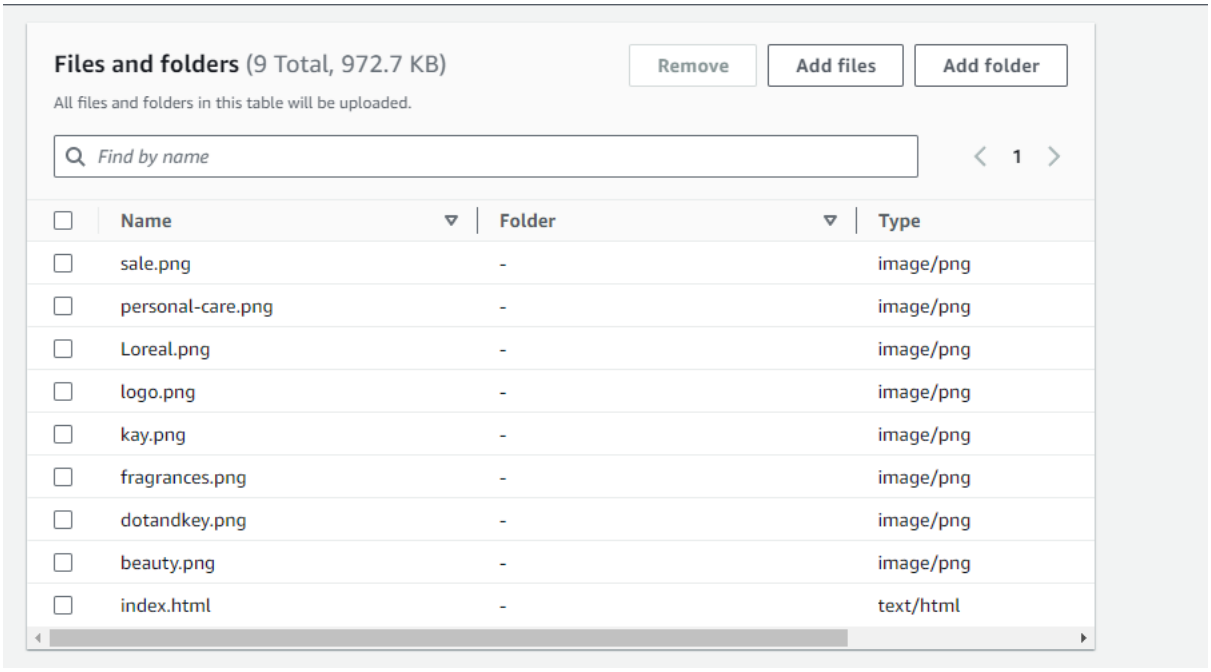
Index document

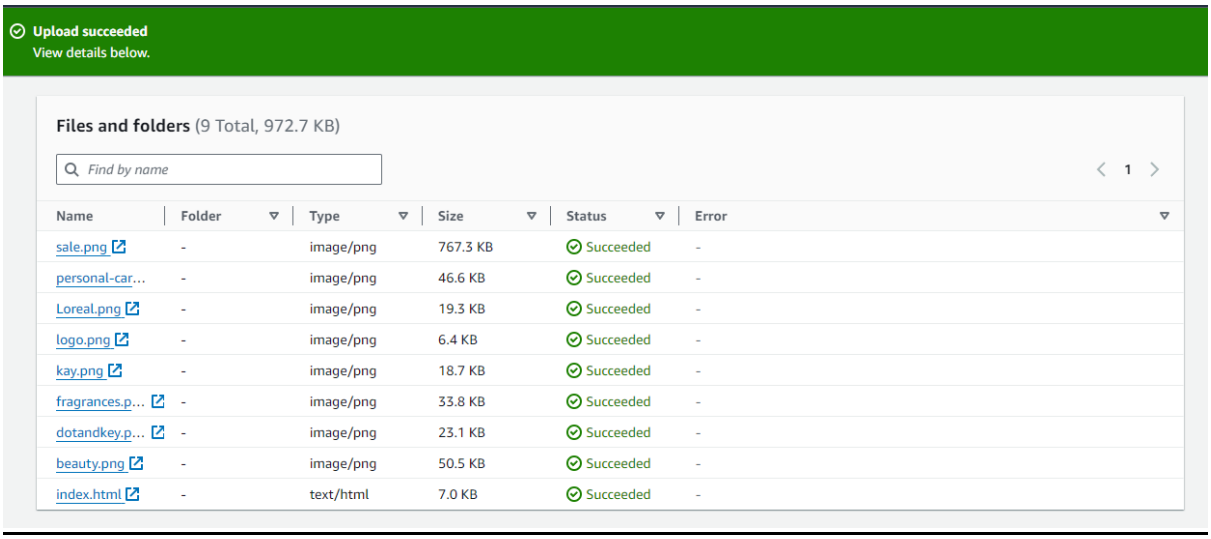
Specify the home or default page of the website.

index.html

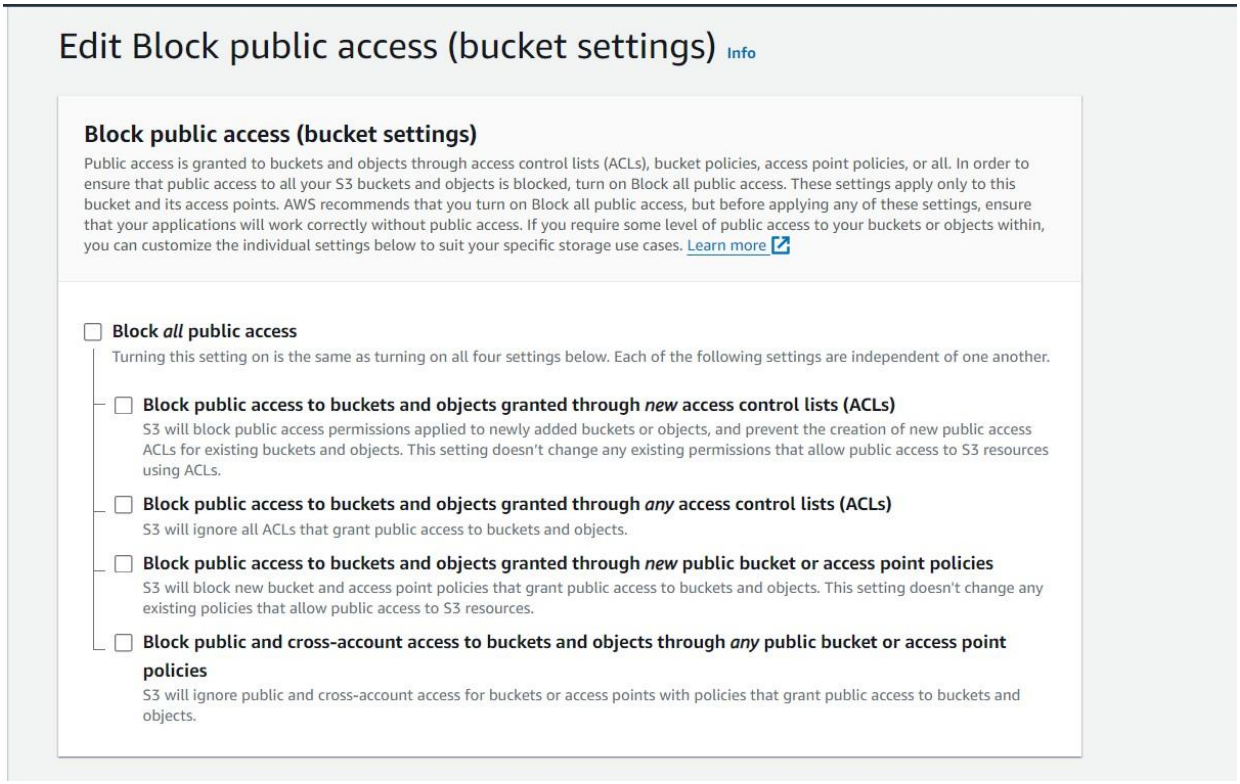


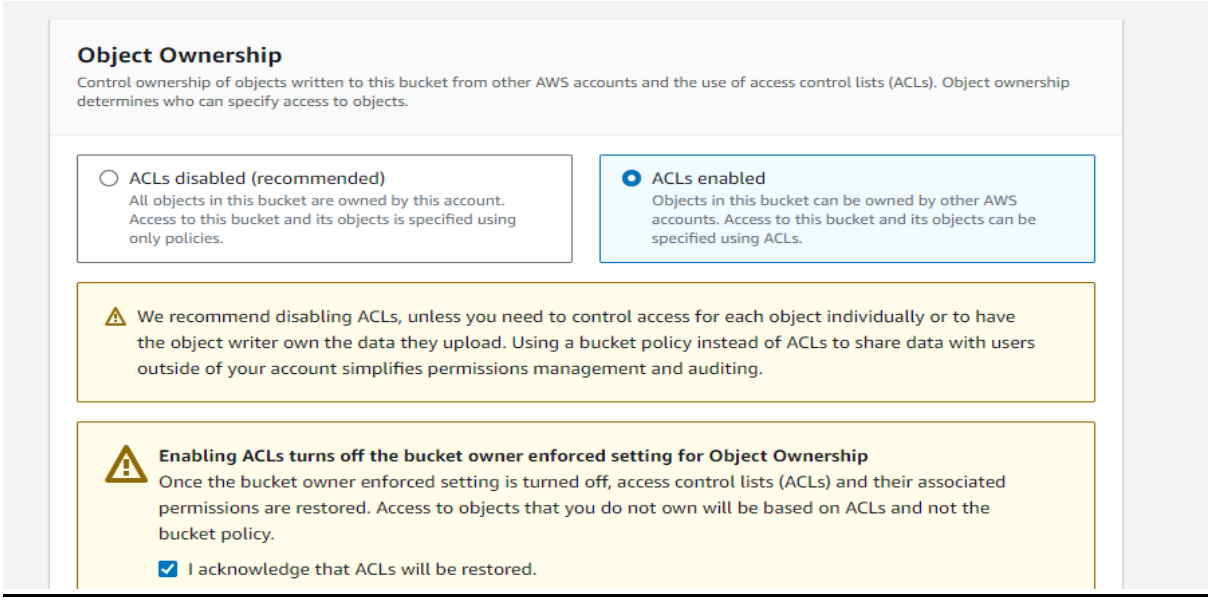
Step 2: Add resources

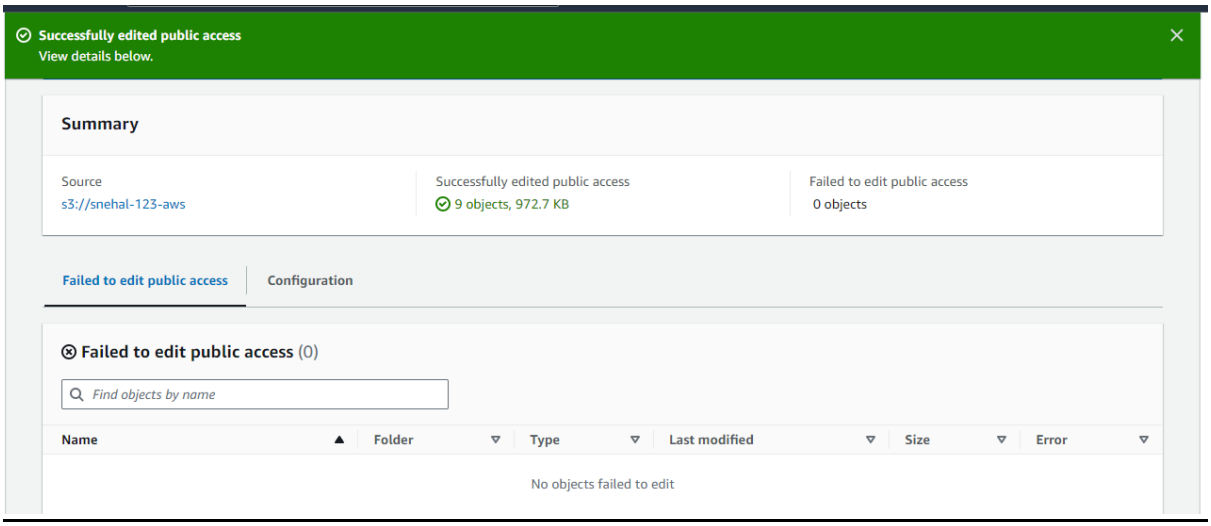


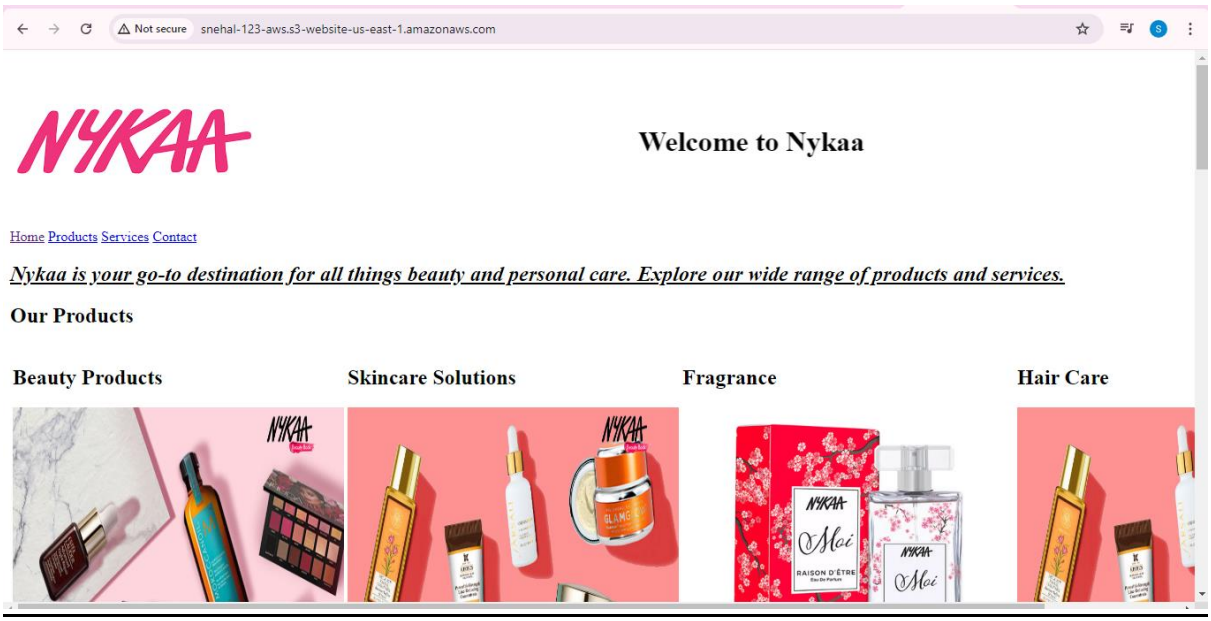


Step 3 : Provide public access









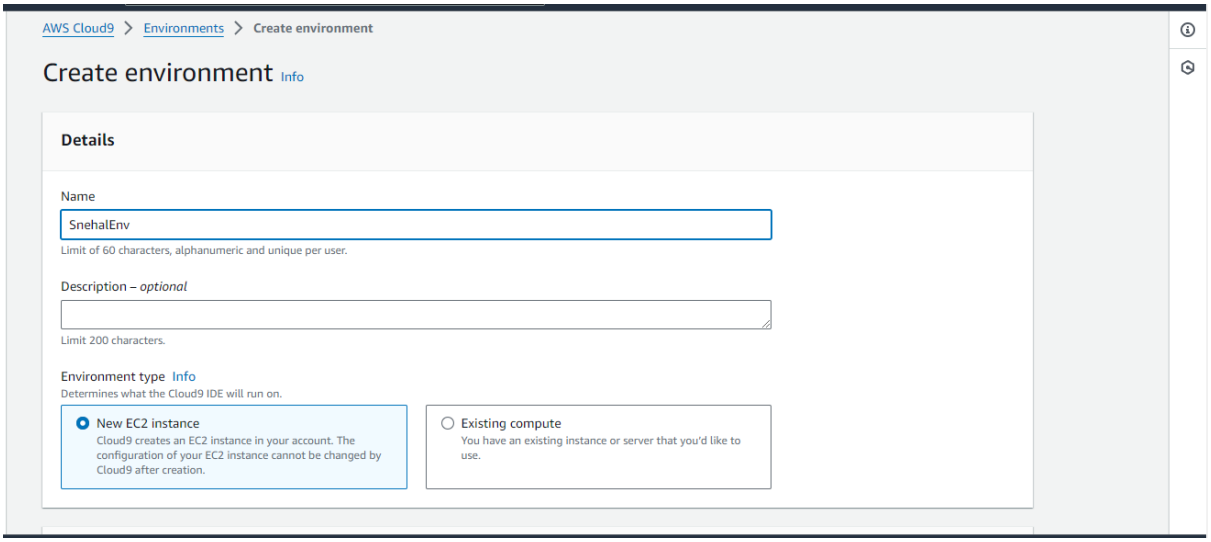
EC2 Dynamic Site Hosting:

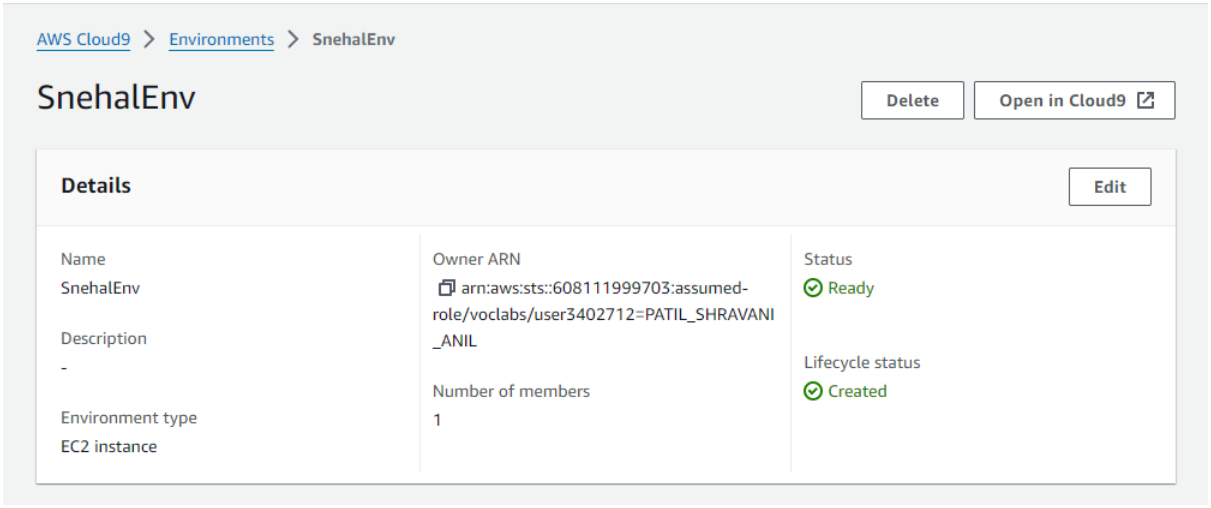
```
root@ip-172-31-55-145:/home/ubuntu/dynamic/dyanamic_site# npm i
(                    ) ⠙ reify:define-data-property: http fetch GET 200 https://registry.npmjs.org/define-data-property

added 93 packages, and audited 94 packages in 3s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
root@ip-172-31-55-145:/home/ubuntu/dynamic/dyanamic_site# npm start

> hosting-dynamic-website@1.0.0 start
> nodemon index.js

[nodemon] 3.1.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Server is running on port 3000
```

Hey this is about page.

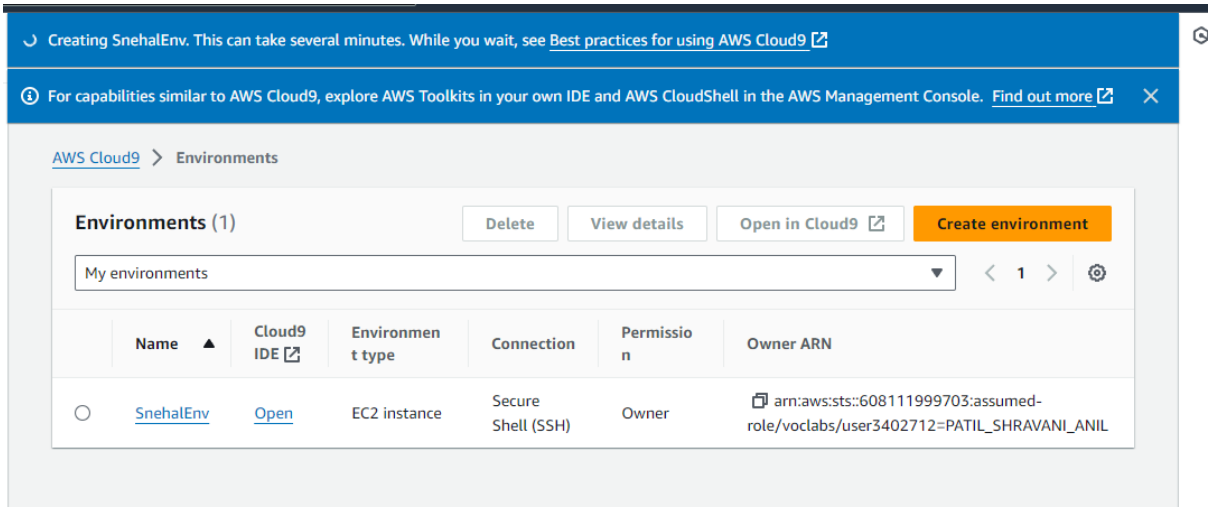
Cloud 9 IDE Site Hosting:

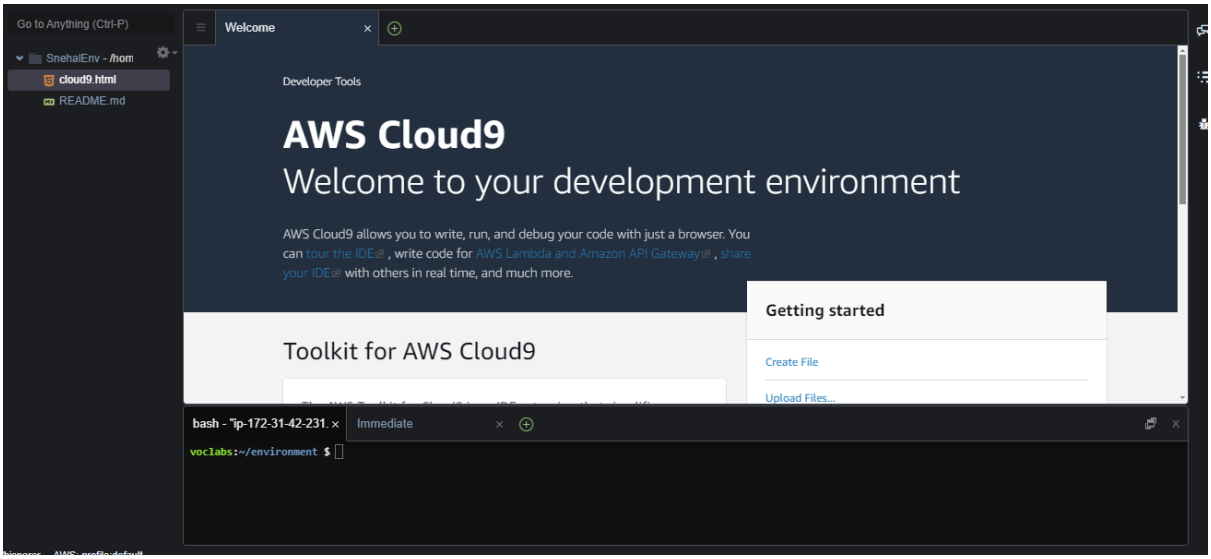
Step 1: Create Environment



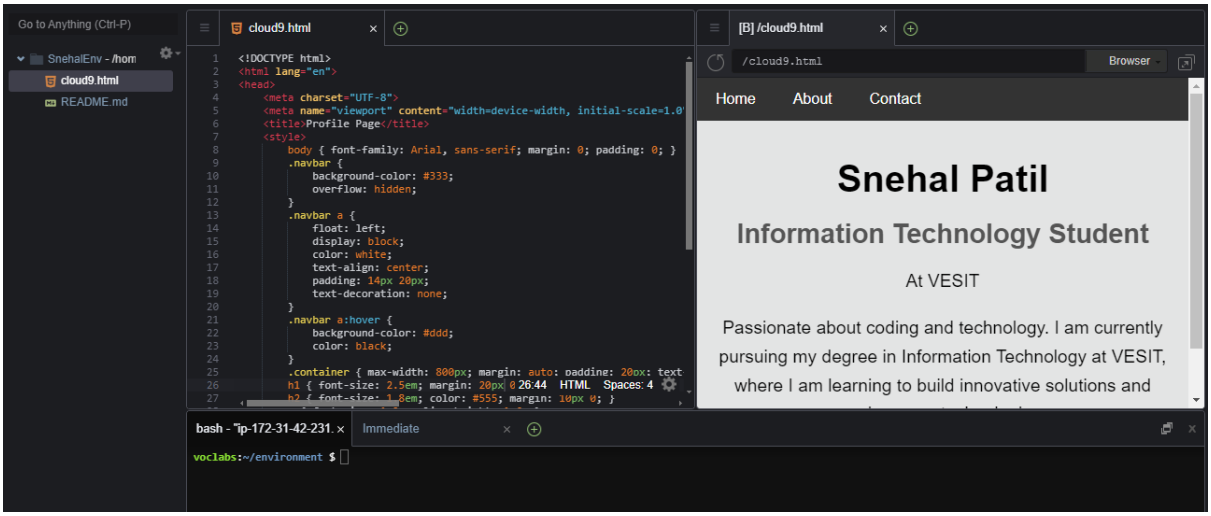


Step 2 : Open the Environment IDE





Step 3: Add the code and preview the website



User details

User name

Snehal

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 (0-9), and symbols ! @ # $ % ^ & * ( ) _ + - (hyphen) = [ ] { } | '

Permissions options

◉ Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

○ Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

○ Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Get started with groups
Create a group and select policies to attach to the group. We recommend using groups to manage user permissions by job function, AWS service access, or custom permissions. Learn more

Create group

▸ **Set permissions boundary - *optional***

Cancel Previous Next

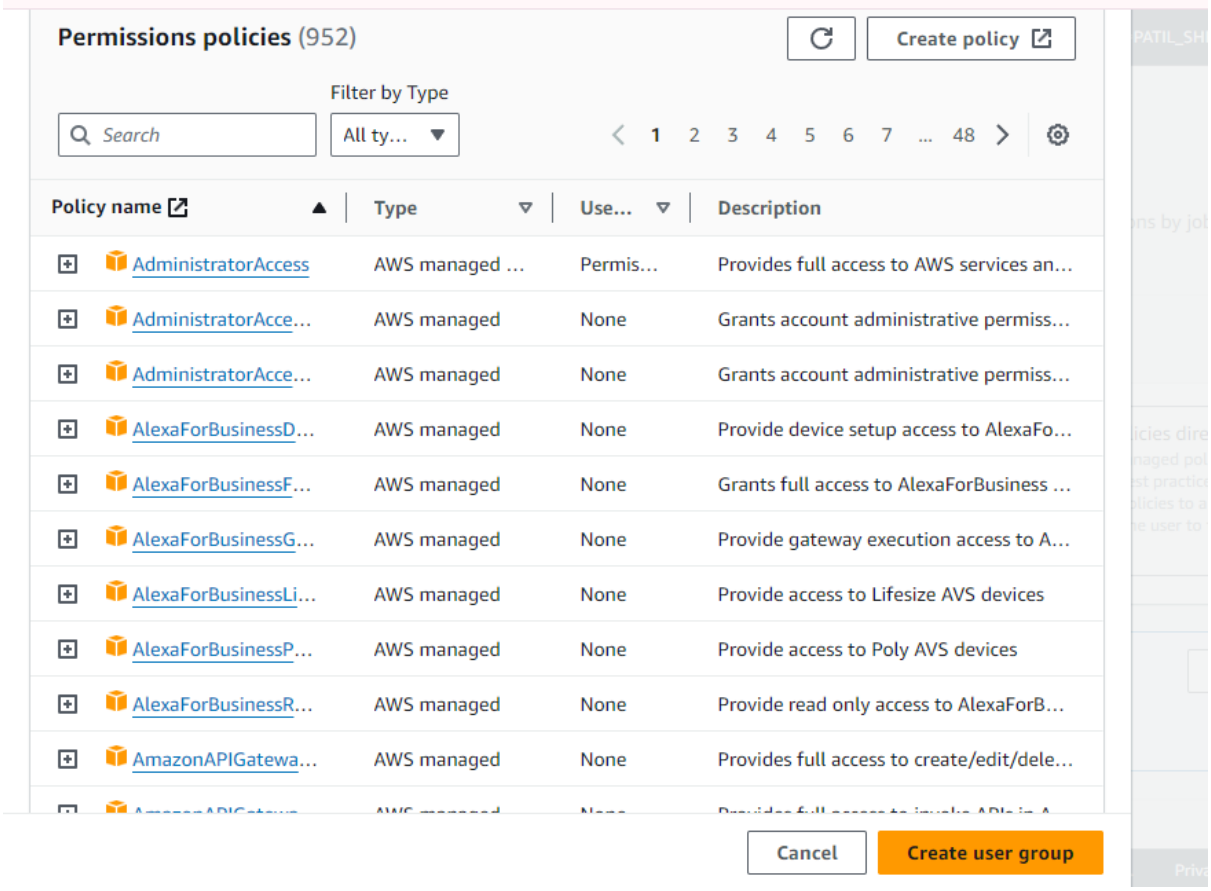
Create user group

Create a user 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

User group name
Enter a meaningful name to identify this group.

awsgrp

Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.


Permissions policies (952)
Create policy
Filter by Type
Search
All ty...
1 2 3 4 5 6 7 ... 48
Policy name
Type
Use...
Description
AdministratorAccess
AWS managed ...
Permis...
Provides full access to AWS services an...
AdministratorAcce...
AWS managed
None
Grants account administrative permiss...
AdministratorAcce...
AWS managed
None
Grants account administrative permiss...
AlexaForBusinessD...
AWS managed
None
Provide device setup access to AlexaFo...
AlexaForBusinessF...
AWS managed
None
Grants full access to AlexaForBusiness ...
AlexaForBusinessG...
AWS managed
None
Provide gateway execution access to A...
AlexaForBusinessLi...
AWS managed
None
Provide access to Lifesize AVS devices
AlexaForBusinessP...
AWS managed
None
Provide access to Poly AVS devices
AlexaForBusinessR...
AWS managed
None
Provide read only access to AlexaForB...
AmazonAPIGatewa...
AWS managed
None
Provides full access to create/edit/dele...
Cancel
Create user group

SNEHAL A. PATIL D15A 39

EXPERIMENT NO:2

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

Theory:

AWS Elastic Beanstalk

AWS Elastic Beanstalk is a versatile Platform as a Service (PaaS) offering from Amazon Web Services (AWS) that simplifies the deployment and management of applications in the cloud. It abstracts much of the underlying infrastructure complexity, allowing developers to focus on building and maintaining their applications while AWS takes care of provisioning resources like EC2 instances, load balancers, and databases.

Key Features of Elastic Beanstalk:

1. **Simplified Application Deployment:**
   - Elastic Beanstalk streamlines the deployment process by automatically handling the setup, configuration, and maintenance of the infrastructure required to run your application. This allows developers to concentrate on writing and refining their code rather than managing servers.
2. **Broad Language and Framework Support:**
   - Elastic Beanstalk supports multiple programming languages and frameworks, such as Java, .NET, Node.js, Python, Ruby, PHP, Go, and Docker. This flexibility makes it easy for developers to deploy applications built with their preferred technologies.
3. **Automatic Resource Scaling:**
   - Elastic Beanstalk includes automatic scaling capabilities, dynamically adjusting the number of instances running your application based on current demand. This ensures that your application remains responsive and cost-efficient during traffic fluctuations.
4. **Robust Health Monitoring:**
   - The platform continuously monitors the health of your application and provides detailed metrics and logs. It can automatically replace any unhealthy instances, ensuring high availability and reliability of your application.
5. **Multiple Environment Support:**
   - Elastic Beanstalk allows you to manage different environments, such as development, staging, and production. You can deploy updates to one environment without impacting others, providing a controlled and organized deployment process.

AWS CodeBuild

AWS CodeBuild is a fully managed continuous integration (CI) service that compiles your source code, runs tests, and produces deployable artifacts. It automates the build process, ensuring consistent compilation and testing across various environments.

Key Features of AWS CodeBuild:

1. **Managed Build Infrastructure:**

- With AWS CodeBuild, there's no need to manage your own build servers. AWS handles all infrastructure management, allowing you to focus on developing and testing your applications.

2. **Automatic Scaling:**
   - CodeBuild automatically scales to meet your needs, handling multiple builds simultaneously. This ensures that even during peak times, your builds are processed quickly and efficiently.
3. **Customizable Build Environments:**
   - You can define custom build environments using Docker images, providing the flexibility to tailor the build environment to meet the specific needs of your application.
4. **Seamless AWS Integration:**
   - CodeBuild integrates effortlessly with other AWS services, such as CodePipeline, CodeCommit, and S3, enabling you to create a complete CI/CD pipeline that automates your entire build and deployment process.
5. **Cost-Effective Pricing:**
   - CodeBuild uses a pay-as-you-go pricing model, charging you only for the build time you consume. This makes it an economical solution for automating your build processes.

Deploying on S3 Using AWS CodePipeline

AWS CodePipeline is a fully managed continuous delivery service that automates the steps required to release your software, from building and testing to deployment. By integrating with services like CodeBuild and S3, CodePipeline provides a streamlined process for deploying applications.

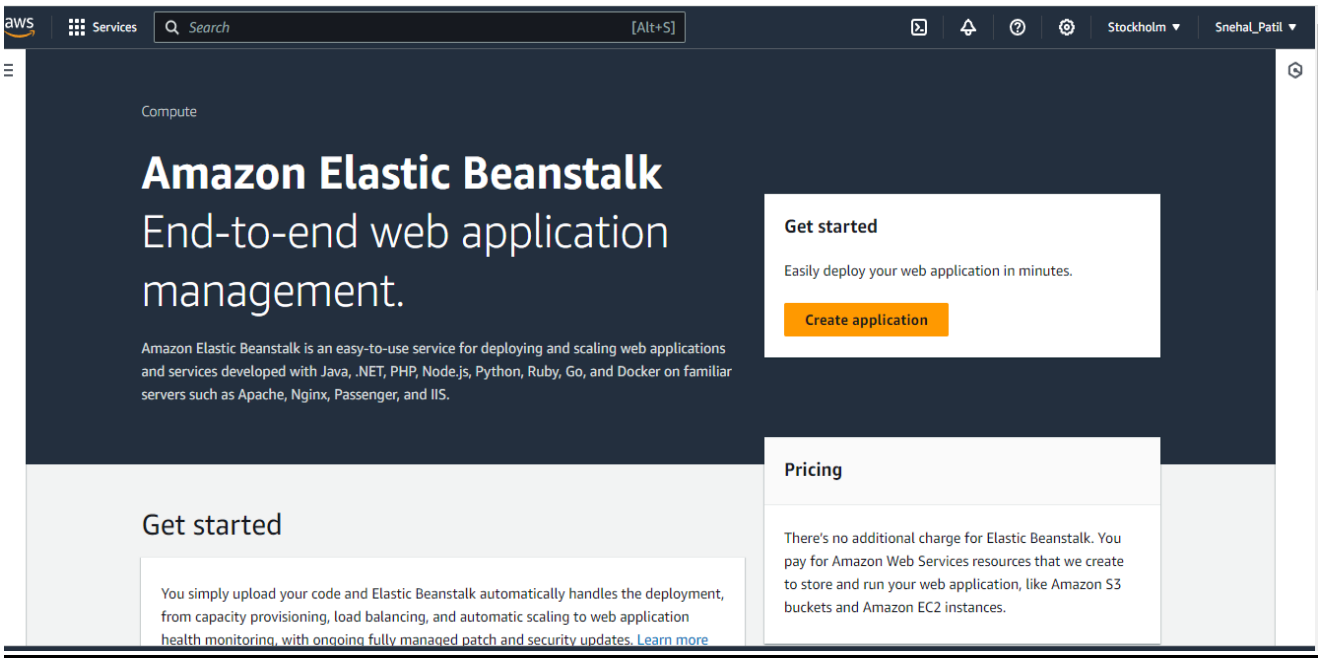
Key Steps to Deploy on S3 Using AWS CodePipeline:

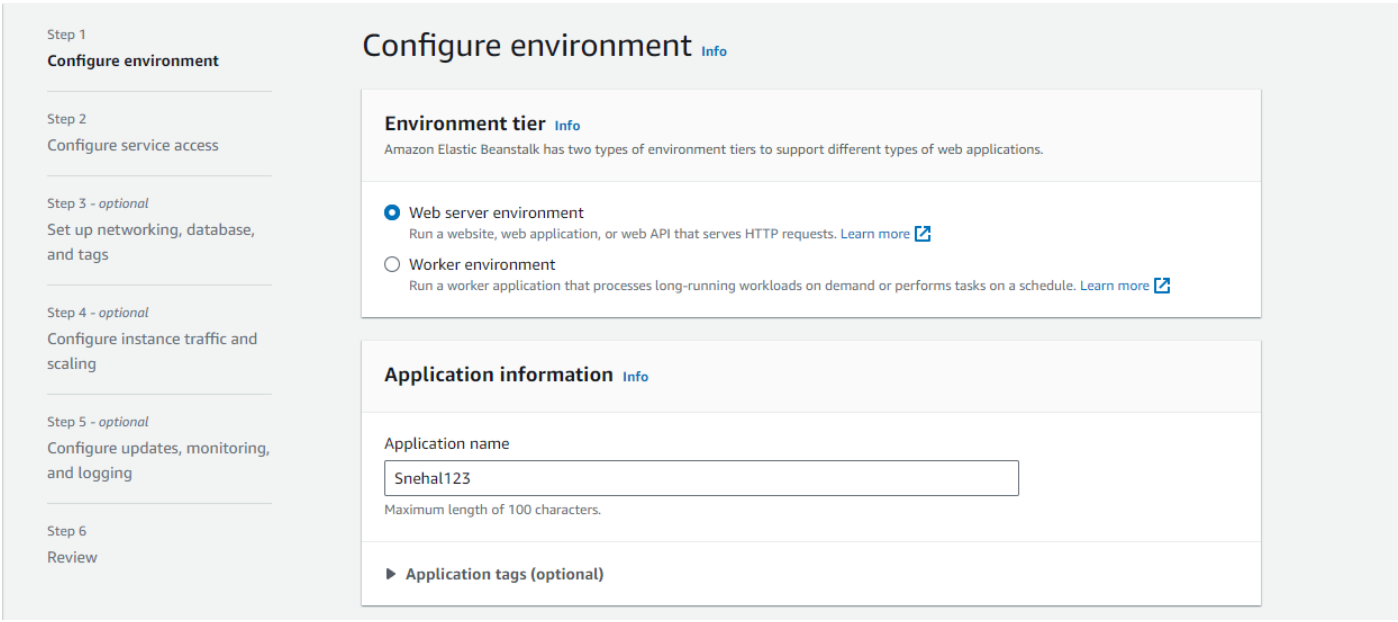
1. **Source Code Retrieval:**
   - The pipeline begins with a source stage, where the source code is retrieved from a repository such as AWS CodeCommit, GitHub, or an S3 bucket. This source code is then used in subsequent stages for building and deploying the application.
2. **Build and Package with CodeBuild:**
   - In the build stage, AWS CodePipeline triggers CodeBuild to compile the source code, run tests, and package the application. The output of this process is a deployable artifact stored in an S3 bucket.
3. **Deployment to S3:**
   - In the deployment stage, CodePipeline automatically deploys the artifacts generated during the build stage to an S3 bucket. The S3 bucket can be configured to host a static website or store files accessed by your application.
4. **Automation and Notifications:**
   - CodePipeline can be configured to automatically trigger builds and deployments when changes are made to the source code repository. Additionally, it can send notifications via Amazon SNS to keep you informed about the status of your pipeline.
5. **Pipeline Monitoring and Logging:**
   - CodePipeline provides tools for monitoring and logging the progress of each stage in your pipeline. This allows you to quickly identify and address any issues that arise during the build or deployment process, ensuring a smooth release cycle.

Implementation :

Elastic Beanstalk

Step 1: create environment





Environment information Info

Choose the name, subdomain and description for your environment. These cannot be changed later.

Environment name

Snehal123-env

Must be from 4 to 40 characters in length. The name can contain only letters, numbers, and hyphens. It can't start or end with a hyphen. This name must be unique within a region in your account.

Domain

Leave blank for autogenerated value .eu-north-1.elasticbeanstalk.com Check availability

Environment description

Platform Info

Platform type

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

Platform

Python

Platform branch

Python 3.11 running on 64bit Amazon Linux 2023

Platform version

4.1.3 (Recommended)

Application code Info

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

Presets Info

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

Configuration presets

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

Cancel Next

IAM > Dashboard

IAM Dashboard

Security recommendations 1

⚠ Add MFA for root user
Add MFA for root user - Enable multi-factor authentication (MFA) for the root user to improve security for this account.
Add MFA

✓ Root user has no active access keys
Using access keys attached to an IAM user instead of the root user improves security.

IAM resources

Resources in this AWS Account

User groups	Users	Roles	Policies	Identity providers
0	0	2	0	0

AWS Account

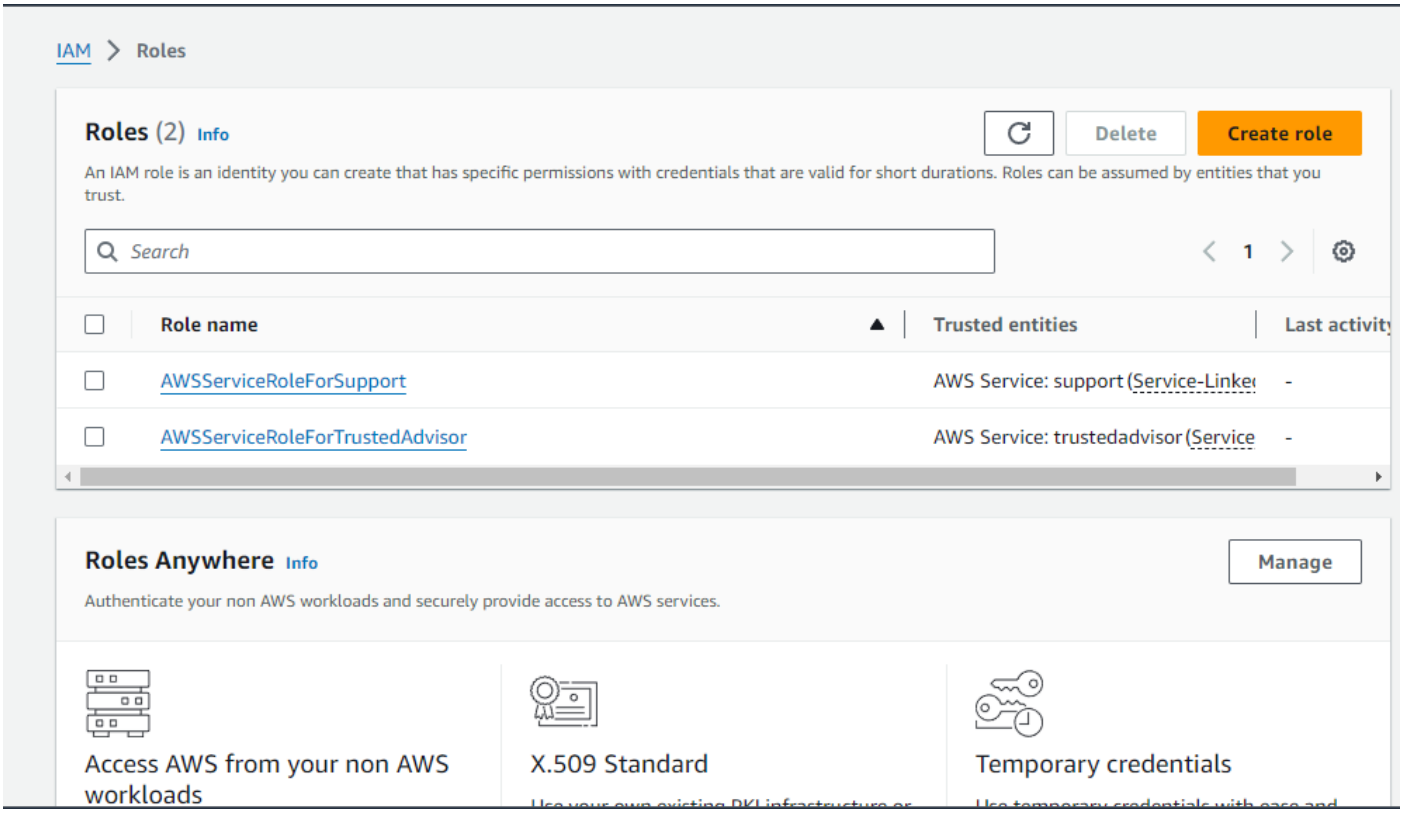
Account ID
825765388229

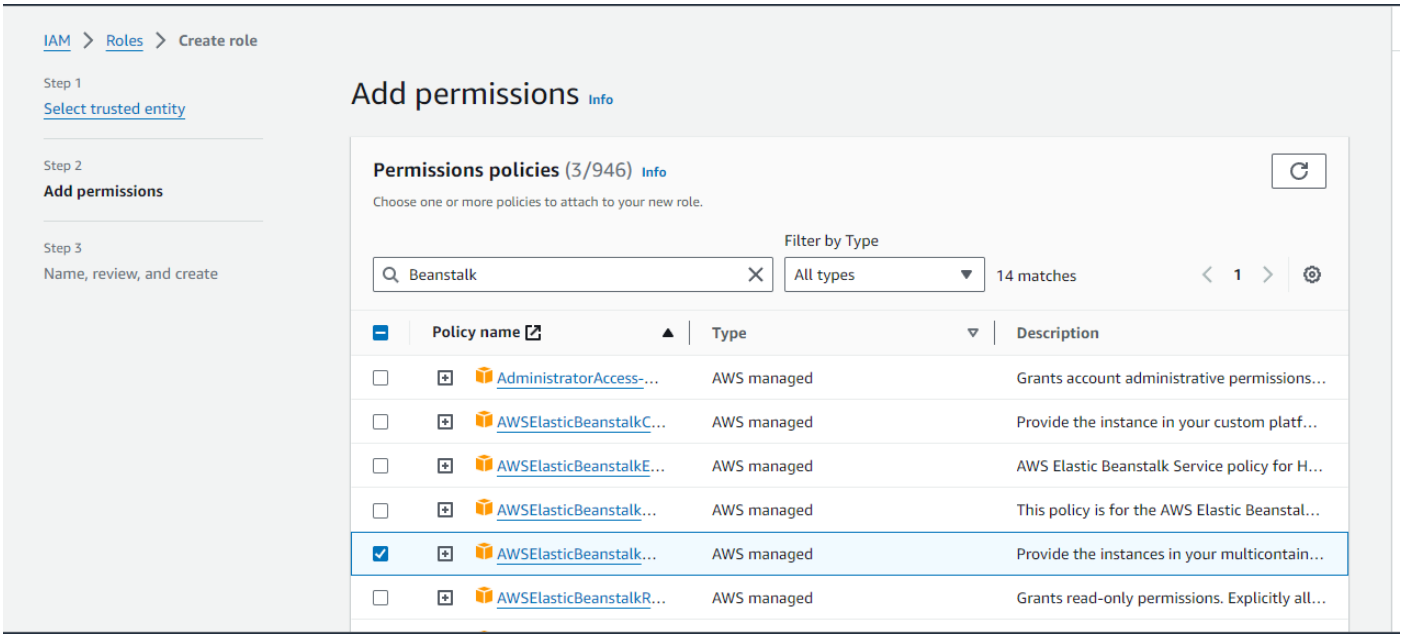
Account Alias
Create

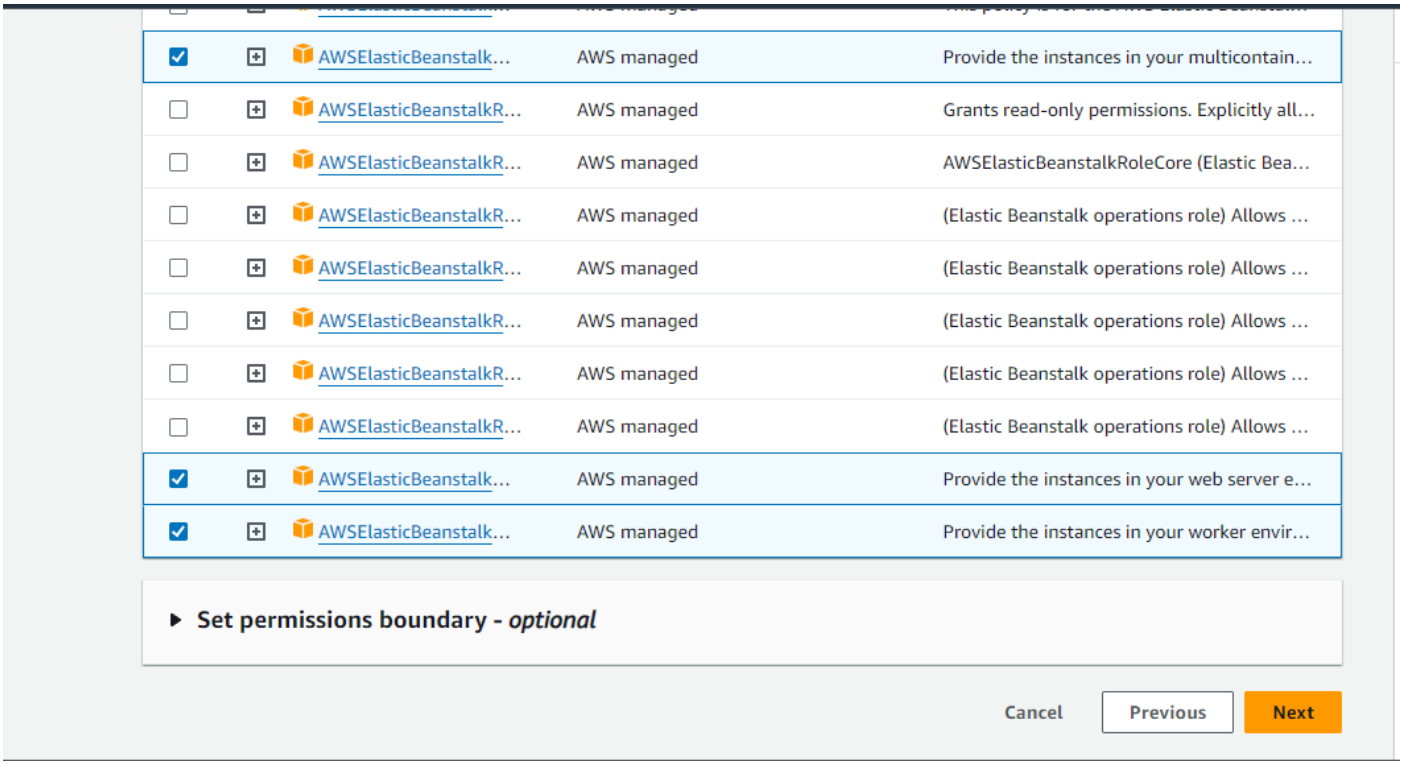
Sign-in URL for IAM users in this account
https://825765388229.signin.aws.amazon.com/console

