Advanced DevOps Experiment-1

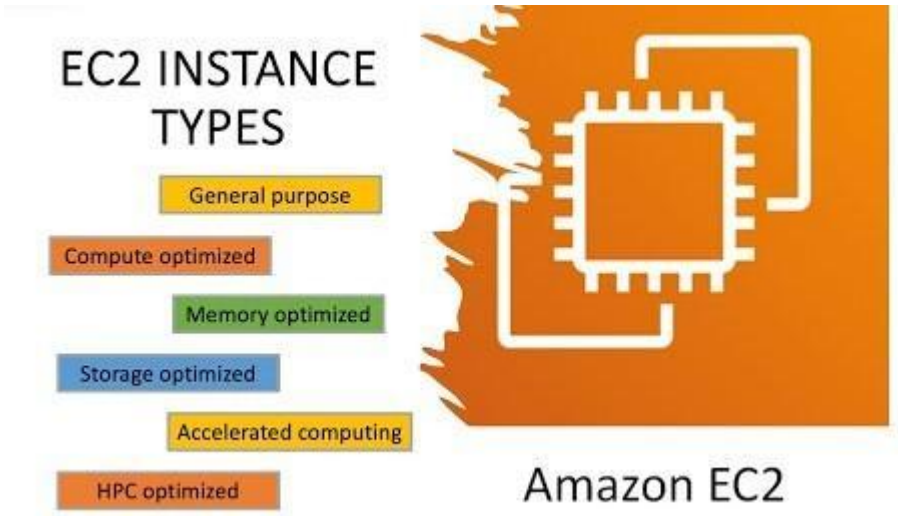
Aim: Using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

Theory:-

Amazon Elastic Compute Cloud (Amazon EC2) offers flexible, on-demand computing capacity in the Amazon Web Services (AWS) Cloud, enabling faster development and deployment of applications without the need for upfront hardware investments. EC2 allows you to launch and manage virtual servers based on your workload needs. You can adjust security settings, configure networking, and manage storage efficiently.

EC2's scalable nature lets you easily increase capacity (scale up) to handle compute-intensive tasks such as data analysis, seasonal traffic spikes, or resource-heavy applications. Conversely, when demand decreases, you can reduce capacity (scale down), ensuring cost-effective resource management.

Each EC2 instance is a virtual server, and the instance type you select determines the combination of compute, memory, network, and storage resources available. This flexibility ensures you can tailor your environment to meet specific performance and cost requirements. For further details, refer to the Amazon EC2 Instance Types Guide.



Key Features of Amazon EC2

Amazon EC2 offers a range of powerful features to enhance your cloud computing experience:

Instances: Virtual servers that you can configure and launch to meet your application needs.

Amazon Machine Images(AMIs): Preconfigured templates containing the necessary components for your server, such as the operating system and additional software.

Instance Types: A variety of configurations offering different combinations of CPU, memory, storage, networking capacity, and graphics hardware, allowing you to choose the best fit for your workloads.

Amazon EBS Volumes: Persistent storage solutions using Amazon Elastic Block Store (Amazon EBS) that provide durable storage for your data.

Instance Store Volumes: Temporary storage that offers high-performance local storage for your instance, deleted automatically when the instance is stopped, hibernated, or terminated.

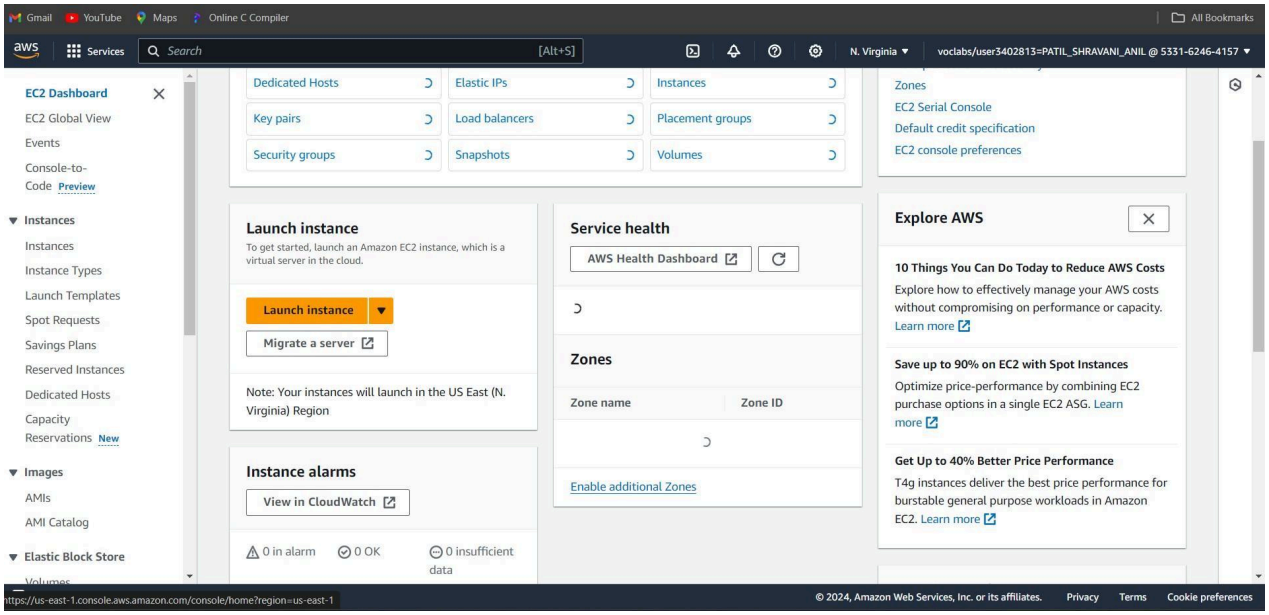
Key Pairs: Secure login credentials for your instances, with AWS storing the public key and you keeping the private key in a secure location.

Security Groups: Virtual firewalls that enable you to control inbound and outbound traffic by specifying allowed protocols, ports, and IP ranges.