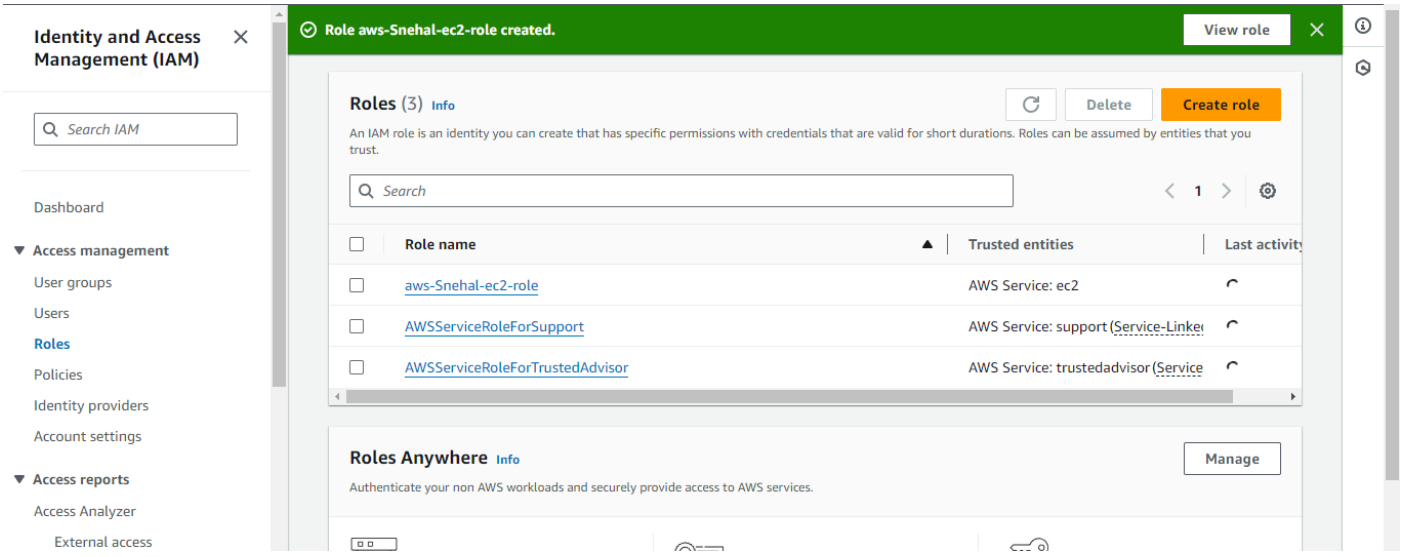
Quick Links

My security credentials

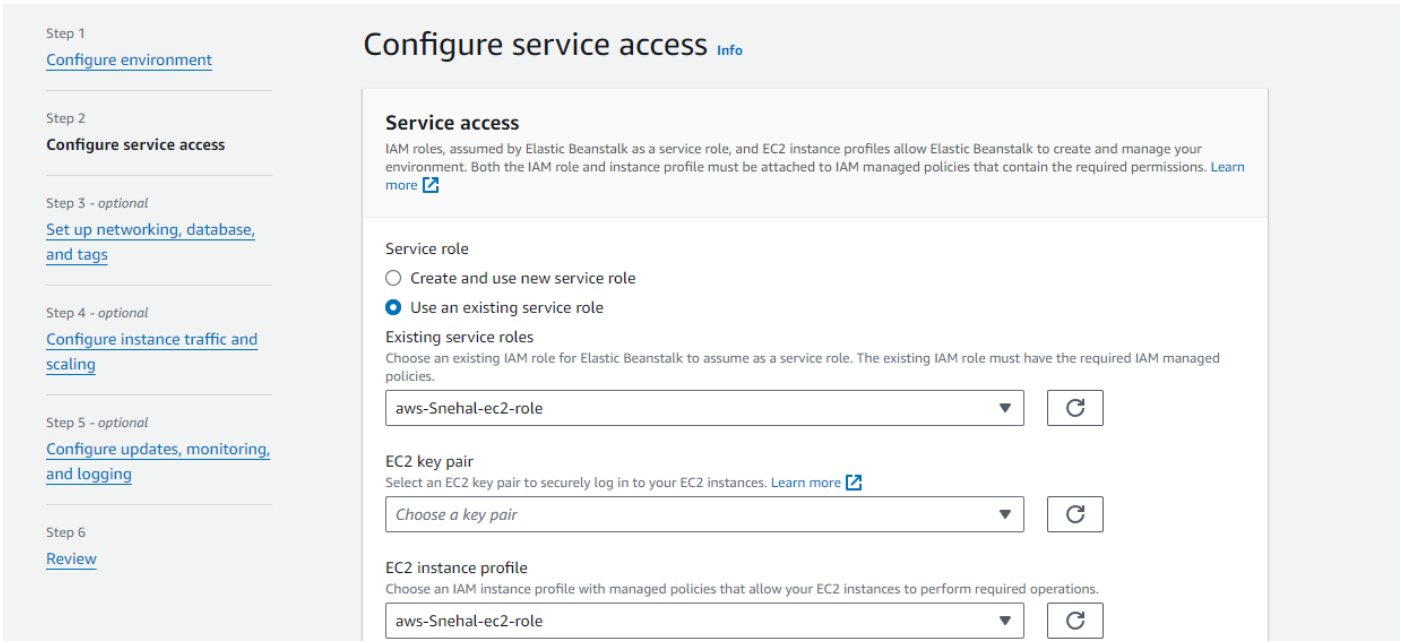

IAM > Roles
Roles (2) Info
Delete
Create role
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.
Search
1
Role name
Trusted entities
Last activity
AWSServiceRoleForSupport
AWS Service: support (Service-Linke
-
AWSServiceRoleForTrustedAdvisor
AWS Service: trustedadvisor (Service
-
Roles Anywhere Info
Manage
Authenticate your non AWS workloads and securely provide access to AWS services.
Access AWS from your non AWS workloads
X.509 Standard
Temporary credentials

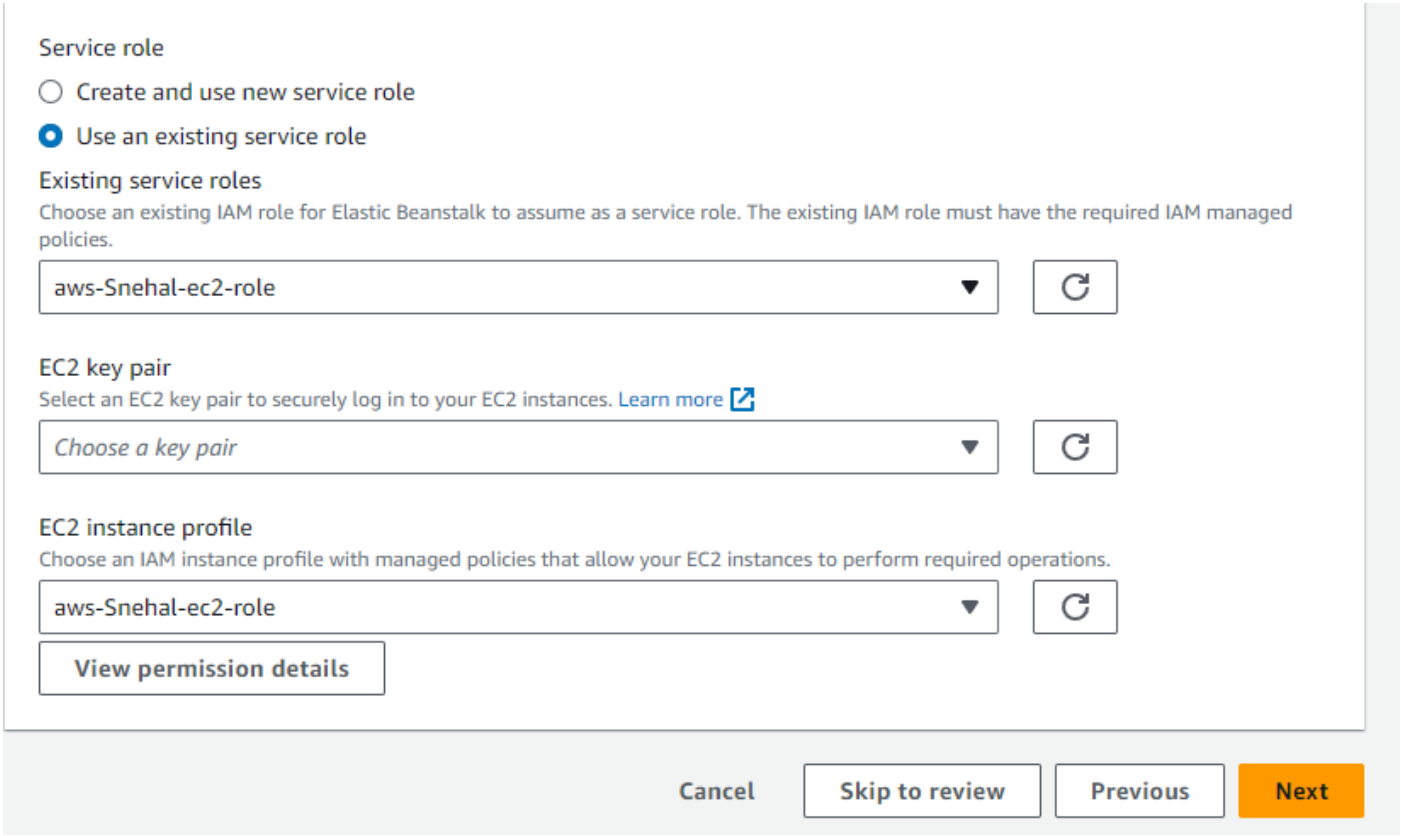

IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Add permissions Info
Permissions policies (3/946) Info
Choose one or more policies to attach to your new role.
Filter by Type
Beanstalk
All types
14 matches
1
Policy name
Type
Description
AdministratorAccess-...
AWS managed
Grants account administrative permissions...
AWSElasticBeanstalkC...
AWS managed
Provide the instance in your custom platf...
AWSElasticBeanstalkE...
AWS managed
AWS Elastic Beanstalk Service policy for H...
AWSElasticBeanstalk...
AWS managed
This policy is for the AWS Elastic Beanstal...
AWSElasticBeanstalk...
AWS managed
Provide the instances in your multicontain...
AWSElasticBeanstalkR...
AWS managed
Grants read-only permissions. Explicitly all...


AWSElasticBeanstalk... AWS managed Provide the instances in your multicontain...
AWSElasticBeanstalkR... AWS managed Grants read-only permissions. Explicitly all...
AWSElasticBeanstalkR... AWS managed AWSElasticBeanstalkRoleCore (Elastic Bea...
AWSElasticBeanstalkR... AWS managed (Elastic Beanstalk operations role) Allows ...
AWSElasticBeanstalkR... AWS managed (Elastic Beanstalk operations role) Allows ...
AWSElasticBeanstalkR... AWS managed (Elastic Beanstalk operations role) Allows ...
AWSElasticBeanstalkR... AWS managed (Elastic Beanstalk operations role) Allows ...
AWSElasticBeanstalkR... AWS managed (Elastic Beanstalk operations role) Allows ...
AWSElasticBeanstalk... AWS managed Provide the instances in your web server e...
AWSElasticBeanstalk... AWS managed Provide the instances in your worker envir...
Set permissions boundary - optional
Cancel
Previous
Next


Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Access reports
Access Analyzer
External access
Role aws-Snehal-ec2-role created.
View role
Roles (3) Info
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.
Delete
Create role
Search
Role name
Trusted entities
Last activity
aws-Snehal-ec2-role
AWS Service: ec2
AWSServiceRoleForSupport
AWS Service: support (Service-Linke
AWSServiceRoleForTrustedAdvisor
AWS Service: trustedadvisor (Service
Roles Anywhere Info
Authenticate your non AWS workloads and securely provide access to AWS services.
Manage

Step 2 : add your Ec2 key pair and instance profile





Review Info

Step 1: Configure environment

Edit

Environment information

Environment tier
Web server environment

Application name
Snehal123

Environment name
Snehal123-env

Application code
Sample application

Platform
arn:aws:elasticbeanstalk:eu-north-1::platform/Python 3.11 running on 64bit Amazon Linux 2023/4.1.3

Step 2: Configure service access

Edit

Service access Info

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
arn:aws:iam::825765388229:role/aws-Snehal-ec2-role

EC2 instance profile
aws-Snehal-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.

No options configured

Tags

Step 4: Configure instance traffic and scaling

Edit

Instance traffic and scaling Info

Customize the capacity and scaling for your environment's instances. Select security groups to control instance traffic.Configure the software that runs on your environment's instances by setting platform-specific options.

Instances

IMDSv1

Deactivated

Capacity

Environment type	Fleet composition	On-demand base
Single instance	On-Demand instance	0
On-demand above base	**Capacity rebalancing**	**Scaling cooldown**
0	Deactivated	360
Processor type	**Instance types**	**AMI ID**
x86_64	t3.micro,t3.small	ami-030d0ebd08fe18778

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
Command timeout	**Deployment policy**	**Health threshold**

Platform software

Lifecycle false	Log streaming Deactivated	NumProcesses 1
NumThreads 15	WSGIPath application	Proxy server nginx
Logs retention 7	Rotate logs Deactivated	Update level minor
X-Ray enabled Deactivated		

Environment properties

Key	Value
PYTHONPATH	/var/app/venv/staging-LQM1lest/bin

Cancel | Previous | Submit

Environment successfully launched.

Elastic Beanstalk > Environments > Snehal123-env

Snehal123-env Info

Actions | Upload and deploy

Environment overview

Health
Warning

Environment ID
e-nm5tuux6pa

Domain
Snehal123-env.eba-rpipmwt9.eu-north-1.elasticbeanstalk.com

Application name
Snehal123

Platform

Change version

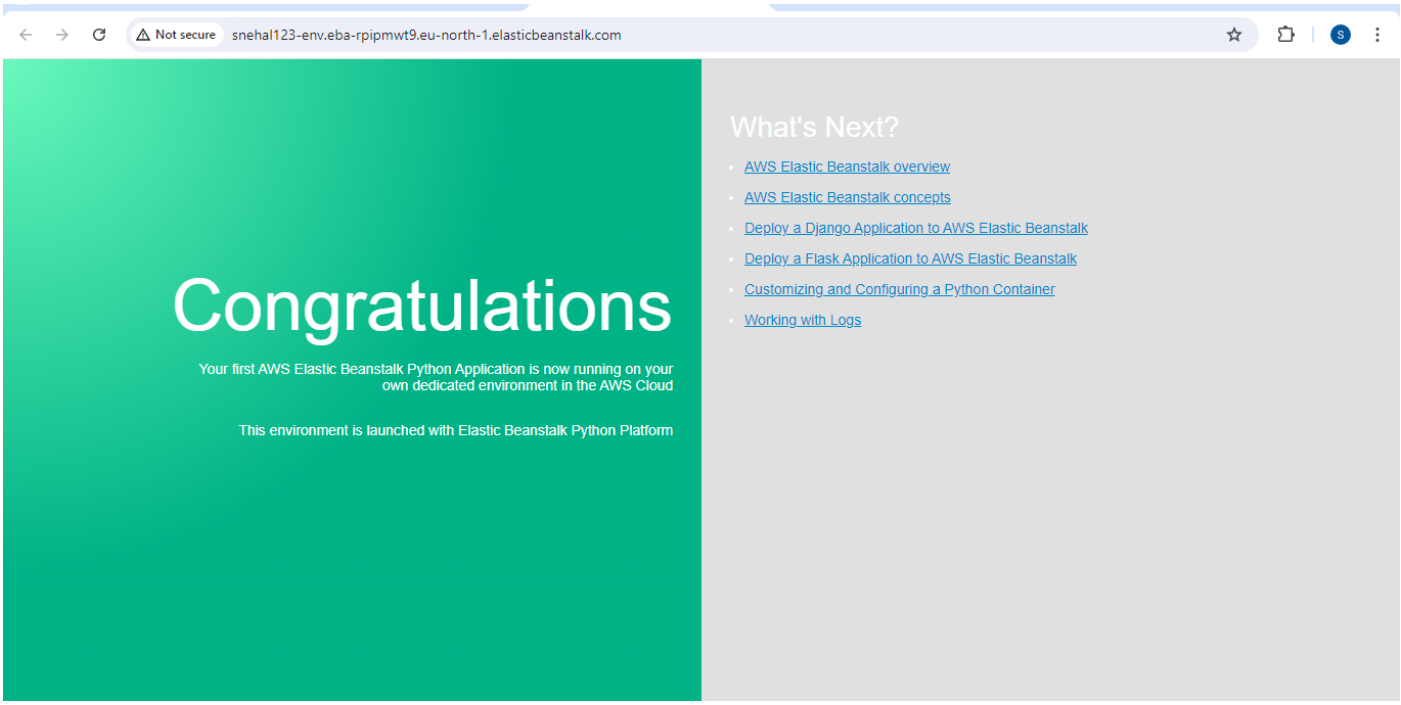
Platform
Python 3.11 running on 64bit Amazon Linux 2023/4.1.3

Running version
-

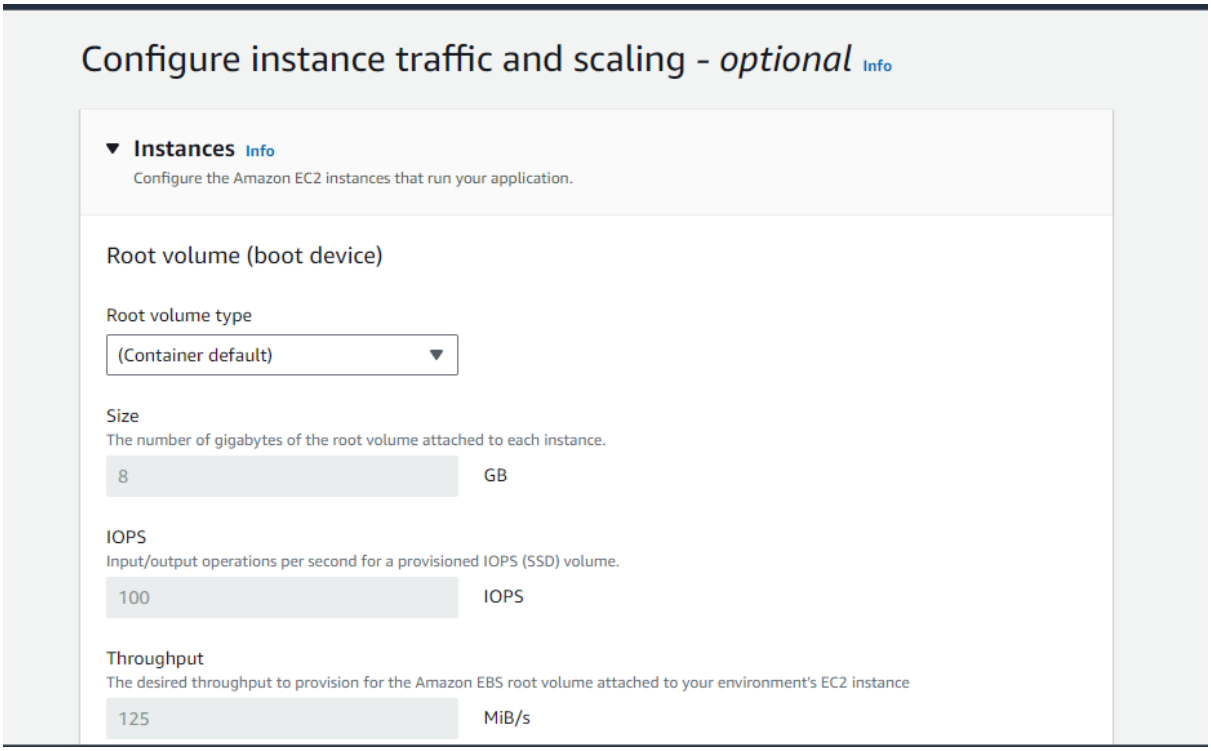
Platform state
Supported

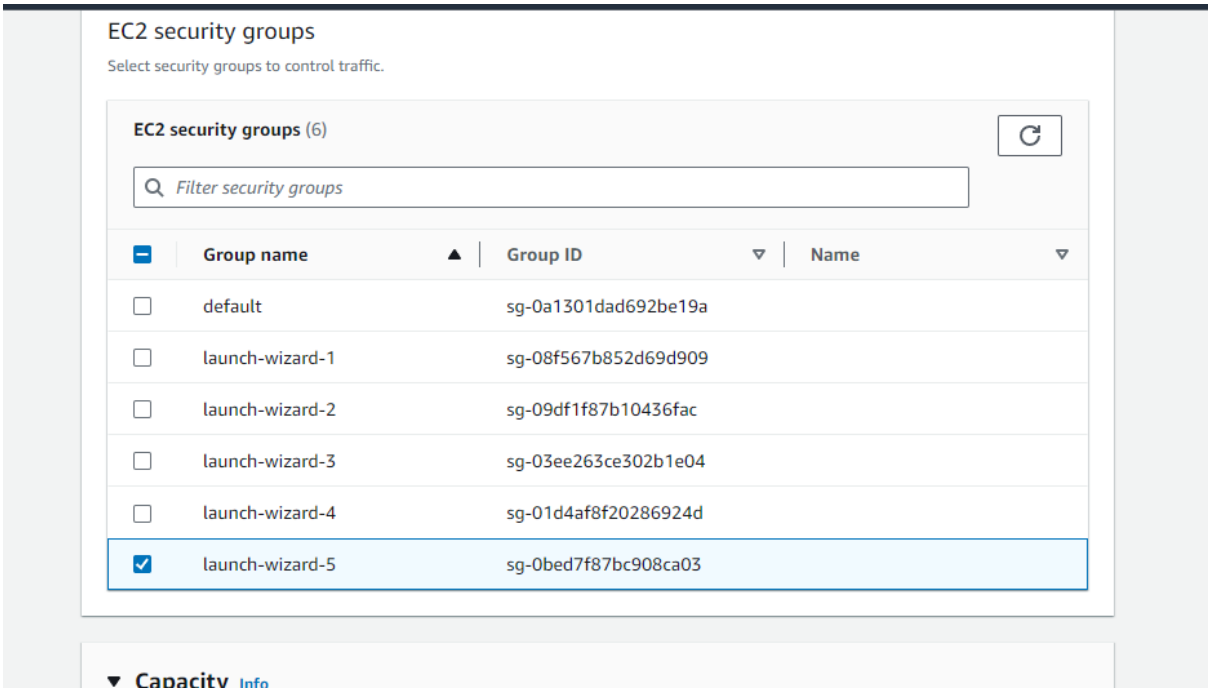
Events | Health | Logs | Monitoring | Alarms | Managed updates | Tags

Step 3: Beanstalk environment is created



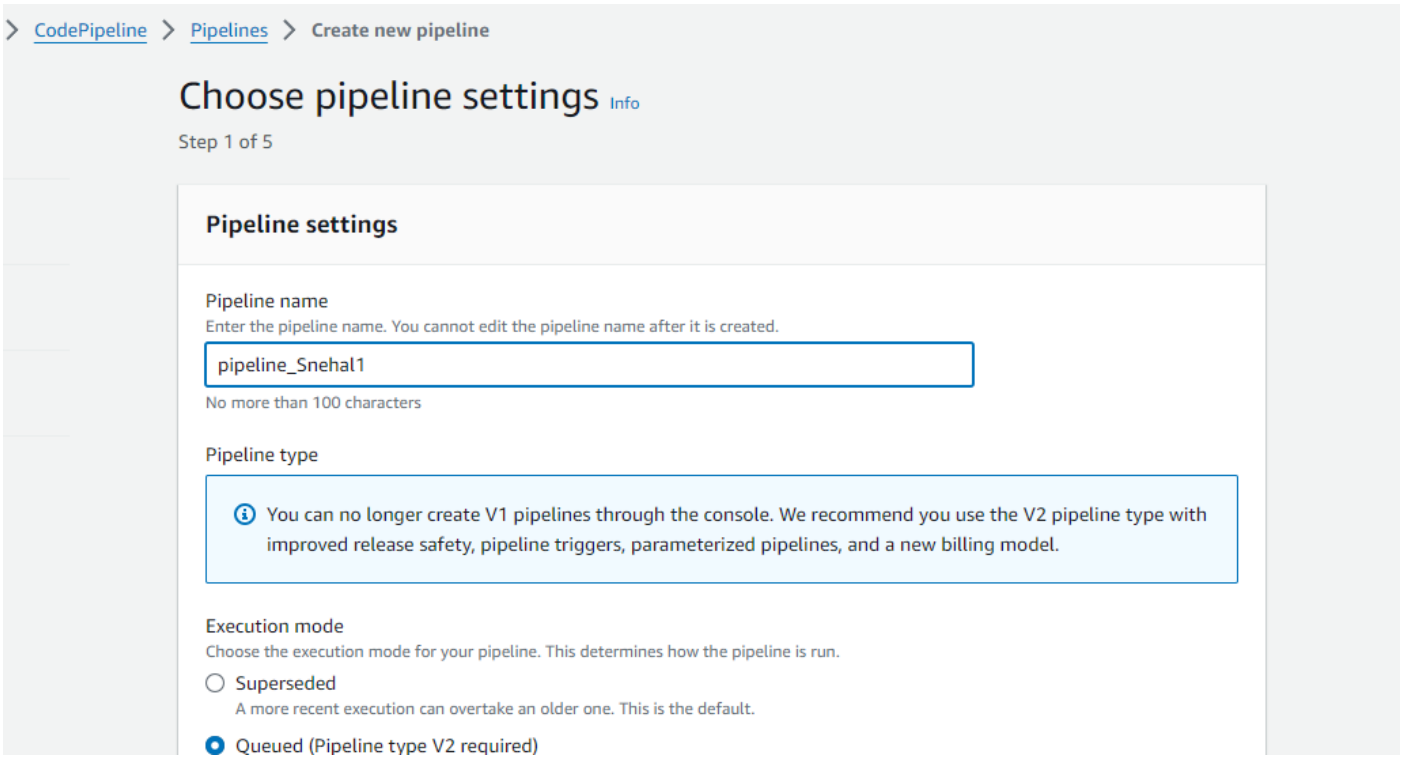
Step 4 : add security config and review all settings



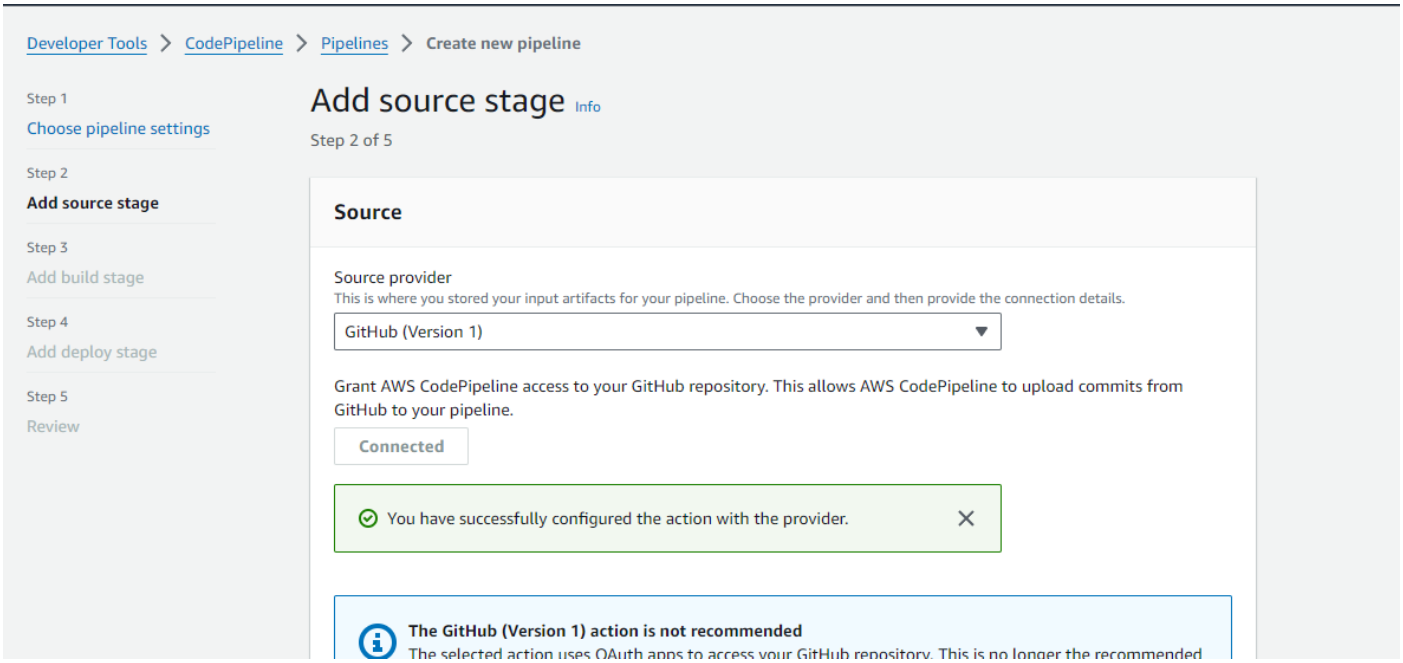


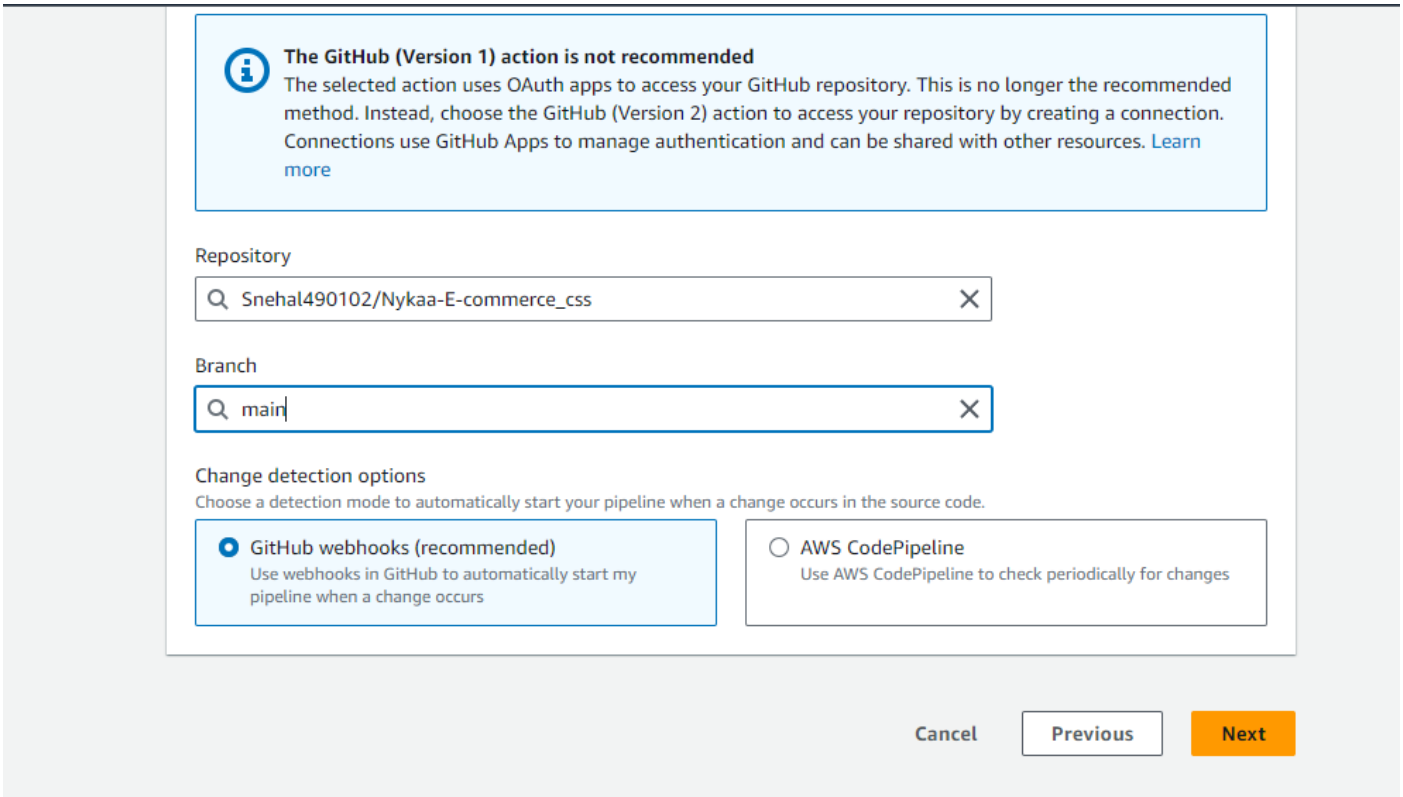
PIPELINE CREATION:

Step 1 : click on create pipeline and give name

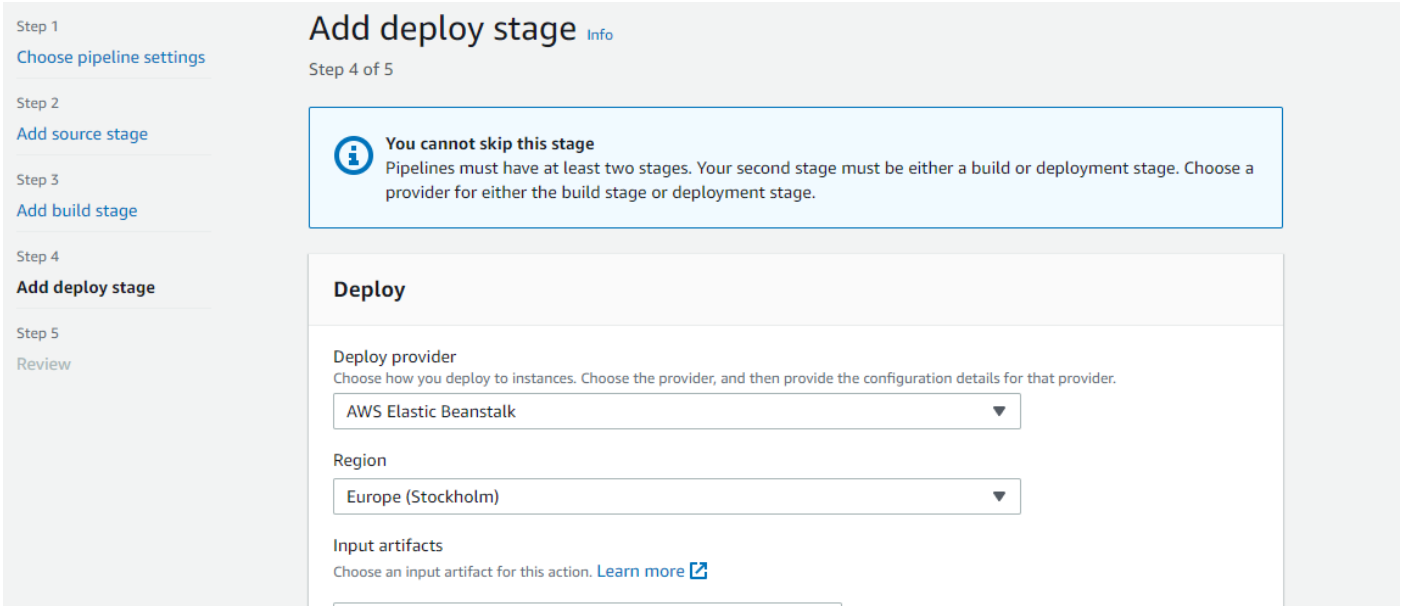


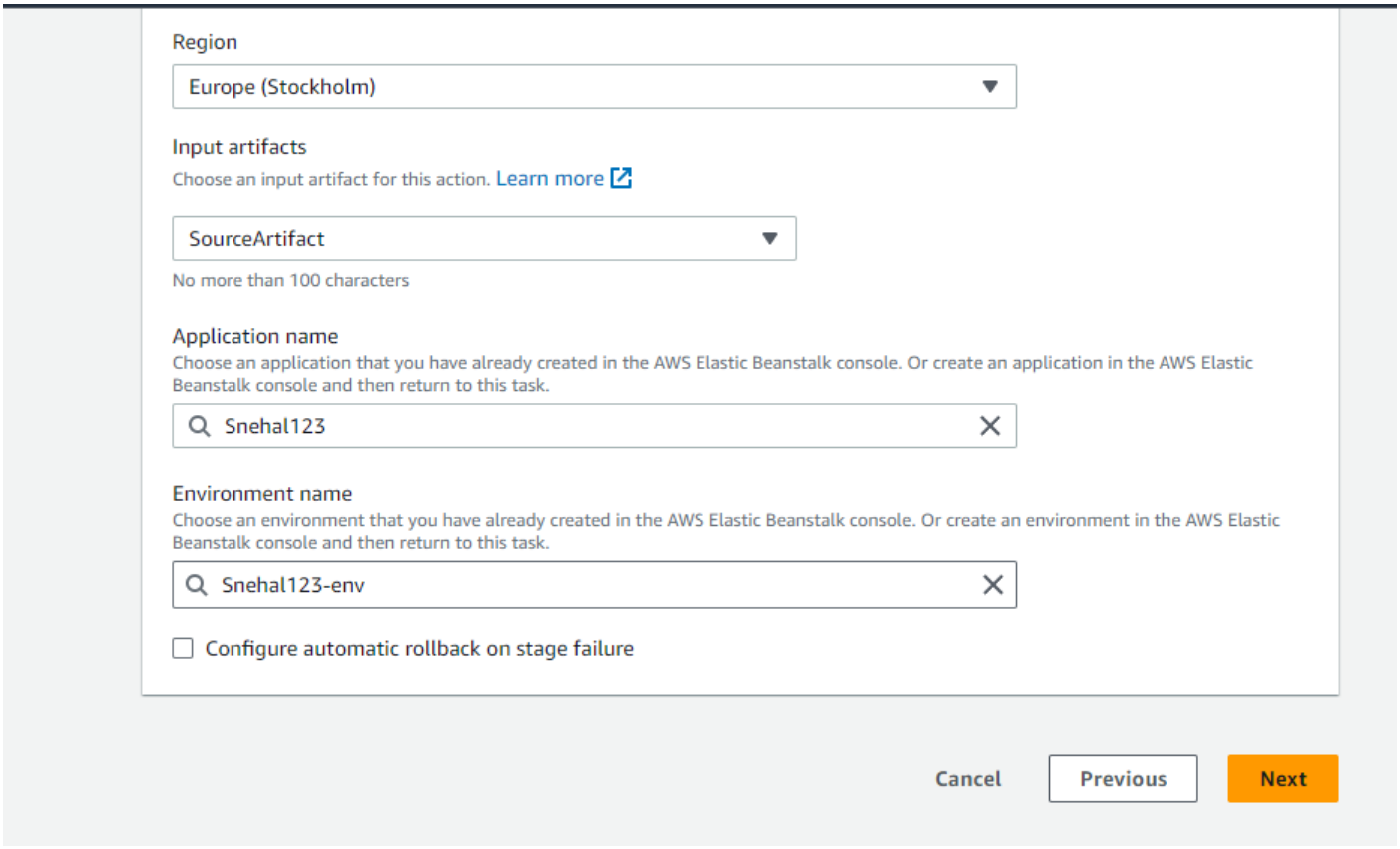
Step 2 : Add Your github account and add the file to add to pipeline deployment



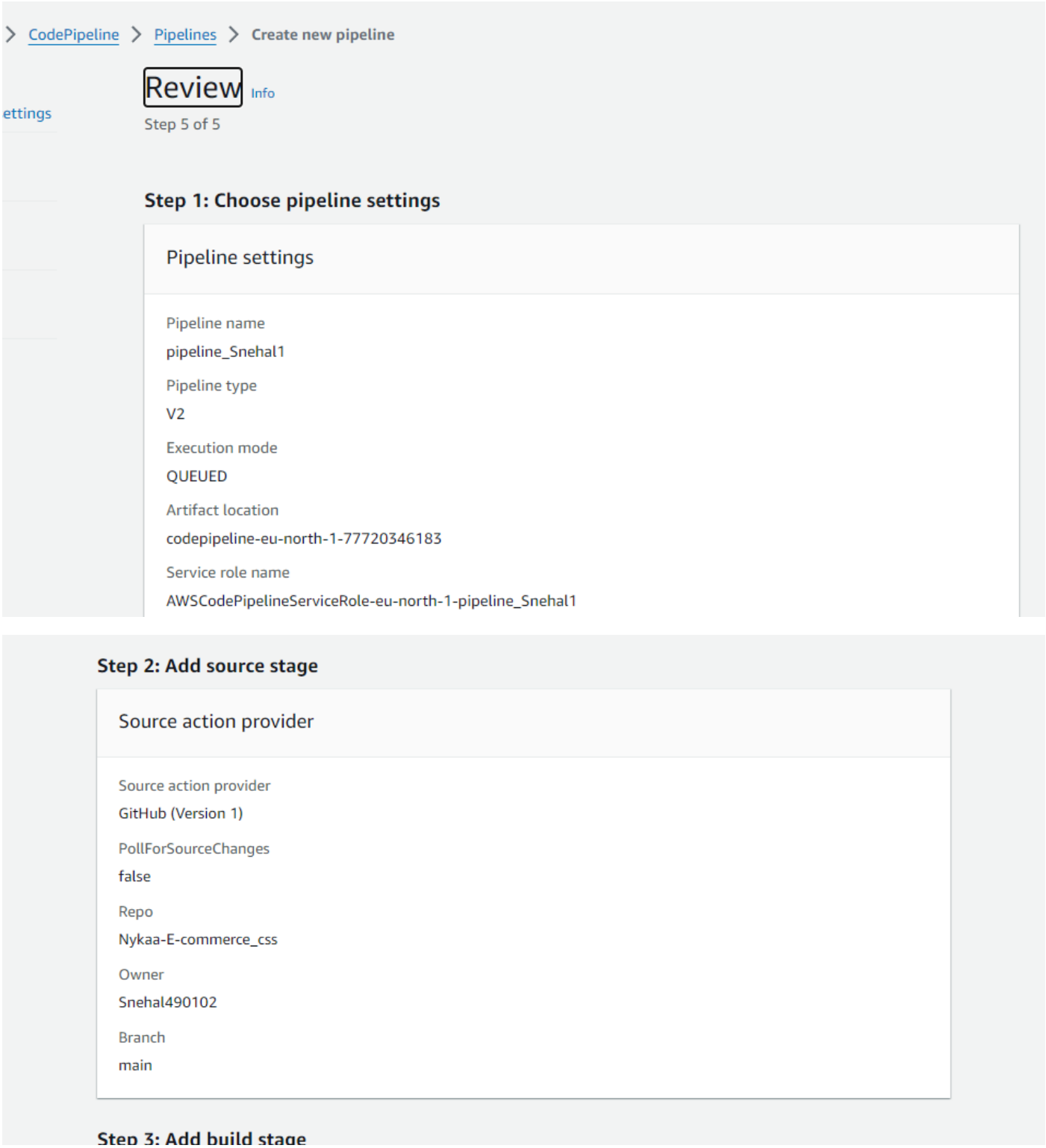


Step 3 : Add deploy config choosing the elastic beanstalk





Step 4 : review changes and submit



Step 2: Add source stage

Source action provider

Source action provider
GitHub (Version 1)

PollForSourceChanges
false

Repo
Nykaa-E-commerce_css

Owner
Snehal490102

Branch
main

Step 4: Add deploy stage

Deploy action provider

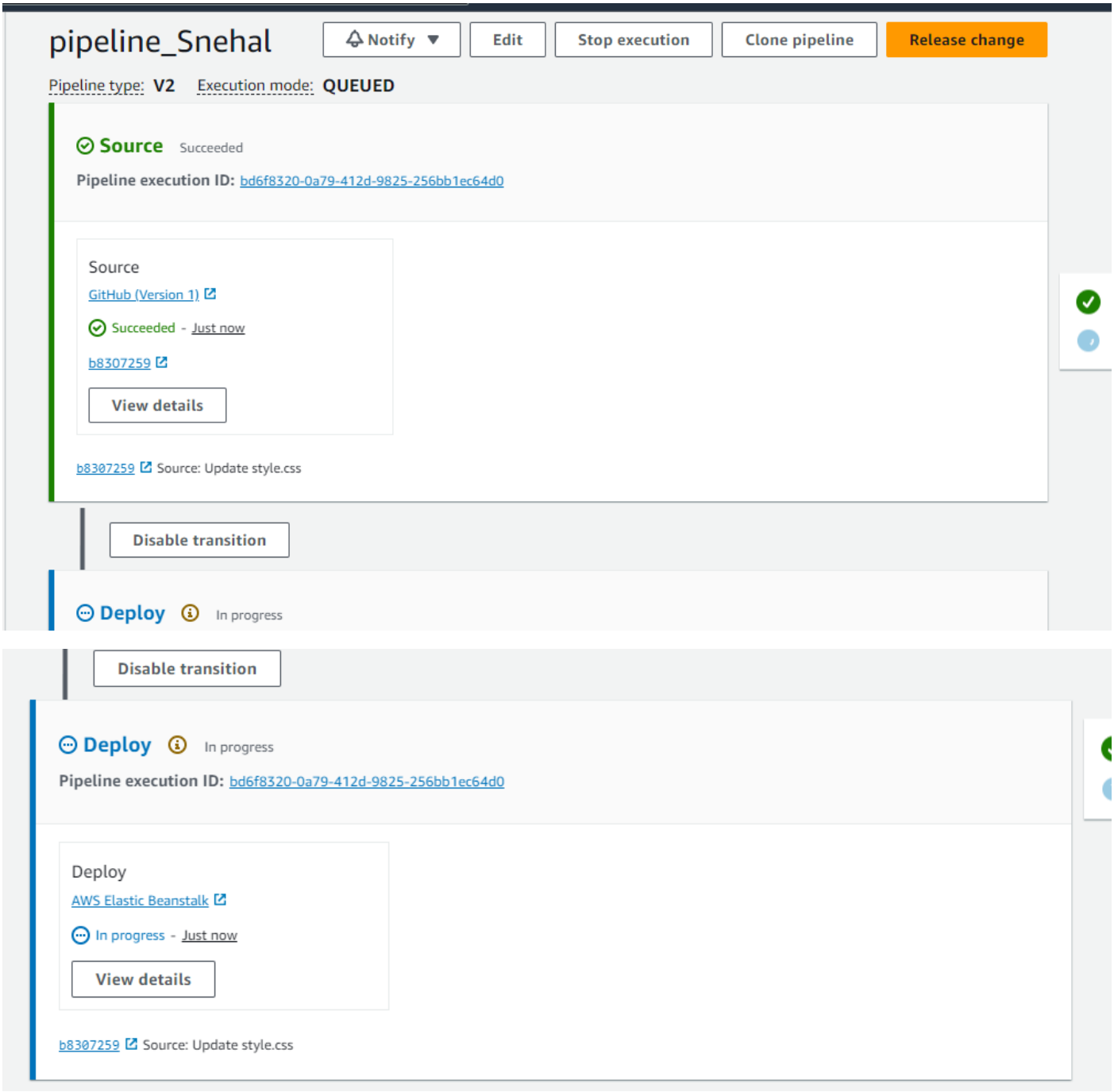
Deploy action provider
AWS Elastic Beanstalk

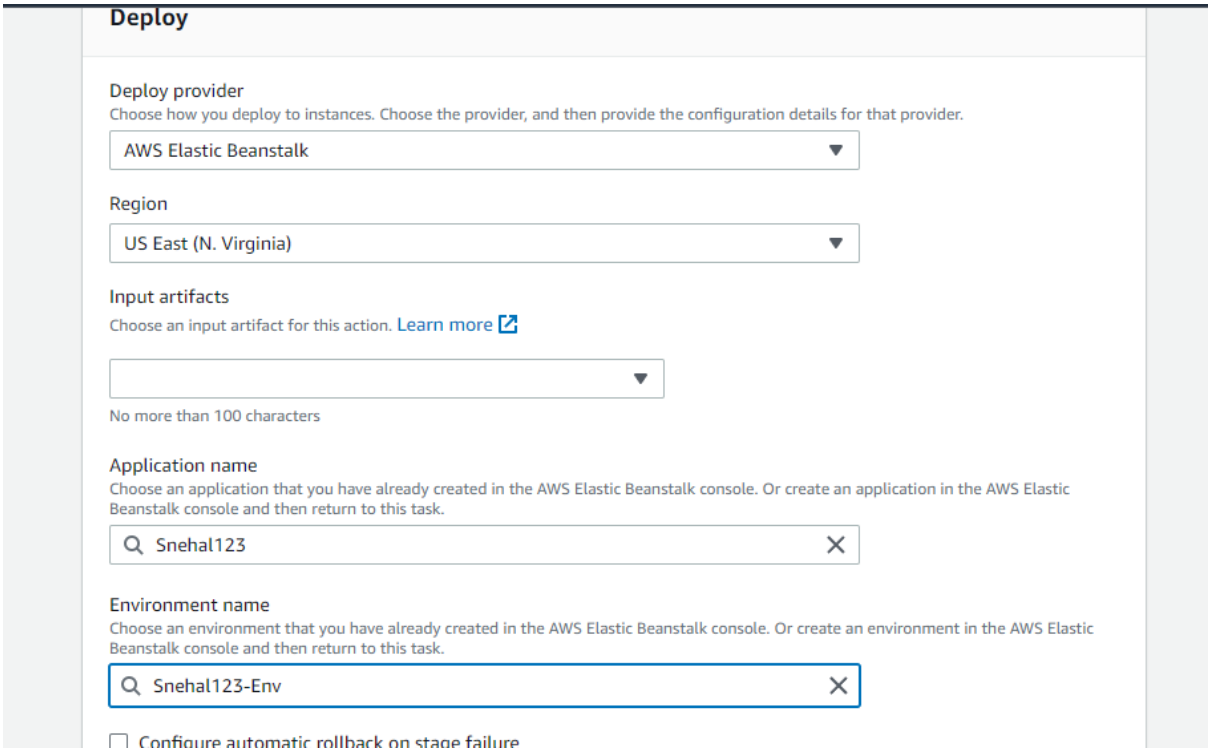
ApplicationName
Snehal123

EnvironmentName
Snehal123-env

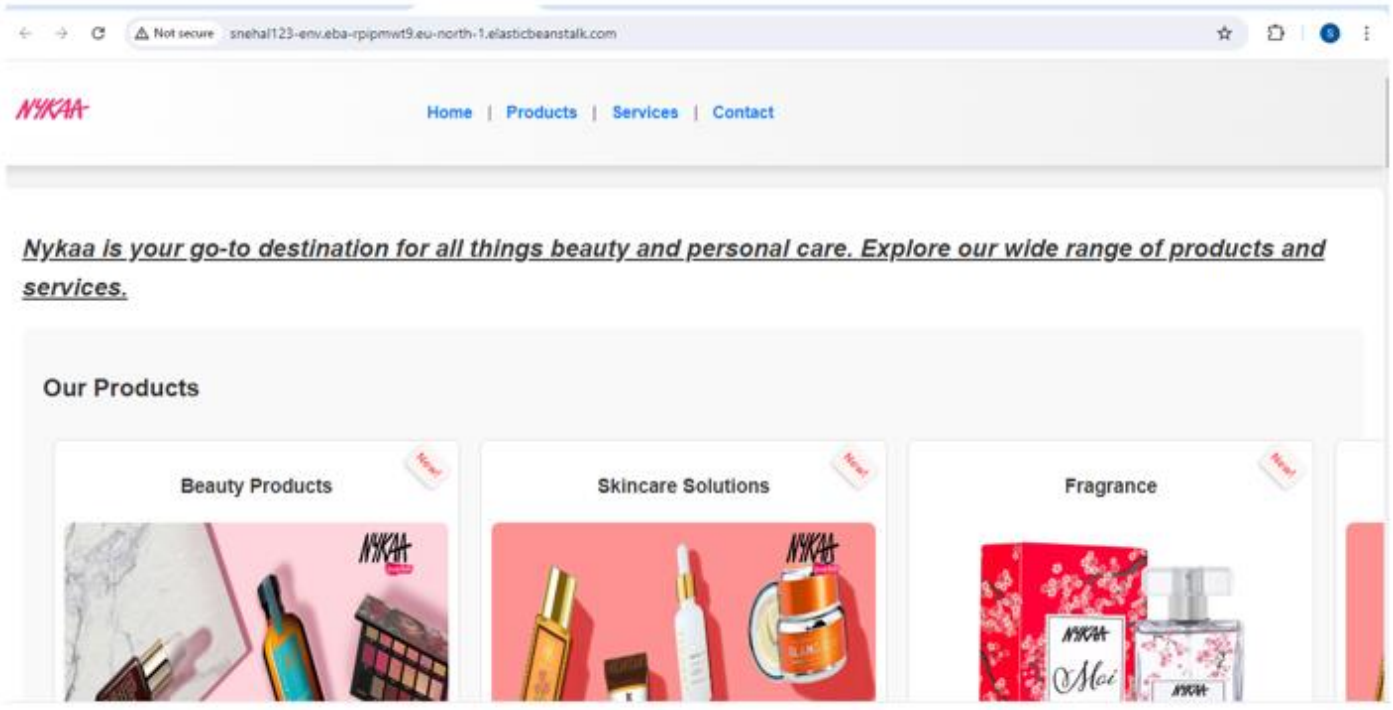
Configure automatic rollback on stage failure
Disabled

Cancel | Previous | Create pipeline


pipeline_Snehal
Notify
Edit
Stop execution
Clone pipeline
Release change
Pipeline type: V2
Execution mode: QUEUED
Source
Succeeded
Pipeline execution ID: bd6f8320-0a79-412d-9825-256bb1ec64d0
Source
GitHub (Version 1)
Succeeded - Just now
b8307259
View details
b8307259 Source: Update style.css
Disable transition
Deploy
In progress
Disable transition
Deploy
In progress
Pipeline execution ID: bd6f8320-0a79-412d-9825-256bb1ec64d0
Deploy
AWS Elastic Beanstalk
In progress - Just now
View details
b8307259 Source: Update style.css



Step 6 : Check the deployed website at beanstalk link



Name: Snehal Patil D15A 39

Advanced DevOps Experiment:3

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

Theory:

To understand Kubernetes Cluster Architecture and how to install and spin up a Kubernetes cluster on Linux machines or cloud platforms, it's essential to grasp the fundamental components and design principles of Kubernetes.

Overview of Kubernetes

Kubernetes is an open-source container orchestration platform developed by Google, designed to automate the deployment, scaling, and management of containerized applications. It provides a robust infrastructure that supports microservices architecture, offering features such as
self-healing, scaling, and zero-downtime deployments. Kubernetes can run on various environments, including public clouds (like AWS and Azure), private clouds, and bare metal servers.

Kubernetes Architecture

Kubernetes architecture is primarily composed of two main components: the Control Plane and the Data Plane.

Control Plane

The Control Plane manages the overall state of the Kubernetes cluster and includes several key components:

- kube-apiserver: The API server acts as the gateway for all interactions with the cluster, processing REST requests and managing the state of the cluster.
- etcd: A distributed key-value store that holds the configuration data and state of the cluster, ensuring consistency and availability.
- kube-scheduler: Responsible for assigning Pods to worker nodes based on resourceavailability and other constraints.
- kube-controller-manager: Manages controllers that regulate the state of the cluster, ensuring that the desired state matches the actual state.
- cloud-controller-manager (optional): Integrates with cloud provider APIs to manage resources specific to the cloud environment.

Data Plane

The Data Plane consists of the worker nodes that run the containerized applications. Each worker node includes:

- kubelet: An agent that ensures containers are running in Pods. It communicates with the Control Plane to receive instructions.
- kube-proxy: Maintains network rules and facilitates communication between Pods and services.
- Container Runtime: Software responsible for running containers, such as Docker or containerd.

Core Concepts

Key concepts in Kubernetes include:

- Pods: The smallest deployable units in Kubernetes, which can contain one or more containers.
- Services: Abstracts a set of Pods, providing a stable network endpoint for accessing them.
- Deployments: Define the desired state for Pods and manage their lifecycle, including scaling and updates.

Installing and Spinning Up a Kubernetes Cluster

To install and set up a Kubernetes cluster, follow these general steps:

1. Choose an Environment: Decide whether to deploy on local machines or a cloud platform. For cloud platforms, services like Google Kubernetes Engine (GKE), Amazon EKS, or Azure AKS can simplify the process.
2. Install Prerequisites: Ensure that you have the necessary tools installed, such as kubectl (the command-line tool for interacting with the cluster) and a container runtime.
3. Set Up the Control Plane: This can be done using tools like kubeadm, which helps bootstrap the cluster by initializing the Control Plane components.
4. Join Worker Nodes: Once the Control Plane is set up, you can join worker nodes to the cluster using the token generated during the initialization.
5. Deploy Applications: After the cluster is up and running, you can deploy your applications using YAML configuration files that define the desired state of your Pods and Services.

Best Practices

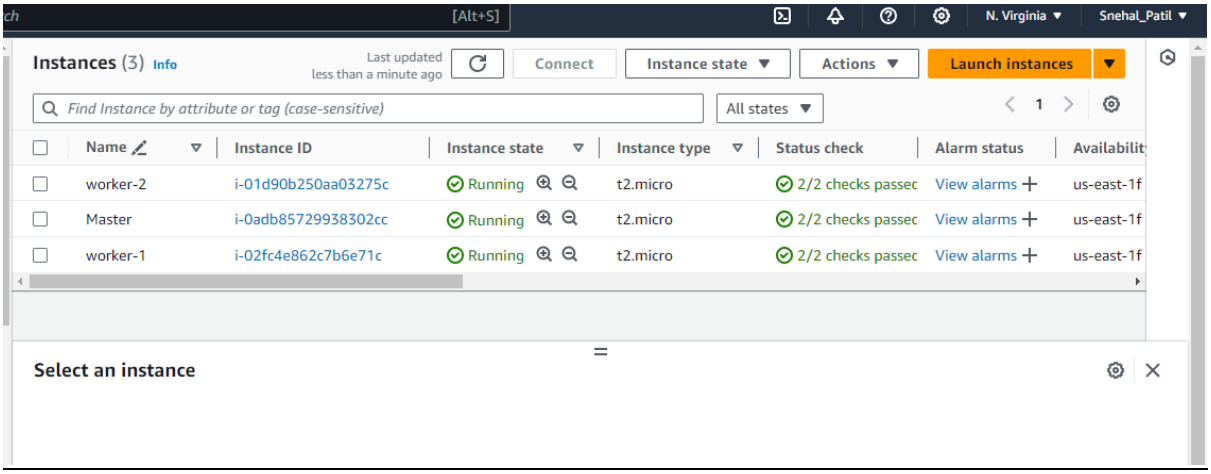
When setting up a Kubernetes cluster, consider the following best practices:

- Resource Management: Define resource requests and limits for Pods to ensure efficient utilization of cluster resources.
- High Availability: Use multiple Control Plane nodes to avoid single points of failure.
- Networking: Implement network policies to secure communication between Pods and manage external access.
- Monitoring and Logging: Integrate monitoring tools and logging solutions to keep track of cluster performance and troubleshoot issues.

By understanding the architecture and following the installation steps and best practices, you can effectively manage a Kubernetes cluster, enabling efficient deployment and scaling of containerized applications.

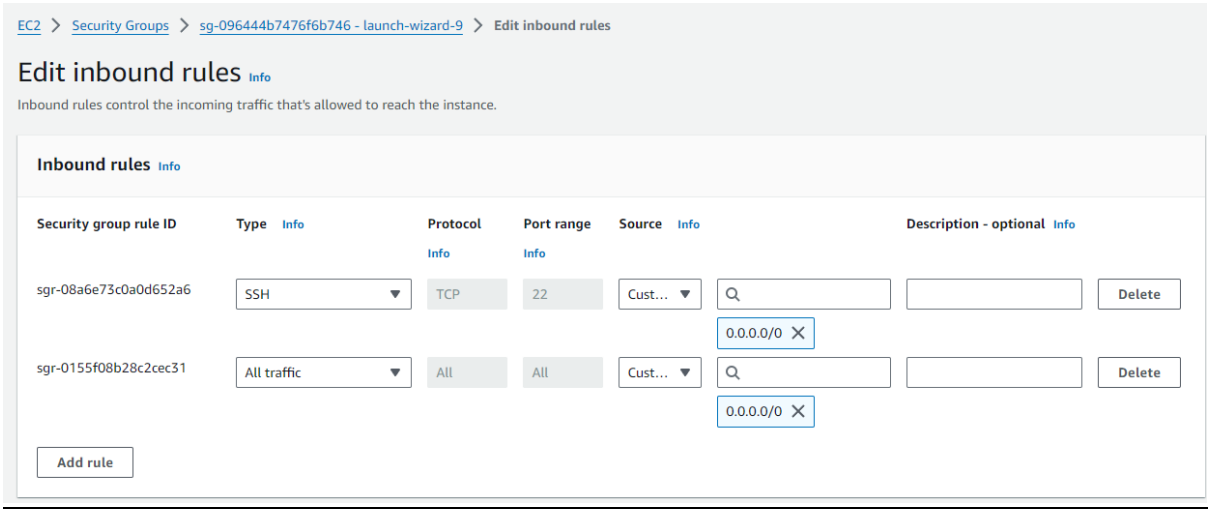
Steps:

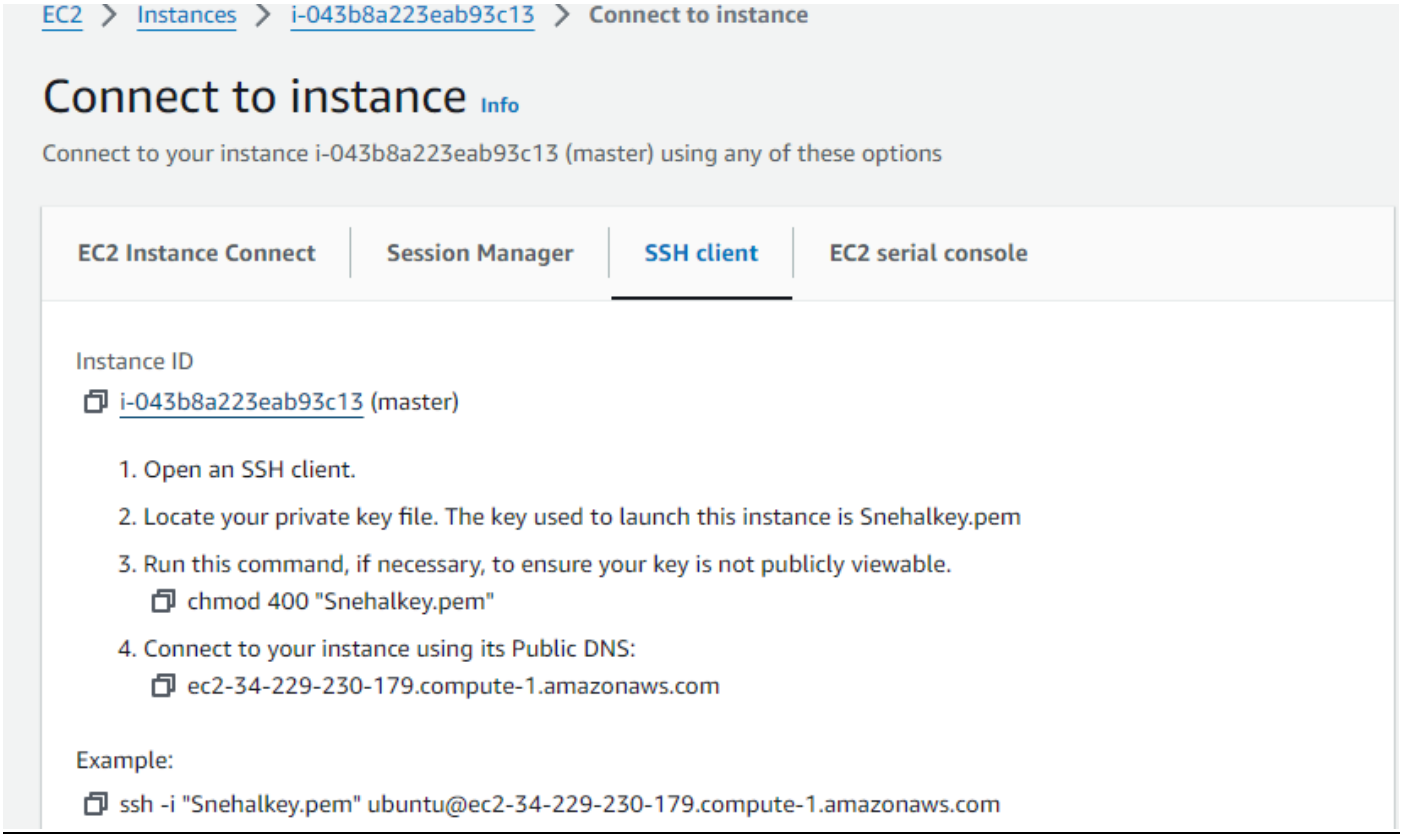
1. Create 3 EC2 Ubuntu Instances on AWS.



2. Now click on connect to instance, then click on SSH client.

Now copy the ssh from the example and paste it on command prompt.





Prerequisites:

Ensure the following requirements are met before starting the Kubernetes cluster setup:

- **Ubuntu OS**: Ubuntu Xenial or later is recommended.
- **sudo Privileges**: Administrative access to execute commands.
- **Instance Type**: Use t2.medium or higher for adequate resources.

Commands to Execute on Both Master and Worker Nodes:

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

```
Windows@DESKTOP-I925HTO MINGW64 ~/Downloads
$ ssh -i "Snehalkey.pem" ubuntu@ec2-44-204-227-4.compute-1.amazonaws.com
The authenticity of host 'ec2-44-204-227-4.compute-1.amazonaws.com (44.204.227.4
)' can't be established.
ED25519 key fingerprint is SHA256:IGnMIHNbyAzruaz0ospRsy1CcVXZPSJ1iGisJHkwTqE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-204-227-4.compute-1.amazonaws.com' (ED25519)
to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Wed Sep 18 09:48:04 UTC 2024

  System load:  0.0               Processes:             114
  Usage of /:   22.8% of 6.71GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for enX0: 172.31.87.198
  Swap usage:   0%
```

```
ubuntu@ip-172-31-87-198:~$ sudo su
root@ip-172-31-87-198:/home/ubuntu# Yum install docker -y
Command 'Yum' not found, did you mean:
  command 'gum' from snap gum (0.13.0)
  command 'uum' from deb freewnn-jserver (1.1.1~a021+cvs20130302-7build1)
  command 'num' from deb quickcal (2.4-1)
  command 'zum' from deb perforate (1.2-5.3)
  command 'sum' from deb coreutils (9.4-2ubuntu2)
See 'snap info <snapname>' for additional versions.
```

Run the following commands on both the master and worker nodes to prepare them for kubeadm.

1. **Disable Swap**:

- Command: sudo swapoff -a
- Explanation: Disables swap memory, which is required by Kubernetes for proper functioning.

2. **Create Configuration File for Kernel Modules:**

- Commands:

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

overlay

br_netfilter

EOF

- Explanation: Loads necessary kernel modules for Kubernetes networking.

```
root@ip-172-31-87-198:/home/ubuntu# sudo swapoff -a
root@ip-172-31-87-198:/home/ubuntu# cat <<EOF | sudo tee /etc/modules-load.d/k8s
.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter
overlay
br_netfilter
```

Configure Sysctl Parameters:

- Command:

bash

Copy code

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-iptables = 1

net.bridge.bridge-nf-call-ip6tables = 1

net.ipv4.ip_forward   = 1

EOF

- Explanation: Sets necessary network parameters for Kubernetes. These settings enable IP forwarding and ensure iptables rules are applied to bridge traffic.

```
root@ip-172-31-87-198:/home/ubuntu# cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
>
> ^C
root@ip-172-31-87-198:/home/ubuntu# cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
```

Apply Sysctl Parameters:

- Command: sudo sysctl --system
- Explanation: Applies the sysctl parameters without rebooting the system.

```
root@ip-172-31-87-198:/home/ubuntu# sudo sysctl --system
* Applying /usr/lib/sysctl.d/10-apparmor.conf ...
* Applying /etc/sysctl.d/10-console-messages.conf ...
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...
* Applying /etc/sysctl.d/10-magic-sysrq.conf ...
* Applying /etc/sysctl.d/10-map-count.conf ...
* Applying /etc/sysctl.d/10-network-security.conf ...
* Applying /etc/sysctl.d/10-ptrace.conf ...
```

Install CRI-O Runtime:

- **Commands**:

  bash
  Copy code
  sudo apt-get update -y
  sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates gpg

  sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg
  echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" | sudo tee /etc/apt/sources.list.d/cri-o.list

  sudo apt-get update -y
  sudo apt-get install -y cri-o

  sudo systemctl daemon-reload
  sudo systemctl enable crio --now
  sudo systemctl start crio.service

- **Explanation**: Installs and configures CRI-O, a container runtime compatible with Kubernetes.

```
root@ip-172-31-87-198:/home/ubuntu# sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https ca-c
ertificates gpg
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [12
6 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [
126 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packag
es [15.0 MB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-
en [5982 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Compon
ents [3871 kB]
```

Add Kubernetes APT Repository and Install Packages:

- **Commands**:

  bash
  Copy code
  curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
  echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

  sudo apt-get update -y
  sudo apt-get install -y kubelet="1.29.0-*" kubectl="1.29.0-*" kubeadm="1.29.0-*"
  sudo apt-get update -y
  sudo apt-get install -y jq

  sudo systemctl enable --now kubelet
  sudo systemctl start kubelet

- **Explanation**: Adds the Kubernetes APT repository, installs Kubernetes tools (kubelet, kubectl, kubeadm), and ensures the kubelet service is running.

```
root@ip-172-31-87-198:/home/ubuntu# sudo curl -fsSL https://pkgs.k8s.io/addons:/
cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyring
s/cri-o-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.i
o/addons:/cri-o:/prerelease:/main/deb/ /" | sudo tee /etc/apt/sources.list.d/cri
-o.list
deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addo
ns:/cri-o:/prerelease:/main/deb/ /
```

```
root@ip-172-31-87-198:/home/ubuntu# sudo apt-get update -y
sudo apt-get install -y cri-o
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [12
6 kB]
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri
-o:/prerelease:/main/deb  InRelease [1206 B]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri
-o:/prerelease:/main/deb  Packages [2454 B]
Fetched 130 kB in 0s (282 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Recommended packages:
  kubernetes-cni
The following NEW packages will be installed:
  cri-o
0 upgraded, 1 newly installed, 0 to remove and 130 not upgraded.
Need to get 19.9 MB of archives.
After this operation, 76.5 MB of additional disk space will be used.
Get:1 https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb  cri-o 1.32.0~dev-
2.1 [19.9 MB]
```

```
root@ip-172-31-87-198:/home/ubuntu# sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl start crio.service
root@ip-172-31-87-198:/home/ubuntu# echo "CRI runtime installed successfully"
CRI runtime installed successfully
root@ip-172-31-87-198:/home/ubuntu# curl -fsSL https://pkgs.k8s.io/core:/stable:
/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-
keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.
k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes
.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io
/core:/stable:/v1.29/deb/ /
root@ip-172-31-87-198:/home/ubuntu# sudo apt-get update -y
sudo apt-get install -y kubelet="1.29.0-*" kubectl="1.29.0-*" kubeadm="1.29.0-*"

sudo apt-get update -y
sudo apt-get install -y jq
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri
```

```
root@ip-172-31-87-198:/home/ubuntu# sudo systemctl enable --now kubelet
sudo systemctl start kubelet
root@ip-172-31-87-198:/home/ubuntu# sudo kubeadm config images pull
I0918 09:59:24.968795    3577 version.go:256] remote version is much newer: v1.3
1.0; falling back to: stable-1.29
[config/images] Pulled registry.k8s.io/kube-apiserver:v1.29.9
[config/images] Pulled registry.k8s.io/kube-controller-manager:v1.29.9
[config/images] Pulled registry.k8s.io/kube-scheduler:v1.29.9
[config/images] Pulled registry.k8s.io/kube-proxy:v1.29.9
[config/images] Pulled registry.k8s.io/coredns/coredns:v1.11.1
[config/images] Pulled registry.k8s.io/pause:3.9
[config/images] Pulled registry.k8s.io/etcd:3.5.10-0
root@ip-172-31-87-198:/home/ubuntu# sudo kubeadm init
I0918 09:59:41.619600    3915 version.go:256] remote version is much newer: v1.3
1.0; falling back to: stable-1.29
[init] Using Kubernetes version: v1.29.9
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
```

Commands to execute on master node:

1. **Pull Kubernetes Control Plane Images**:
   - Command: sudo kubeadm config images pull
   - Explanation: Downloads the required container images for the Kubernetes Control Plane components.

```
root@ip-172-31-87-198:/home/ubuntu# sudo kubeadm config images pull
I0918 09:59:24.968795    3577 version.go:256] remote version is much newer: v1.3
1.0; falling back to: stable-1.29
[config/images] Pulled registry.k8s.io/kube-apiserver:v1.29.9
[config/images] Pulled registry.k8s.io/kube-controller-manager:v1.29.9
[config/images] Pulled registry.k8s.io/kube-scheduler:v1.29.9
[config/images] Pulled registry.k8s.io/kube-proxy:v1.29.9
[config/images] Pulled registry.k8s.io/coredns/coredns:v1.11.1
[config/images] Pulled registry.k8s.io/pause:3.9
[config/images] Pulled registry.k8s.io/etcd:3.5.10-0
```

2. **Initialize the Kubernetes Cluster**:
   - Command: sudo kubeadm init
   - Explanation: Initializes the Kubernetes Control Plane on the Master node

```
root@ip-172-31-87-198:/home/ubuntu# sudo kubeadm init
I0918 09:59:41.619600    3915 version.go:256] remote version is much newer: v1.3
1.0; falling back to: stable-1.29
[init] Using Kubernetes version: v1.29.9
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your inte
rnet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config
 images pull'
W0918 09:59:41.935262    3915 checks.go:835] detected that the sandbox image "re
gistry.k8s.io/pause:3.10" of the container runtime is inconsistent with that use
d by kubeadm. It is recommended that using "registry.k8s.io/pause:3.9" as the CR
I sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-87-198 kuberne
tes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.loc
```

Configure kubectl:

- Commands:

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

- Explanation: Sets up kubectl on the Master node to interact with the cluster.

```
root@ip-172-31-87-198:/home/ubuntu# mkdir -p "$HOME"/.kube
sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config
sudo chown "$(id -u)":"$(id -g)" "$HOME"/.kube/config
```

Install Network Plugin (Calico):

- Command: kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml
- Explanation: Deploys Calico as the network plugin for the cluster.

```
root@ip-172-31-87-198:/home/ubuntu# mkdir -p "$HOME"/.kube
sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config
sudo chown "$(id -u)":"$(id -g)" "$HOME"/.kube/config
root@ip-172-31-87-198:/home/ubuntu# kubectl apply -f https://raw.githubuserconte
nt.com/projectcalico/calico/v3.26.0/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
serviceaccount/calico-cni-plugin created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalic
```

Generating Join Command for Worker Nodes:

- Command: kubeadm token create --print-join-command
- Explanation: Generates the command to join worker nodes to the Kubernetes cluster.

```
root@ip-172-31-87-198:/home/ubuntu# kubeadm token create --print-join-command
kubeadm join 172.31.87.198:6443 --token xqllei.w91ed3g6i8omszpr --discovery-toke
n-ca-cert-hash sha256:750e1245a8b1c658b62daad553dbcd27c89ac2bb81aca75f1b686c4da2
838885
root@ip-172-31-87-198:/home/ubuntu# kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-80-64    Ready    <none>          15s     v1.29.0
ip-172-31-81-208   Ready    <none>          49s     v1.29.0
ip-172-31-87-198   Ready    control-plane   3m53s   v1.29.0
root@ip-172-31-87-198:/home/ubuntu#
```

Commands to execute on worker nodes:

1. **Reset Pre-flight Checks**:
   - Command**:** sudo kubeadm reset
   - Explanation**:** Resets the Kubernetes setup on worker nodes, useful for ensuring a clean state before joining.

```
root@ip-172-31-81-208:/home/ubuntu# sudo kubeadm reset pre-flight checks
W0918 10:01:40.885084    3738 preflight.go:56] [reset] WARNING: Changes made to
this host by 'kubeadm init' or 'kubeadm join' will be reverted.
[reset] Are you sure you want to proceed? [y/N]: yes
[preflight] Running pre-flight checks
W0918 10:01:46.735217    3738 removeetcdmember.go:106] [reset] No kubeadm config
, using etcd pod spec to get data directory
[reset] Deleted contents of the etcd data directory: /var/lib/etcd
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
[reset] Deleting contents of directories: [/etc/kubernetes/manifests /var/lib/ku
belet /etc/kubernetes/pki]
[reset] Deleting files: [/etc/kubernetes/admin.conf /etc/kubernetes/super-admin.
conf /etc/kubernetes/kubelet.conf /etc/kubernetes/bootstrap-kubelet.conf /etc/ku
bernetes/controller-manager.conf /etc/kubernetes/scheduler.conf]
```

2. **Join the Cluster**:

- Command: sudo kubeadm join 172.31.87.198:6443 --token xqllei.w91ed3g6i8omszpr --discovery-token-ca-cert-hash sha256:750e1245a8b1c658b62daad553dbcd27c89ac2bb81aca75f1b686c4da2838885 --v=5
- Explanation**:** Joins the worker node to the Kubernetes cluster using the join command generated from the Master node.

```
root@ip-172-31-81-208:/home/ubuntu# sudo kubeadm join 172.31.87.198:6443 --token
 xqllei.w91ed3g6i8omszpr --discovery-token-ca-cert-hash sha256:750e1245a8b1c658b
62daad553dbcd27c89ac2bb81aca75f1b686c4da2838885 --v=5
I0918 10:02:52.014281    3760 join.go:413] [preflight] found NodeName empty; usi
ng OS hostname as NodeName
I0918 10:02:52.015029    3760 initconfiguration.go:122] detected and using CRI s
ocket: unix:///var/run/crio/crio.sock
[preflight] Running pre-flight checks
I0918 10:02:52.015127    3760 preflight.go:93] [preflight] Running general check
s
I0918 10:02:52.015154    3760 checks.go:280] validating the existence of file /e
tc/kubernetes/kubelet.conf
I0918 10:02:52.015223    3760 checks.go:280] validating the existence of file /e
tc/kubernetes/bootstrap-kubelet.conf
I0918 10:02:52.015235    3760 checks.go:104] validating the container runtime
I0918 10:02:52.034714    3760 checks.go:639] validating whether swap is enabled
or not
I0918 10:02:52.034785    3760 checks.go:370] validating the presence of executab
le crictl
```

Verify Cluster Connection

1. **Check Nodes**:
    - **Command**: kubectl get nodes
    - **Explanation**: Verifies that all nodes (Master and Worker) have successfully joined the cluster and are in a ready state.

```
root@ip-172-31-87-198:/home/ubuntu# kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-80-64    Ready    <none>          15s     v1.29.0
ip-172-31-81-208   Ready    <none>          49s     v1.29.0
ip-172-31-87-198   Ready    control-plane   3m53s   v1.29.0
root@ip-172-31-87-198:/home/ubuntu#
```

Name: Snehal Patil D15A 39

ADVANCE DEVOPS EXP: 4

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

Theory:

What is kubectl?

kubectl is the command-line interface (CLI) used to interact with a Kubernetes cluster. It allows users to manage cluster resources, deploy applications, inspect and manage cluster components, and much more. Using kubectl, you can communicate with the Kubernetes API server to issue commands and queries.

Common kubectl commands:

- kubectl get: View information about resources.
- kubectl describe: Detailed description of resources.
- kubectl create/apply: Create or update resources.
- kubectl delete: Delete resources.

kubectl plays a crucial role in the day-to-day operation of a Kubernetes cluster.

Basic Concepts in Kubernetes

Before diving into the application deployment process, it's important to understand a few key Kubernetes objects:

1. **Pods**: The smallest deployable unit in Kubernetes. A pod encapsulates one or more containers (usually a single container) that share the same network namespace and storage.
2. **Deployments**: A Kubernetes resource that defines how to create and manage pods. It ensures the specified number of pod replicas are running at any given time and handles updates and rollbacks.
3. **Services**: An abstraction that defines how to access the pods. A service allows you to expose your pods to internal or external clients.
4. **ReplicaSets**: Ensures that a specified number of pod replicas are running at all times. It is managed by a Deployment, but can also be used independently.

1.1 Install prerequisites:

sudo apt-get update

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

```
root@ip-172-31-87-198:/home/ubuntu# sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https ca-c
ertificates gpg

sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.
key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.i
o/addons:/cri-o:/prerelease:/main/deb/ /" | sudo tee /etc/apt/sources.list.d/cri
-o.list

sudo apt-get update -y
sudo apt-get install -y cri-o
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri
-o:/prerelease:/main/deb  InRelease
Ign:5 https://packages.cloud.google.com/apt kubernetes-focal InRelease
Err:7 https://packages.cloud.google.com/apt kubernetes-focal Release
  404  Not Found [IP: 172.253.122.100 443]
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 disa
bled by default.
N: See apt-secure(8) manpage for repository creation and user configuration deta
ils.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
software-properties-common is already the newest version (0.99.48).
curl is already the newest version (8.5.0-2ubuntu10.4).
apt-transport-https is already the newest version (2.7.14build2).
ca-certificates is already the newest version (20240203).
gpg is already the newest version (2.4.4-2ubuntu17).
0 upgraded, 0 newly installed, 0 to remove and 130 not upgraded.
File '/etc/apt/keyrings/cri-o-apt-keyring.gpg' exists. Overwrite? (y/N) y
deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addo
ns:/cri-o:/prerelease:/main/deb/ /
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
```

1. **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

```
root@ip-172-31-87-198:/home/ubuntu# sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-ke
https://packages.cloud.google.com/apt/doc/apt-key.gpg
curl: (2) no URL specified
curl: try 'curl --help' or 'curl --manual' for more information
hash: https://packages.cloud.google.com/apt/doc/apt-key.gpg: No such file or directory
```

2. **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

```
root@ip-172-31-87-198:/home/ubuntu# echo "deb [signed-by=/usr/share/keyrings/kubernetes-archiv
https://apt.kubernetes.io/ kubernetes-focal main" | sudo tee
/etc/apt/sources.list.d/kubernetes.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

```
root@ip-172-31-87-198:/home/ubuntu# sudo apt-get update
sudo apt-get install -y kubectl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri-o:/prerelease
Ign:6 https://packages.cloud.google.com/apt kubernetes-focal InRelease
Err:7 https://packages.cloud.google.com/apt kubernetes-focal Release
  404  Not Found [IP: 172.253.122.102 443]
Reading package lists... Done
E: The repository 'https://apt.kubernetes.io kubernetes-focal Release' does not have a Release
N: Updating from such a repository can't be done securely, and is therefore disabled by defaul
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 130 not upgraded.
```

```
root@ip-172-31-87-198:/home/ubuntu# nano nginx-deployment.yaml
root@ip-172-31-87-198:/home/ubuntu# nano nginx-service.yaml
```

Verifying the installation:

Kubectl version --client

```
root@ip-172-31-87-198:/home/ubuntu# kubectl version --client
Client Version: v1.29.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
```

Step 2: Deploying the Application on Kubernetes

2.1 Setting 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 acluster running.
2. Once your cluster is ready,confirm that all the nodes

   are successfully connected and operational.

   Command: kubectl get nodes

```
root@ip-172-31-87-198:/home/ubuntu# kubectl get nodes
NAME               STATUS   ROLES           AGE   VERSION
ip-172-31-80-64    Ready    <none>          18s   v1.29.0
ip-172-31-81-208   Ready    <none>          35s   v1.29.0
ip-172-31-87-198   Ready    control-plane   43m   v1.29.0
```

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

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

```
root@ip-172-31-87-198:/home/ubuntu# nano nginx-deployment.yaml
root@ip-172-31-87-198:/home/ubuntu# nano nginx-service.yaml
root@ip-172-31-87-198:/home/ubuntu# kubectl apply of nginx-deployment.yaml
error: Unexpected args: [of nginx-deployment.yaml]
See 'kubectl apply -h' for help and examples
root@ip-172-31-87-198:/home/ubuntu# kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
root@ip-172-31-87-198:/home/ubuntu# kubectl apply -f nginx-service.yaml
service/nginx-service created
```

b)Adding the Deployment Configuration to nginx-deployment.yaml and nginx-service.yaml

root@ip-172-31-87-198: /home/ubuntu

```
  GNU nano 7.2                    nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx

spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.21.3
                          [ Read 22 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

root@ip-172-31-87-198: /home/ubuntu

```
  GNU nano 7.2                    nginx-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

Step 4:Applying the YAML Files

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

```
root@ip-172-31-87-198:/home/ubuntu# kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
root@ip-172-31-87-198:/home/ubuntu# kubectl apply -f nginx-service.yaml
service/nginx-service created
```

Verifying the Deployment and also describing the deployment:

Check the status of your Deployment,Pods and Services.

```
root@ip-172-31-87-198:/home/ubuntu# kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2            2           5m57s
root@ip-172-31-87-198:/home/ubuntu# kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2            2           6m39s
```

```
root@ip-172-31-87-198:/home/ubuntu# kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2            2           28m
root@ip-172-31-87-198:/home/ubuntu# kubectl describe deployment
Name:                   nginx-deployment
Namespace:              default
CreationTimestamp:      Wed, 18 Sep 2024 12:14:59 +0000
Labels:                 app=nginx
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=nginx
Replicas:               2 desired | 2 updated | 2 total | 2 available | 0 unavai
lable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx:1.21.3
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
```

```
root@ip-172-31-87-198:/home/ubuntu# kubectl get service
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP      10.96.0.1        <none>        443/TCP        55m
nginx-service   LoadBalancer   10.109.148.186   <pending>     80:30162/TCP   31m
root@ip-172-31-87-198:/home/ubuntu#
```

Step 6: Ensure Service is Running

6.1 **Verify Service**: Running the following commands to check the services running in our cluster:

Command: kubectl get service

```
root@ip-172-31-87-198:/home/ubuntu# kubectl get service
NAME            TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP      10.96.0.1      <none>        443/TCP        101m
nginx-service   LoadBalancer   10.106.17.37   <pending>     80:31687/TCP   105s
root@ip-172-31-87-198:/home/ubuntu# kubectl port-forward service/nginx-service 8
080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
^Croot@ip-172-31-87-198:/home/ubuntu# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6b4d6fdbf-n52mq    1/1     Running   0          9m20s
nginx-deployment-6b4d6fdbf-w9qjv    1/1     Running   0          9m20s
root@ip-172-31-87-198:/home/ubuntu# kubectl logs nginx-deployment-6b4d6fdbf-n52m
q
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perfo
rm configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-defau
lt.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: Launching /docker-entrypoint.d/20-envsubst-on-templates.s
h
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.s
h
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/09/18 11:34:24 [notice] 1#1: using the "epoll" event method
2024/09/18 11:34:24 [notice] 1#1: nginx/1.21.3
2024/09/18 11:34:24 [notice] 1#1: built by gcc 8.3.0 (Debian 8.3.0-6)
2024/09/18 11:34:24 [notice] 1#1: OS: Linux 6.8.0-1012-aws
2024/09/18 11:34:24 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/09/18 11:34:24 [notice] 1#1: start worker processes
2024/09/18 11:34:24 [notice] 1#1: start worker process 26
2024/09/18 11:34:24 [notice] 1#1: start worker process 27
```

```
root@ip-172-31-87-198:/home/ubuntu# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6b4d6fdbf-bqqg2    1/1     Running   0          35m
nginx-deployment-6b4d6fdbf-ptgmg    1/1     Running   0          35m
```

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.

Command:

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

```
root@ip-172-31-87-198:/home/ubuntu# kubectl port-forward service/nginx-service 8
080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

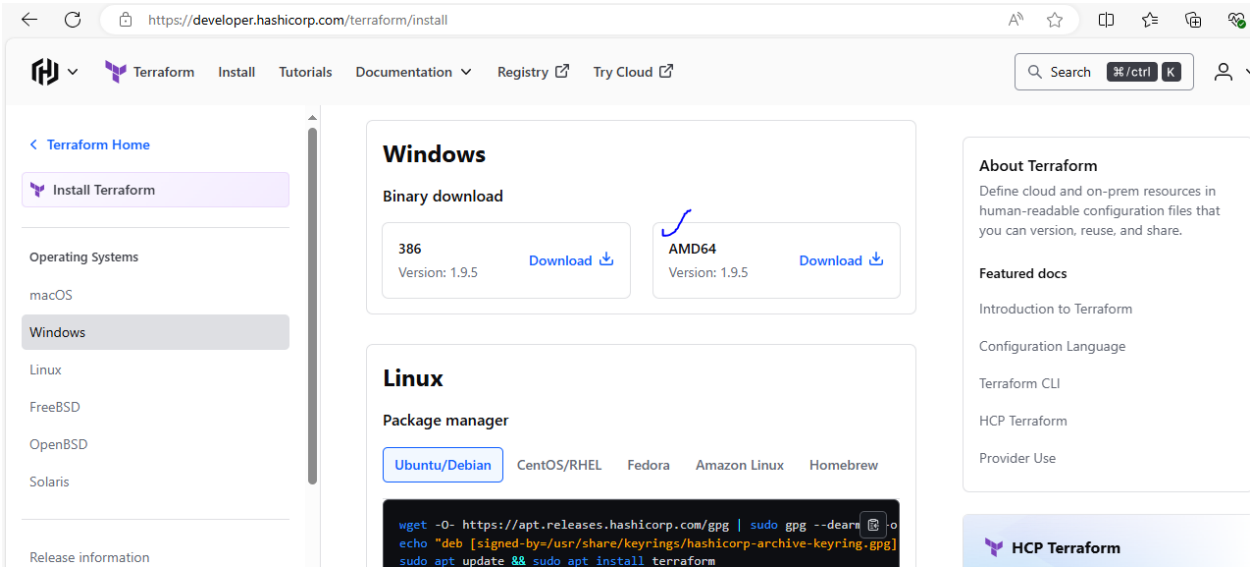
```
root@ip-172-31-87-198:/home/ubuntu# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6b4d6fdbf-bqqg2    1/1     Running   0          35m
nginx-deployment-6b4d6fdbf-ptgmg    1/1     Running   0          35m
root@ip-172-31-87-198:/home/ubuntu# kubectl logs nginx-deployment-6b4d6fdbf-bqqg
2
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perfo
rm configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-defau
lt.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: Launching /docker-entrypoint.d/20-envsubst-on-templates.s
h
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.s
h
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/09/18 12:34:16 [notice] 1#1: using the "epoll" event method
2024/09/18 12:34:16 [notice] 1#1: nginx/1.21.3
2024/09/18 12:34:16 [notice] 1#1: built by gcc 8.3.0 (Debian 8.3.0-6)
2024/09/18 12:34:16 [notice] 1#1: OS: Linux 6.8.0-1012-aws
2024/09/18 12:34:16 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/09/18 12:34:16 [notice] 1#1: start worker processes
2024/09/18 12:34:16 [notice] 1#1: start worker process 26
2024/09/18 12:34:16 [notice] 1#1: start worker process 27
```

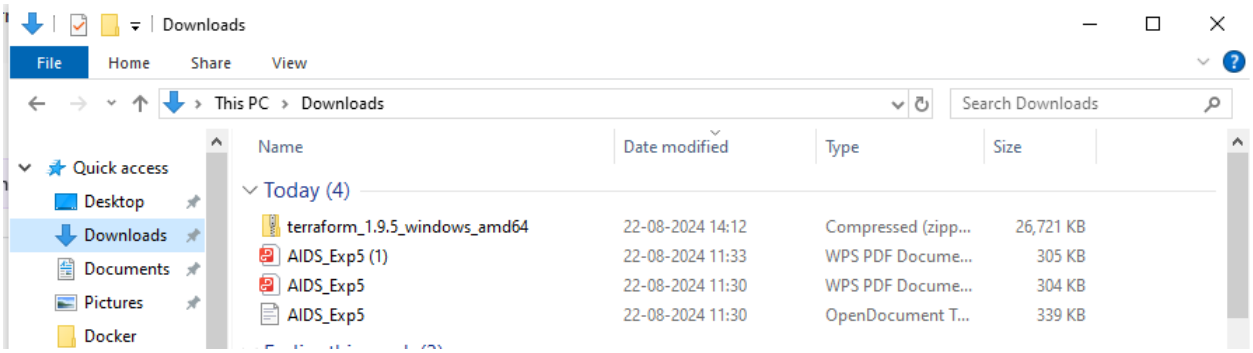
Step 8: Access the Application

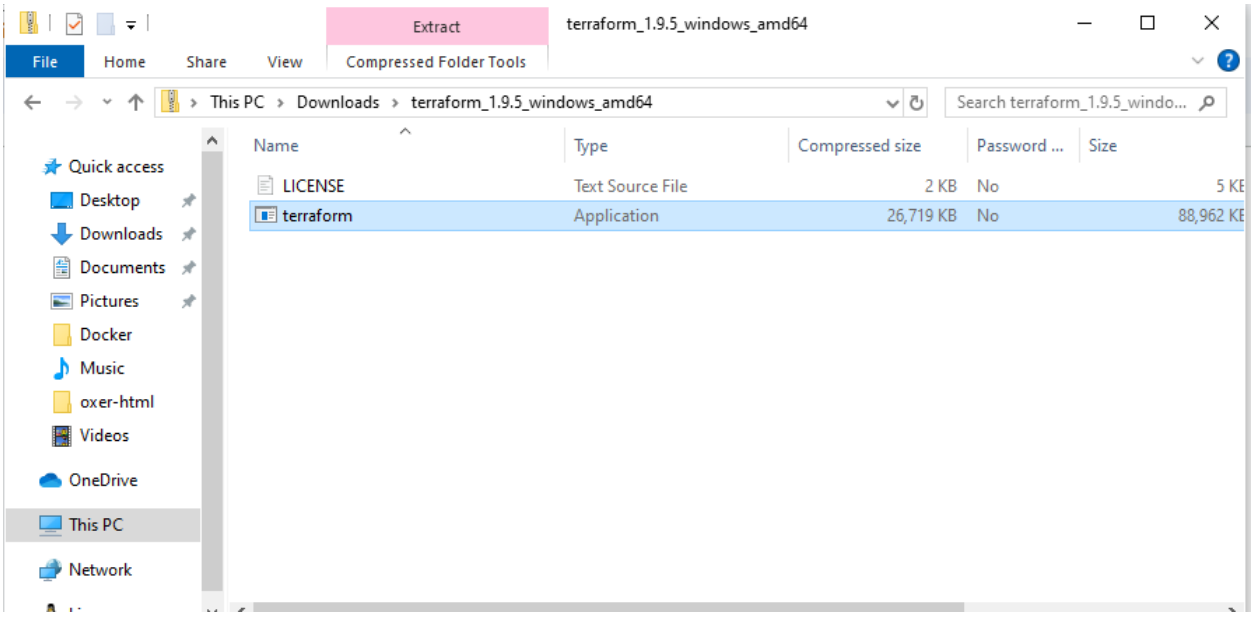
- Open a web browser and navigate to `http://<Node-IP>:<Port>`. You should see the NGINX application running in the Kubernetes cluster.

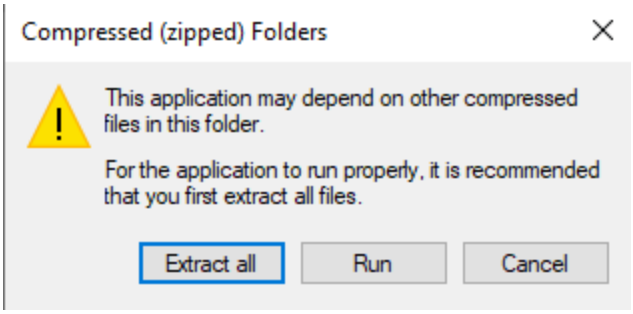
Experiment No 5 : Terraform

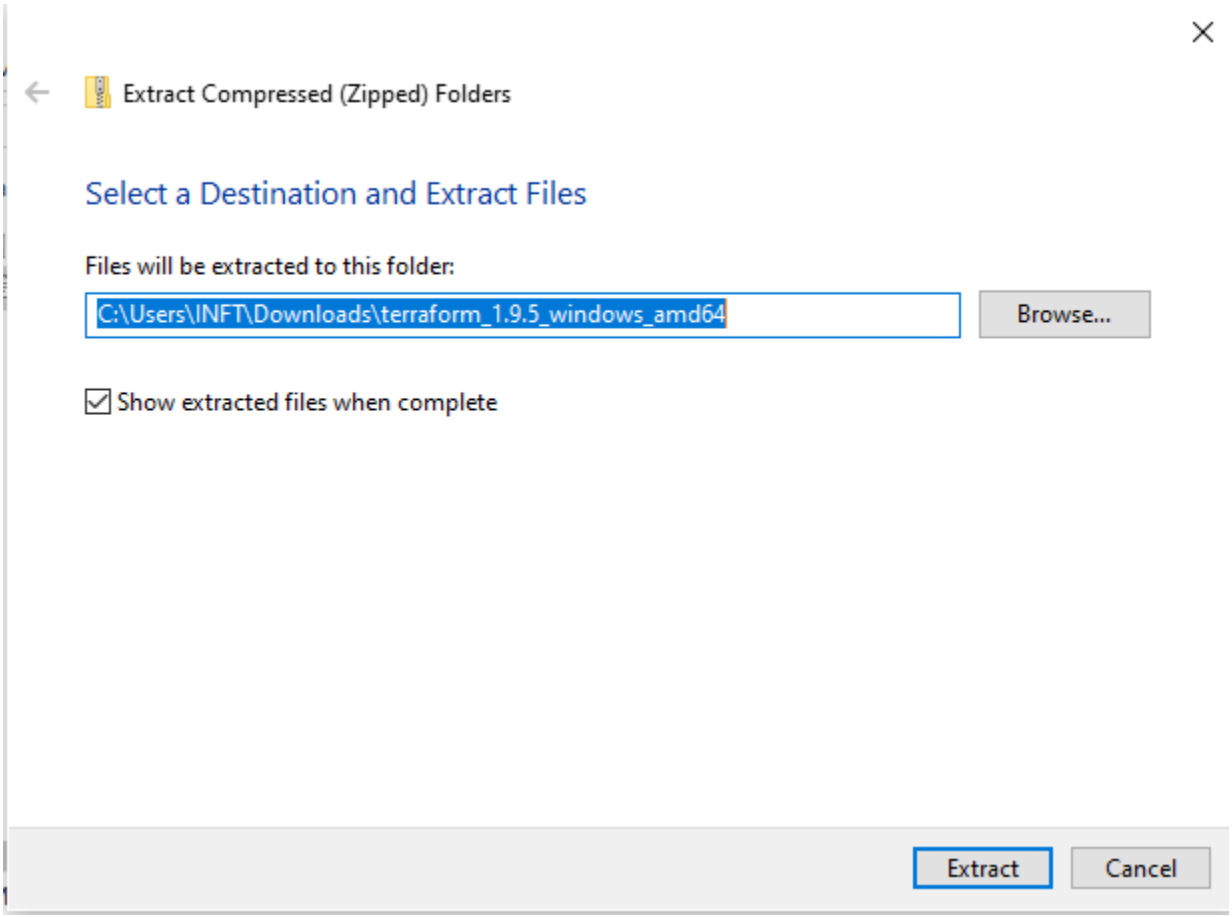
Aim : Installation and Configuration of Terraform in Windows

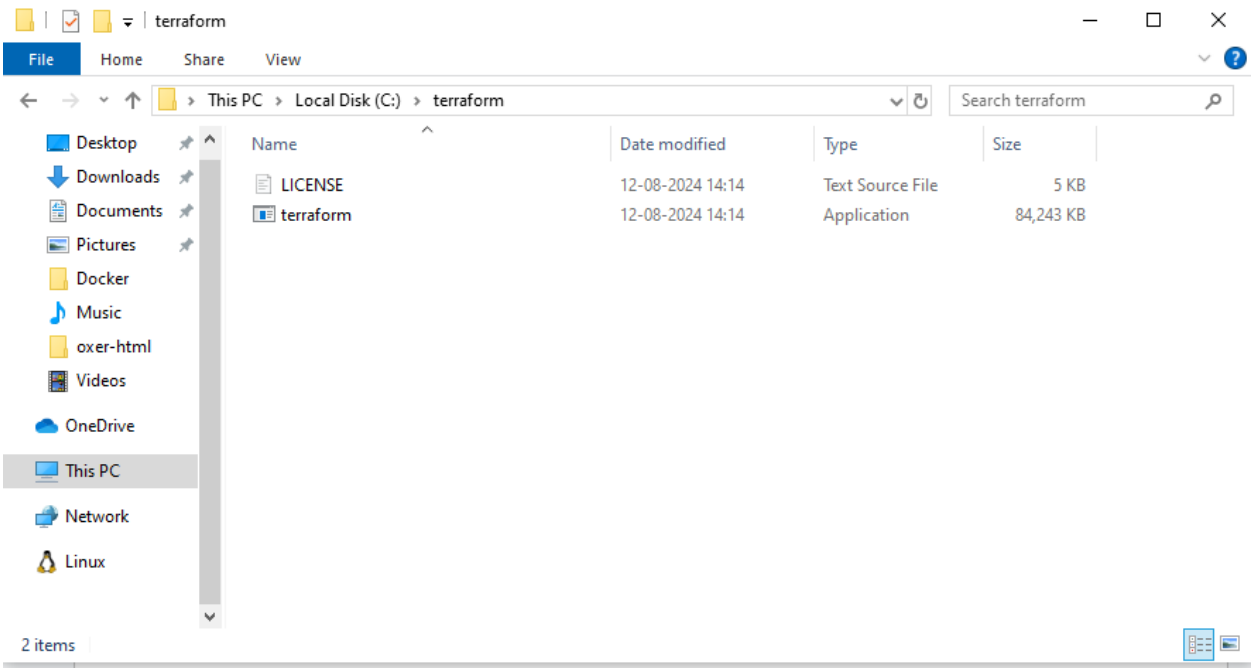


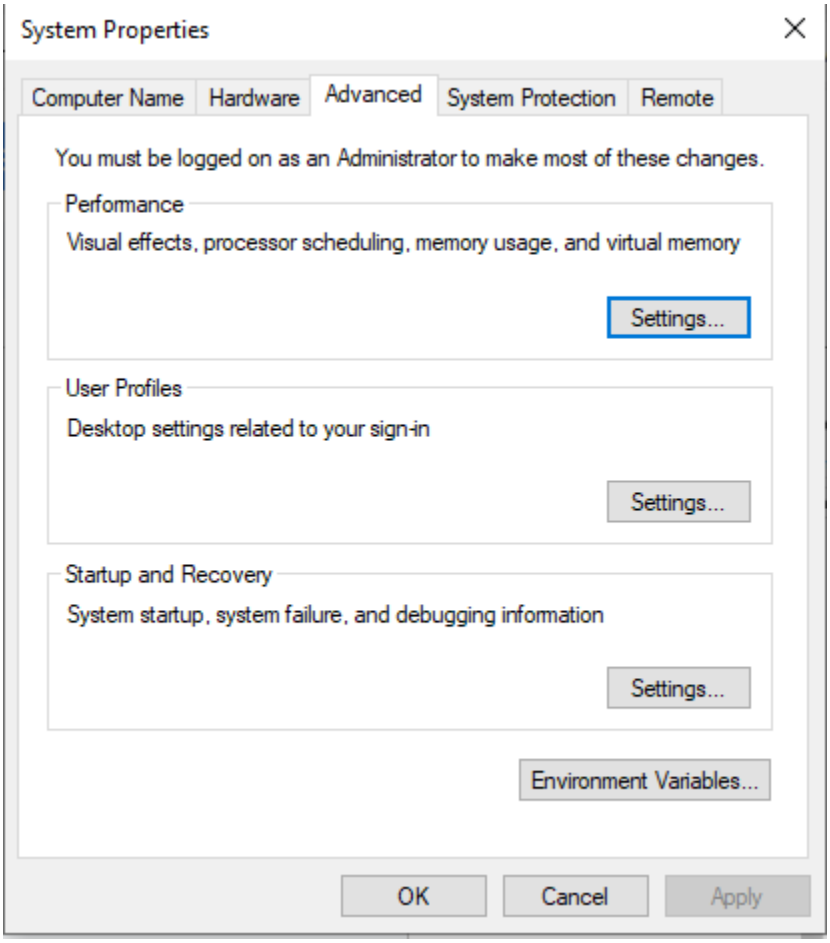


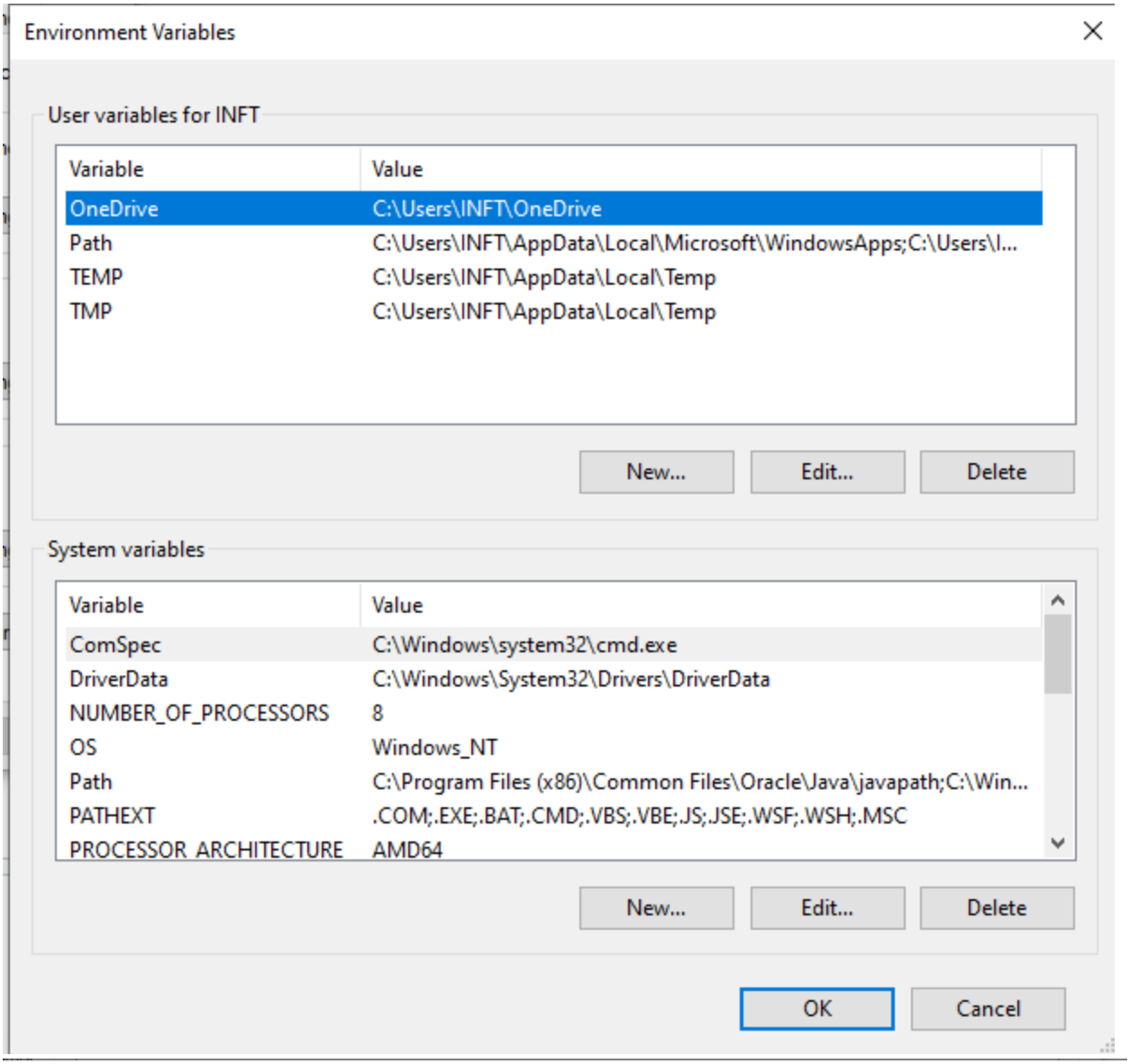


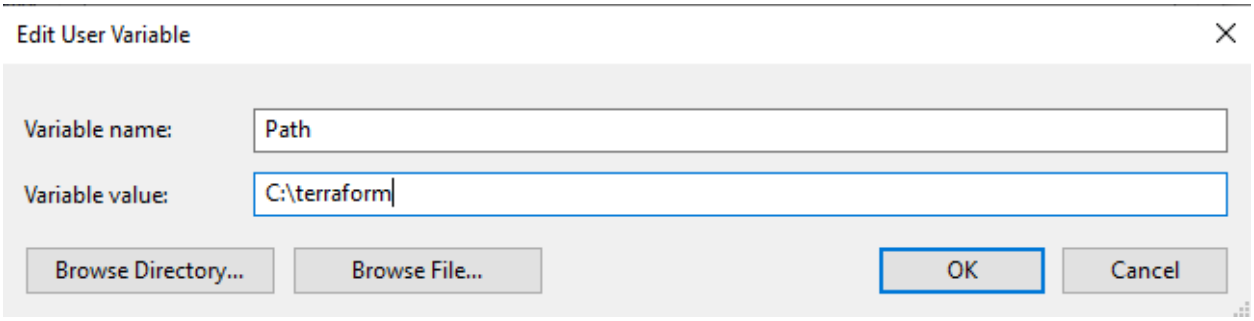

Compressed (zipped) Folders
This application may depend on other compressed files in this folder.
For the application to run properly, it is recommended that you first extract all files.
Extract all
Run
Cancel

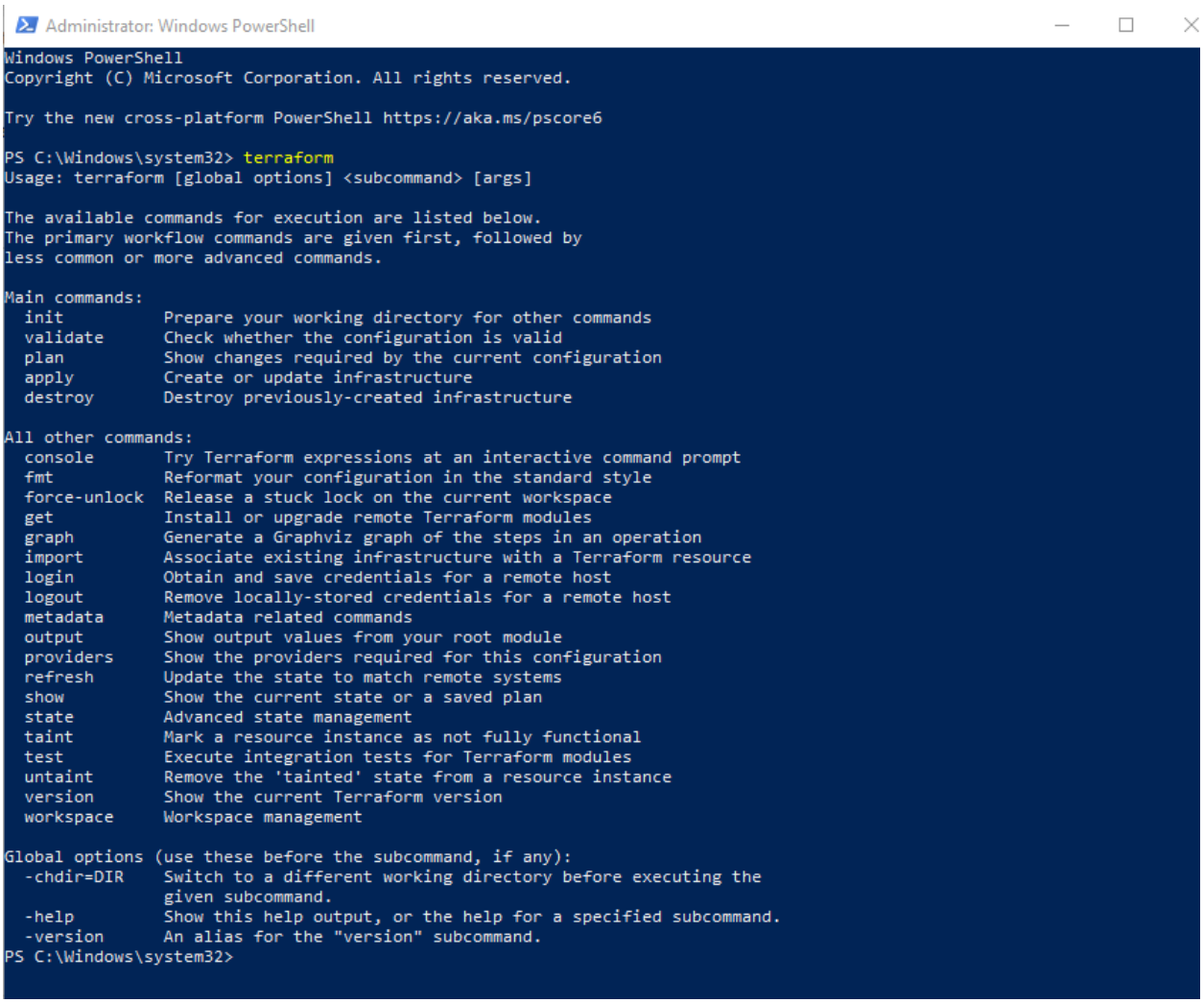

Extract Compressed (Zipped) Folders
Select a Destination and Extract Files
Files will be extracted to this folder:
C:\Users\INFT\Downloads\terraform_1.9.5_windows_amd64
Browse...
Show extracted files when complete
Extract
Cancel


terraform
File
Home
Share
View
This PC
Local Disk (C:)
terraform
Search terraform
Desktop
Downloads
Documents
Pictures
Docker
Music
oxer-html
Videos
OneDrive
This PC
Network
Linux
Name
Date modified
Type
Size
LICENSE
12-08-2024 14:14
Text Source File
5 KB
terraform
12-08-2024 14:14
Application
84,243 KB
2 items


System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...
User Profiles
Desktop settings related to your sign-in
Settings...
Startup and Recovery
System startup, system failure, and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply


Environment Variables
User variables for INFT
Variable
Value
OneDrive
C:\Users\INFT\OneDrive
Path
C:\Users\INFT\AppData\Local\Microsoft\WindowsApps;C:\Users\I...
TEMP
C:\Users\INFT\AppData\Local\Temp
TMP
C:\Users\INFT\AppData\Local\Temp
New...
Edit...
Delete
System variables
Variable
Value
ComSpec
C:\Windows\system32\cmd.exe
DriverData
C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS
8
OS
Windows_NT
Path
C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Win...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE
AMD64
New...
Edit...
Delete
OK
Cancel


Edit User Variable
Variable name:
Path
Variable value:
C:\terraform
Browse Directory...
Browse File...
OK
Cancel


Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Windows\system32> 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:\Windows\system32>

SNEHAL A. PATIL **D15A - 39**

EXPERIMENT NO. 6

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

Theory :

Terraform is an open-source tool that enables developers and operations teams to define,provision, and manage cloud infrastructure through code. It uses a declarative language to specify the desired state of infrastructure, which can include servers, storage, networking components, and more. With Terraform, infrastructure changes can be automated, versioned, and tracked efficiently.

Building Infrastructure

When you build infrastructure using Terraform, you define the desired state of your infrastructure in configuration files. For example, you may want to create an S3 bucket or deploy a Docker container on an EC2 instance. Terraform reads these configuration files and, using the specified cloud provider (such as AWS), it provisions the necessary resources to match the desired state.

- **S3 Buckets:** Terraform can create and manage S3 buckets, which are used to store and retrieve data objects in the cloud. You can define the properties of the bucket, such as its name, region, access permissions, and versioning.
- **Docker on AWS:** Terraform can deploy Docker containers on AWS infrastructure.This often involves setting up an EC2 instance and configuring it to run Docker containers, which encapsulate applications and their dependencies.

Changing Infrastructure

As your needs evolve, you may need to modify the existing infrastructure. Terraform makes it easy to implement changes by updating the configuration files to reflect the newdesired state. For instance, you might want to change the storage settings of an S3 bucket, add new security policies, or modify the Docker container's configuration.

Terraform's "plan" command helps you preview the changes that will be made to your infrastructure before applying them. This step ensures that you understand the impact of your changes and can avoid unintended consequences.

Destroying Infrastructure

When certain resources are no longer needed, Terraform allows you to destroy them in a controlled manner. This might involve deleting an S3 bucket or terminating an EC2 instance running Docker containers. By running the "destroy" command, Terraform ensures that all associated resources are properly de-provisioned and removed.

Destroying infrastructure with Terraform is beneficial because it helps avoid unnecessary costs associated

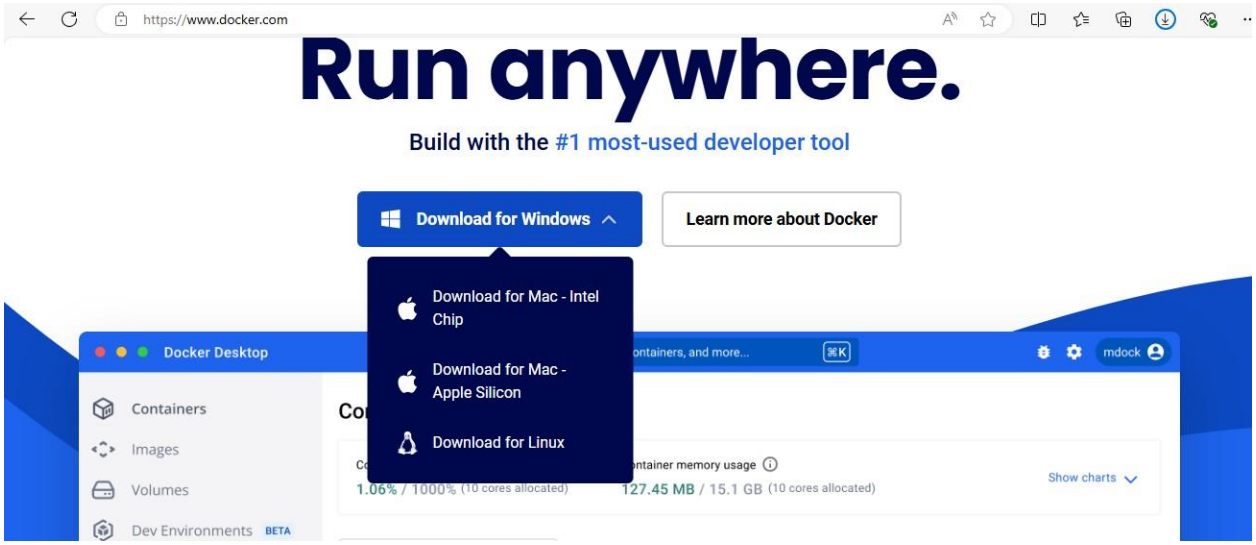
with unused resources and ensures that the environment remains clean and free of clutter.

Benefits of Using Terraform for AWS Infrastructure

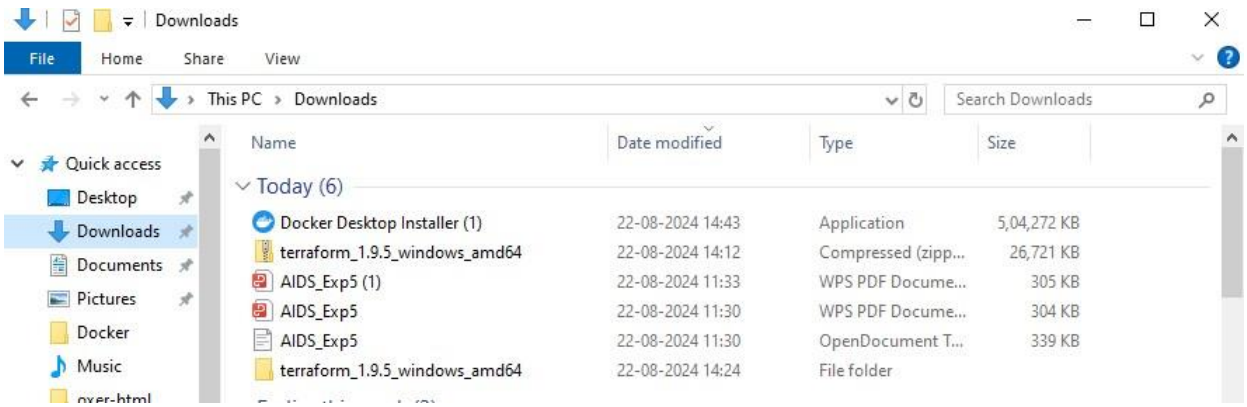
1. **Consistency:** Terraform ensures that infrastructure is consistent across environments by applying the same configuration files.
2. **Automation:** Manual processes are reduced, and infrastructure is provisioned, updated, and destroyed automatically based on code.
3. **Version Control:** Infrastructure configurations can be stored in version control systems (like Git), allowing teams to track changes, collaborate, and roll back if necessary.
4. **Scalability:** Terraform can manage complex infrastructures, scaling them up or down as needed, whether for small projects or large-scale applications.
5. **Modularity:** Terraform configurations can be broken down into reusable modules, making it easier to manage and scale infrastructure.

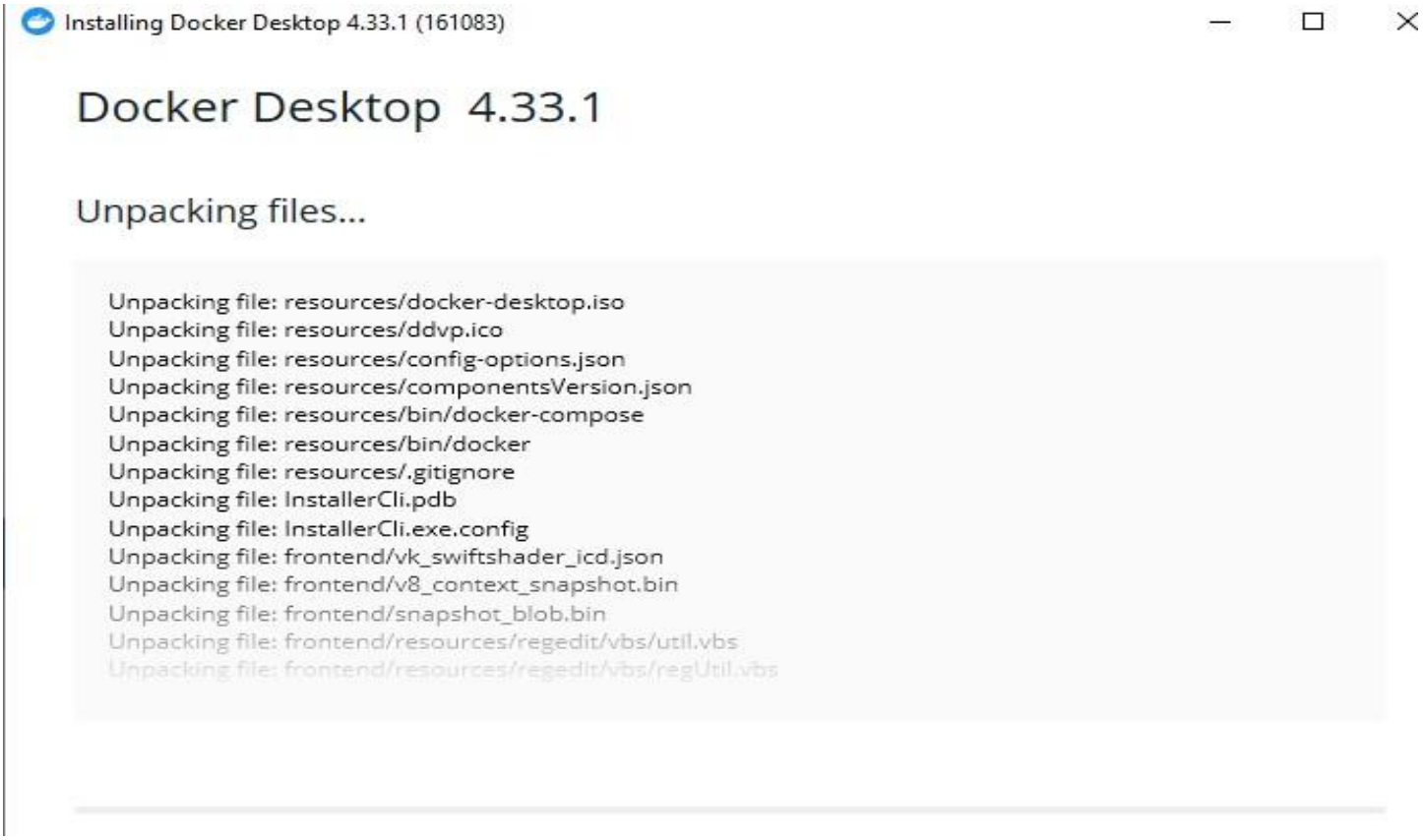
Installing and Setting Up Docker with Terraform:

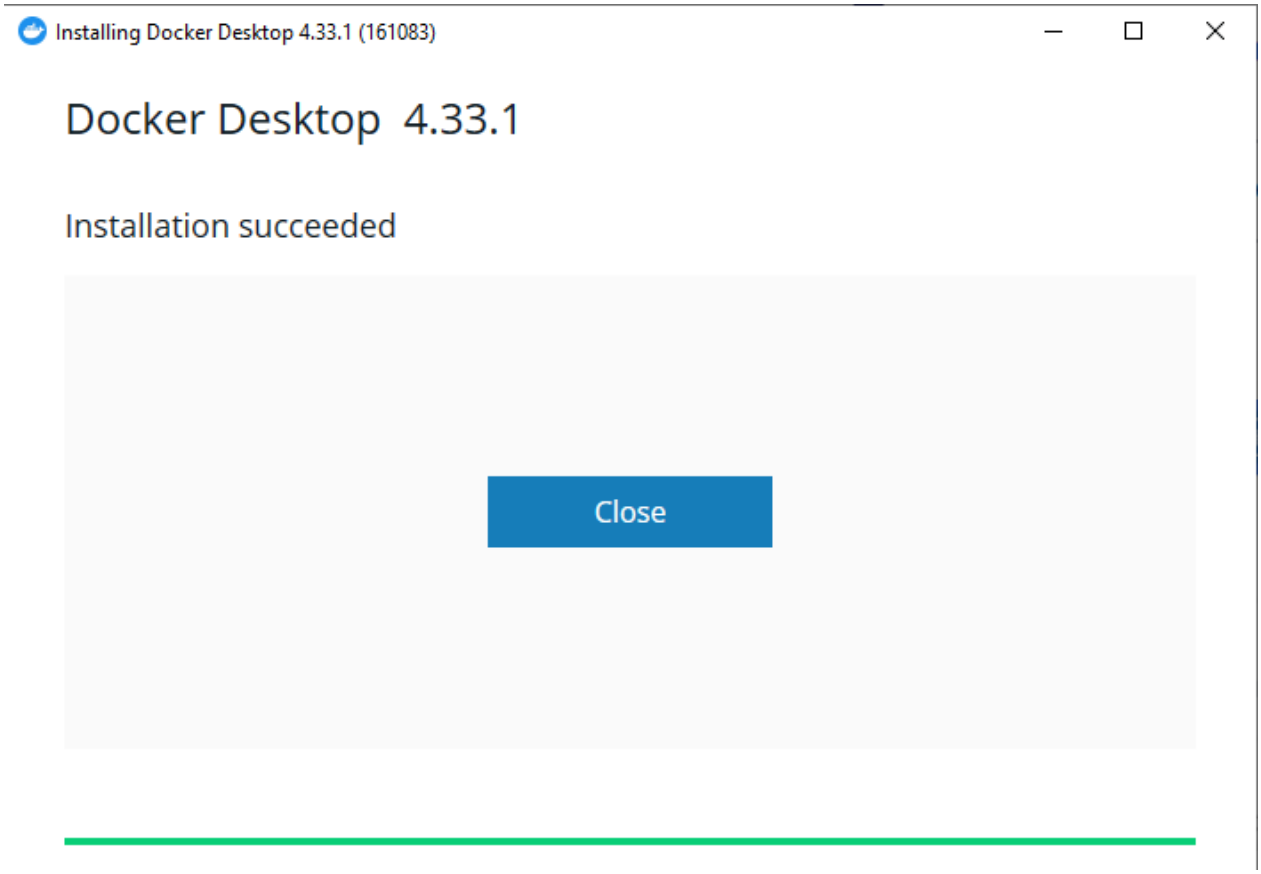
Step 1: Download Docker



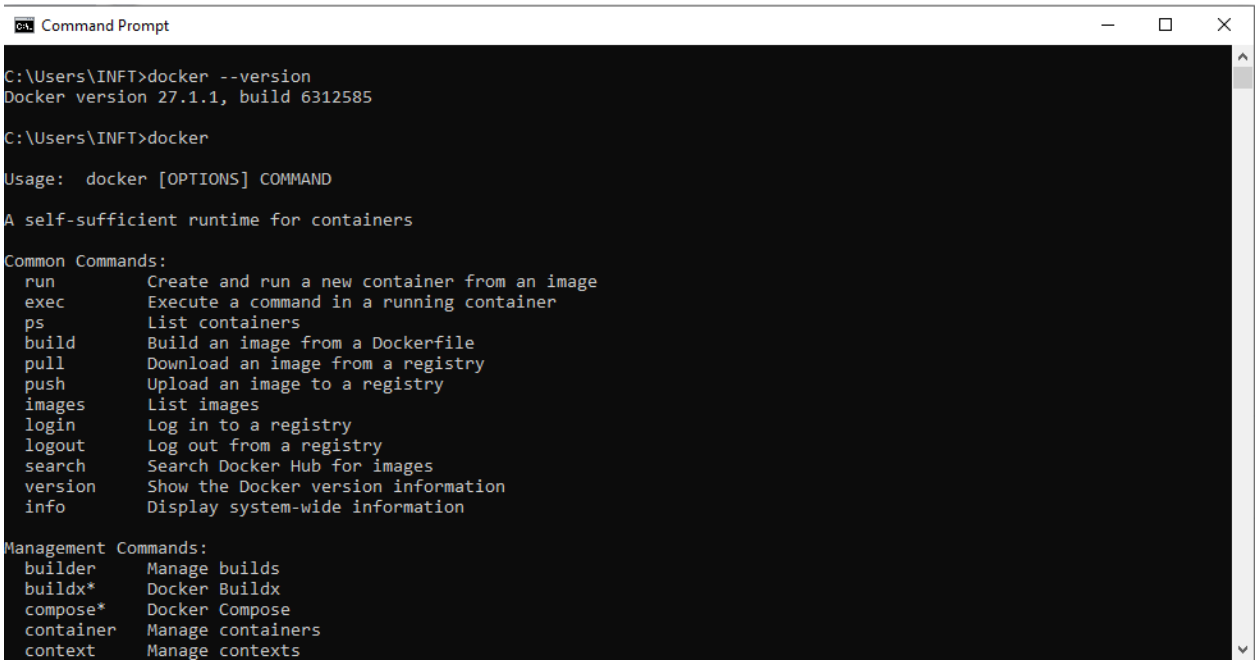
Step 2: Run the Docker installer and complete the installation process.


Downloads
File
Home
Share
View
This PC › Downloads
Search Downloads
Quick access
Desktop
Downloads
Documents
Pictures
Docker
Music
Name
Date modified
Type
Size
Today (6)
Docker Desktop Installer (1) 22-08-2024 14:43 Application 5,04,272 KB
terraform_1.9.5_windows_amd64 22-08-2024 14:12 Compressed (zipp... 26,721 KB
AIDS_Exp5 (1) 22-08-2024 11:33 WPS PDF Docume... 305 KB
AIDS_Exp5 22-08-2024 11:30 WPS PDF Docume... 304 KB
AIDS_Exp5 22-08-2024 11:30 OpenDocument T... 339 KB
terraform_1.9.5_windows_amd64 22-08-2024 14:24 File folder

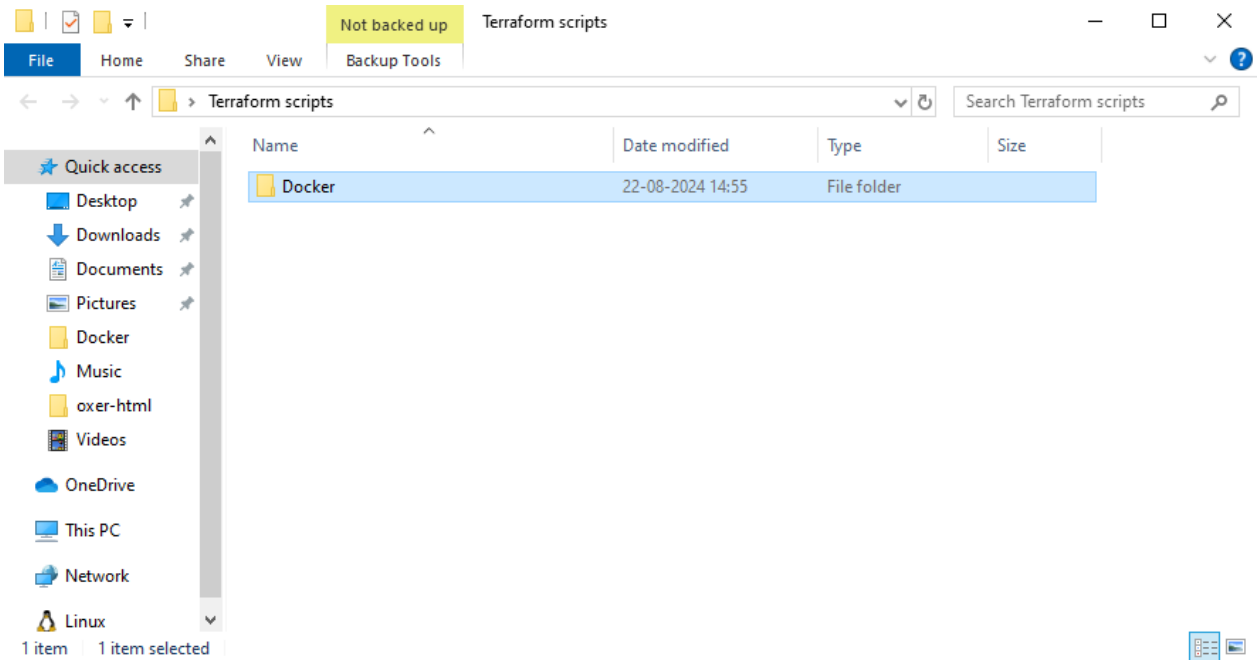

Installing Docker Desktop 4.33.1 (161083)
Docker Desktop 4.33.1
Unpacking files...
Unpacking file: resources/docker-desktop.iso
Unpacking file: resources/ddvp.ico
Unpacking file: resources/config-options.json
Unpacking file: resources/componentsVersion.json
Unpacking file: resources/bin/docker-compose
Unpacking file: resources/bin/docker
Unpacking file: resources/.gitignore
Unpacking file: InstallerCli.pdb
Unpacking file: InstallerCli.exe.config
Unpacking file: frontend/vk_swiftshader_icd.json
Unpacking file: frontend/v8_context_snapshot.bin
Unpacking file: frontend/snapshot_blob.bin
Unpacking file: frontend/resources/regedit/vbs/util.vbs
Unpacking file: frontend/resources/regedit/vbs/regUtil.vbs



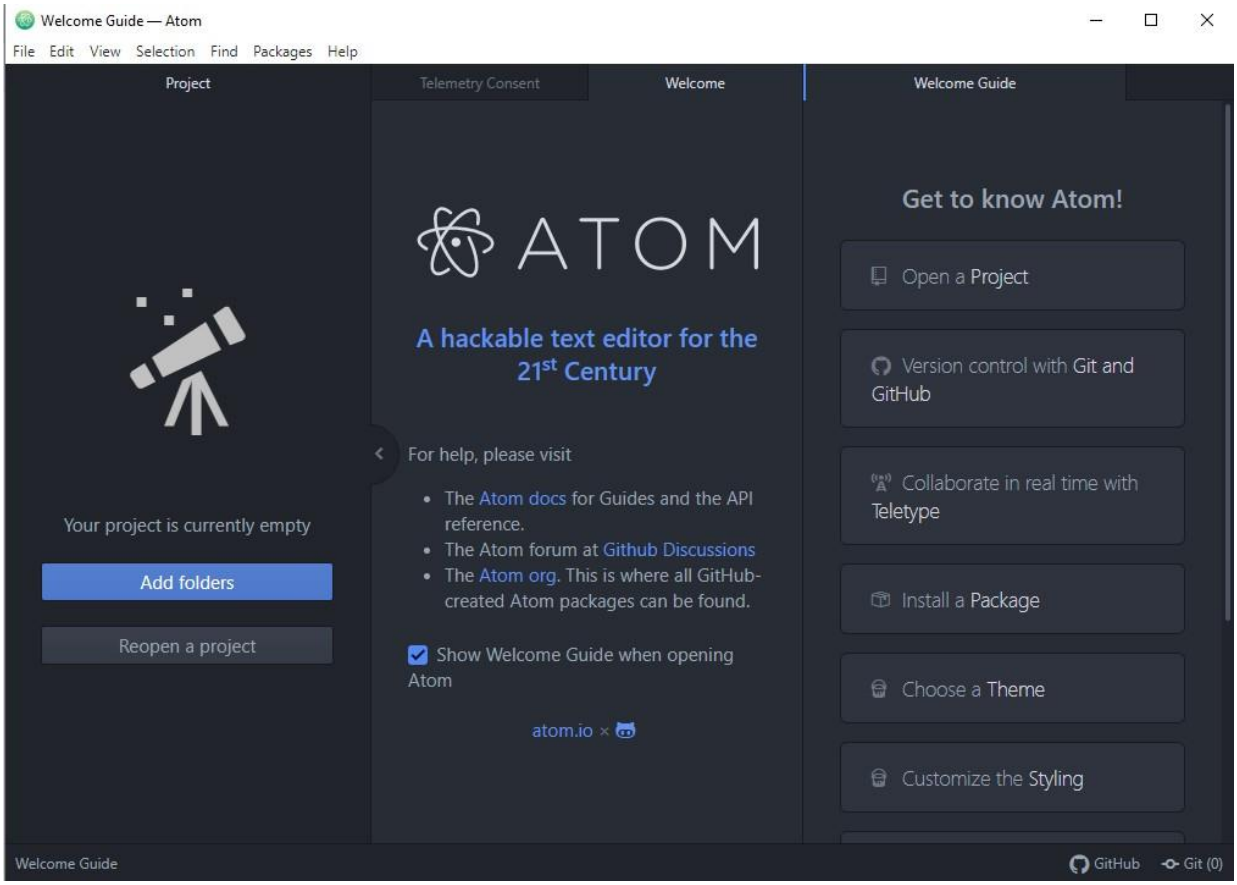
Step 3: Open Command Prompt as an administrator and enter `docker --version` to verify installation.



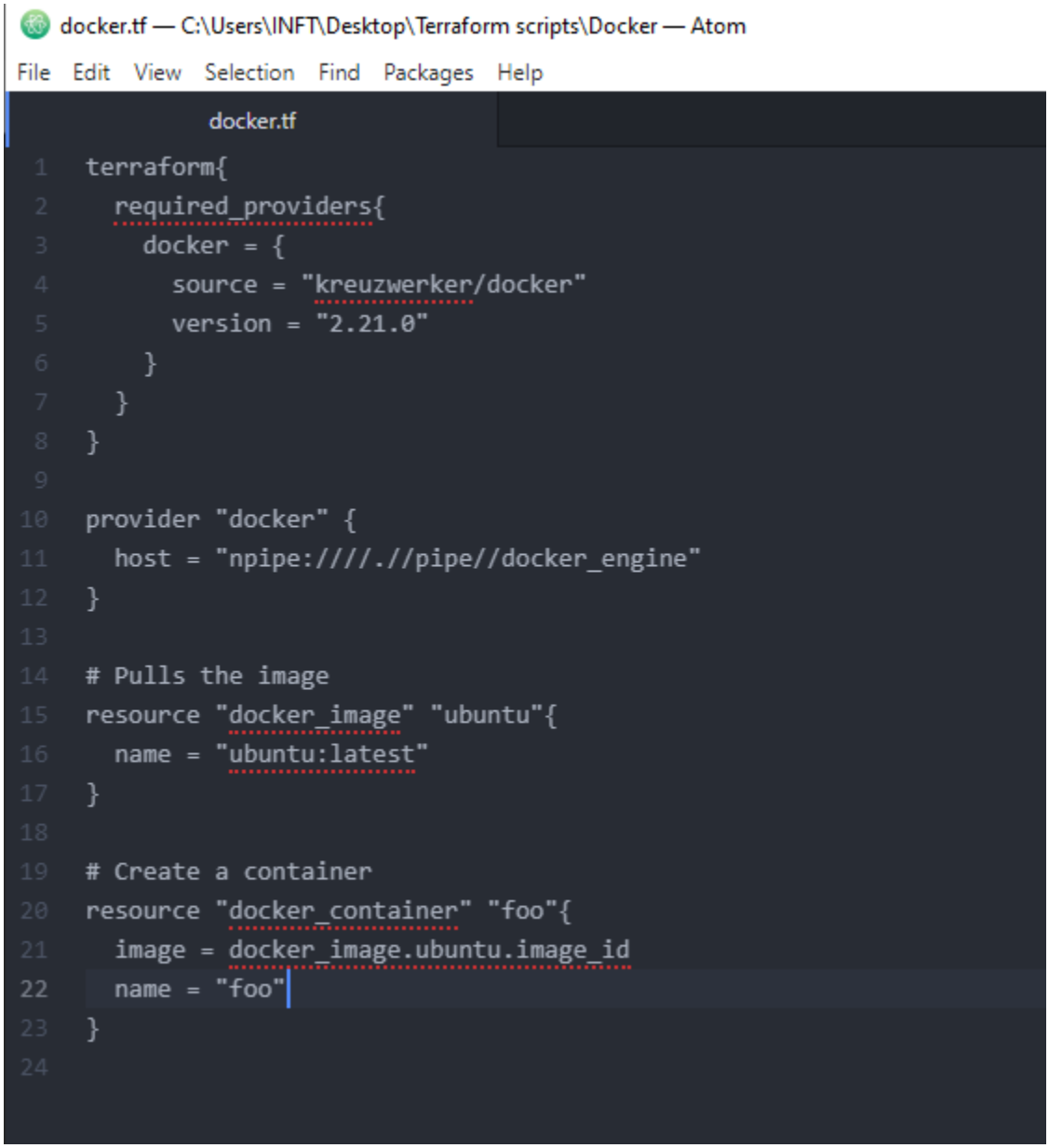
Step 4: Create a directory called Terraform_Scripts and within it, a subdirectory named Docker



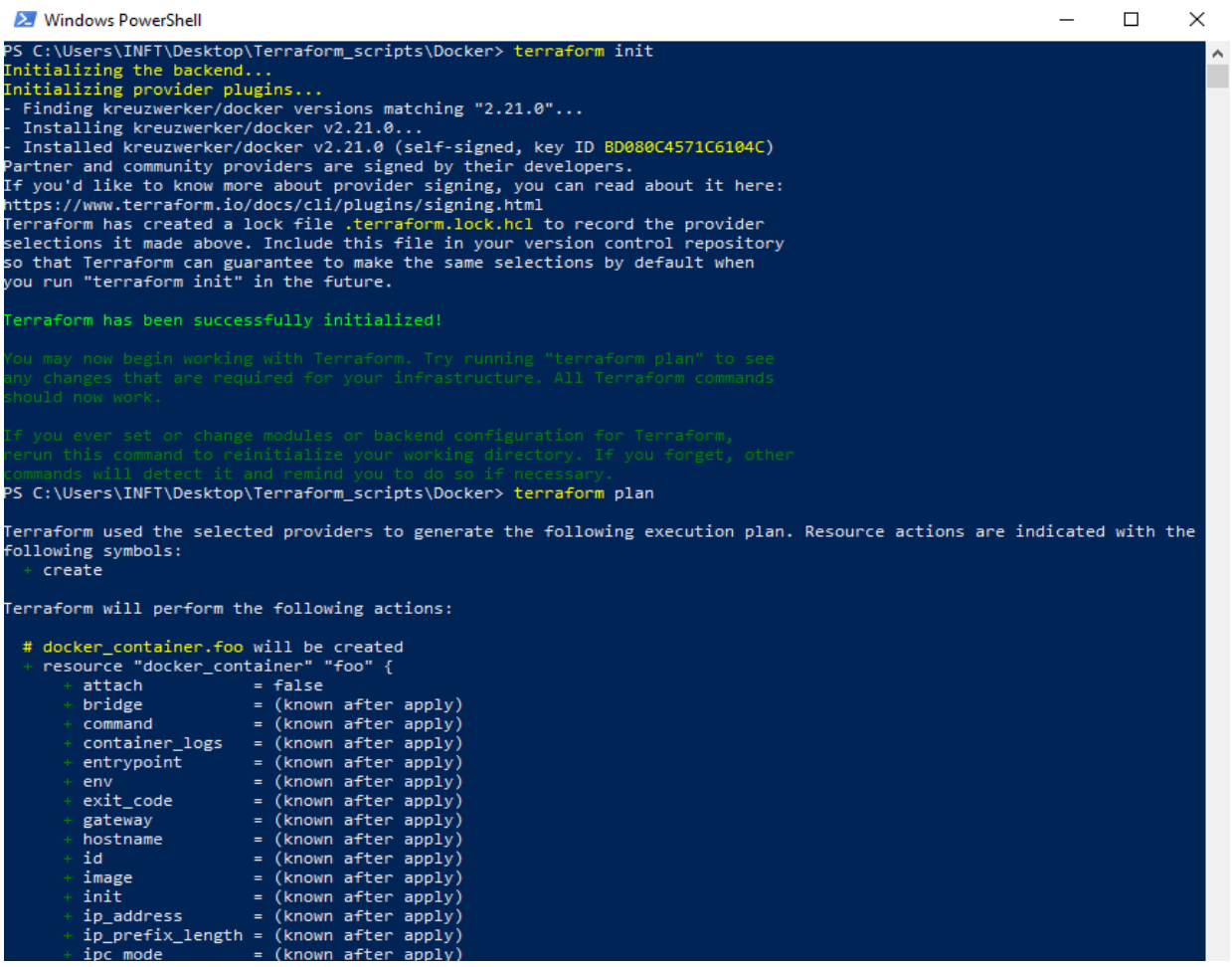
Step 5: Download and install the Atom Editor from Atom's official site.

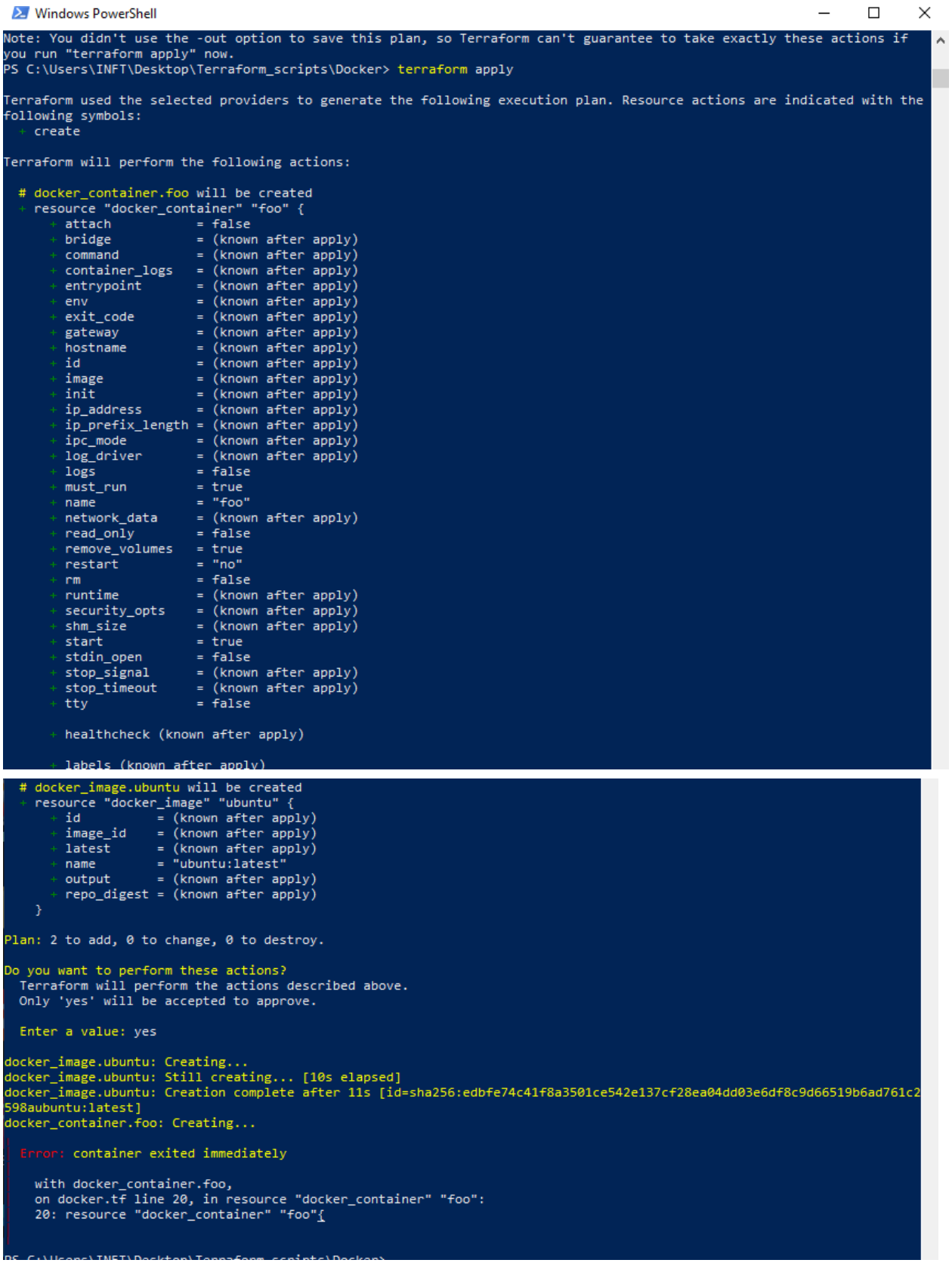


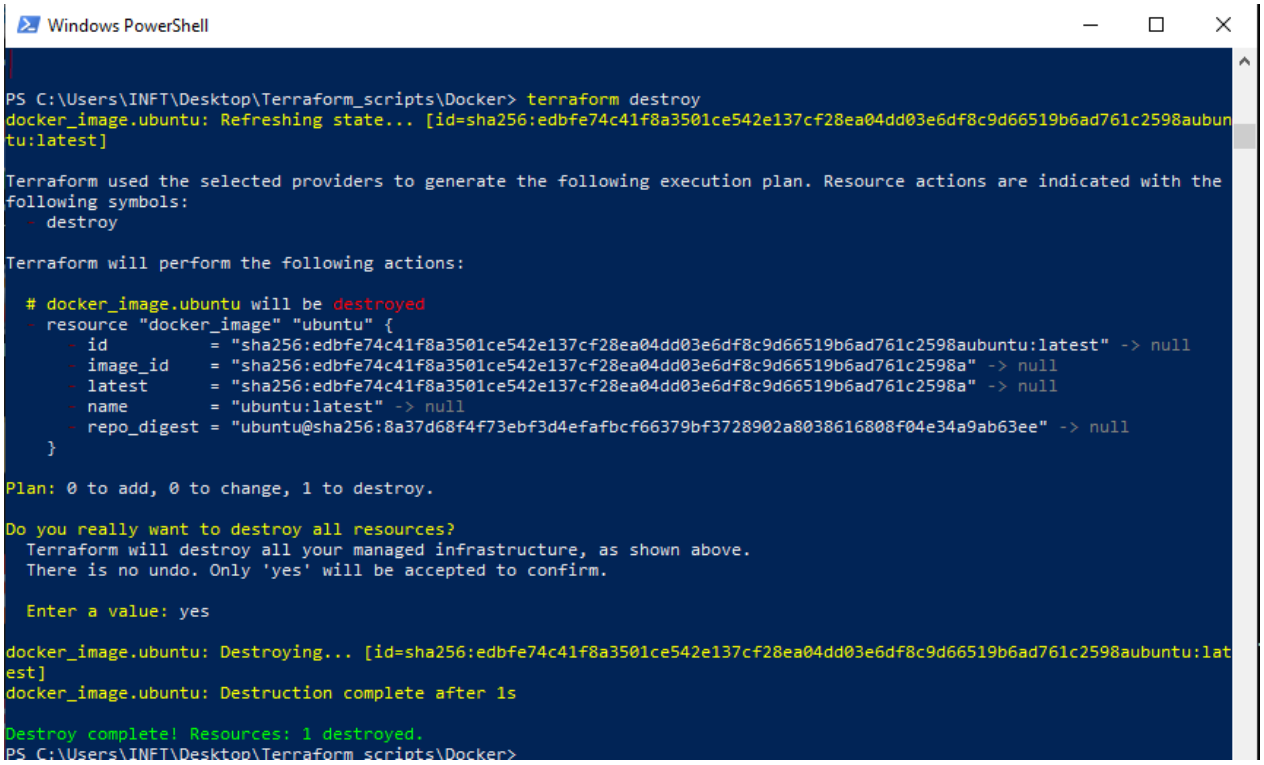
Step 6: Open Atom Editor, create a new document, and input or paste your script.

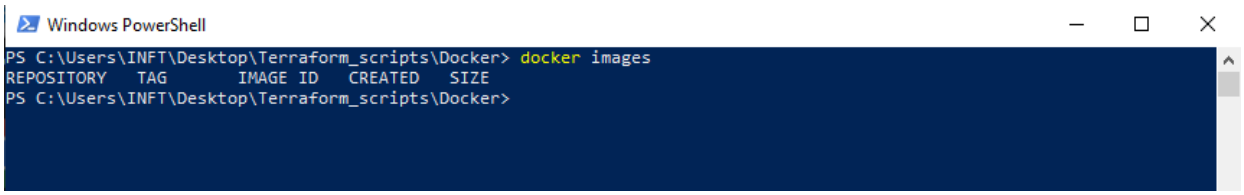


Step 7: In Command Prompt, go to the Terraform_Scripts directory and execute terraform init, terraform plan, terraform apply, terraform destroy, and docker images.




Windows PowerShell
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
PS C:\Users\INFT\Desktop\Terraform_scripts\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 = (known after apply)
+ container_logs = (known after apply)
+ entrypoint = (known after apply)
+ env = (known after apply)
+ exit_code = (known after apply)
+ gateway = (known after apply)
+ hostname = (known after apply)
+ id = (known after apply)
+ image = (known after apply)
+ init = (known after apply)
+ ip_address = (known after apply)
+ ip_prefix_length = (known after apply)
+ ipc_mode = (known after apply)
+ log_driver = (known after apply)
+ logs = false
+ must_run = true
+ name = "foo"
+ network_data = (known after apply)
+ read_only = false
+ remove_volumes = true
+ restart = "no"
+ rm = false
+ runtime = (known after apply)
+ security_opts = (known after apply)
+ shm_size = (known after apply)
+ start = true
+ stdin_open = false
+ stop_signal = (known after apply)
+ stop_timeout = (known after apply)
+ tty = false
+ healthcheck (known after apply)
+ labels (known after apply)
# docker_image.ubuntu will be created
+ resource "docker_image" "ubuntu" {
+ id = (known after apply)
+ image_id = (known after apply)
+ latest = (known after apply)
+ name = "ubuntu:latest"
+ output = (known after apply)
+ repo_digest = (known after apply)
}
Plan: 2 to add, 0 to change, 0 to destroy.
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.
Enter a value: yes
docker_image.ubuntu: Creating...
docker_image.ubuntu: Still creating... [10s elapsed]
docker_image.ubuntu: Creation complete after 11s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Creating...
Error: container exited immediately
with docker_container.foo,
on docker.tf line 20, in resource "docker_container" "foo":
20: resource "docker_container" "foo"{


Windows PowerShell
PS C:\Users\INFT\Desktop\Terraform_scripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
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_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, 1 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_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 1s
Destroy complete! Resources: 1 destroyed.
PS C:\Users\INFT\Desktop\Terraform_scripts\Docker>


Windows PowerShell
PS C:\Users\INFT\Desktop\Terraform_scripts\Docker> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
PS C:\Users\INFT\Desktop\Terraform_scripts\Docker>

Snehal A. Patil D15A 39

EXPERIMENT NO:7

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

Theory:

Static Application Security Testing (SAST) is a method of debugging by examining source code before a program is run. It involves analyzing the application's source code, bytecode, or binary code to identify vulnerabilities and security flaws. SAST tools scan code for common security vulnerabilities such as SQL injection, cross-site scripting (XSS), and buffer overflows, among others.

Problems SAST Solves:

1. **Early Detection of Vulnerabilities**: SAST enables developers to find security flaws early in the development lifecycle, reducing the cost and effort required to fix them later.
2. **Compliance with Security Standards**: It helps organizations comply with various security regulations and standards, such as PCI DSS, OWASP Top Ten, and ISO 27001, by identifying security weaknesses that need to be addressed.
3. **Integration into CI/CD Pipelines**: SAST tools can be integrated into Continuous Integration/Continuous Deployment (CI/CD) pipelines, allowing for automated security checks during the development process.
4. **Comprehensive Coverage**: It scans all code paths and identifies vulnerabilities that may not be detected during dynamic testing (which tests the application while it runs).
5. **Reduction of Technical Debt**: By catching vulnerabilities early, SAST helps prevent the accumulation of technical debt related to security issues, making the codebase more maintainable.
6. **Improved Code Quality**: Besides security, SAST tools often identify coding best practices and help improve overall code quality.
7. **Enhanced Collaboration**: By providing clear reports and insights, SAST tools foster better communication between development and security teams.
8. **Risk Mitigation**: It helps organizations manage risks associated with software vulnerabilities, thereby protecting against data breaches and cyberattacks.

Prerequisites:

- Jenkins installed
- Docker Installed (for SonarQube)

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 -

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

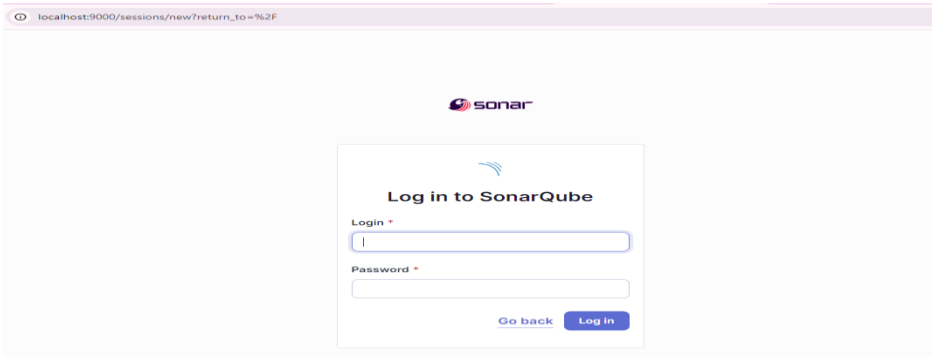
```
PS C:\Users\Windows> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
7df3e28058c6bfc74d745f9f18f0923c82c1fc4058967a5b33907e0010b01ee2
PS C:\Users\Windows>
```

```
PS C:\Users\Windows> docker ps
>>
CONTAINER ID   IMAGE              COMMAND                  CREATED         STATUS         PORTS                    NAMES
7df3e28058c6   sonarqube:latest   "/opt/sonarqube/dock…"   5 minutes ago   Up 5 minutes   0.0.0.0:9000->9000/tcp   sonarqube
PS C:\Users\Windows>
```

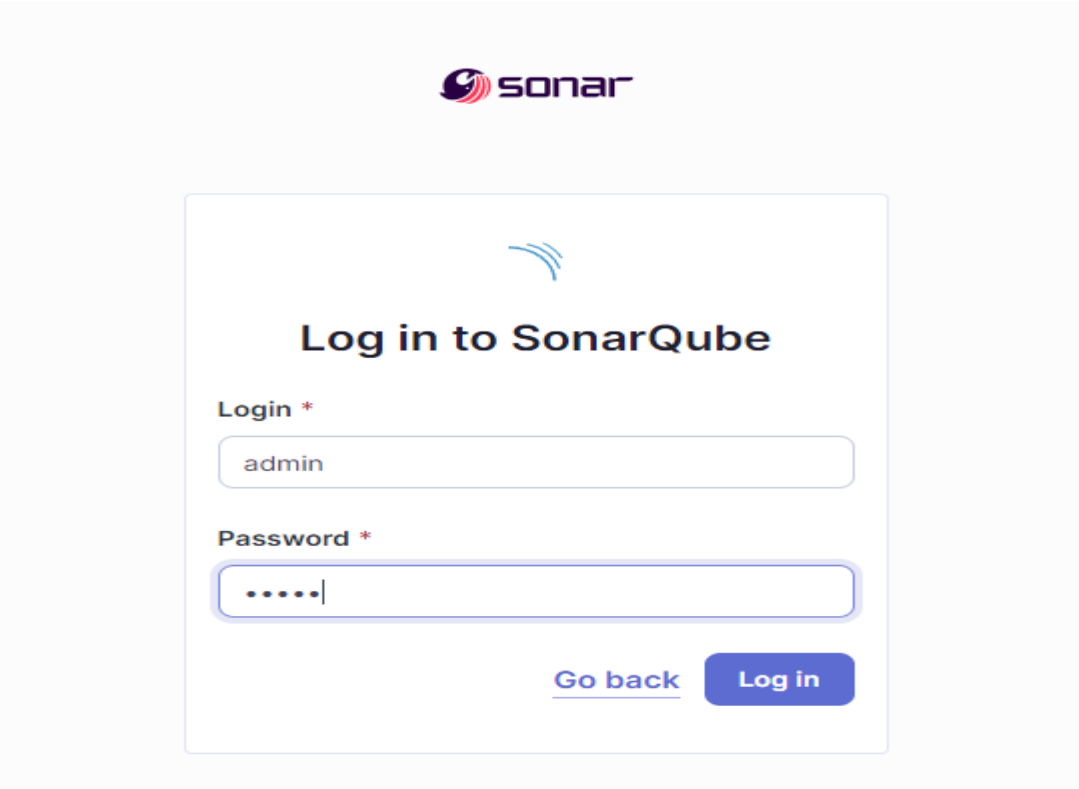
```
# User credentials.
# Permissions to create tables, indices and triggers must be granted to JDBC user.
# The schema must be created first.
sonar.jdbc.username=snehalsonar
sonar.jdbc.password=snehalsonar
```

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

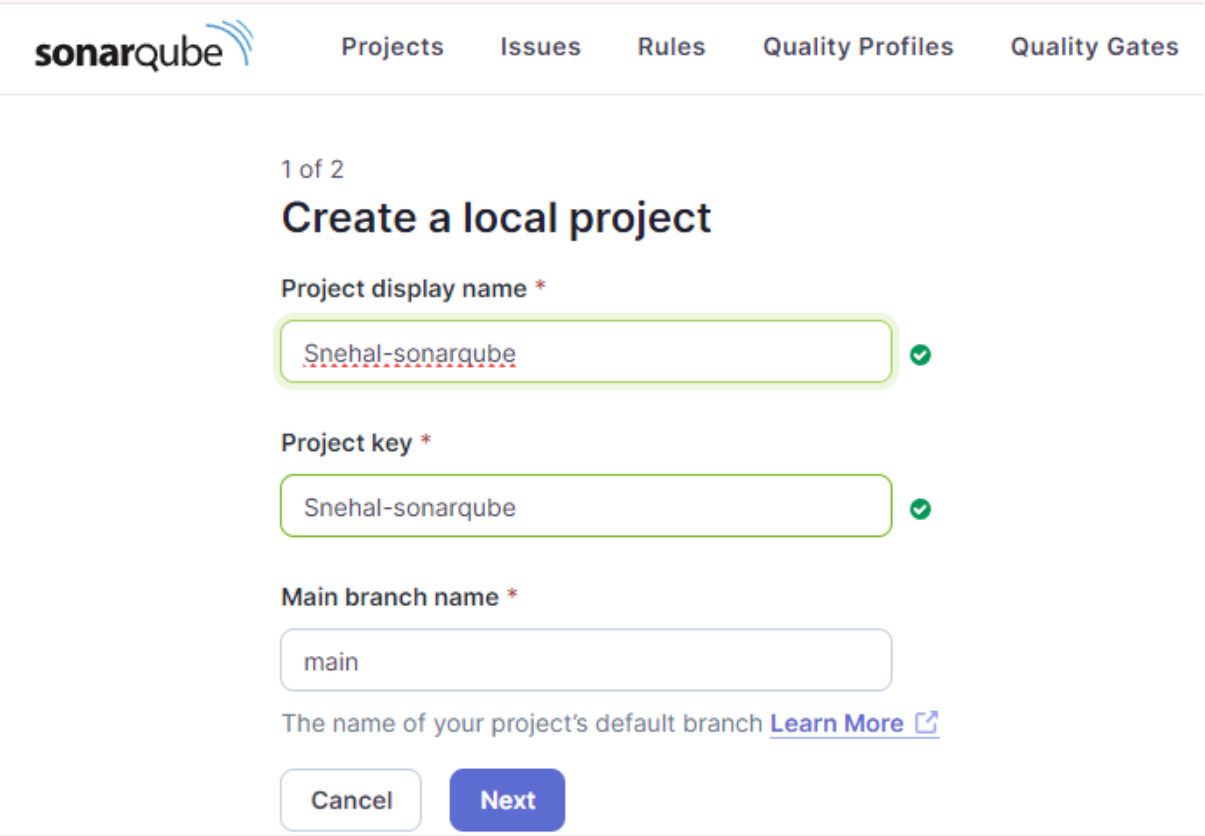
localhost port 9000.

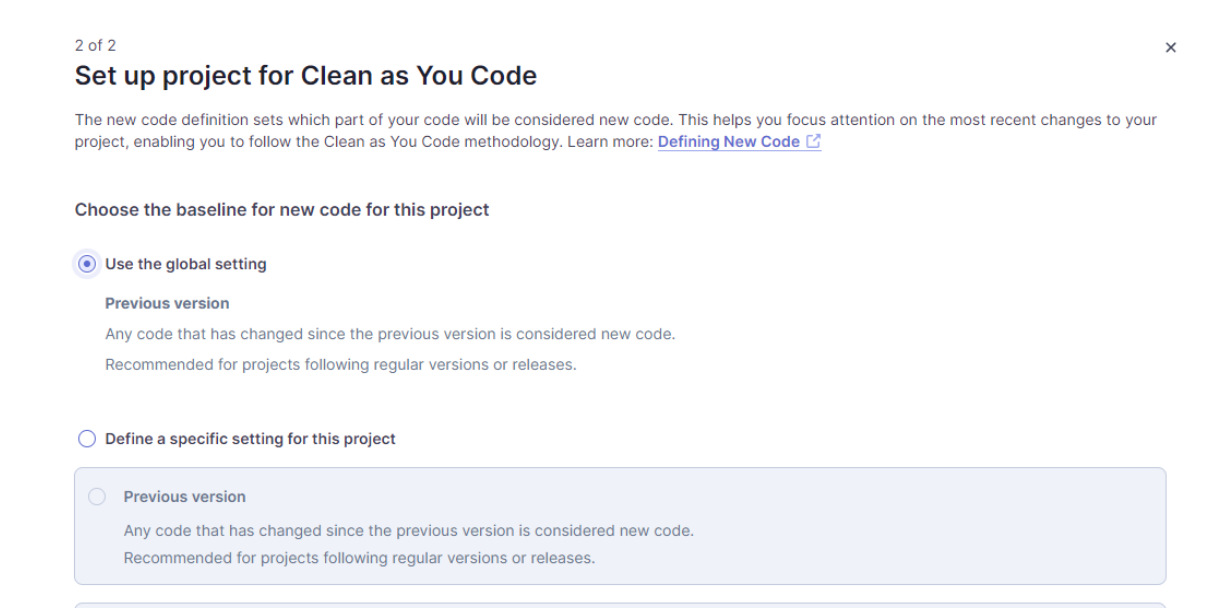


4. Login to SonarQube using username admin and password admin.

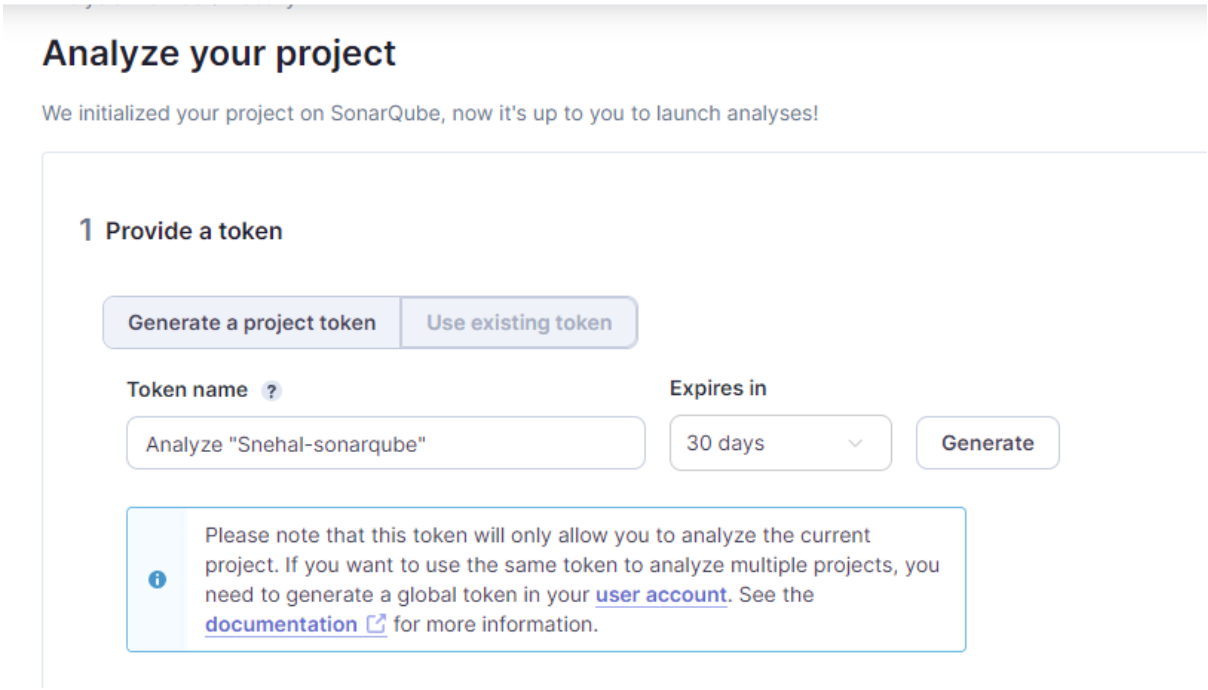


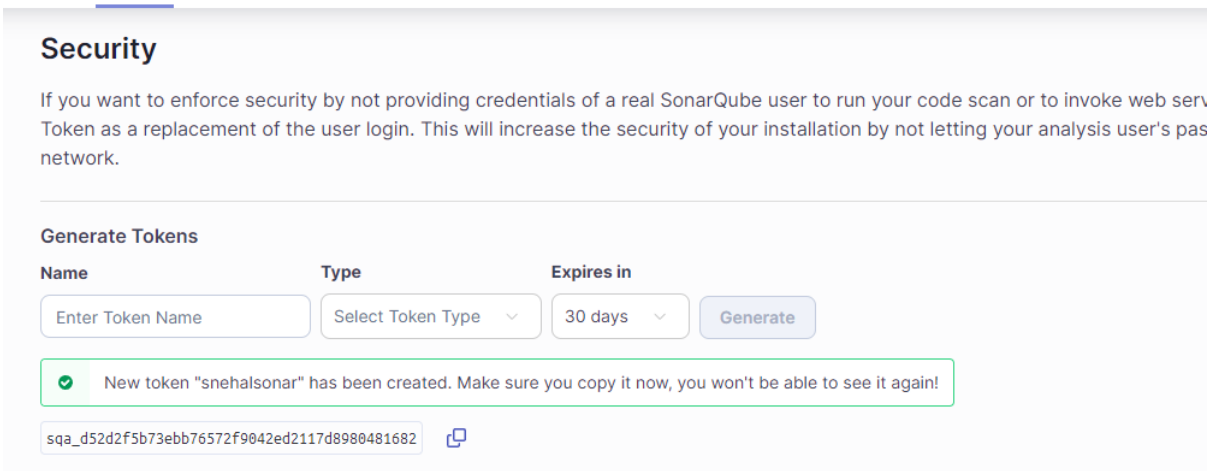
5.Create a manual project in SonarQube with the name sonarqube





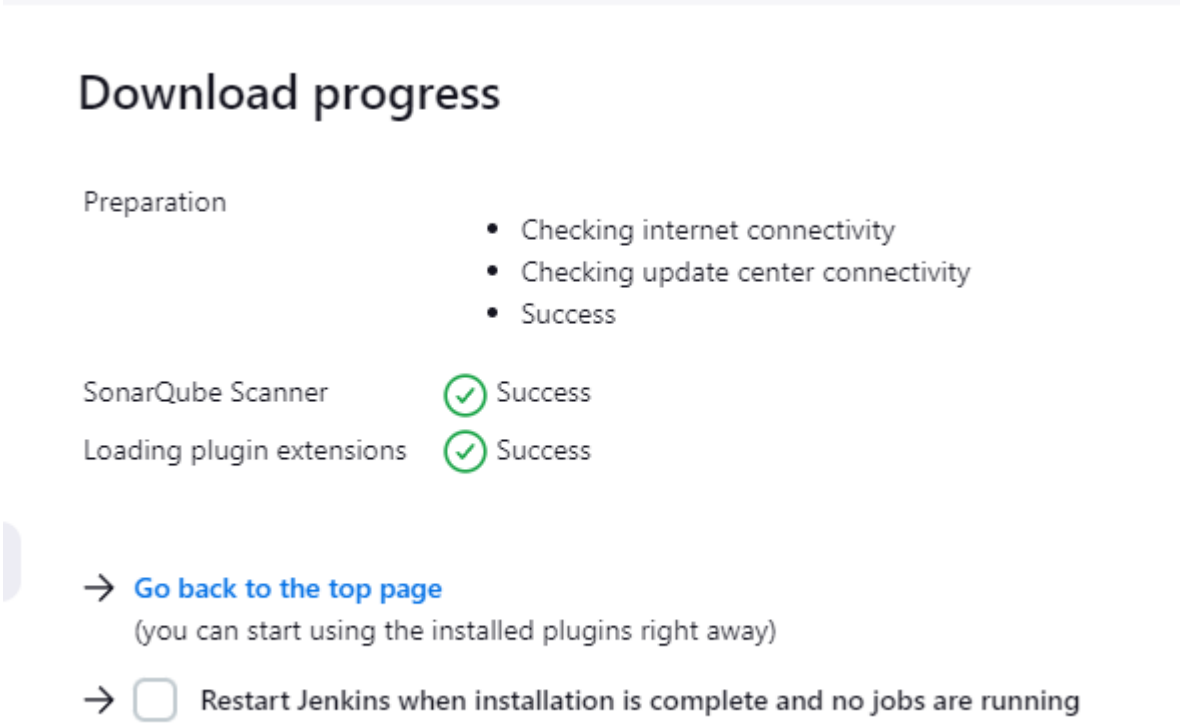
6. Generate a token





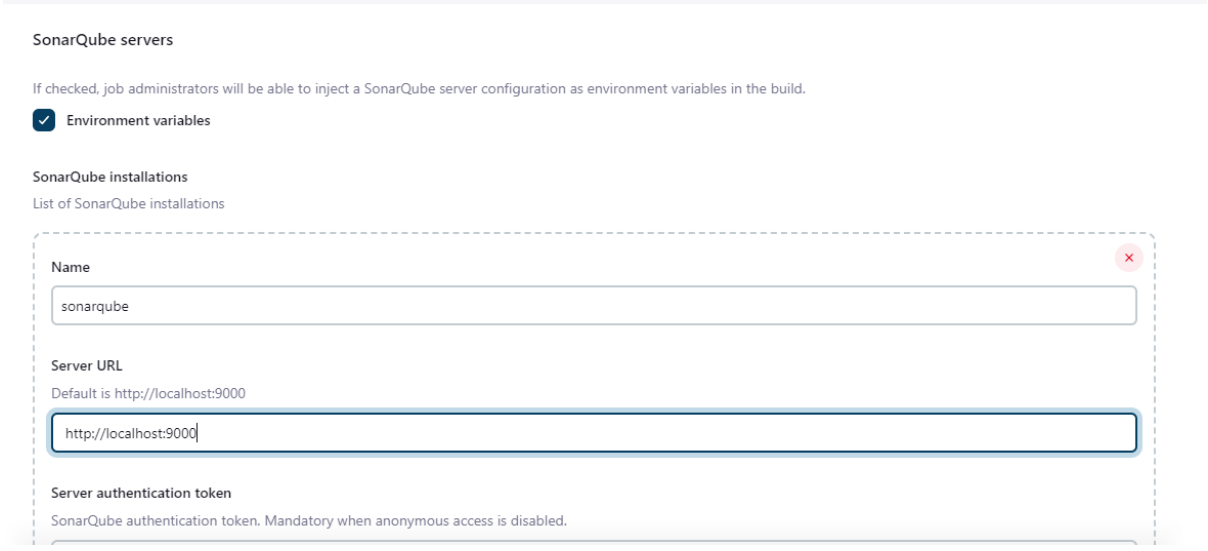
7. Setup the project and come back to Jenkins Dashboard.

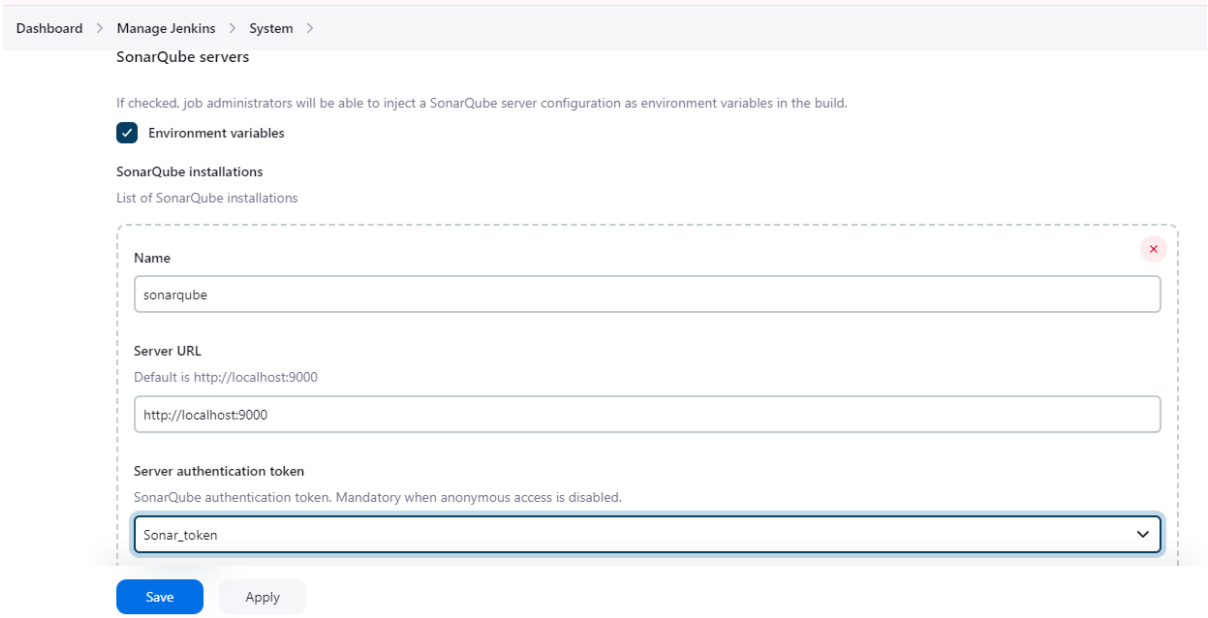
Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.



8. Under Jenkins ‘Configure System’, look for SonarQube Servers and enter the details.

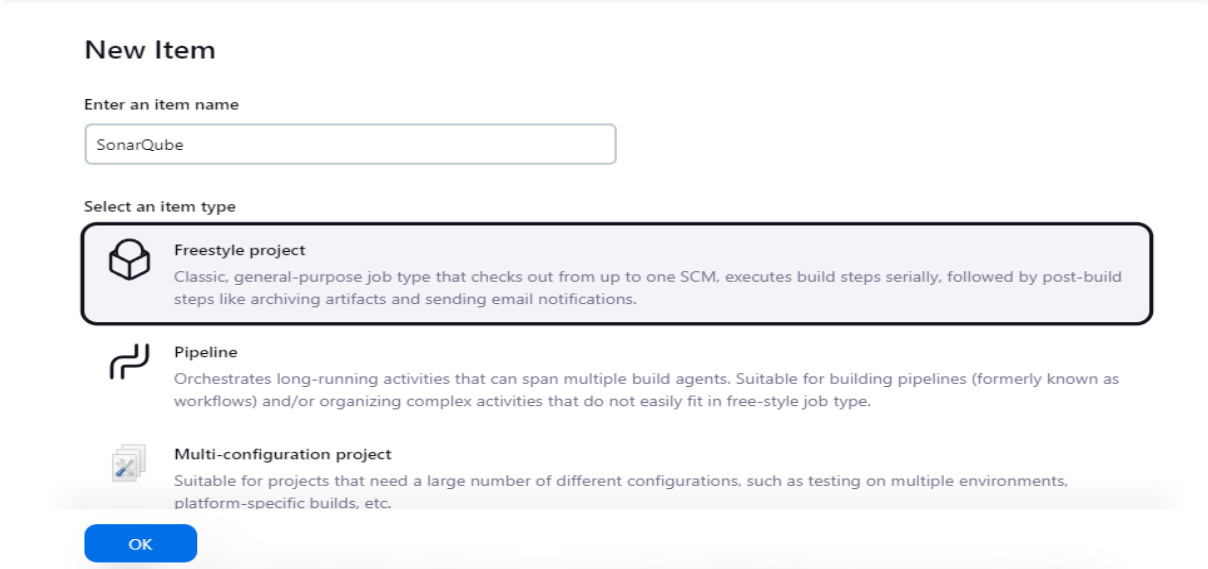
Enter the Server Authentication token if needed.





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

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

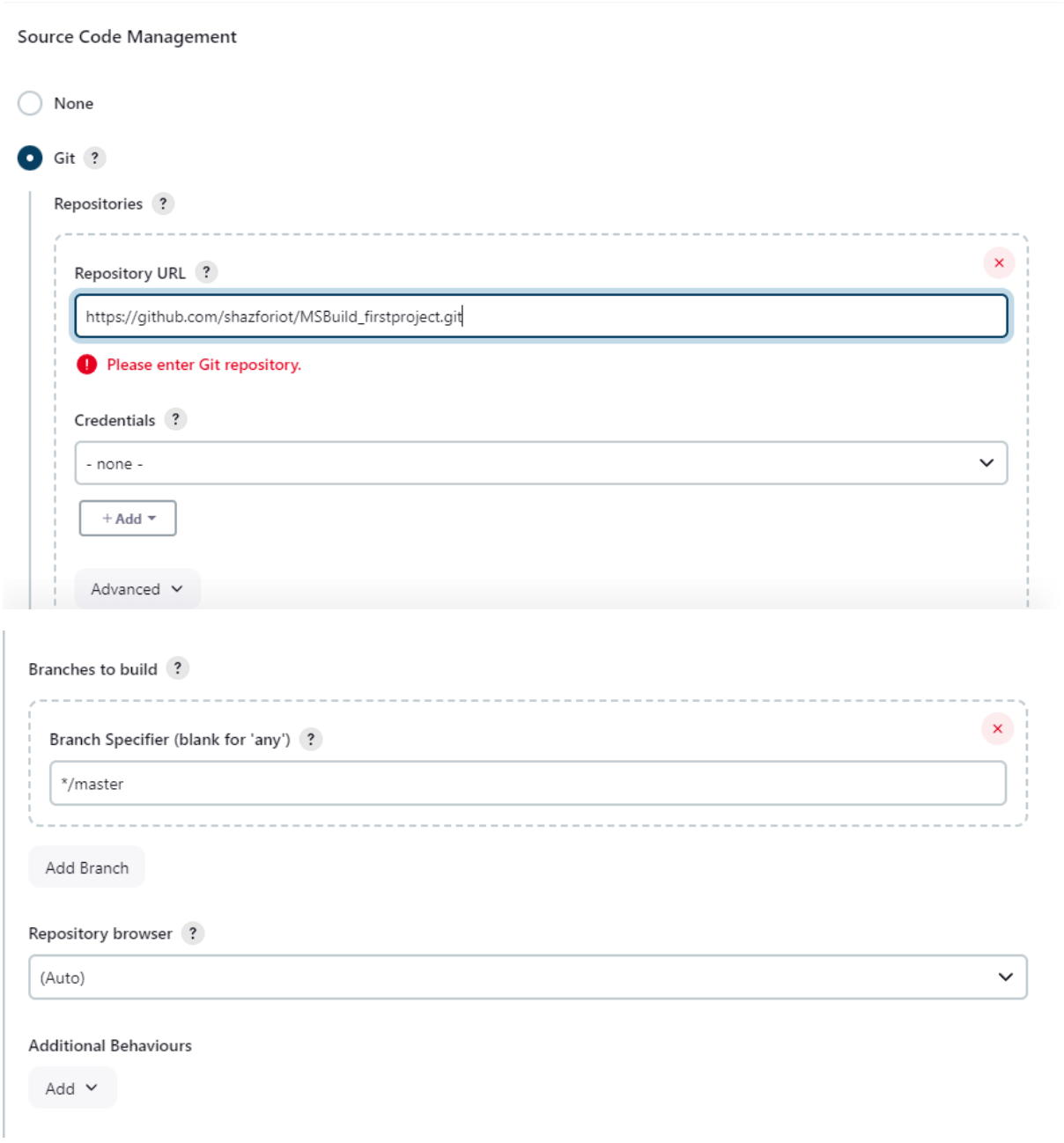


11.Choose this GitHub repository in Source Code

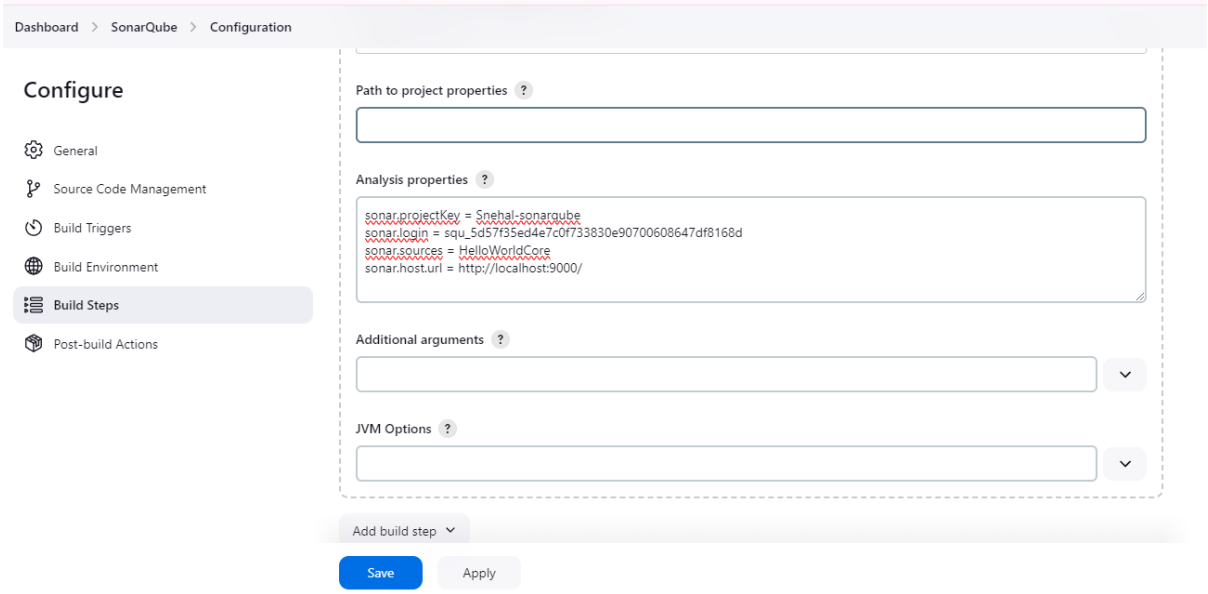
Management.

https://github.com/shazforiot/MSBuild_firstproject.git

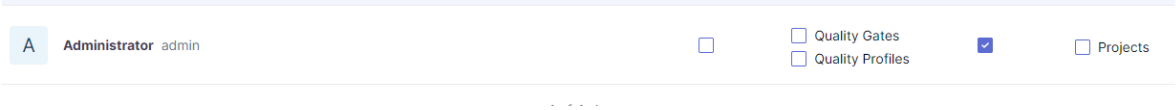
It is a sample hello-world project with no vulnerabilities and issues, just to test



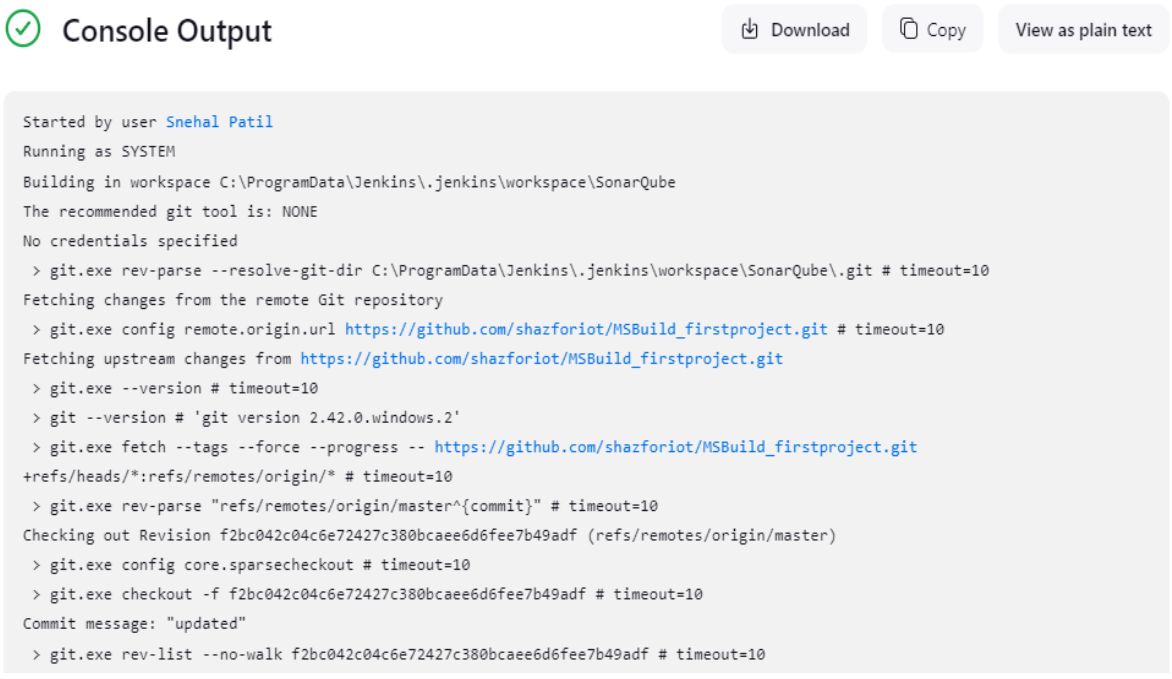
12. Under Build-> Execute SonarQube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.



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



Check the console output.

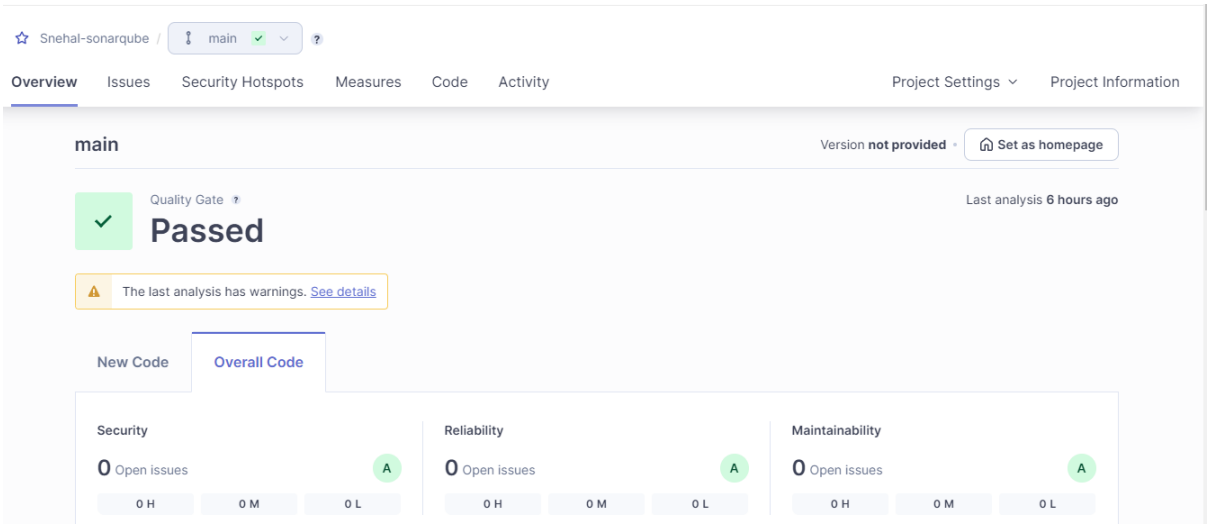


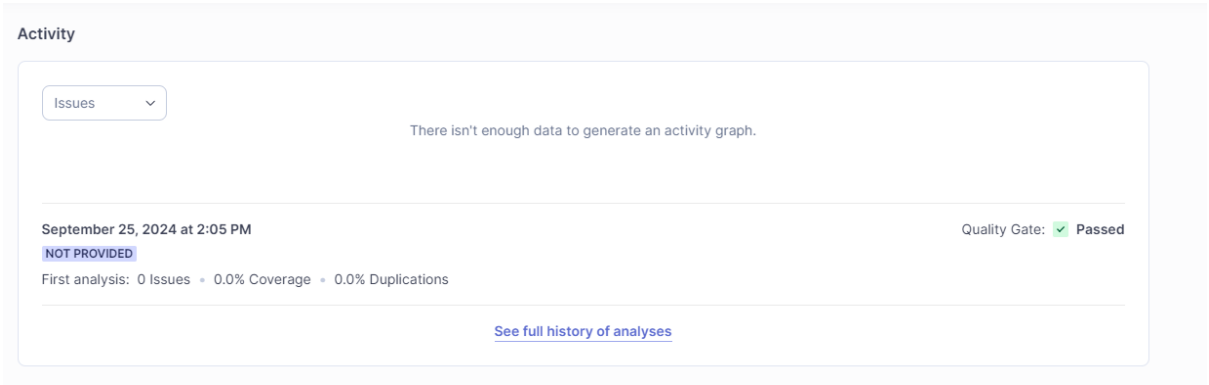
```
14:04:53.398 INFO  Project root configuration file: NONE
14:04:53.455 INFO  SonarScanner CLI 6.2.0.4584
14:04:53.459 INFO  Java 21.0.4 Eclipse Adoptium (64-bit)
14:04:53.471 INFO  Windows 10 10.0 amd64
14:04:53.542 INFO  User cache: C:\WINDOWS\system32\config\systemprofile\.sonar\cache
14:04:55.391 INFO  JRE provisioning: os[windows], arch[amd64]
14:04:56.116 INFO  Communicating with SonarQube Server 10.6.0.92116
14:04:57.650 INFO  Starting SonarScanner Engine...
14:04:57.651 INFO  Java 17.0.11 Eclipse Adoptium (64-bit)
14:05:01.587 INFO  Load global settings
14:05:01.991 INFO  Load global settings (done) | time=402ms
14:05:02.005 INFO  Server id: 147B411E-AZIoO70pNro_dTnC3uoH
14:05:02.035 INFO  Loading required plugins
14:05:02.036 INFO  Load plugins index
14:05:02.166 INFO  Load plugins index (done) | time=127ms
14:05:02.169 INFO  Load/download plugins
14:05:07.332 INFO  Load/download plugins (done) | time=5168ms
14:05:08.489 INFO  Process project properties
14:05:08.518 INFO  Process project properties (done) | time=29ms
14:05:08.560 INFO  Project key: Snehal-sonarqube
14:05:08.562 INFO  Base dir: C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube
14:05:08.565 INFO  Working dir: C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube\.scannerwork
14:05:08.601 INFO  Load project settings for component key: 'Snehal-sonarqube'
14:05:08.694 INFO  Load project settings for component key: 'Snehal-sonarqube' (done) | time=91ms
14:05:08.906 INFO  Load quality profiles
14:05:09.907 INFO  Load quality profiles (done) | time=1001ms
```

```
14:05:39.719 INFO  Load analysis cache (404) | time=45ms
14:05:39.875 WARN  The property 'sonar.login' is deprecated and will be removed in the future. Please use the 'sonar.token'
property instead when passing a token.
14:05:39.954 INFO  Preprocessing files...
14:05:43.668 INFO  2 languages detected in 23 preprocessed files
14:05:43.671 INFO  0 files ignored because of scm ignore settings
14:05:43.678 INFO  Loading plugins for detected languages
14:05:43.679 INFO  Load/download plugins
14:05:44.555 INFO  Load/download plugins (done) | time=876ms
14:05:44.835 INFO  Executing phase 2 project builders
14:05:44.838 INFO  Executing phase 2 project builders (done) | time=3ms
14:05:44.870 INFO  Load project repositories
14:05:44.941 INFO  Load project repositories (done) | time=72ms
14:05:45.005 INFO  Indexing files...
14:05:45.007 INFO  Project configuration:
14:05:45.083 INFO  23 files indexed
14:05:45.087 INFO  Quality profile for cs: Sonar way
14:05:45.088 INFO  Quality profile for json: Sonar way
14:05:45.091 INFO  ------------ Run sensors on module Snehal-sonarqube
14:05:45.236 INFO  Load metrics repository
14:05:45.321 INFO  Load metrics repository (done) | time=84ms
14:05:47.290 INFO  Sensor C# Project Type Information [csharp]
14:05:47.294 INFO  Sensor C# Project Type Information [csharp] (done) | time=5ms
14:05:47.296 INFO  Sensor C# Analysis Log [csharp]
14:05:47.343 INFO  Sensor C# Analysis Log [csharp] (done) | time=48ms
14:05:47.344 INFO  Sensor C# Properties [csharp]
```

```
14:05:51.146 WARN  Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider
setting 'sonar.projectBaseDir' property.
14:05:51.147 INFO  Sensor C# File Caching Sensor [csharp] (done) | time=1ms
14:05:51.147 INFO  Sensor Zero Coverage Sensor
14:05:51.162 INFO  Sensor Zero Coverage Sensor (done) | time=18ms
14:05:51.170 INFO  SCM Publisher SCM provider for this project is: git
14:05:51.174 INFO  SCM Publisher 2 source files to be analyzed
14:05:52.968 INFO  SCM Publisher 2/2 source files have been analyzed (done) | time=1792ms
14:05:52.973 INFO  CPD Executor Calculating CPD for 0 files
14:05:52.984 INFO  CPD Executor CPD calculation finished (done) | time=0ms
14:05:52.998 INFO  SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
14:05:53.651 INFO  Analysis report generated in 374ms, dir size=199.1 kB
14:05:53.845 INFO  Analysis report compressed in 123ms, zip size=20.5 kB
14:05:54.150 INFO  Analysis report uploaded in 299ms
14:05:54.156 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=Snehal-sonarqube
14:05:54.160 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted
analysis report
14:05:54.162 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=1b0e8114-e5a0-4256-ba7f-
e9eb666d5810
14:05:54.202 INFO  Analysis total time: 46.510 s
14:05:54.211 INFO  SonarScanner Engine completed successfully
14:05:54.357 INFO  EXECUTION SUCCESS
14:05:54.360 INFO  Total time: 1:00.973s
Finished: SUCCESS
```

14. Once the build is complete, check the project in SonarQube.





Snehal Patil D15A 39

Advanced DevOps Expt No:08

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.

Theory:

Static Application Security Testing (SAST) is a methodology that analyzes source code for security vulnerabilities before the code is compiled, often referred to as white box testing.

Problems SAST Solves:

- **Early Detection**: Identifies vulnerabilities in the early stages of the Software Development Life Cycle (SDLC), allowing developers to fix issues without breaking builds or passing vulnerabilities to production.
- **Real-Time Feedback**: Provides immediate insights while coding, which helps in addressing issues proactively.
- **Visual Guidance**: Offers graphical representations of vulnerabilities, indicating their locations and providing detailed guidance on remediation.

Importance of SAST:

- **Efficiency**: Can analyze 100% of the codebase quickly, scanning millions of lines in minutes, unlike manual reviews that are time-consuming.
- **Scalability**: Addresses the challenge of limited security staff by automating vulnerability detection, identifying critical issues like SQL injection and cross-site scripting with high accuracy.

What is SonarQube?

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications.

Benefits of SonarQube

- Sustainability - Reduces complexity, possible vulnerabilities, and code duplications, optimising the life of applications.
- Increase productivity - Reduces the scale, cost of maintenance, and risk of the application; as such, it removes the need to spend more time changing the code
- Quality code - Code quality control is an inseparable part of the process of software development.
- Detect Errors - Detects errors in the code and alerts developers to fix them automatically before submitting them for output.

Integrating Jenkins with SonarQube:

Prerequisites:

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

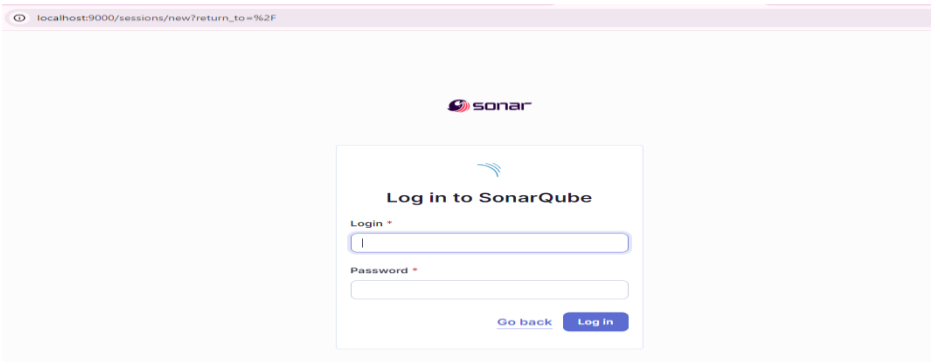
Steps to create a Jenkins CI/CD Pipeline and use SonarQube to perform

SAST

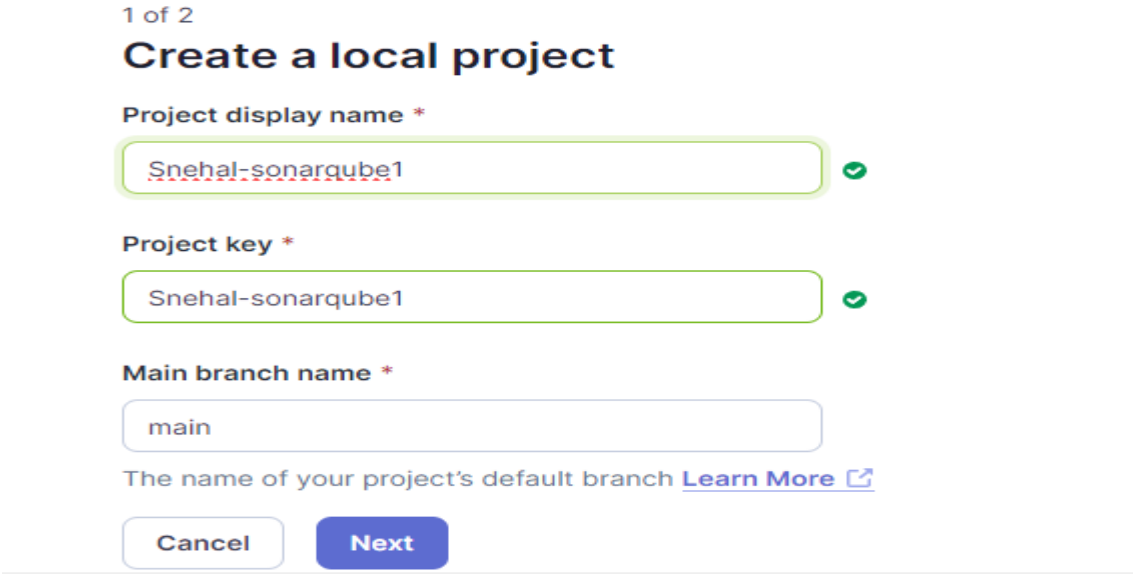
1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.
2. Run SonarQube in a Docker container using this command –

```
PS C:\Users\Windows> 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
7df3e28058c6bfc74d745f9f18f0923c82c1fc4058967a5b33907e0010b01ee2
PS C:\Users\Windows>
```

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.

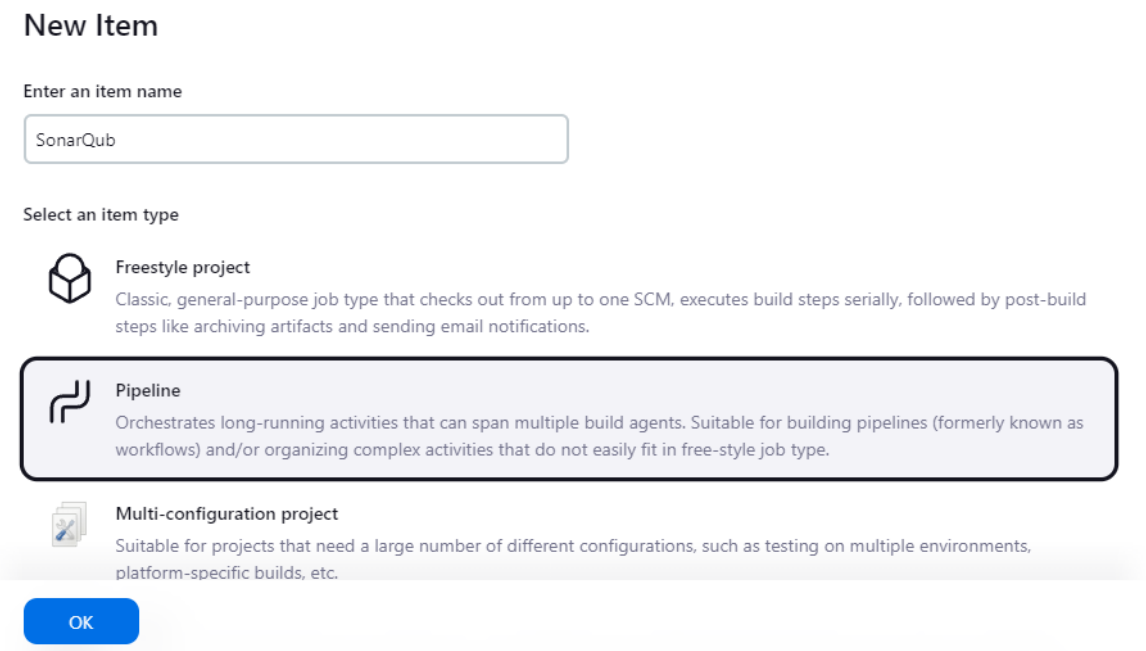


5. Create a manual project in SonarQube with the name sonarqube-test



5. Setup the project and come back to Jenkins Dashboard.

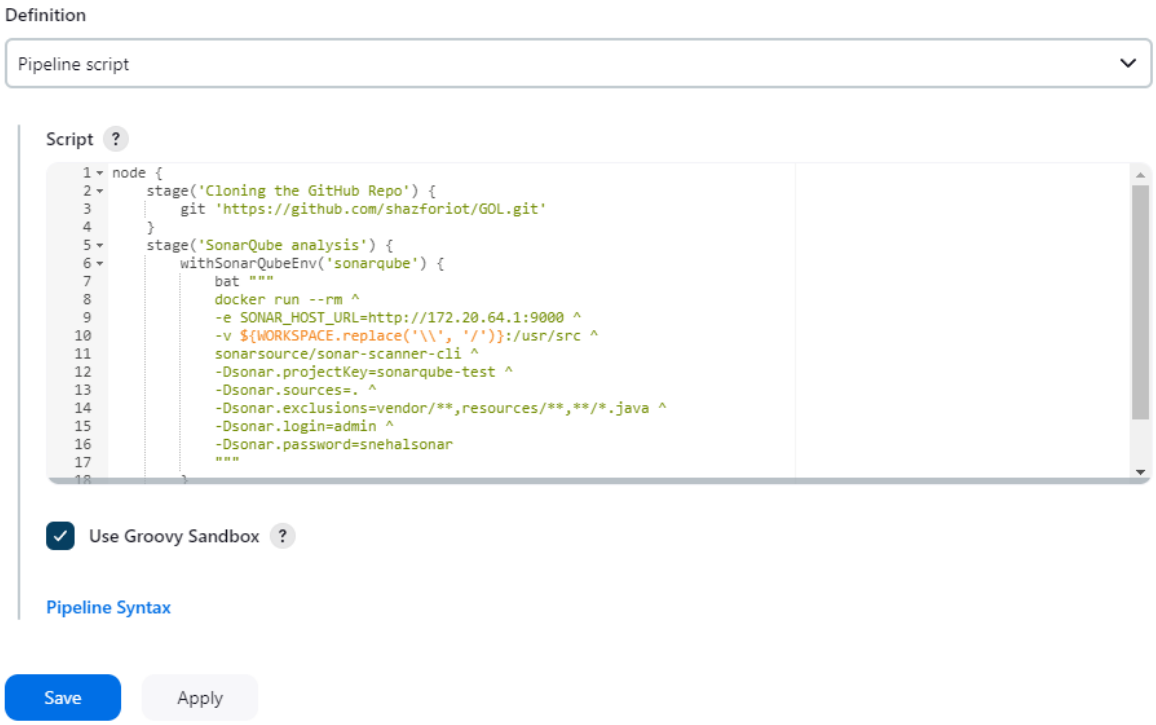
6. Create a New Item in Jenkins, choose Pipeline.



7. Under Pipeline Script, enter the following -

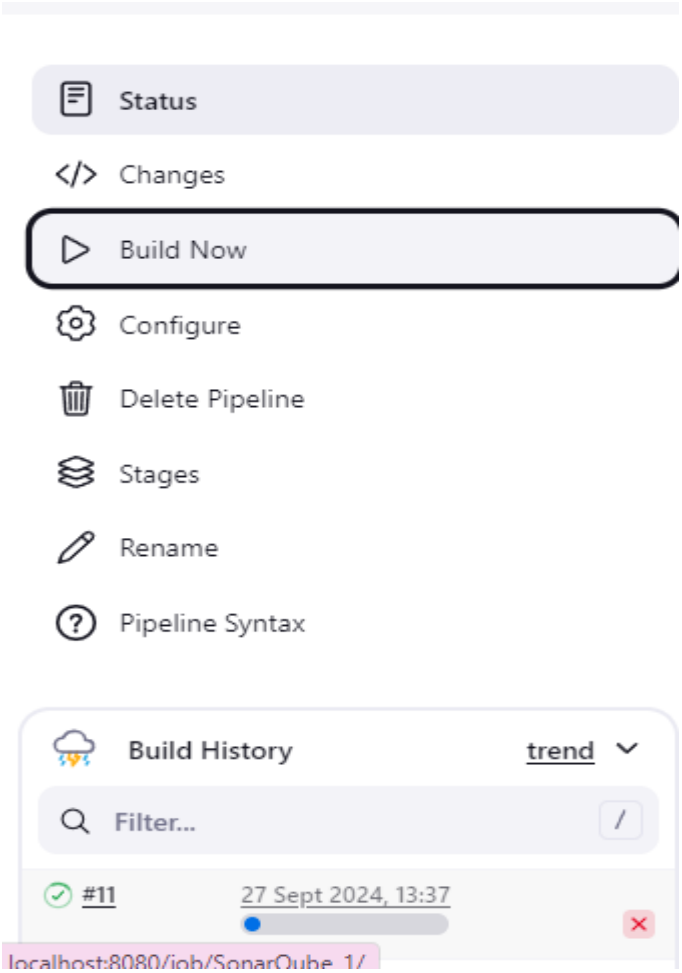
Under Pipeline Script, enter the following -

```
node {
stage('Cloning the GitHub Repo') {
git 'https://github.com/shazforiot/GOL.git'
}
stage('SonarQube analysis') {
withSonarQubeEnv('sonarqube') {
sh "<PATH_TO_SONARQUBE_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=http://127.0.0.1:9000/"
}
}
}
```

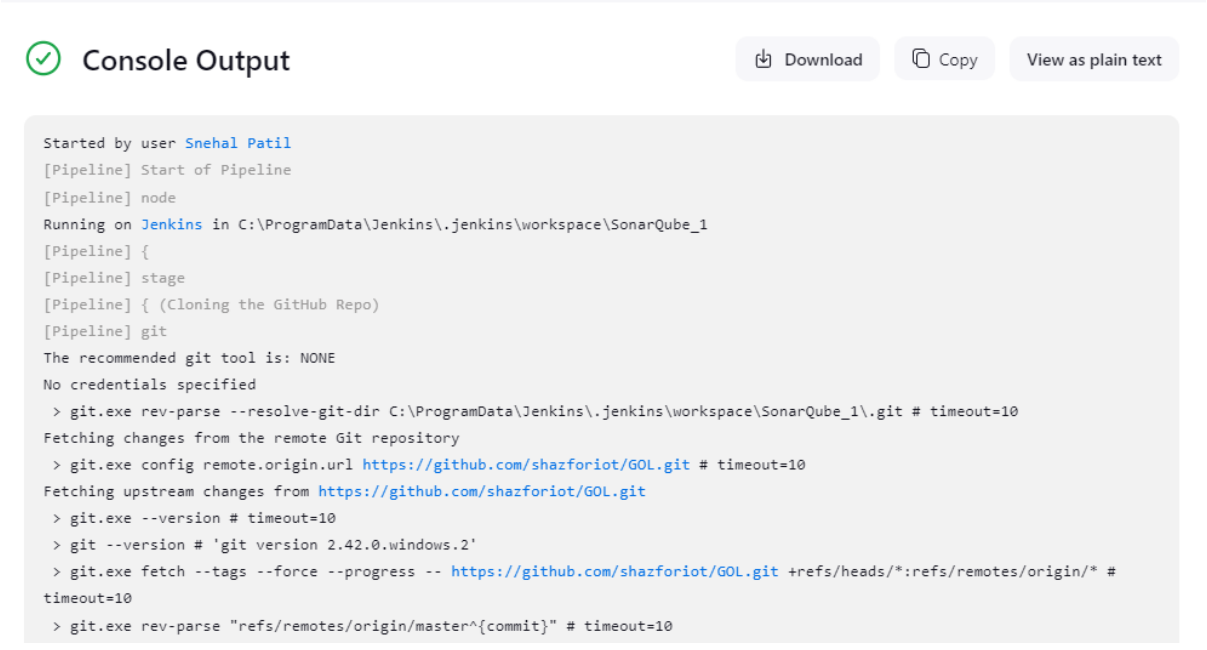


It is a java sample project which has a lot of repetitions and issues that will be detected by SonarQube.

8.Run The Build.

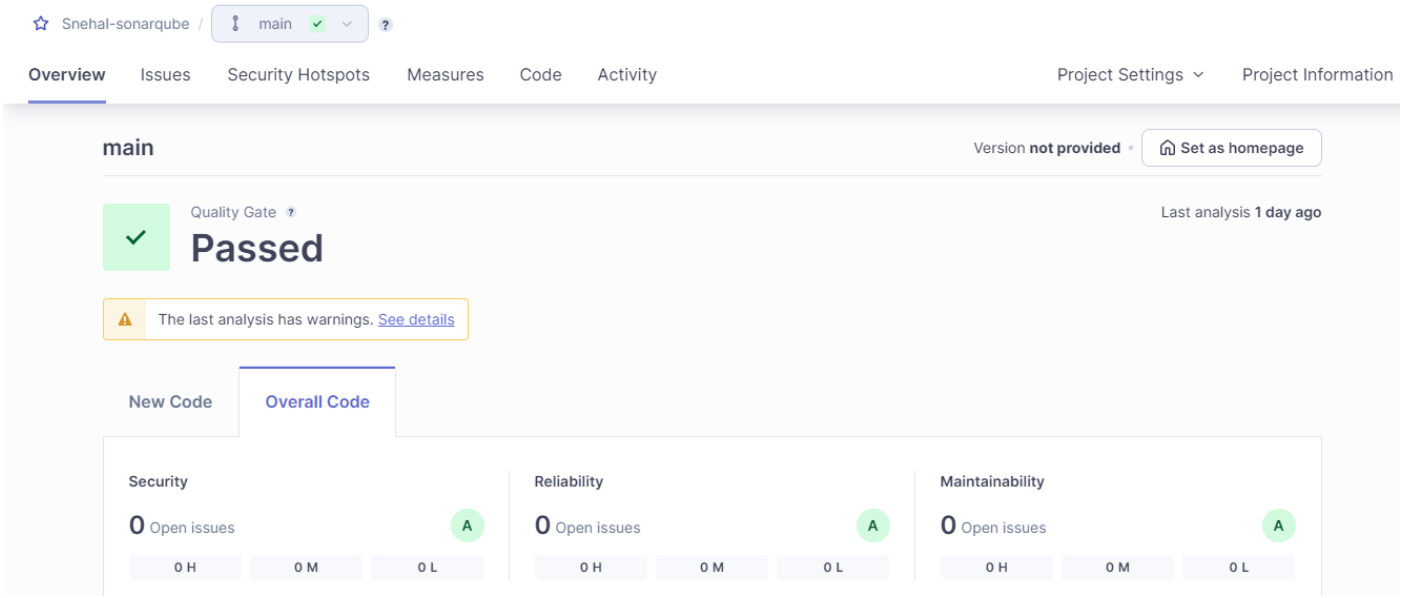


9. Check the console output once the build is complete.

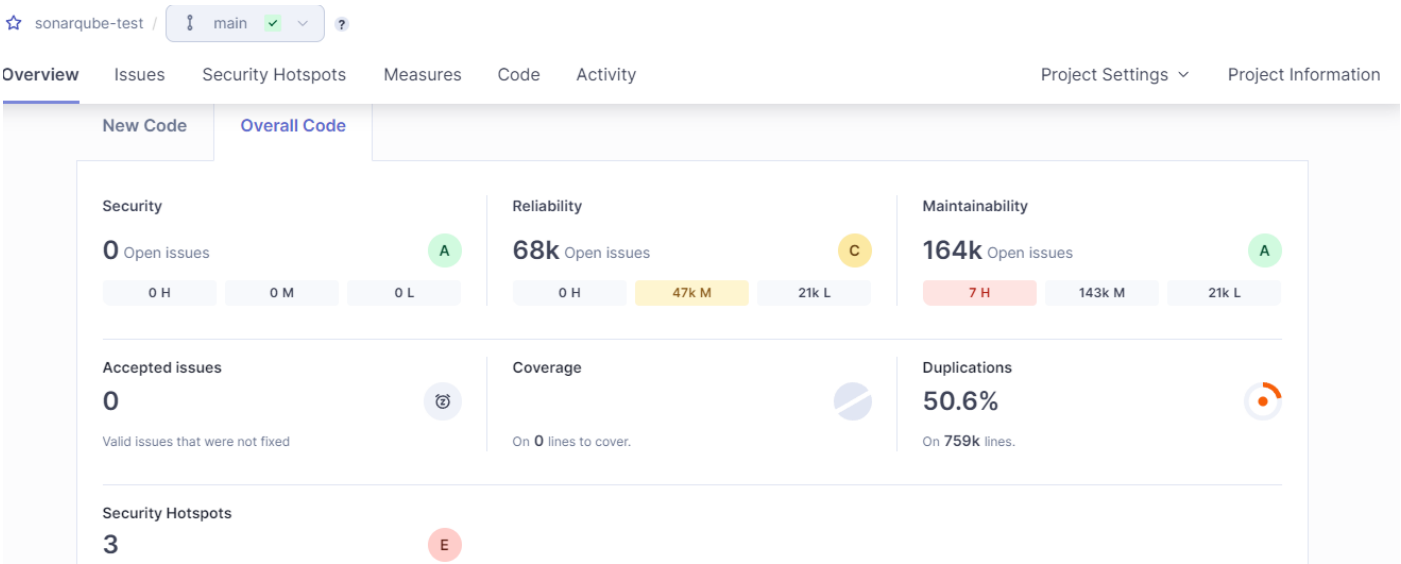


```
09:10:39.180 INFO  Analysis report generated in 42838ms, dir size=127.2 MB
09:11:11.618 INFO  Analysis report compressed in 32408ms, zip size=29.6 MB
09:11:36.343 INFO  Analysis report uploaded in 24600ms
09:11:36.349 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://172.20.64.1:9000/dashboard?id=sonarqube-test
09:11:36.350 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted
analysis report
09:11:36.350 INFO  More about the report processing at http://172.20.64.1:9000/api/ce/task?id=176cf1be-9b99-439a-a09b-
9a9d6f49a11c
09:11:53.206 INFO  Analysis total time: 31:25.145 s
09:11:53.223 INFO  SonarScanner Engine completed successfully
09:11:54.386 INFO  EXECUTION SUCCESS
09:11:56.253 INFO  Total time: 32:07.446s
[Pipeline] }
WARN: Unable to locate 'report-task.txt' in the workspace. Did the SonarScanner succeed?
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

10. After that, check the project in SonarQube.

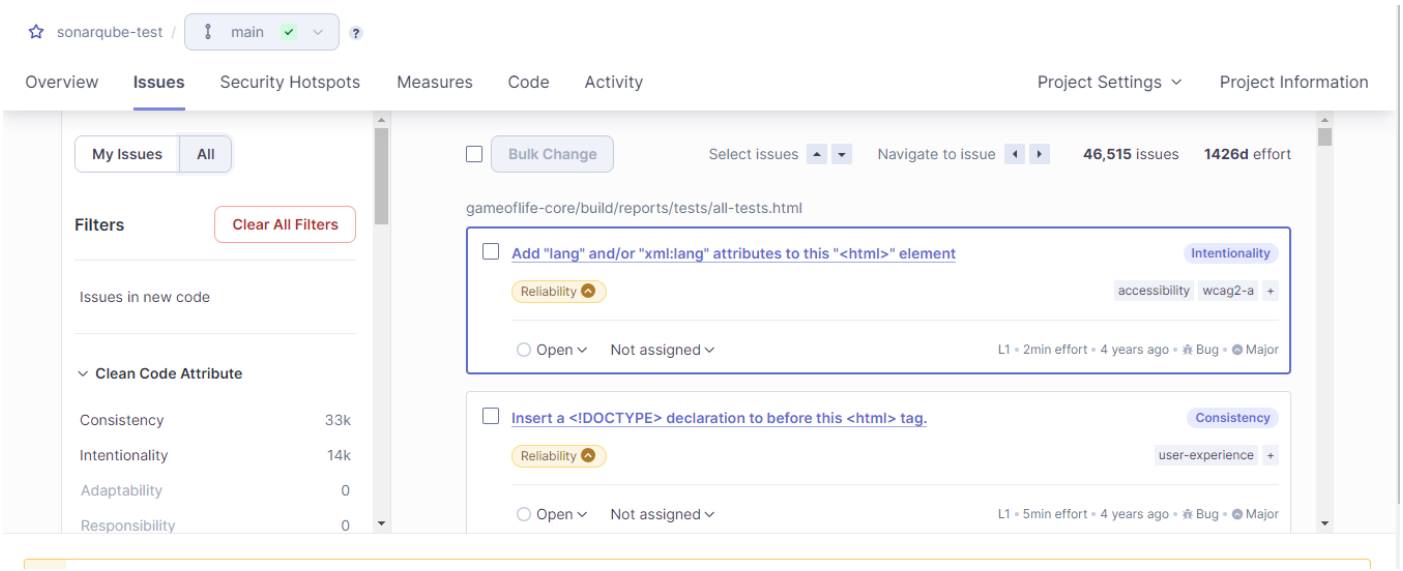


Under different tabs, check all different issues with the code.

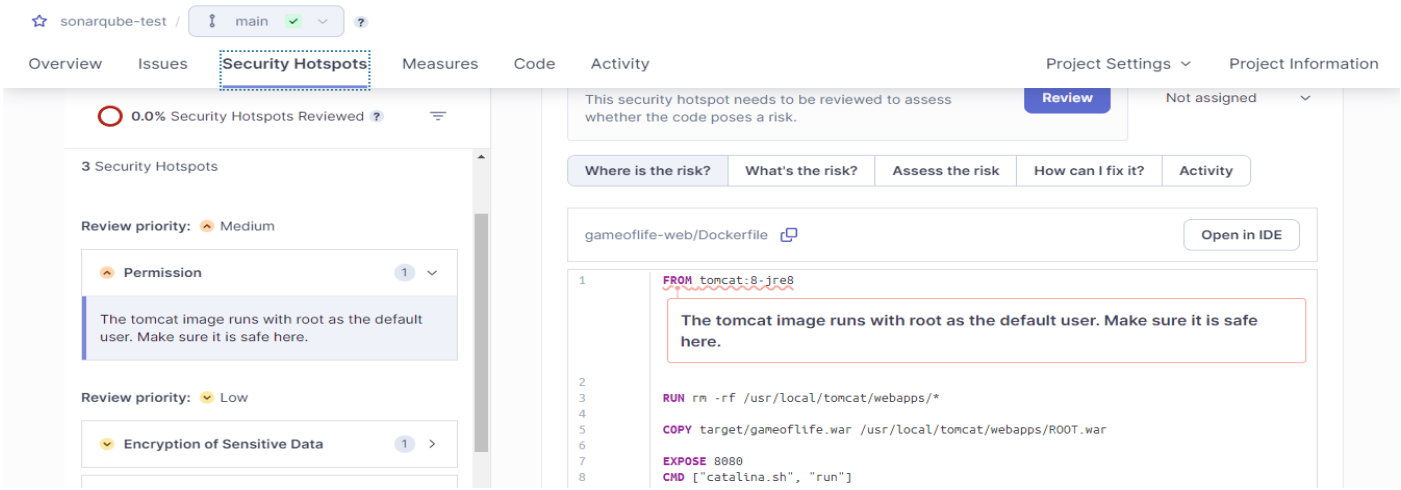


11. Code Problems –

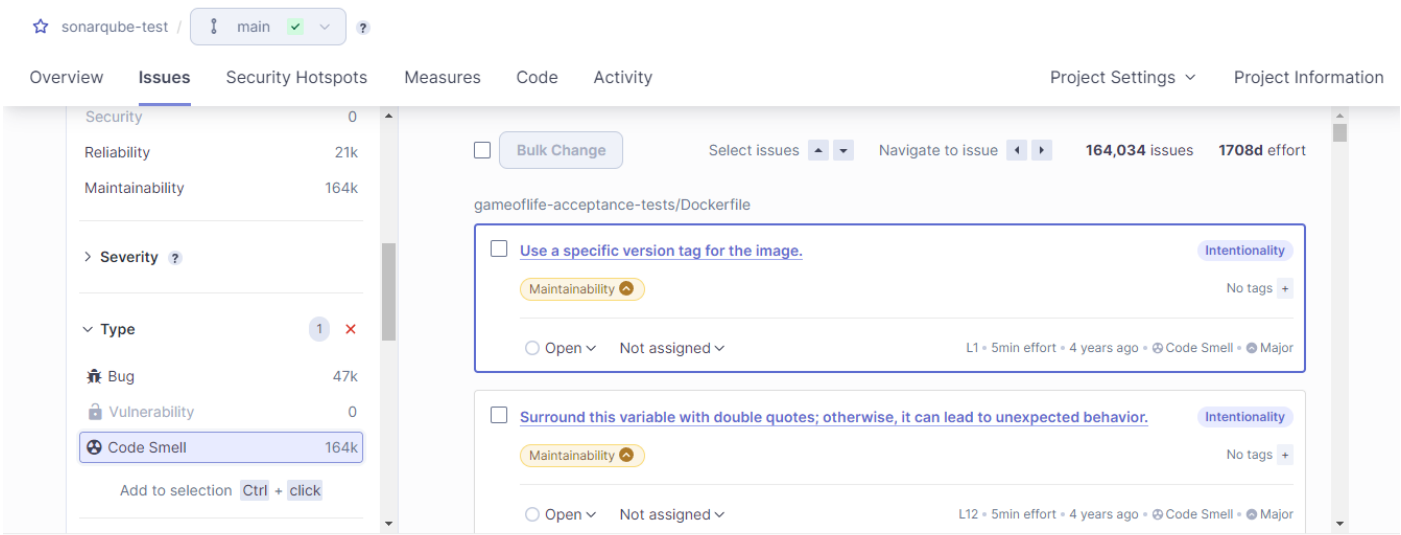
Issues:



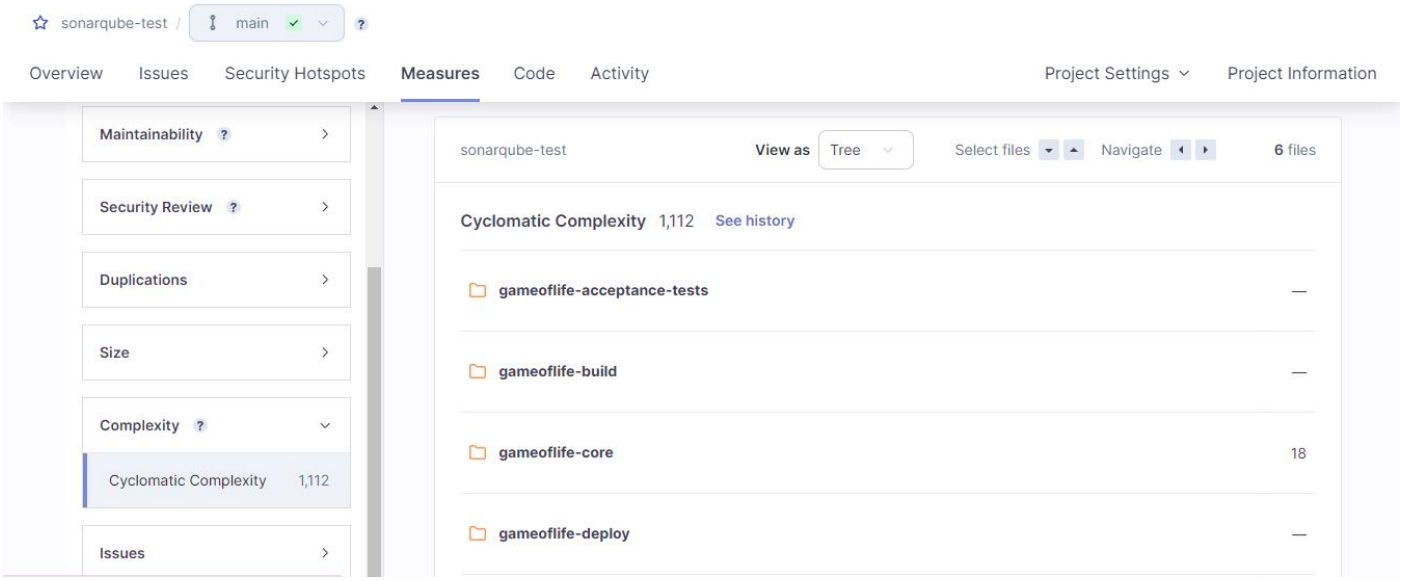
Security hotspots:



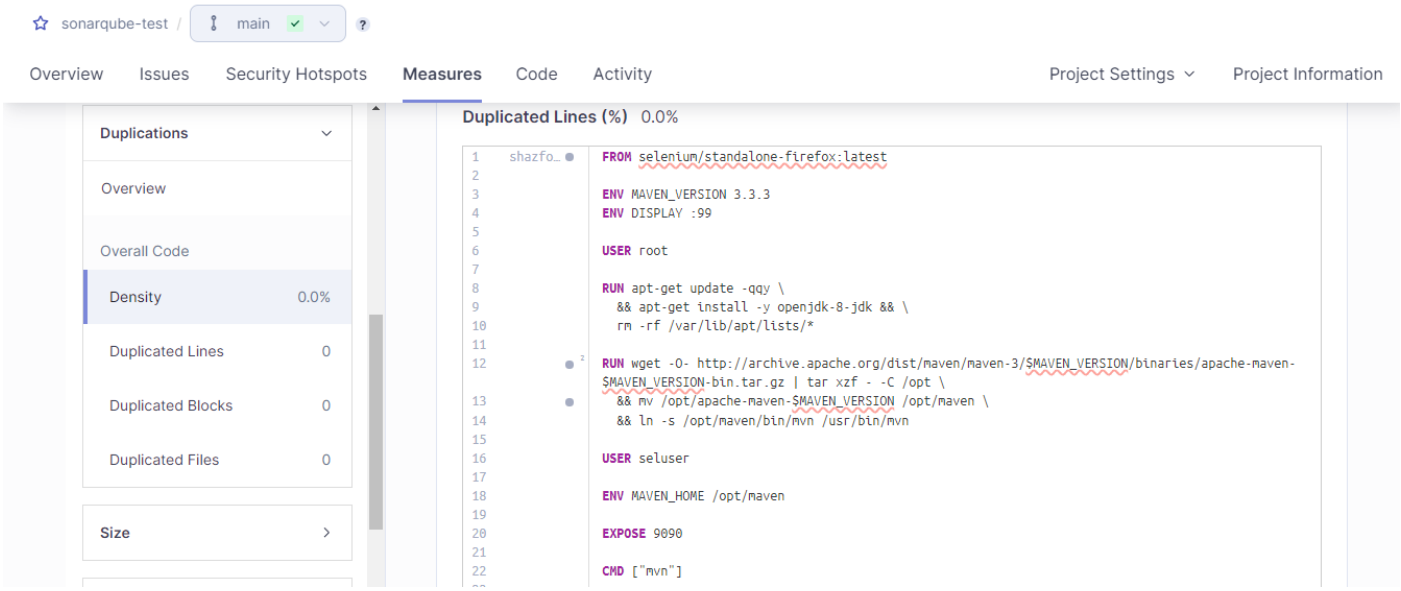
Codesmells:



Complexity:



Duplications:



Snehal Patil D15A 39

Advanced DevOps Lab

Experiment No: 9

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

Theory:

What is Nagios?

Nagios is an open-source software for continuous monitoring of systems, networks, and infrastructures. It runs plugins stored on a server that is connected with a host or another server on your network or the Internet. In case of any failure, Nagios alerts about the issues so that the technical team can perform the recovery process immediately.

Nagios is used for continuous monitoring of systems, applications, service and business processes in a DevOps culture.

Why We Need Nagios tool?

Here are the important reasons to use Nagios monitoring tool:

- Detects all types of network or server issues
- Helps you to find the root cause of the problem which allows you to get the permanent solution to the problem
- Active monitoring of your entire infrastructure and business processes
- Allows you to monitor and troubleshoot server performance issues
- Helps you to plan for infrastructure upgrades before outdated systems create failures
- You can maintain the security and availability of the service
- Automatically fix problems in a panic situation

Features of Nagios:

Following are the important features of Nagios monitoring tool:

- Relatively scalable, Manageable, and Secure
- Good log and database system
- Informative and attractive web interfaces
- Automatically send alerts if condition changes
- If the services are running fine, then there is no need to do check that host is an alive
- Helps you to detect network errors or server crashes
- You can troubleshoot the performance issues of the server.

● The issues, if any, can be fixed automatically as they are identified during the monitoring process

● You can monitor the entire business process and IT infrastructure with a single pass

● The product's architecture is easy to write new plugins in the language of your choice

● Nagios allows you to read its configuration from an entire directory which helps you to decide how to define individual files

● Utilizes topology to determine dependencies

● Monitor network services like HTTP, SMTP, HTTP, SNMP, FTP, SSH, POP, etc.

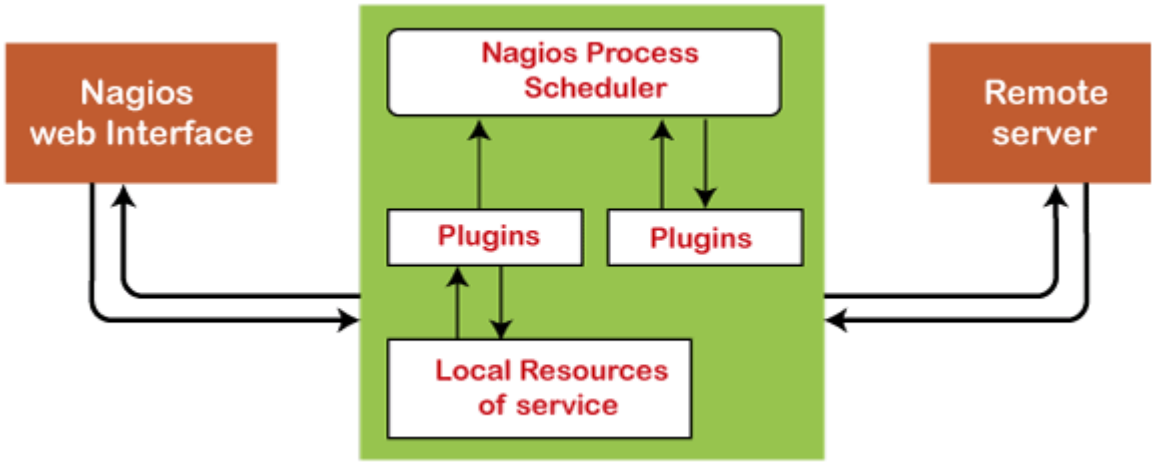
● Helps you to define network host hierarchy using parent hosts

● Ability to define event handlers that runs during service or host events for proactive problem resolution

● Support for implementing redundant monitoring hosts

Nagios Architecture

Nagios is a client-server architecture. Usually, on a network, a Nagios server is running on a host, and plugins are running on all the remote hosts which should be monitored.



1. The scheduler is a component of the server part of Nagios. It sends a signal to execute the plugins at the remote host.

2. The plugin gets the status from the remote host

3. The plugin sends the data to the process scheduler

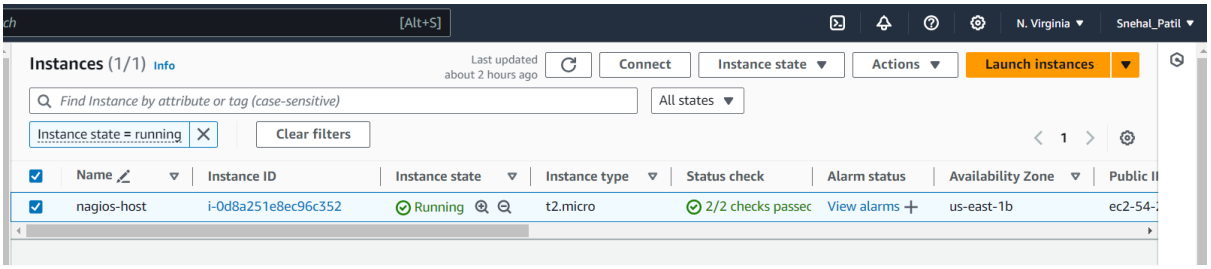
4. The process scheduler updates

Installation of Nagios

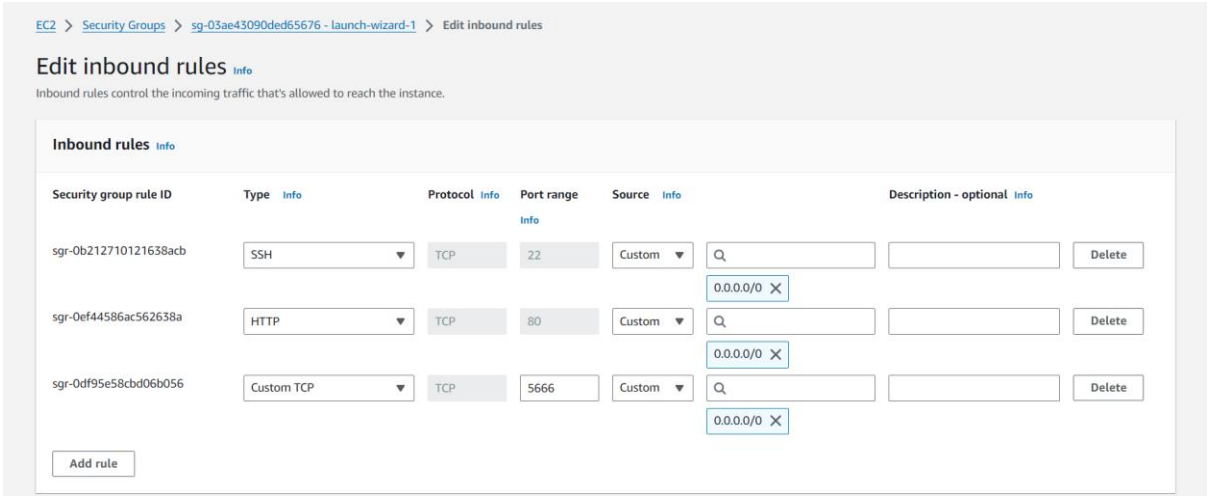
Prerequisites: AWS Free Tier

Steps:

1. Create an Amazon Linux EC2 Instance in AWS and name it - nagios-host

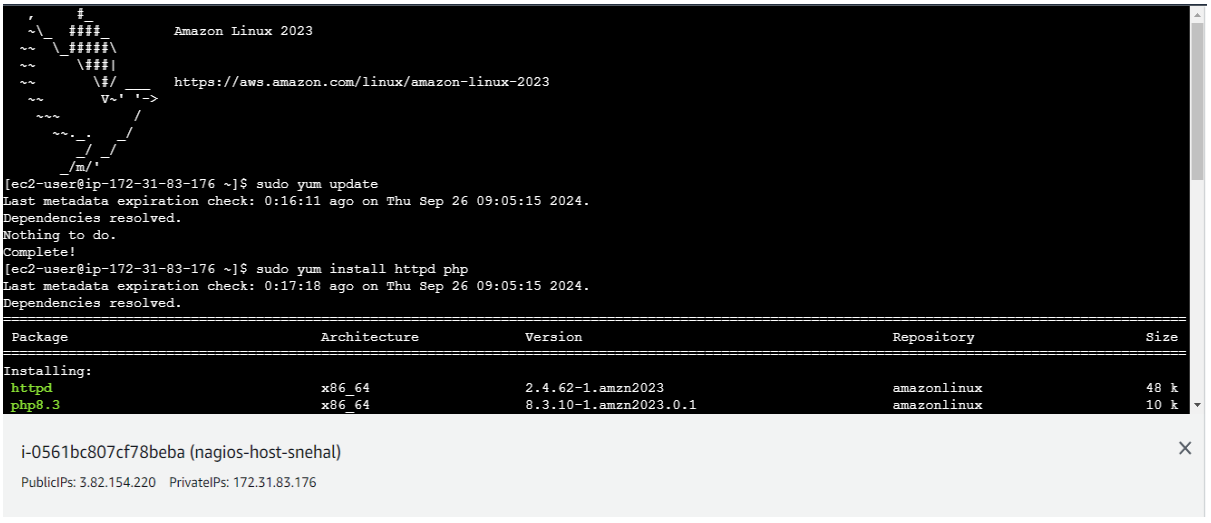


2. Under Security Group, make sure HTTP, HTTPS, SSH, ICMP are open from everywhere.



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

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



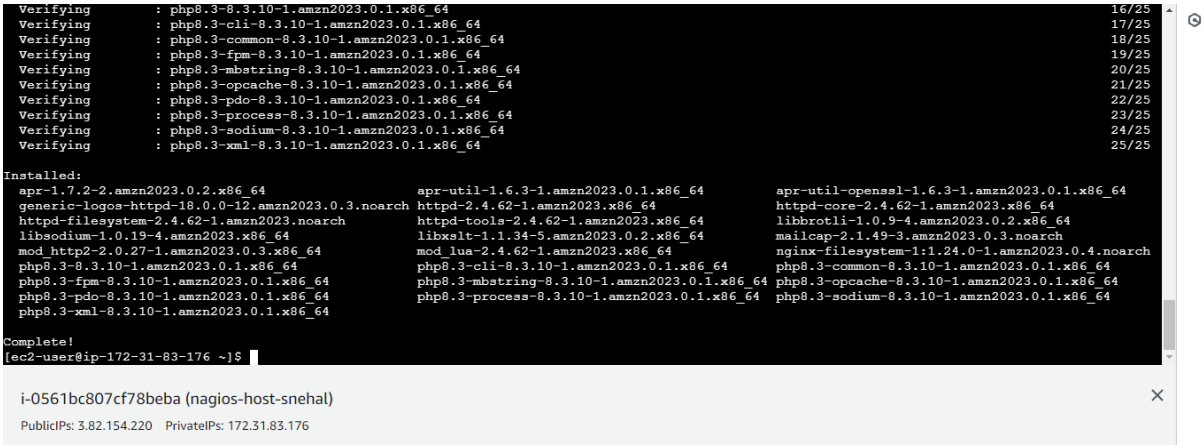
4. Update the package indices and install the following packages using yum

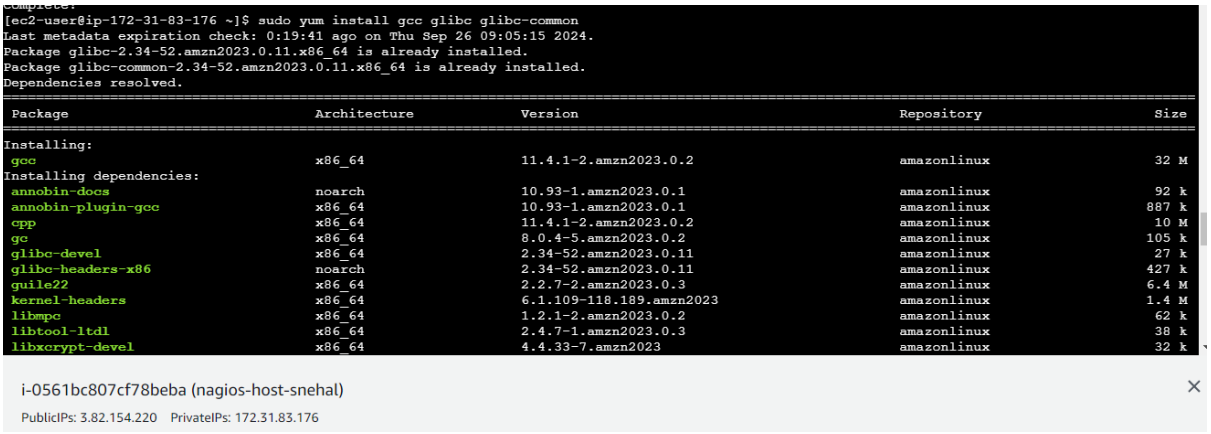
sudo yum update

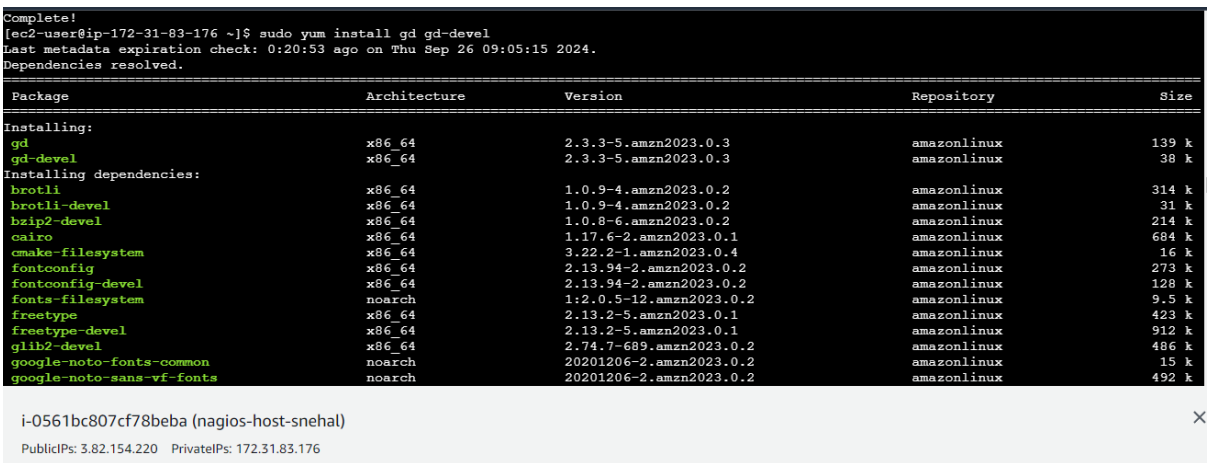
sudo yum install httpd php

sudo yum install gcc glibc glibc-common

sudo yum install gd gd-devel







5. Create a new Nagios User with its password. You'll have to enter the password twice for confirmation.

```
[ec2-user@ip-172-31-83-176 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-83-176 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
[ec2-user@ip-172-31-83-176 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
Sorry, passwords do not match.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-83-176 ~]$
```

6. Create a new user group

sudo groupadd nagcmd

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

sudo usermod -a -G nagcmd nagios

sudo usermod -a -G nagcmd apache

```
[ec2-user@ip-172-31-83-176 ~]$ sudo usermod -a -G nagcmd nagios
[ec2-user@ip-172-31-83-176 ~]$ sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-83-176 ~]$
```

8. Create a new directory for Nagios downloads

mkdir ~/downloads

cd ~/downloads

```
[ec2-user@ip-172-31-83-176 ~]$ mkdir ~/downloads
[ec2-user@ip-172-31-83-176 ~]$ cd ~/downloads
```

9. Use wget to download the source zip files.

wget

http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.0.8.tar.

gz

wget http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz

```
[ec2-user@ip-172-31-83-176 downloads]$ wget
http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.0.8.tar.
gz
wget http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz
wget: missing URL
Usage: wget [OPTION]... [URL]...

Try `wget --help' for more options.
-bash: http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.0.8.tar.: No such file or directory
-bash: gz: command not found
--2024-09-26 09:33:05--  http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2659772 (2.5M) [application/x-gzip]
Saving to: 'nagios-plugins-2.0.3.tar.gz'

nagios-plugins-2.0.3.tar.gz         100%[=================================================================>]   2.54M  7.18MB/s    in 0.4s

2024-09-26 09:33:06 (7.18 MB/s) - 'nagios-plugins-2.0.3.tar.gz' saved [2659772/2659772]
```

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

tar zxvf nagios-4.0.8.tar.gz

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

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

12. Compile the source code.

make all

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

command directory.

sudo make install

sudo make install-init

sudo make install-config

sudo make install-commandmode

```
[ec2-user@ip-172-31-91-100 nagioscore-nagios-4.4.6]$ cd /tmp/nagioscore-nagios-4.4.6
make all
sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
cd ./base && make
make[1]: Entering directory '/tmp/nagioscore-nagios-4.4.6/base'
make -C ../lib
make[2]: Entering directory '/tmp/nagioscore-nagios-4.4.6/lib'
make[2]: Nothing to be done for 'all'.
```

14.Edit the config file and change the email address.

sudo nano /usr/local/nagios/etc/objects/contacts.cfg

```
[ec2-user@ip-172-31-83-176 nagios-4.0.8]$ sudo nano /usr/local/nagios/etc/objects/contacts.cfg
[ec2-user@ip-172-31-83-176 nagios-4.0.8]$ sudo nano /usr/local/nagios/etc/objects/contacts.cfg
```

```
##############################################################################
##############################################################################

# Just one contact defined by default - the Nagios admin (that's you)
# This contact definition inherits a lot of default values from the 'generic-contact'
# template which is defined elsewhere.

define contact{
        contact_name                    nagiosadmin             ; Short name of user
        use                             generic-contact         ; Inherit default values from generic-contact template (defined above)
        alias                           Nagios Admin            ; Full name of user

        email                           snehalpatil302004@gmail.com     ; <<***** CHANGE THIS TO YOUR EMAIL ADDRESS ******
        }


##############################################################################
##############################################################################
```

15. Configure the web interface.

sudo make install-webconf

```
[ec2-user@ip-172-31-83-176 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 ***

[ec2-user@ip-172-31-83-176 nagios-4.0.8]$
```

16. Create a nagiosadmin account for nagios login along with password. You'll have to

specify the password twice.

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

17. Restart Apache

sudo service httpd restart

```
[ec2-user@ip-172-31-83-176 nagios-4.0.8]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-83-176 nagios-4.0.8]$
```

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

cd ~/downloads

tar zxvf nagios-plugins-2.0.3.tar.gz

```
[ec2-user@ip-172-31-83-176 nagios-4.0.8]$ cd ~/downloads
tar zxvf nagios-plugins-2.0.3.tar.gz
nagios-plugins-2.0.3/
nagios-plugins-2.0.3/perlmods/
nagios-plugins-2.0.3/perlmods/Config-Tiny-2.14.tar.gz
nagios-plugins-2.0.3/perlmods/parent-0.226.tar.gz
nagios-plugins-2.0.3/perlmods/Test-Simple-0.98.tar.gz
nagios-plugins-2.0.3/perlmods/Makefile.in
nagios-plugins-2.0.3/perlmods/version-0.9903.tar.gz
nagios-plugins-2.0.3/perlmods/Makefile.am
nagios-plugins-2.0.3/perlmods/Module-Runtime-0.013.tar.gz
nagios-plugins-2.0.3/perlmods/Module-Metadata-1.000014.tar.gz
nagios-plugins-2.0.3/perlmods/Params-Validate-1.08.tar.gz
```

19. Compile and install plugins

cd nagios-plugins-2.0.3

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

make

sudo make install

20. Start Nagios

Add Nagios to the list of system services

sudo chkconfig --add nagios

sudo chkconfig nagios on

Verify the sample configuration files

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

If there are no errors, you can go ahead and start Nagios.

sudo service nagios start

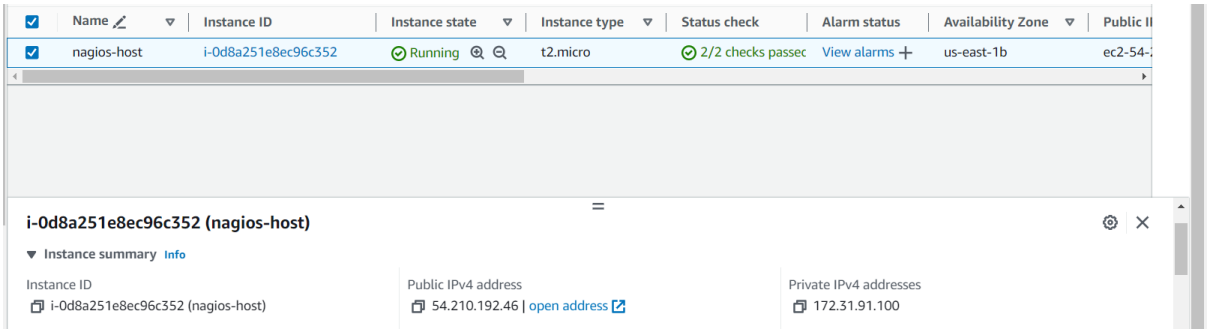
```
[ec2-user@ip-172-31-83-176 ~]$ sudo service nagios start
Starting nagios (via systemctl):                           [  OK  ]
[ec2-user@ip-172-31-83-176 ~]$
[ec2-user@ip-172-31-83-176 ~]$ sudo systemctl status nagios
● nagios.service - LSB: Starts and stops the Nagios monitoring server
```

21. Check the status of Nagios

sudo systemctl status nagios

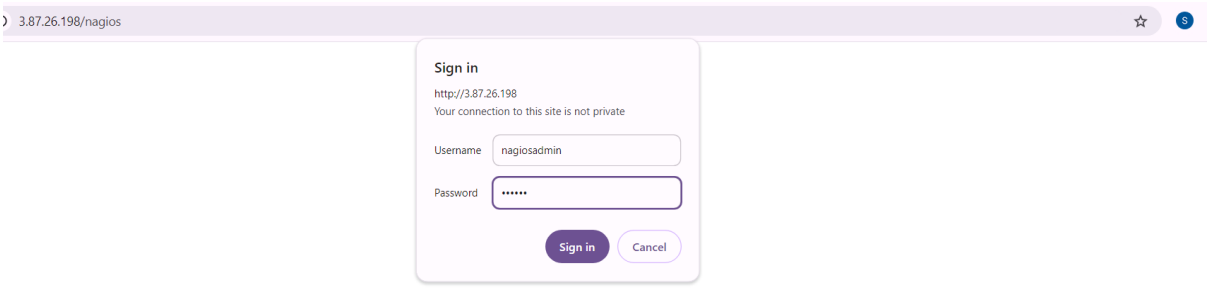
```
[ec2-user@ip-172-31-91-100 nagioscore-nagios-4.4.6]$ sudo systemctl status nagios
sudo systemctl status httpd
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: di>
     Active: active (running) since Sat 2024-09-28 17:23:08 UTC; 5min ago
       Docs: https://www.nagios.org/documentation
   Main PID: 34305 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 2.0M
        CPU: 68ms
     CGroup: /system.slice/nagios.service
             ├─34305 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagio>
```

22. Go back to EC2 Console and copy the Public IP address of this instance



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

Enter username as nagiosadmin and password which you set in Step 16.



Extra:

Faced Problem

Permission Forbidden Error

Problem: While accessing the Nagios web interface, I encountered "403 Forbidden" errors.

Solution: This issue typically arises due to improper file permissions or SELinux settings. To resolve it:

Adjust File Permissions: Ensure that the Nagios directory and its files have the correct permissions. Execute the following commands:

sudo chown -R nagios:nagios /usr/local/nagios

sudo chmod -R 755 /usr/local/nagios

```
        _/m/'
[ec2-user@ip-172-31-35-105 ~]$ sudo chown -R nagios:nagios /usr/local/nagios
sudo chmod -R 755 /usr/local/nagios
```

Configure SELinux (if applicable): If SELinux is enabled on your instance, it might block access. To check the status and modify it:

sestatus # Check SELinux status

sudo setenforce 0 # Temporarily disable SELinux

```
[ec2-user@ip-172-31-35-105 ~]$ sestatus  # Check SELinux status
sudo setenforce 0  # Temporarily disable SELinux
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   permissive
Mode from config file:          permissive
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      33
```

Apache Configuration: Ensure that the Apache configuration allows access to the Nagios directory. You may need to add the following lines to your /etc/httpd/conf.d/nagios.conf:

<Directory "/usr/local/nagios/share">

Options None

AllowOverride None

Require all granted

</Directory>

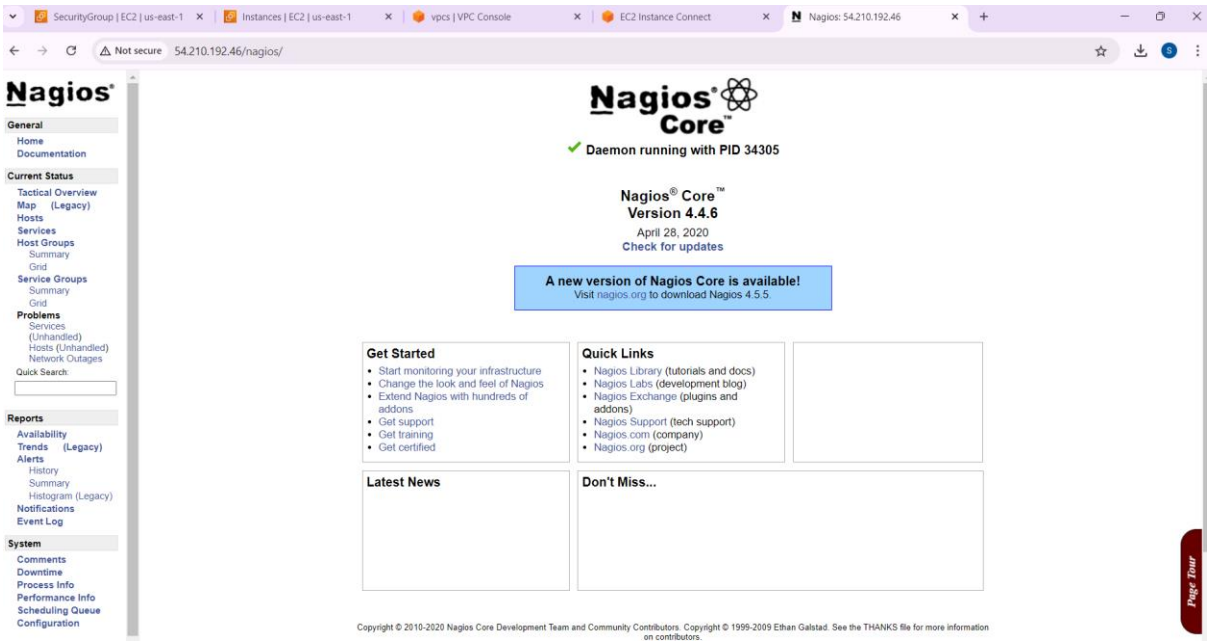
```
<Directory "/usr/local/nagios/share">
    Options None
    AllowOverride None
    Require all granted
</Directory>
```

After making changes, restart Apache:

sudo service httpd restart

24. After entering the correct credentials, I am able to

see this page.



This means that Nagios was correctly installed and configured with its plugins so far.

Conclusion:

We successfully installed Nagios on an Amazon Linux EC2 instance in the AWS Free Tier. By configuring the security group, setting up users and groups, and installing the necessary packages, we prepared the environment for Nagios. After downloading and compiling the software, we configured the web interface and authenticated the admin account. Accessing the Nagios dashboard confirmed a successful setup, enabling effective monitoring of systems and services. This installation lays the groundwork for a robust monitoring solution tailored to your needs.

Snehal Patil D15A 39

Advanced DevOps Lab

Experiment No: 10

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

Steps:

Prerequisites: AWS Free Tier, Nagios Server running on Amazon Linux Machine.

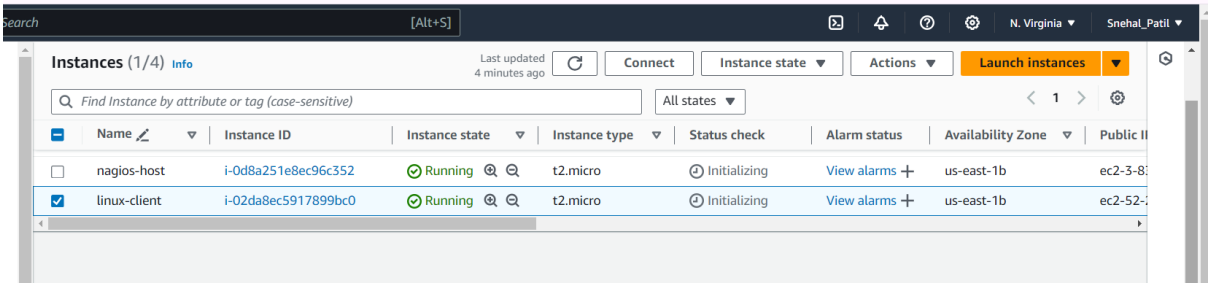
1. To Confirm that Nagios is running on the server side, run this sudo systemctl status nagios on the “NAGIOS HOST”.

```
[ec2-user@ip-172-31-91-100 ~]$ sudo systemctl status
nagios
● ip-172-31-91-100.ec2.internal
    State: running
    Units: 298 loaded (incl. loaded aliases)
     Jobs: 0 queued
   Failed: 0 units
    Since: Sun 2024-09-29 05:39:54 UTC; 25s ago
  systemd: 252.23-2.amzn2023
   CGroup: /
           ├─init.scope
           │ └─1 /usr/lib/systemd/systemd --switched-root --system --deserialize=32
           ├─system.slice
           │ ├─acpid.service
           │ │ ├─1957 /usr/bin/systemd-inhibit --what=handle-suspend-key:handle-hibernate
           │ │ └─1995 /usr/sbin/acpid -f
```

You can proceed if you get this message.

Before we begin,

2. To monitor a Linux machine, create an Ubuntu 20.04 server EC2 Instance in AWS. Provide it with the same security group as the Nagios Host and name it ‘linux-client’ alongside the host.



For now, leave this machine as is, and go back to your nagios HOST machine.

Extra:

Apache Server was Not Running

Problem: Unable to access the Nagios web interface due to the Apache HTTP server not running.

```
[ec2-user@ip-172-31-84-219 ~]$ sudo systemctl status httpd  # For CentOS/RHEL
# or
sudo systemctl status apache2  # For Ubuntu/Debian
o httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─php-fpm.conf
     Active: inactive (dead)
       Docs: man:httpd.service(8)
Unit apache2.service could not be found.
[ec2-user@ip-172-31-84-219 ~]$ sudo systemctl status httpd  # For CentOS/RHEL
# or
sudo systemctl status apache2  # For Ubuntu/Debian
```

```
Unit apache2.service could not be found.
[ec2-user@ip-172-31-84-219 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-84-219 ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─php-fpm.conf
     Active: active (running) since Wed 2024-10-02 06:48:57 UTC; 8s ago
       Docs: man:httpd.service(8)
   Main PID: 3401 (httpd)
     Status: "Started, listening on: port 80"
      Tasks: 177 (limit: 1112)
     Memory: 17.8M
        CPU: 58ms
     CGroup: /system.slice/httpd.service
             ├─3401 /usr/sbin/httpd -DFOREGROUND
             ├─3408 /usr/sbin/httpd -DFOREGROUND
             ├─3409 /usr/sbin/httpd -DFOREGROUND
             ├─3410 /usr/sbin/httpd -DFOREGROUND
             └─3411 /usr/sbin/httpd -DFOREGROUND
```

```
Oct 02 06:48:57 ip-172-31-84-219.ec2.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Oct 02 06:48:57 ip-172-31-84-219.ec2.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Oct 02 06:48:57 ip-172-31-84-219.ec2.internal httpd[3401]: Server configured, listening on: port 80
[ec2-user@ip-172-31-84-219 ~]$ sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-172-31-84-219 ~]$ sudo systemctl restart nagios
[ec2-user@ip-172-31-84-219 ~]$
```

3. On the server, run this command

ps -ef | grep nagios

```
[ec2-user@ip-172-31-91-100 ~]$ ps -ef | grep nagios
nagios      2007       1  0 05:39 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
nagios      2008    2007  0 05:39 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      2009    2007  0 05:39 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      2010    2007  0 05:39 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      2011    2007  0 05:39 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      2012    2007  0 05:39 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
ec2-user    2917    2760  0 05:42 pts/0    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-91-100 ~]$
```

4. Become a root user and create 2 folders

sudo su

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

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

5.

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

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

6. Open linuxserver.cfg using nano and make the following changes

nano

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

Change the hostname to linuxserver (EVERYWHERE ON THE FILE)

Change address to the public IP address of your LINUX CLIENT.

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

# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will inherit all variables that are defined
                                                    ; in (or inherited by) the linux-server host template definition.
    host_name               linuxserver
    alias                   localhost
    address                 52.202.216.168
}



###############################################################################
```

Change hostgroup_name under hostgroup to linux-servers1

```
  GNU nano 5.8                                  /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver
###############################################################################

# 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 hosts that belong to this group
}
```

7. Open the Nagios Config file and add the following line

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

##Add this line

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

8. Verify the configuration files

```
[root@ip-172-31-91-100 ec2-user]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

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

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

```
Running pre-flight check on configuration data...

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

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
```

You are good to go if there are no errors.

9. Restart the nagios service

service nagios restart

10. SSH into the machine or simply use the EC2 Instance Connect feature.

```
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-89-25:~$ sudo apt update -y
```

11. Make a package index update and install gcc, nagios-nrpe-server and the plugins.

sudo apt update -y

sudo apt install gcc -y

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

```
ubuntu@ip-172-31-89-25:~$ sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [380 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [82.9 kB]
Get:10 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4560 B]
```

12. Open nrpe.cfg file to make changes.

sudo nano /etc/nagios/nrpe.cfg

Under allowed_hosts, add your nagios host IP address like so

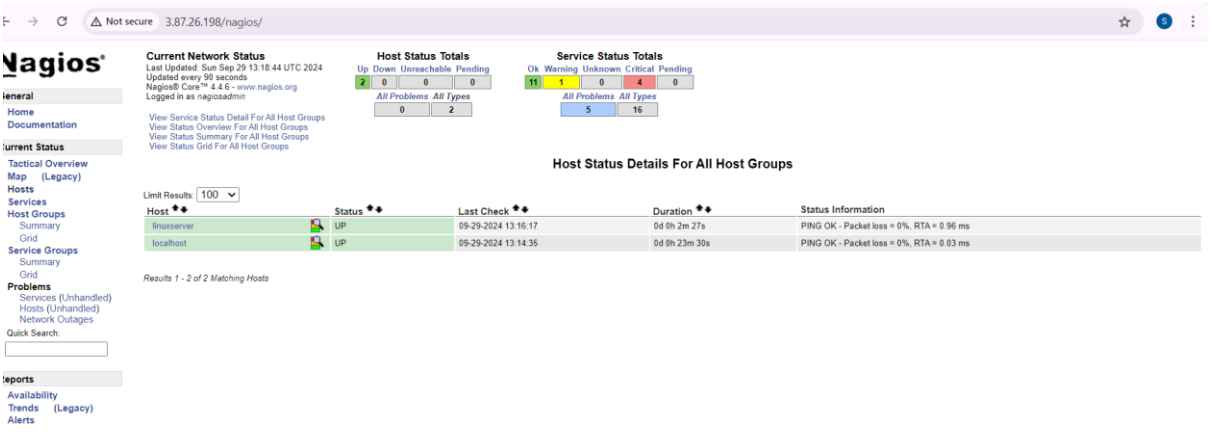
13. Restart the NRPE server

sudo systemctl restart nagios-nrpe-server

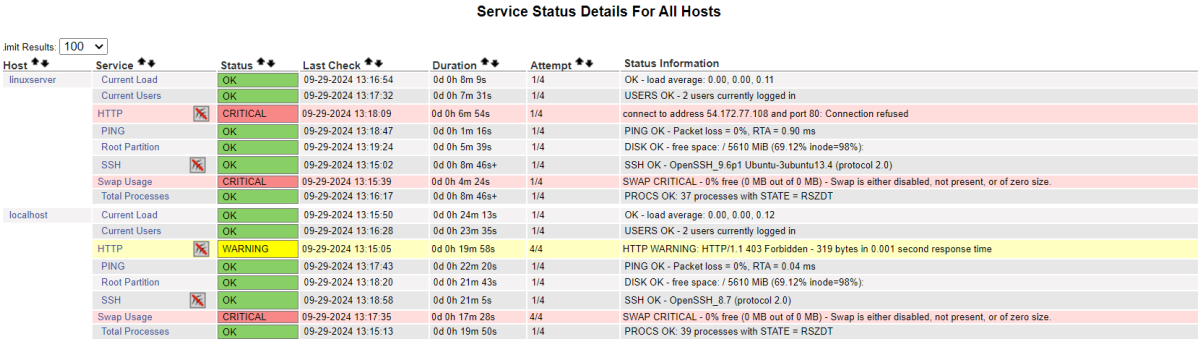
14. Now, check your nagios dashboard and you'll see a new host being added.

Click on Hosts.

Now here we can see there is a host added



You can click Services to see all services and ports being monitored.



As you can see, we have our linuxserver up and running. It is showing critical status on

HTTP due to permission errors and swap because there is no partition created.

In this case, we have monitored -

Servers: 1 linux server

Services: swap

Ports: 22, 80 (ssh, http)

Processes: User status, Current load, total processes, root partition, etc.

Recommended Cleanup:

● Terminate both of your EC-2 instances to avoid charges.

● Delete the security group if you created a new one (it won't affect your bill, you may

avoid it)

Conclusion:

In this experiment, I successfully implemented Nagios for comprehensive port and service monitoring across both Windows and Linux servers. The setup allowed us to effectively track the status of critical services and ports, ensuring optimal performance and availability of server resources.

Name: Snehal A. Patil D15A 39

Advanced DevOps Lab

Experiment No: 11

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

Theory:

AWS Lambda

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

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

Features of AWS Lambda

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

Packaging Functions

Lambda functions need to be packaged and sent to AWS. This is usually a process of compressing the function and all its dependencies and uploading it to an S3 bucket. And letting AWS know that you want to use this package when a specific event takes place. To help us with this process we use the Serverless Stack Framework (SST). We'll go over this in detail later on inthis guide.

Execution Model

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

This has some interesting implications. Firstly, our functions are effectively stateless. Secondly, each request (or event) is served by a single instance of a Lambda function. This means that you are not going to be handling concurrent requests in your code. AWS brings up a container whenever there is a new request. It does make some optimizations here. It will hang on to the container for a few minutes (5 - 15mins depending on the load) so it can respond to subsequent requests without a cold start.

Stateless Functions

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

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

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

Common Use Cases for Lambda

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

1. Individual tasks run for a short time;
2. Each task is generally self-contained;
3. There is a large difference between the lowest and highest levels in the workload of the application.

Some of the most common use cases for AWS Lambda that fit these criteria are: Scalable APIs. When building APIs using AWS Lambda, one execution of a Lambda function can serve a

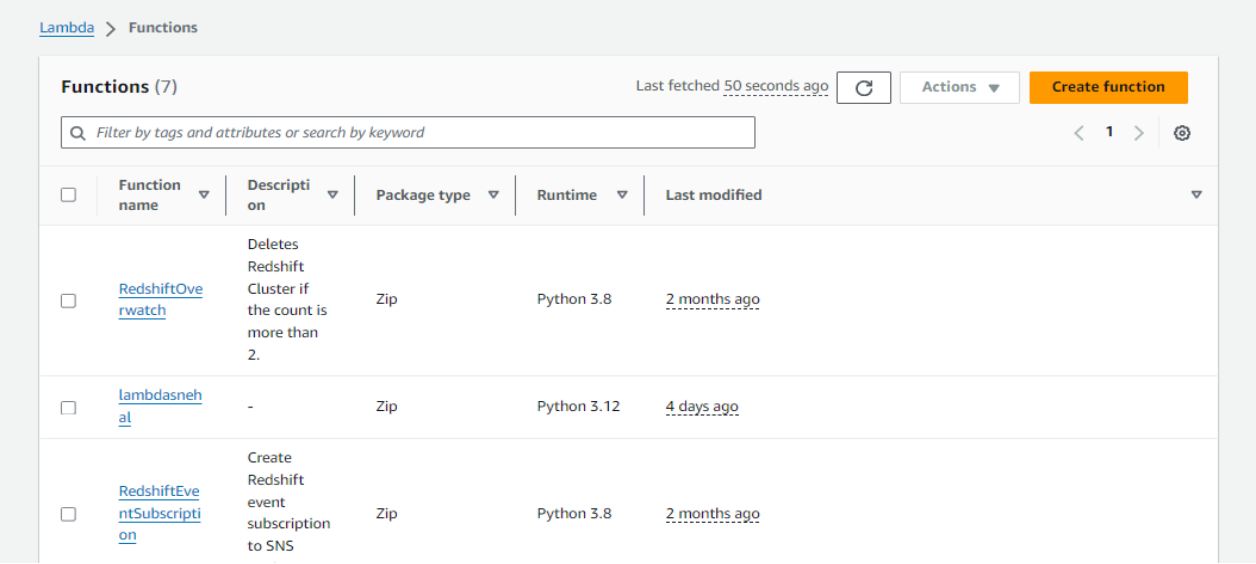
single HTTP request. Different parts of the API can be routed to different Lambda functions via Amazon API Gateway. AWS Lambda automatically scales individual functions according to the demand for them, so different parts of your API can scale differently according to current usage levels. This allows for cost-effective and flexible API setups.

Data processing. Lambda functions are optimized for event-based data processing. It is easy to integrate AWS Lambda with data sources like Amazon DynamoDB and trigger a Lambdafunction for specific kinds of data events. For example, you could employ Lambda to do some work every time an item in DynamoDB is created or updated, thus making it a good fit for things like notifications, counters and analytics.

Steps to create an AWS Lambda function

1. Open up the Lambda Console and click on the Create button.

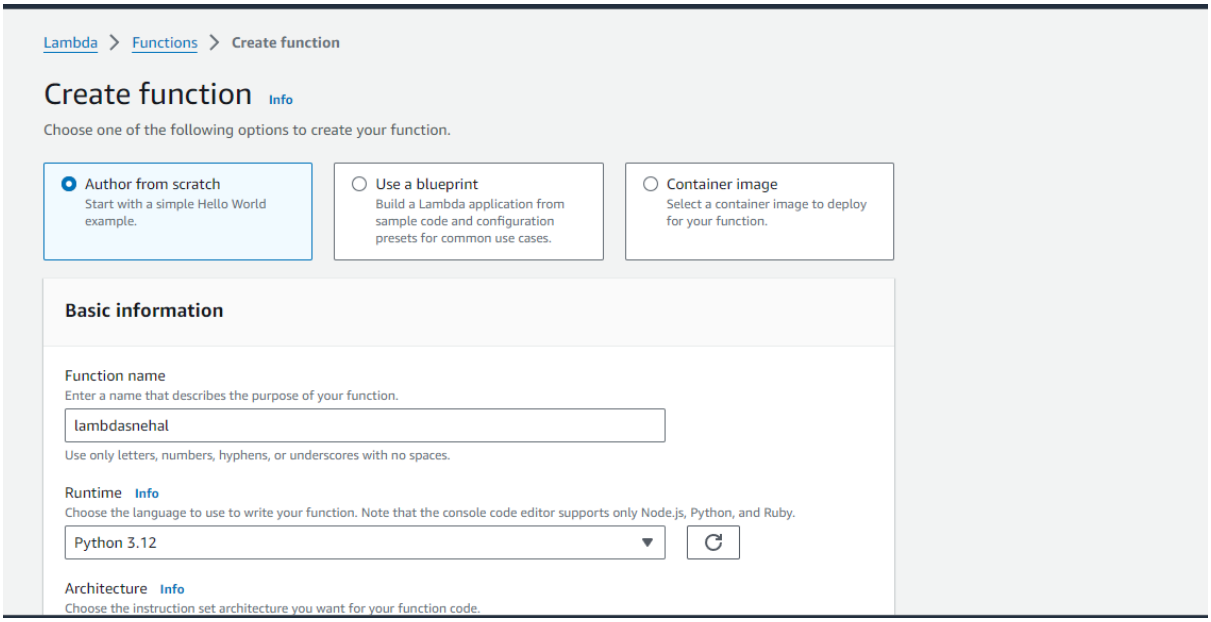
   Be mindful of where you create your functions since Lambda is region-dependent.

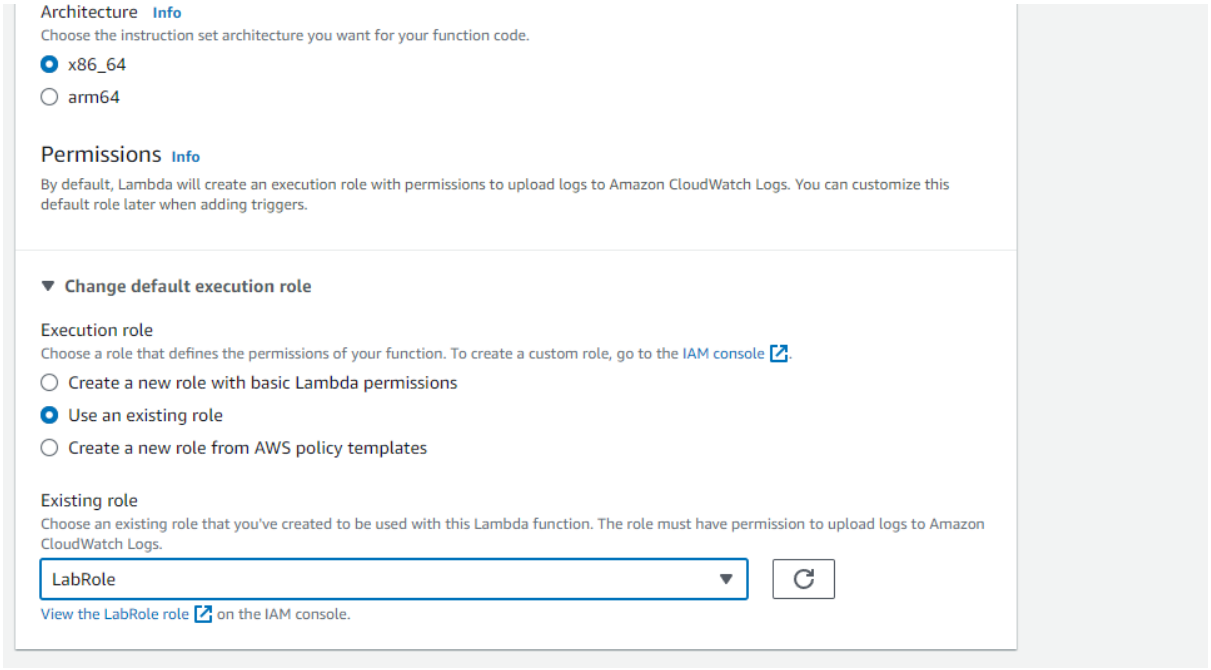


2. Choose to create a function from scratch or use a blueprint, i.e templates defined by AWS for you with all configuration presets required for the most common use cases.

   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.

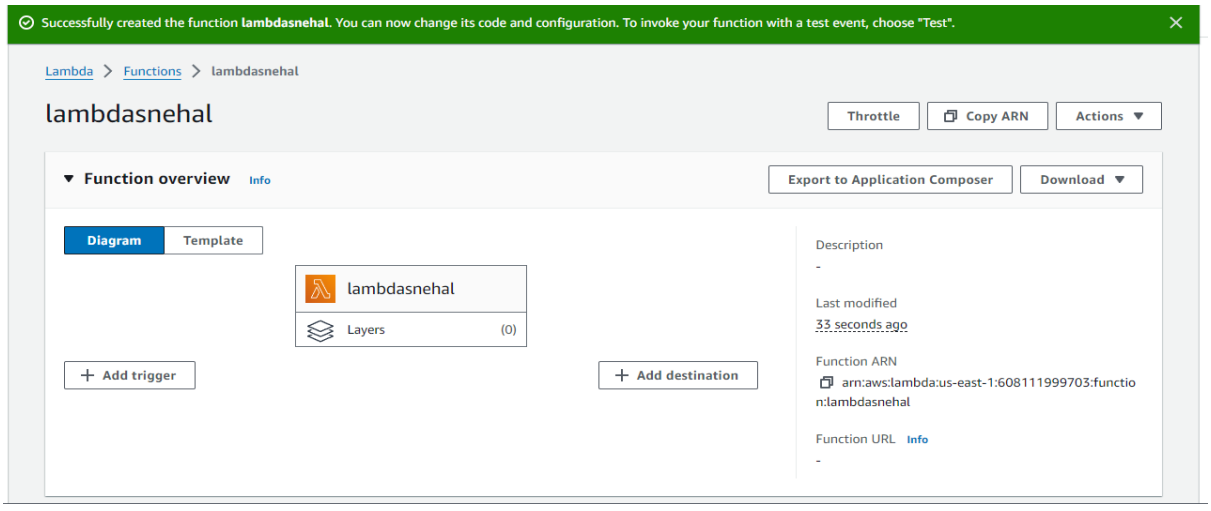
   After that, choose to create a new role with basic Lambda permissions if you don't have an existing one.

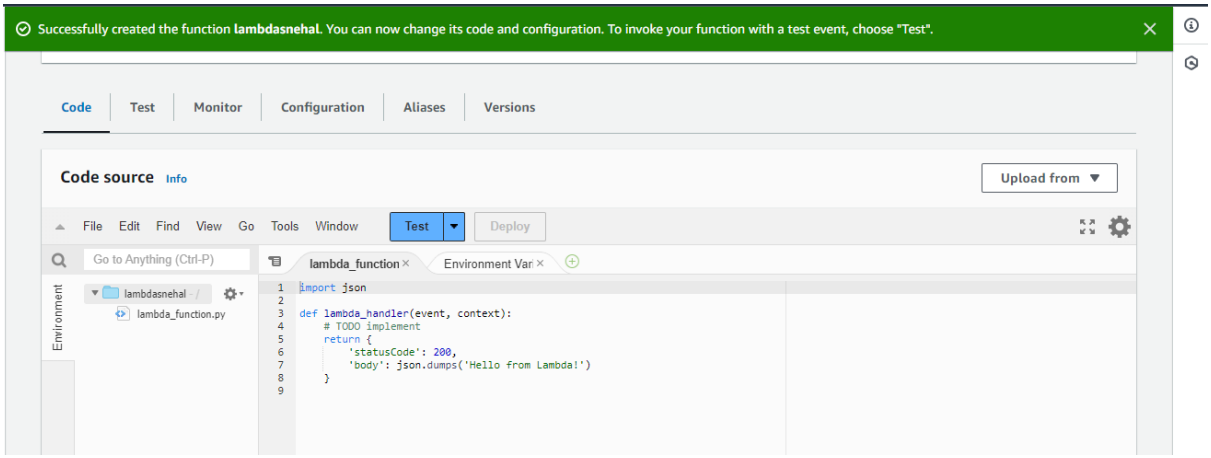




Click on the *Create* button.

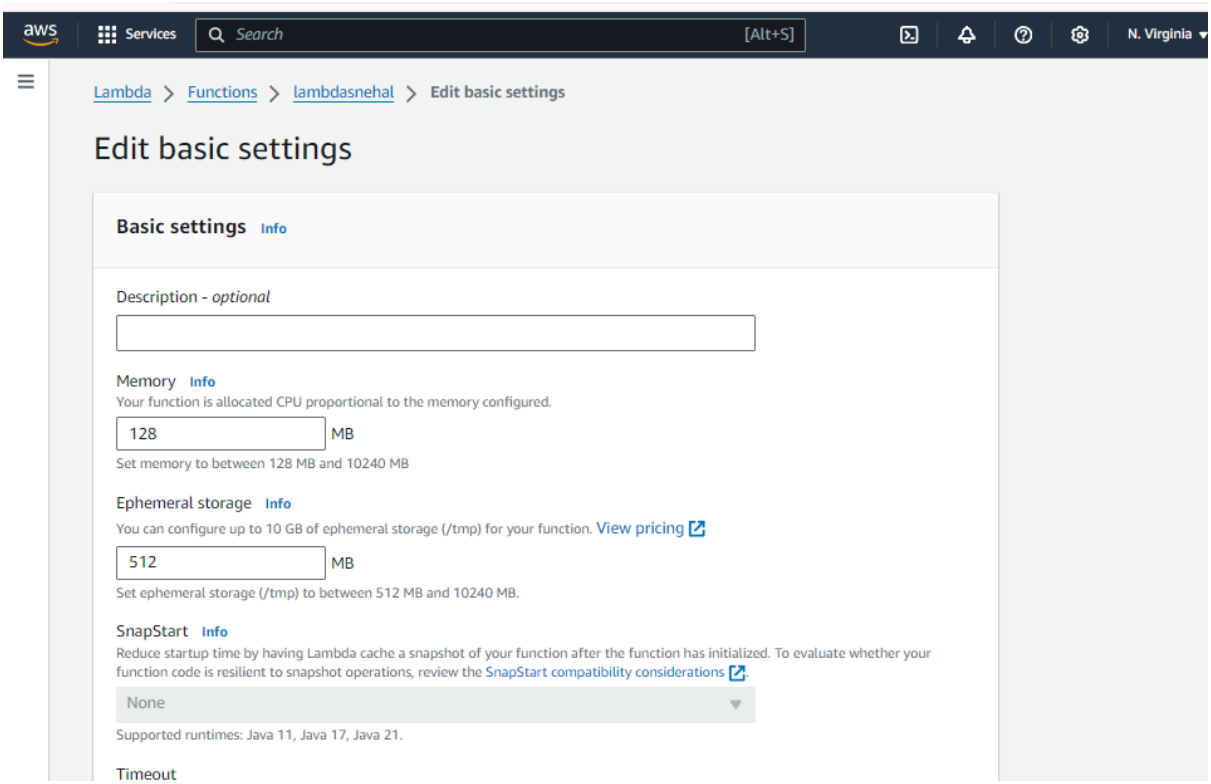
3. This process will take a while to finish and after that, you'll get a message that your function was successfully created.



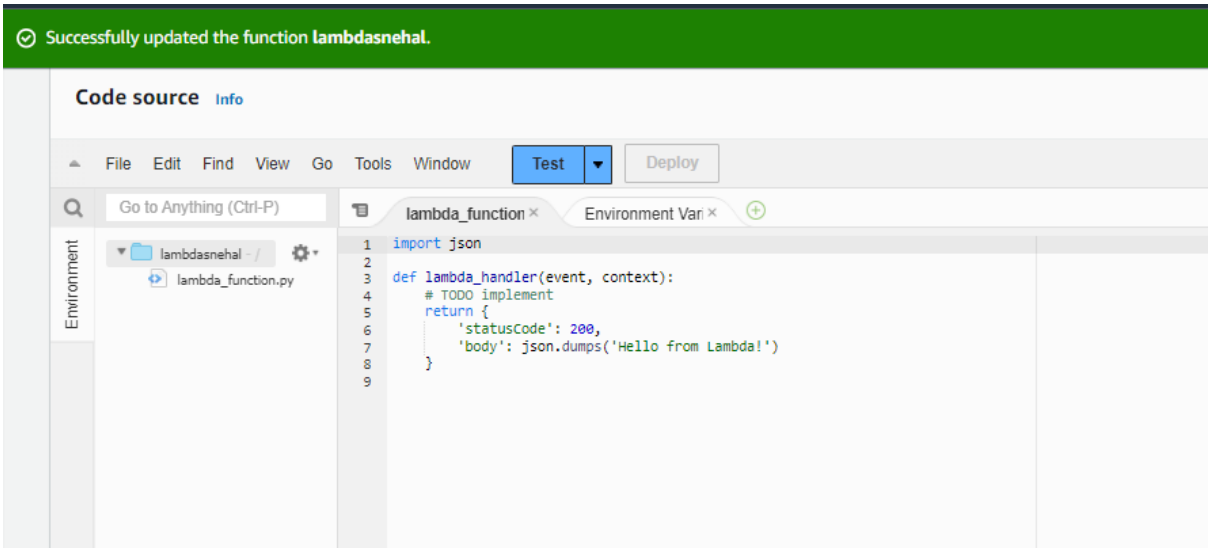


4. To change the configuration, open up the Configuration tab and under General Configuration, choose Edit.

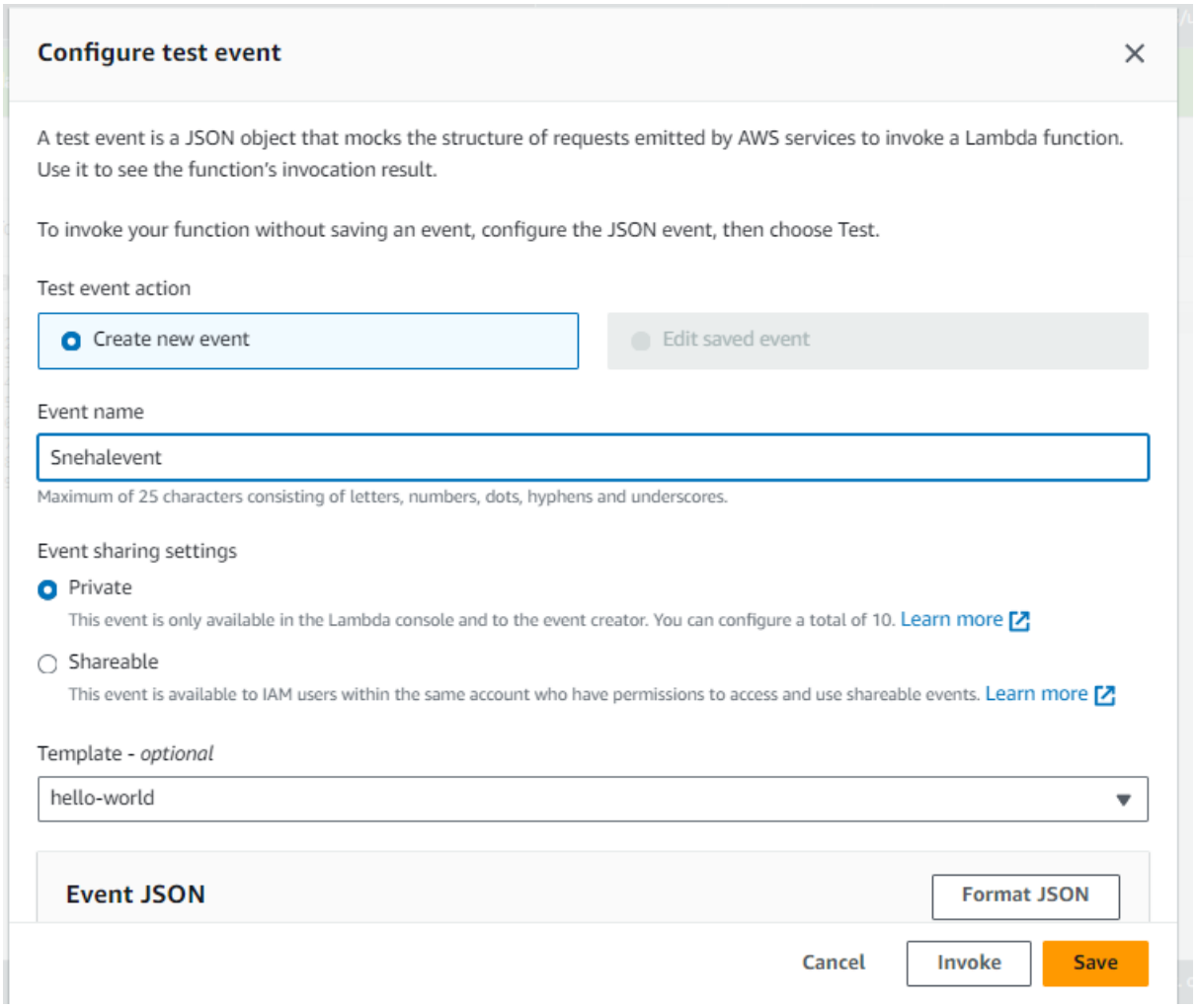
   Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.



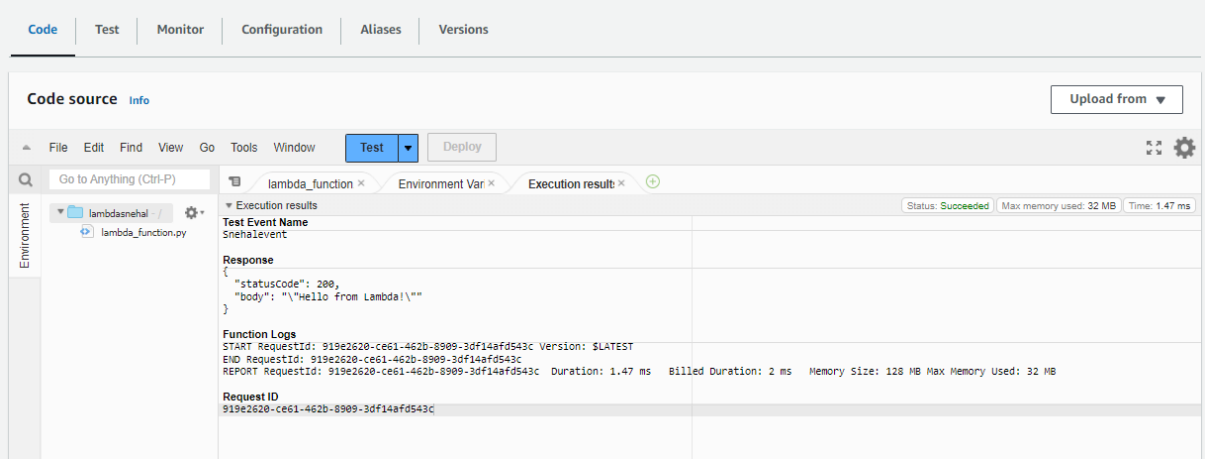
5. You can make changes to your function inside the code editor. You can also upload a zip file of your function or upload one from an S3 bucket if needed.
   Press Ctrl + S to save the file and click Deploy to deploy the changes.



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.



Conclusion:

In conclusion, through this implementation, I learned the fundamentals of AWS Lambda and gained a solid understanding of its workflow and functionalities. By creating my first Lambda functions using Python/Java/Node.js, I was able to see firsthand how serverless architecture operates and its benefits in terms of scalability and efficiency. This hands-on experience has deepened my knowledge and equipped me with the skills to build responsive applications while minimizing infrastructure management. Overall, I feel more confident in leveraging AWS Lambda for future projects and exploring its potential further.

Name: Snehal A. Patil D15A 39

Advanced Devops Lab
Experiment No:12

Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3

Theory:

AWS Lambda and S3 Integration: AWS Lambda allows you to execute code in response to various events, including those triggered by Amazon S3. When an object is added to an S3 bucket, it can trigger a Lambda function to execute, allowing for event-driven processing without managing servers.

Workflow:

1. Create an S3 Bucket:

○ First, create an S3 bucket that will store the objects. This bucket will act as the trigger source for the Lambda function.

2. Create the Lambda Function:

○ Set up a new Lambda function using AWS Lambda’s console. You can choose a runtime environment like Python, Node.js, or Java.

○ Write code that logs a message like “An Image has been added” when triggered.

3. Set Up Permissions:

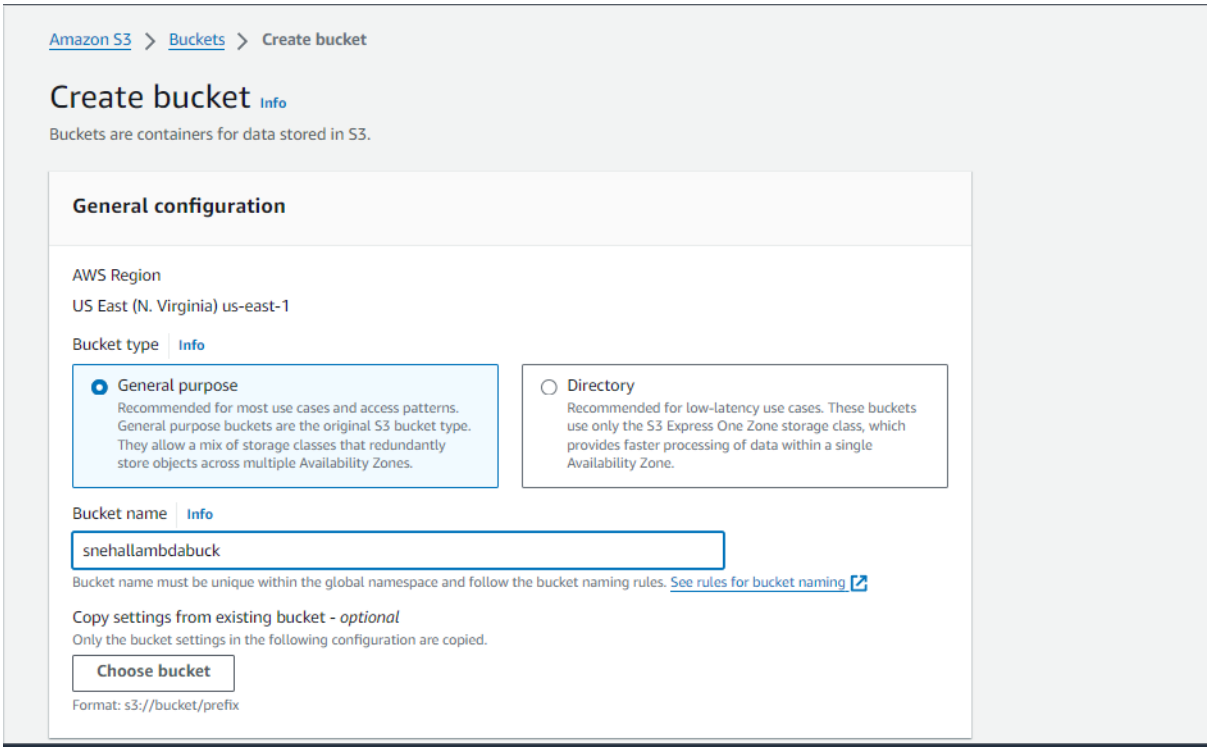
○ Ensure that the Lambda function has the necessary permissions to access S3. You can do this by attaching an IAM role with policies that allow reading from the bucket and writing logs to CloudWatch.

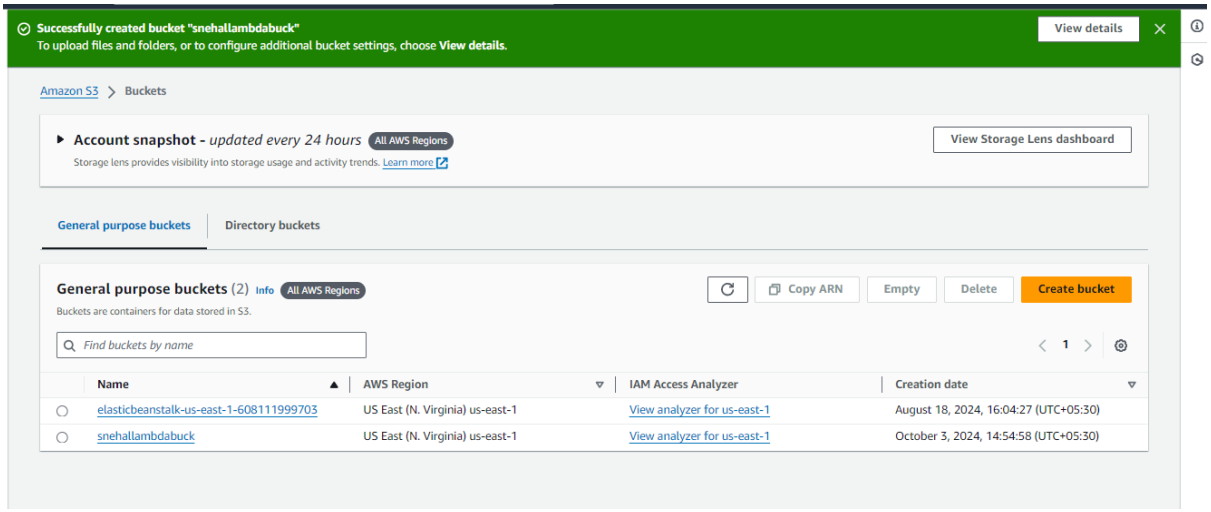
4. Configure S3 Trigger:

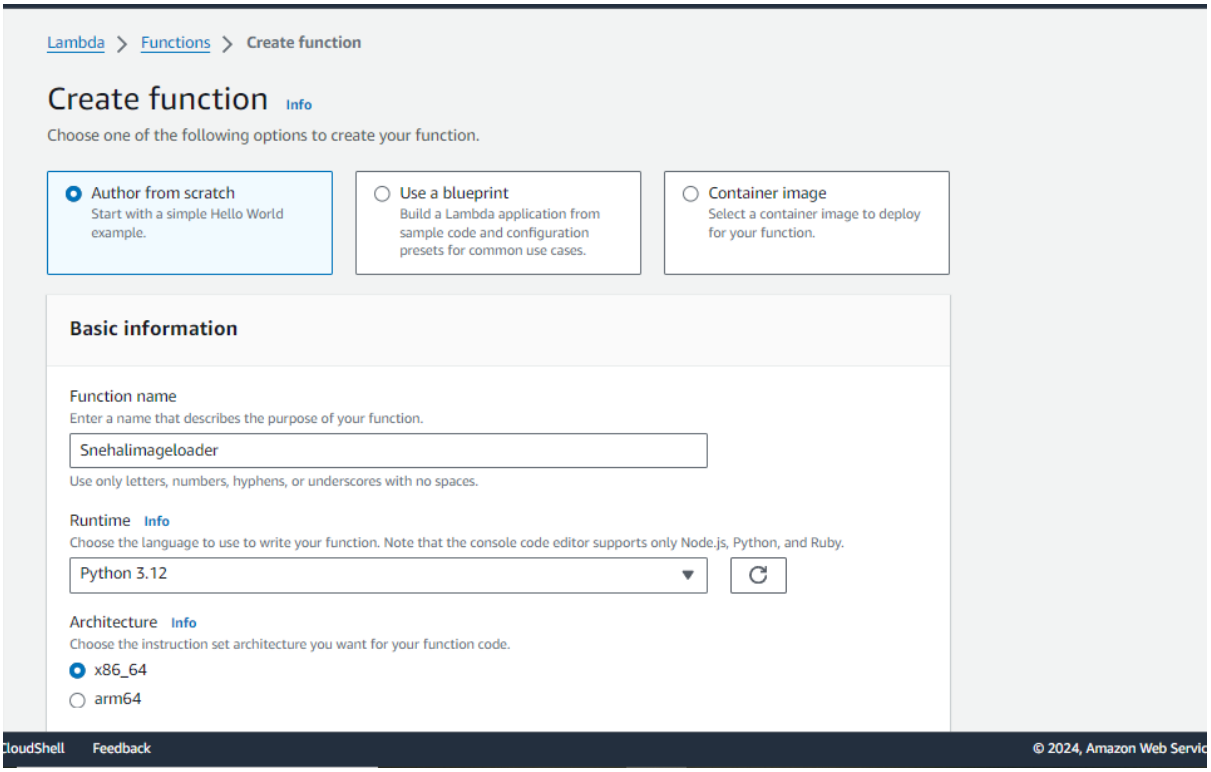
○ Link the S3 bucket to the Lambda function by setting up a trigger. Specify that the function should be triggered when an object is created in the bucket (e.g., when an image is uploaded).

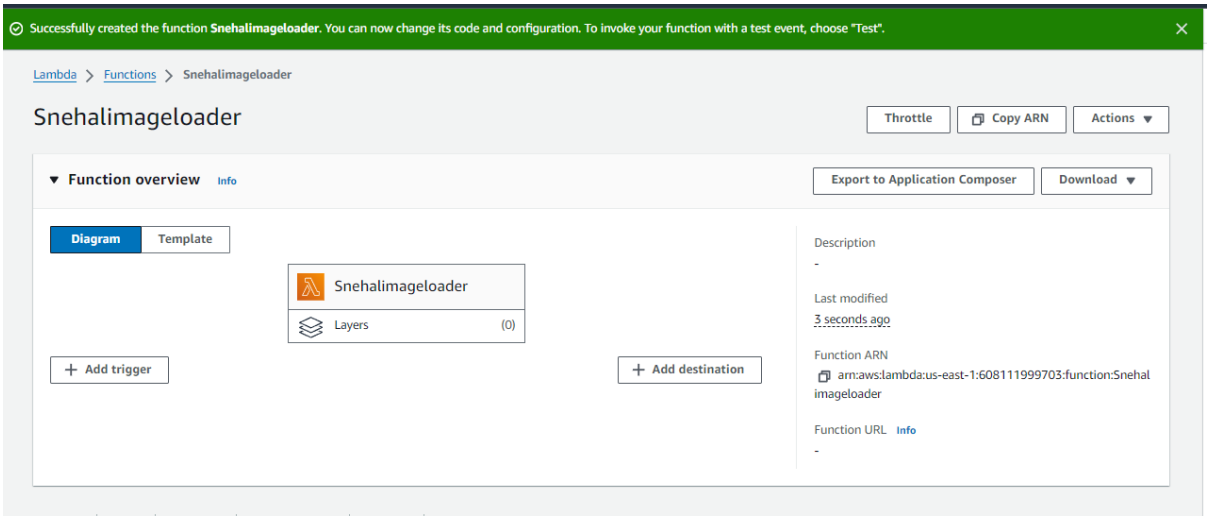
5. Test the Setup:

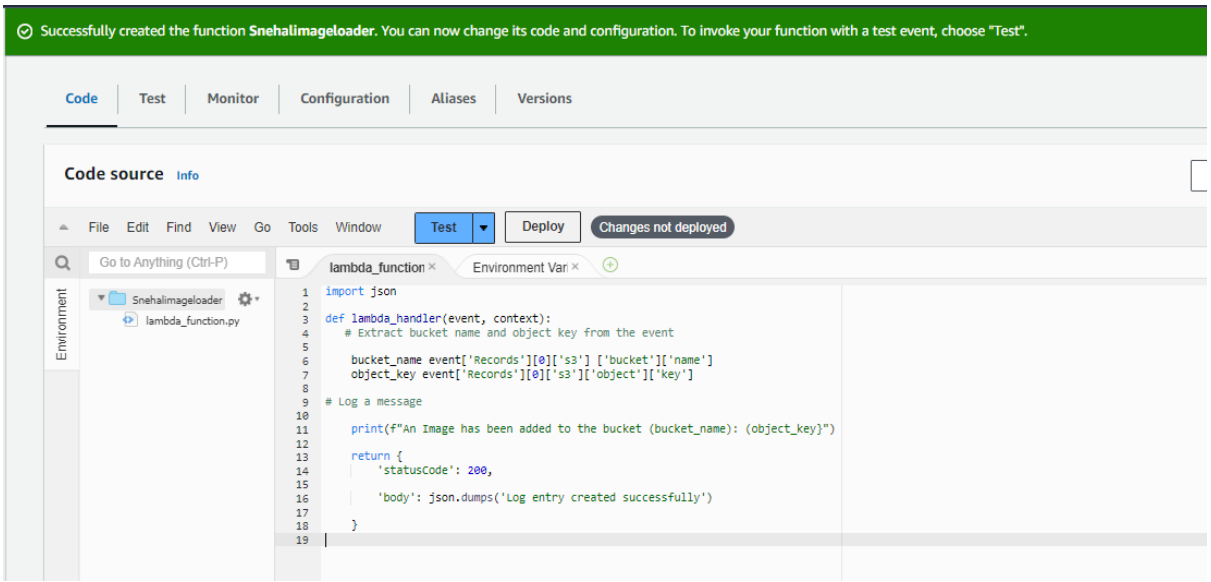
○ Upload an object (e.g., an image) to the S3 bucket to test the trigger. The Lambda function should execute and log the message “An Image has been added” in AWS CloudWatch

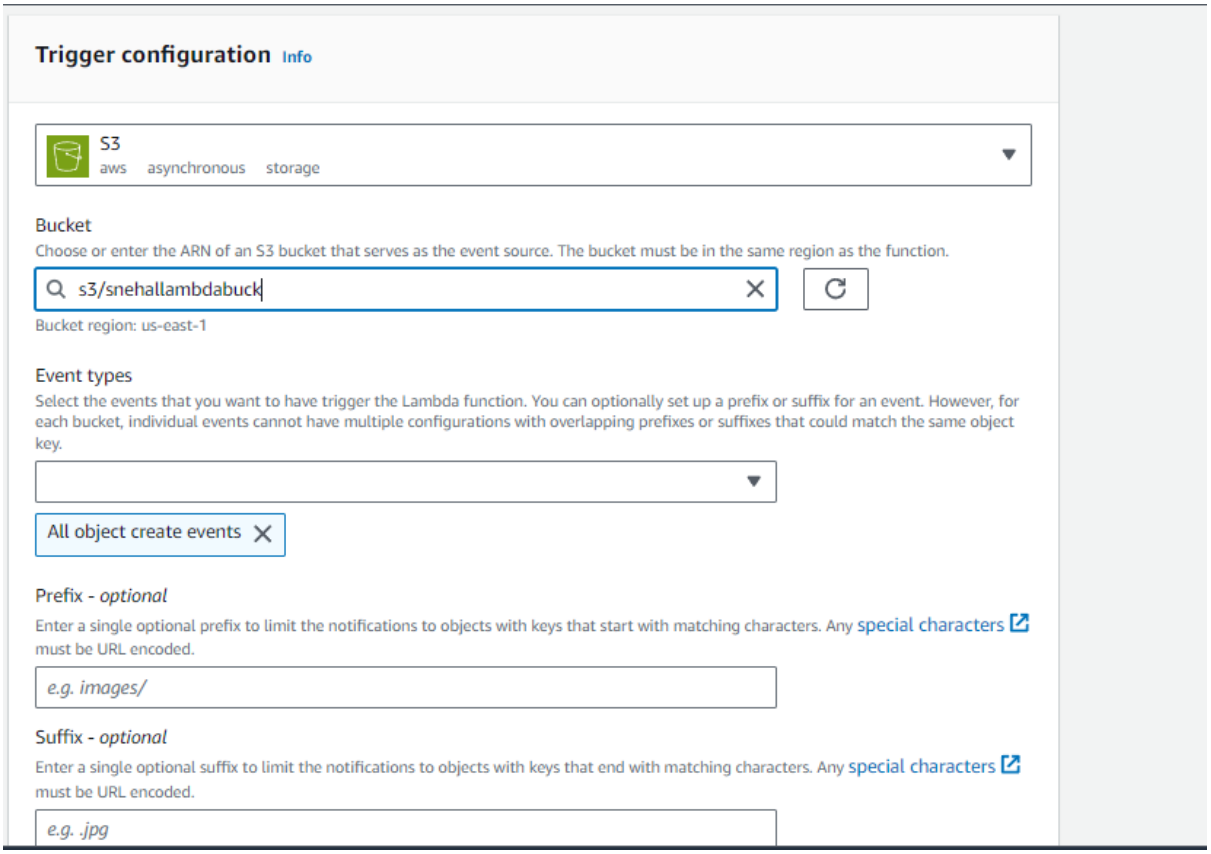

Amazon S3 > Buckets > Create bucket
Create bucket Info
Buckets are containers for data stored in S3.
General configuration
AWS Region
US East (N. Virginia) us-east-1
Bucket type Info
General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.
Bucket name Info
snehallambdabuck
Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming
Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.
Choose bucket
Format: s3://bucket/prefix

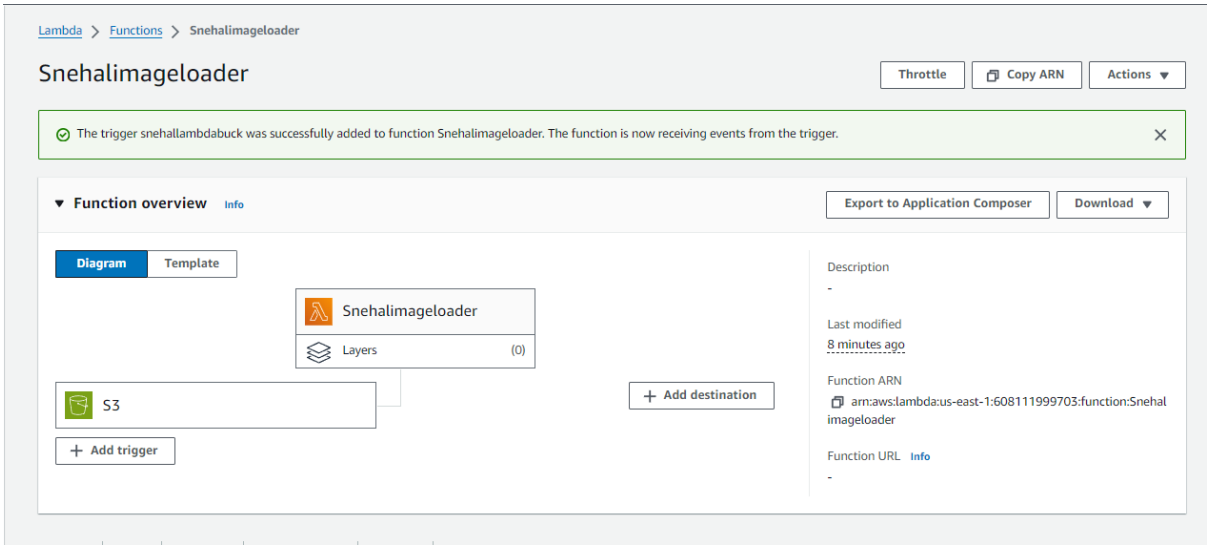

Successfully created bucket "snehallambdabuck"
To upload files and folders, or to configure additional bucket settings, choose View details.
View details
Amazon S3 > Buckets
Account snapshot - updated every 24 hours
All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Learn more
View Storage Lens dashboard
General purpose buckets
Directory buckets
General purpose buckets (2) Info
All AWS Regions
Buckets are containers for data stored in S3.
Copy ARN
Empty
Delete
Create bucket
Find buckets by name
Name
AWS Region
IAM Access Analyzer
Creation date
elasticbeanstalk-us-east-1-608111999703
US East (N. Virginia) us-east-1
View analyzer for us-east-1
August 18, 2024, 16:04:27 (UTC+05:30)
snehallambdabuck
US East (N. Virginia) us-east-1
View analyzer for us-east-1
October 3, 2024, 14:54:58 (UTC+05:30)

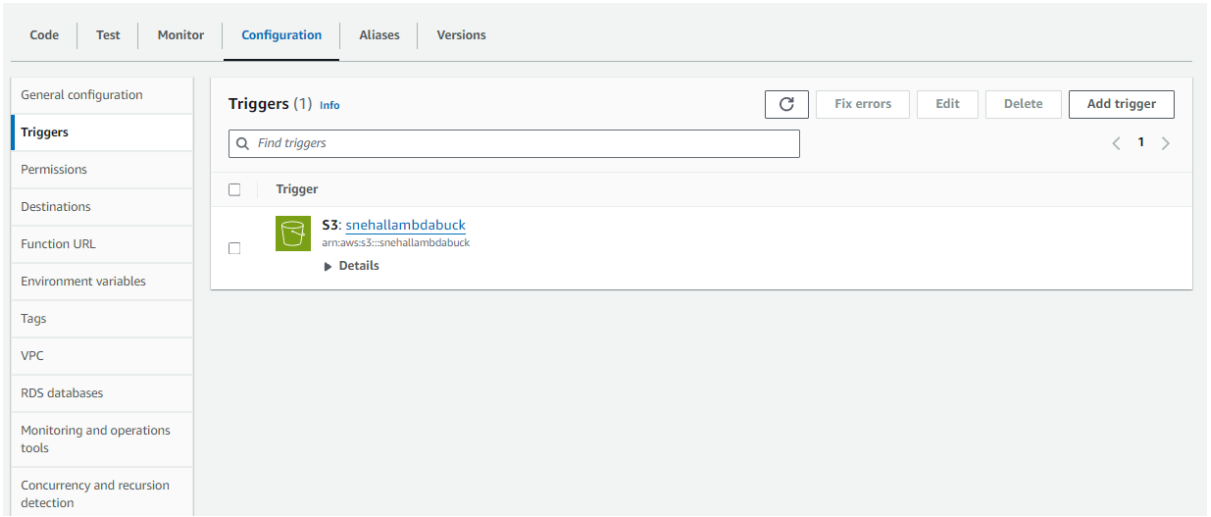

Lambda > Functions > Create function
Create function Info
Choose one of the following options to create your function.
Author from scratch
Start with a simple Hello World example.
Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.
Container image
Select a container image to deploy for your function.
Basic information
Function name
Enter a name that describes the purpose of your function.
Snehalimageloader
Use only letters, numbers, hyphens, or underscores with no spaces.
Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.12
Architecture Info
Choose the instruction set architecture you want for your function code.
x86_64
arm64
CloudShell
Feedback
© 2024, Amazon Web Servic

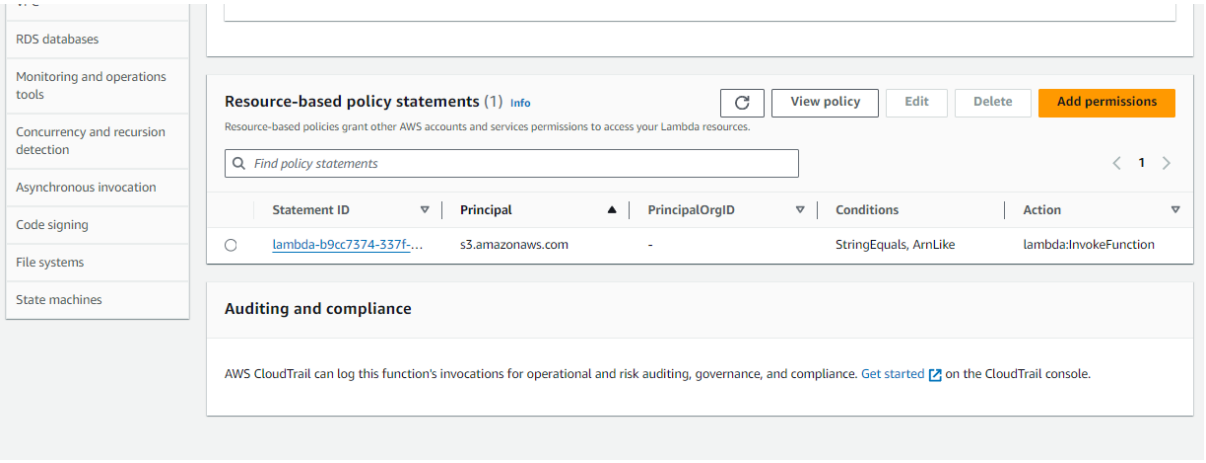

Successfully created the function Snehalimageloader. You can now change its code and configuration. To invoke your function with a test event, choose "Test".
Lambda > Functions > Snehalimageloader
Snehalimageloader
Throttle
Copy ARN
Actions
Function overview Info
Export to Application Composer
Download
Diagram
Template
Snehalimageloader
Layers (0)
Add trigger
Add destination
Description
-
Last modified
3 seconds ago
Function ARN
arn:aws:lambda:us-east-1:608111999703:function:Snehalimageloader
Function URL Info
-

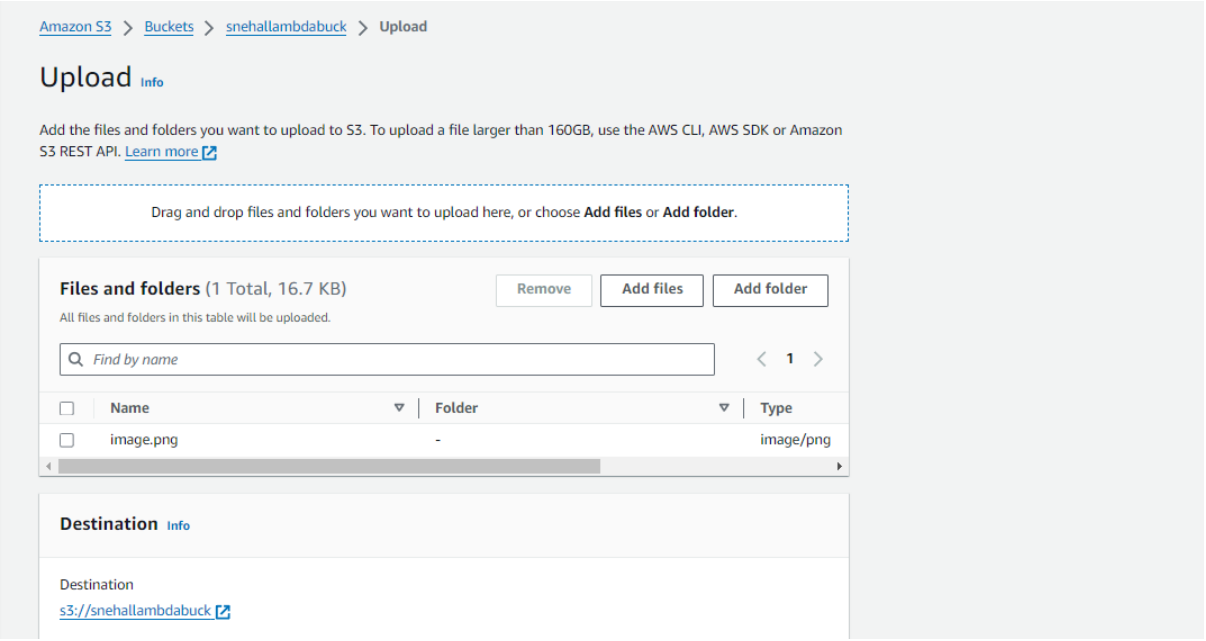

Successfully created the function Snehalimageloader. You can now change its code and configuration. To invoke your function with a test event, choose "Test".
Code
Test
Monitor
Configuration
Aliases
Versions
Code source Info
File Edit Find View Go Tools Window
Test
Deploy
Changes not deployed
Go to Anything (Ctrl-P)
lambda_function
Environment Var
Environment
Snehalimageloader
lambda_function.py
import json
def lambda_handler(event, context):
# Extract bucket name and object key from the event
bucket_name event['Records'][0]['s3'] ['bucket']['name']
object_key event['Records'][0]['s3']['object']['key']
# Log a message
print(f"An Image has been added to the bucket (bucket_name): (object_key)")
return {
'statusCode': 200,
'body': json.dumps('Log entry created successfully')
}

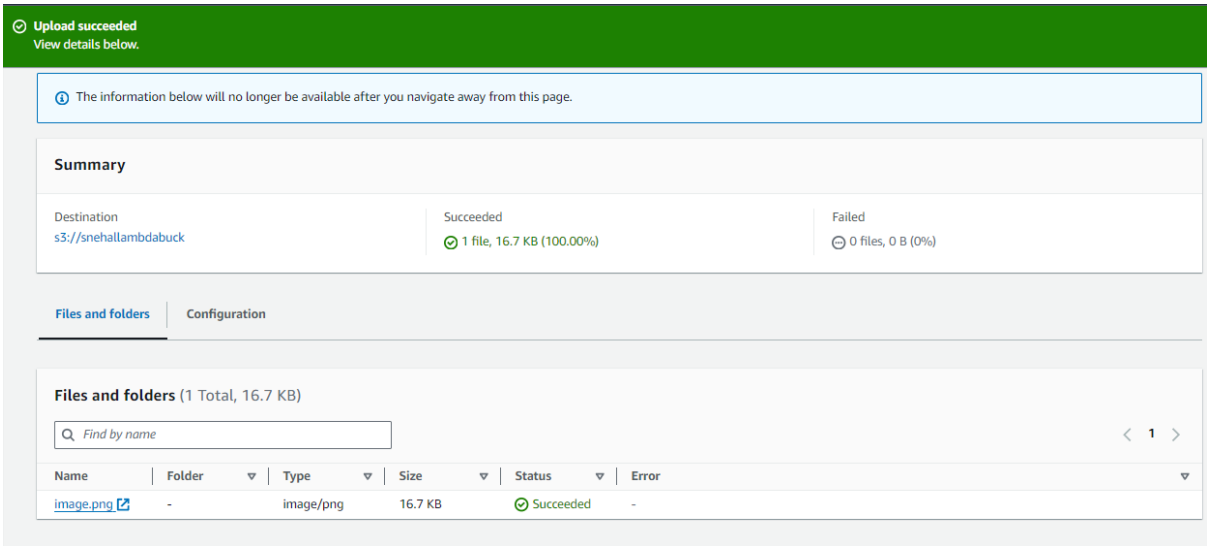

Trigger configuration Info
S3
aws asynchronous storage
Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.
s3/snehallambdabuck
Bucket region: us-east-1
Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.
All object create events
Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any special characters must be URL encoded.
e.g. images/
Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any special characters must be URL encoded.
e.g. .jpg

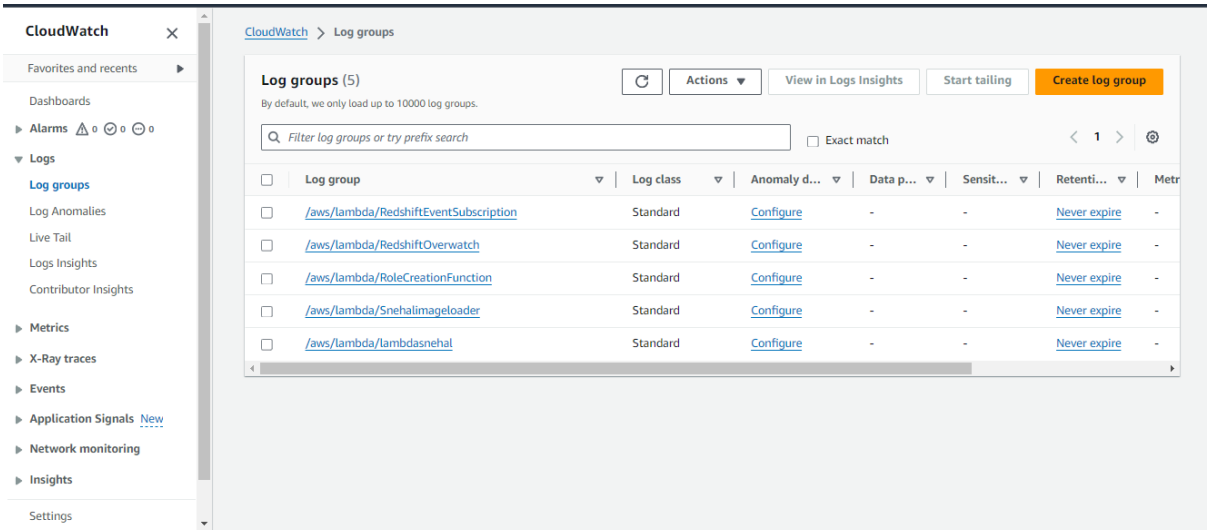

Lambda > Functions > Snehalimageloader
Snehalimageloader
Throttle
Copy ARN
Actions
The trigger snehallambdabuck was successfully added to function Snehalimageloader. The function is now receiving events from the trigger.
Function overview Info
Export to Application Composer
Download
Diagram
Template
Snehalimageloader
Layers (0)
S3
Add trigger
Add destination
Description
-
Last modified
8 minutes ago
Function ARN
arn:aws:lambda:us-east-1:608111999703:function:Snehalimageloader
Function URL Info
-

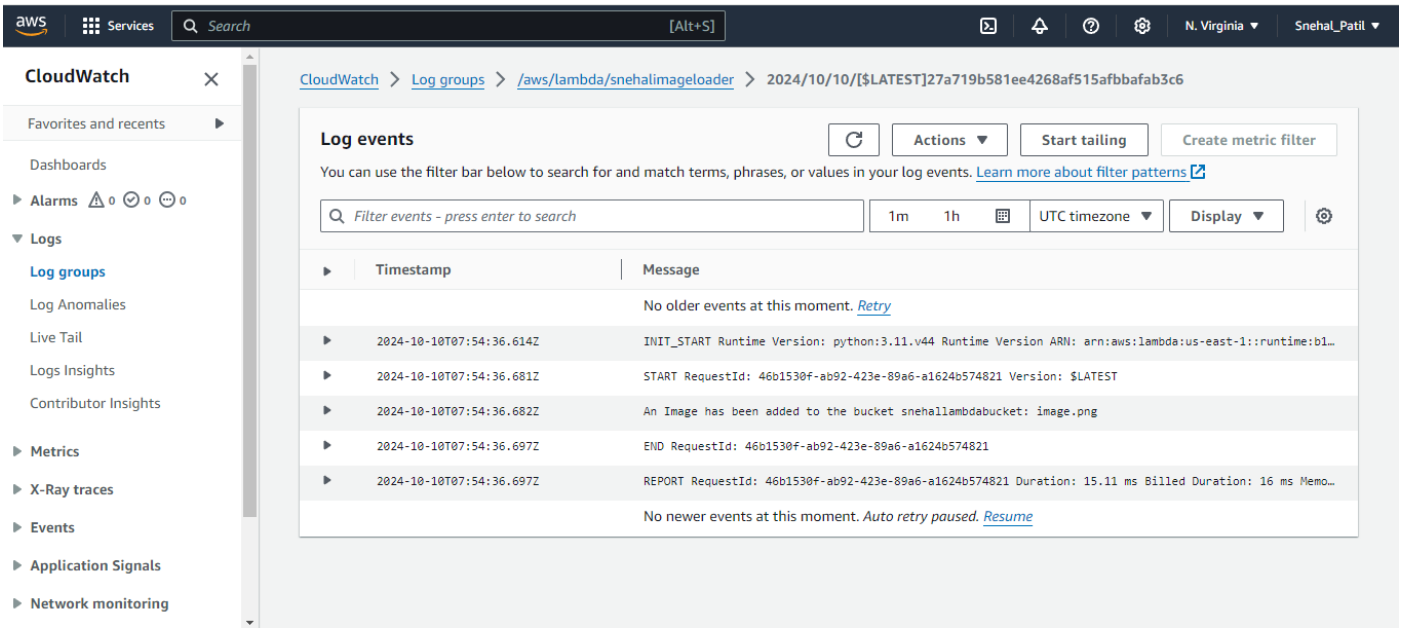

Code
Test
Monitor
Configuration
Aliases
Versions
General configuration
Triggers
Permissions
Destinations
Function URL
Environment variables
Tags
VPC
RDS databases
Monitoring and operations tools
Concurrency and recursion detection
Triggers (1) Info
Fix errors
Edit
Delete
Add trigger
Find triggers
Trigger
S3: snehallambdabuck
arn:aws:s3:::snehallambdabuck
Details


RDS databases
Monitoring and operations tools
Concurrency and recursion detection
Asynchronous invocation
Code signing
File systems
State machines
Resource-based policy statements (1) Info
View policy
Edit
Delete
Add permissions
Resource-based policies grant other AWS accounts and services permissions to access your Lambda resources.
Find policy statements
Statement ID
Principal
PrincipalOrgID
Conditions
Action
lambda-b9cc7374-337f-...
s3.amazonaws.com
-
StringEquals, ArnLike
lambda:InvokeFunction
Auditing and compliance
AWS CloudTrail can log this function's invocations for operational and risk auditing, governance, and compliance. Get started on the CloudTrail console.


Amazon S3 > Buckets > snehallambdabuck > Upload
Upload Info
Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more
Drag and drop files and folders you want to upload here, or choose Add files or Add folder.
Files and folders (1 Total, 16.7 KB)
Remove
Add files
Add folder
All files and folders in this table will be uploaded.
Find by name
Name
Folder
Type
image.png
-
image/png
Destination Info
Destination
s3://snehallambdabuck


Upload succeeded
View details below.
The information below will no longer be available after you navigate away from this page.
Summary
Destination
s3://snehallambdabuck
Succeeded
1 file, 16.7 KB (100.00%)
Failed
0 files, 0 B (0%)
Files and folders
Configuration
Files and folders (1 Total, 16.7 KB)
Find by name
Name
Folder
Type
Size
Status
Error
image.png
-
image/png
16.7 KB
Succeeded
-





Conclusion:

Integrating AWS Lambda with S3 allows for real-time, automated processing of events such as file uploads. In this example, a Lambda function is configured to log a message whenever an image is added to a specific S3 bucket. This setup demonstrates the power and flexibility of serverless computing by automating tasks without requiring manual intervention or server management. By leveraging AWS Lambda, developers can efficiently handle event-driven workflows, reduce operational overhead, and quickly deploy scalable solutions that respond to specific actions within cloud environments.

05/05

Advance DevOps
Assignment No: 2

1. Create a REST API with the Serverless Framework.

Ans: The Serverless Framework is an open source that simplifies the deployment and management of Serverless applications. It allows developers to build and deploy application on cloud platform like AWS without having to manage the underlying infrastructure.

~~Step 1~~:
Prerequisites:
1. Node.js and npm installed on your local machine.
2. Serverless Framework installed globally using npm:
Command: npm install -g serverless.

Step 1: Install the Serverless Framework.
First ensure you have Node.js and npm installed. Then ensure install the serverless Framework globally.

Step 2: create a new serverless Service
serverless create --template aws-nodejs --Path my-service cd my-service.

Step 3: Define function in Serverless.yml
open Serverless.yml file in Project directory.
In this file we define our service Configuratic
including the functions and their triggers/
events.
This file Contains Service, Provider it name
and runtime environment. It also Contains
functions which contains create, read,
update and delete methods.

Step 4: Write Lambda Functions (Handlers)

Open the handler.js file, and write the logic
For the API endPoints.
handler.js Contains the logic For:
- Handling a simple GET request.
- Handling POST request to create a new item

Step 5: Deploy the Service
To deploy the Service and Lambda Functions,
Command: Serverless deploy.
with the 'sls deploy' Command, Serverless
Framework packages your applications,
uploads necessary resources to AW's and
set up the infrastructure.

Step 6: Testing the API:
Once deployed you can test REST API using tools like curl or postman by making POST requests to generated API.

Step 7: Storing data in Dyanmo DB.
To store Submitted Candidate data, you integrate AWS Dynamo DB as a database.

Step 8: AWS IAMpermissions
You need to ensure that Serverless framework is given right permissions to interact with AWS resources like dyanomo DB.

This is the whole process which will create a fully serverless REST API using AWS Lambda, API Gateway the serverless Framework.

Q.2 Case study for Sonarqube:
creating ur own profile in Sonarqube for testing project quality use Sonarqcloud to analyze your Github code.
Install Sonarlint in ur Java intellij ide and analyze Java code Analyze python project with Sonarqube.

Ans: Sonarqube is an open-source platform used for continuous inspection of code quality. It detects bugs, code smells and security vulnerabilities in project across various programming languages.

Soln:
- Create the sonarqube profile for testing project quality.
- Open intellij setting, find Tools > SonarLint entry and select + to open connection wizard
- Enfr a name for this Connection, select sonarcloud or sonarqube.
- Choose the authentication method.

(a) Generate token on sonarqube or Sonarcloud
(b) username + password: This can be used only Sonarqube connection only.

- For Sonarcloud only select organisation that you want to connect.
- Sonarqube and Sonarcloud can push notification to developers.

- Validate the connection creating by selecting Finishing at the end of the wizard.
- Save the connection in global setting by clicking OK.

- BIND PYTHON PROJECT TO SONARQUBE.
- Select SonarLint > Bind Project to Sonarqube/ Sonarcloud.
- Choose the Correct Projects from Sonarqube.
- Analyze the Project. (Python Project)
- Trigger an analysis by going to Code > Analyze Code > SonarLint.

- Analyze Node.js Projects.
- Make sure your Node.js Project is properly Configured with Sonar-Project. Properties file or equivalent for the analysis to run.

Q.3 At large organization, your centralized operations team may get many repetitive infrastructure requests. You can use Terraform modules 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 organisation, allowing teams to efficiently deploy services in compliance with your organisation's practices. Terraform Cloud can also integrate with ticketing systems like Service Now to automatically generate new infrastructure requests.

Ans: Self-service

Self--serve infrastructure Model with Terraform Modules:

At a large organisation, implementation of a self-serve infrastructure model using Terraform can significantly streamline the process of managing infrastructure across different teams. This approach allows product teams to manage their own infrastructure independently while adhering to organizational standards and best practices.

Key aspects of this self-serve model include:

(a) Standardization through Terraform Modules: By creating and utilizing Terraform modules, organizations can codify their infrastructure deployment and management standards. These modules serve as reusable packages of Terraform configurations that encapsulate common patterns and best practices.

(b) Efficient Deployment: Product teams can leverage these standardized modules to quickly deploy services without needing

to reinvent the wheel or wait for the centralized operation team to handle every request.

(c) Compliance
By using predefined modules, teams, ensure that their deployments comply with the organisation's established practices and security gridelines.

(d) Automation: The use of Terraform modules promotes automation. reducing manual intervention and potential human errors in infrastructure management.

(e) Version Control: With modules stored in version control system like Git, teams can track changes, collaborate on improvements, and maintain a history of infrastructure configurations.

INTEGRATION WITH TICKETING SYSTEMS:
terraform cloud offer integration capabilities that further enhance the self-serve model:
(a) Automatic infrastructure Request: Terraform cloud can integrate with ticketing systems like service Now to automatically generate new infrastructure requests. This automation streamlines the process of submitting and tracking infrastructure changes.

(b) Centralized Management: By Centralizing infrastructure management through terrafor Cloud organisation can maintain better control over who can request and approve infrastructure changes.

(c) Goverance: The integration with ticketing syste allows for better goverance of infrastrecture requests, ensuring that all changes go through proper approval processes before deploymen

Collaborative Infrastructure Management.

a) Team Based Performance Permissions.
b) State and Run History.
c) Sensitive Information Protection.
d) Module Registry.

• By implementing these features and practic large organisations can effectively leverage Terraform to build a robust, scalable and Complaint infrastructure management system that supports both Centralized Control and decentralised team Autonomy.

Date
Page

• In large organizations using Terraform modules to build a self-serve infrastructure model allows product teams to manage their own infrastructure independently, reducing repetitive requests to the centralized operations team. Terraform modules codify standards for deploying & managing services, ensuring consistency, compliance and security. By Integrating with ticketing systems like ServiceNow, infra-structure requests can be automated, further streamlining deployment processes. This model increases efficiency, scalability & flexibility, enabling teams to deploy infrastructure quickly while maintaining governance and control.

Benefits of Self-serve infrastructure Model:

1. Codifying standards with Terraform modules:
Terraform modules, which are reusable configurations that define cloud resources, can serve as blue-prints that adhere to your organization's best practices. teams are guided in deploying consistent & compliant services across various environments.

2. Automation and Autonomy:
Once these modules are created, product tean
Can autonomously use them to deploy resourc
without needing approval from the centrali
ops team for every change.

3. Integration with Ticketing Systems:
Integrating this self-service model with
systems like ServiceNow can automate
the generation of infrastructure reques
This reduces manual overhead and
Standardizes the process for new
infrastructure.

Adopting a self-service modules
streamlines infrastructure management in
large organizations. It empowers prod
teams, ensures consistency, & automate
repetitive tasks while maintaining
governance & control. This allows
organizations to scale infrastructure
management while maintaining a high
level of operational excellence.

Teacher's Sign.:

Name: Snehal A. Patil
Class: D15A
ROLL NO: 39

05/05

ASSIGNMENT NO: 1

Q. 1 Use S3 bucket and host video streaming. Give proper each and every step for this in short but all the steps.

Ans: Step 1: create an S3 bucket

1. Sign in to AWS management console
2. Navigate to S3:
   - In AWS Management Console, select S3.
3. Create a Bucket
   - Click on create bucket
   - Enter a unique bucket name.

step 2: Upload video to S3 bucket:

1. Open your Bucket by clicking on bucket name your created.
2. Upload Files.
   - Click on upload.
   - Drag and drop your files and click upload.
3. Set Permissions:
   - For public access under permissions, check Grant Public read access.

step 3: create a CloudFront Distribution.

1. Navigate to CloudFront from aws console.
2. click on create distribution.
   - choose web as delivery method.

3. Configure the distribution:
- Origin Domain name: Select Your S3 bucket
- Viewer protocol Policy: Choose Redirect HTTP to HTTPS for secure access
- Cache Behaviour settings: Configure Caching
- Click create distribution.

Step 4: Configure CloudFront For secure Access:
1. create an origin access Identity (OAI)
- In CloudFront origin console, go to distribution settings.
- Under origins and origin group, click Edit.
- create a new origin access Identity.
2. Update S3 bucket policy.
- Go to your S3 bucket
- Click on permission & then bucket Policy.
- Add to policy to grant access to OAI.

Step 5: Access the video through CloudFront
1. Get the CloudFront URL
- In CloudFront Console, Go to your distribution.
- Copy the Domain Name.
2. Use the URL.
- use this URL in your web application to stream the video.

Q: 2 Discuss BMW and Hot Star case studies using AWS.

Ans: BMW Case Study:

Overview: BMW leverages AWS to enhance its digital transformation, focussing on innovation in connected vehicles and improving operational efficiency.

Key points:

1. Data Analytics:

- BMW uses AWS for big data analytics, enabling real-time processing of vehicle data. This allows for predictive maintenance and improved customer service.

2. Cloud Infrastructure:

- By migrating to AWS, BMW benefits from scalable and flexible cloud infrastructure, reducing IT costs and enhancing agility.

3. Connected vehicles:

- AWS supports BMW's connected car initiatives, allowing for seamless integration

of services like navigation, entertainment, and remote diagnostics.

4. Security and Compliance:
- AWS provides robust security measures that help BMW maintain compliance with automotive industry standards.

Hot Star Case study:
Overview: Hot Star, a popular streaming service in Asia, utilizes AWS to deliver high-quality content to million of users.

Key Points:

1. Scalability:
- During peak events (like sports finals), Hot Star scales its infrastructure dynamically with AWS services to handle massive spikes in user traffic.

2. Content Delivery:
- The use of Amazon CloudFront enhances the delivery of streaming content globally, ensuring low latency and high availability.

3. Machine Learning:

- Hot Star employs AWS machine learning services to personalize user experiences and optimize content recommendations.

4. Cost Management:

- By using AWS's pay-as-you-go model, Hot Star manages operational costs effectively, aligning expenses with user demand.

Conclusion:

Both BMW and Hot Star demonstrate how AWS can drive innovation and operational efficiency in different industries. BMW focuses on enhancing connected vehicle experiences and data management, while Hot Star emphasizes scalability and content delivery for a superior streaming experience. Their successful integration of AWS highlights the platform's versatility and robustness in meeting diverse business needs.

Q. 3 Why Kubernetes and advantages and disadvantages of Kubernetes. Explain How adidas uses Kubernetes.

Ans: Kubernetes is an open-source container orchestration platform designed to automate deploying, scaling and managing Containerized applications.

Advantages of Kubernetes:

1. Scalability: Automatically adjusts resources based on demand.
2. Portability: Consistent performance across cloud and on-premises environments.
3. High Availability: Self-heals and ensures uptime.
4. Load Balancing: Distributes traffic effectively.

Disadvantage Of Kubernetes:

1. Complexity: Steep learning curve and setup time.
2. Resource Intensive: Requires significant computing resources.
3. Operational Overhead: Needs Continuous management and monitoring.
4. Networking Challenges: Complicated Configurations Can be tricky to troubleshoot.

How Adidas Uses Kubernetes:

Adidas uses Kubernetes to scale their e-Commerce platform globally. It allows them to manage microservices efficiently. By adopting a microservices architecture with Kubernetes, Adidas can handle high-traffic events like product launches with ease through auto-scaling, ensuring reliable performance. Kubernetes also supports their Continuous Integration / Continuous Deployment (CI/CD) pipelines, allowing for faster updates and feature rollouts without downtime. Its self-healing capabilities ensure minimal service disruptions, & the platform's flexibility allows Adidas to implement a multi-cloud strategy, optimizing their infrastructure across various cloud providers.

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

Ans: Nagios is an open-source monitoring tool used to oversee systems, networks, & services. It helps detect issues by continuously monitoring resources like servers, applications, and network devices. When problems occur, Nagios sends alerts to administrators, enabling quick action to

prevent downtime or performance degradation.

Key Features of Nagios:

1. Monitoring of Network Services: Nagios monitors services such as HTTP, FTP, SMTP, etc.
2. Monitoring of Host Resources: CPU usage, memory, disk space, etc., can be tracked for servers and network devices.
3. Alerting System: When critical thresholds are reached, Nagios sends alerts via email, SMS.
4. Web Interface: It offers a web-based interface for viewing system statuses, logs, & trends.
5. Scalability: It can be used to monitor both small & large infrastructures, thanks to its Architecture.

How Nagios is used in E-Services:

1. Uptime Monitoring: It monitors the availability of e-services, ensuring that websites, Payment gateways, & other critical components remain operational.
2. Performance Monitoring: Nagios tracks the performance of server resources, databases, and networks, ensuring that service are delivered efficiently to users.

3. Incident Detection: If there is a system failure, service outage or performance degradation, Nagios instantly detects the issue & alerts the IT team to take corrective action.

4. Security Monitoring: Nagios can track suspicious activities, detect unauthorized access, & monitor the health of security systems, helping in safeguarding e-services from cyber threats.

5. Capacity Planning: By monitoring resource usage trends over time, Nagios helps in planning upgrades or resource allocation to avoid slowdowns or outages.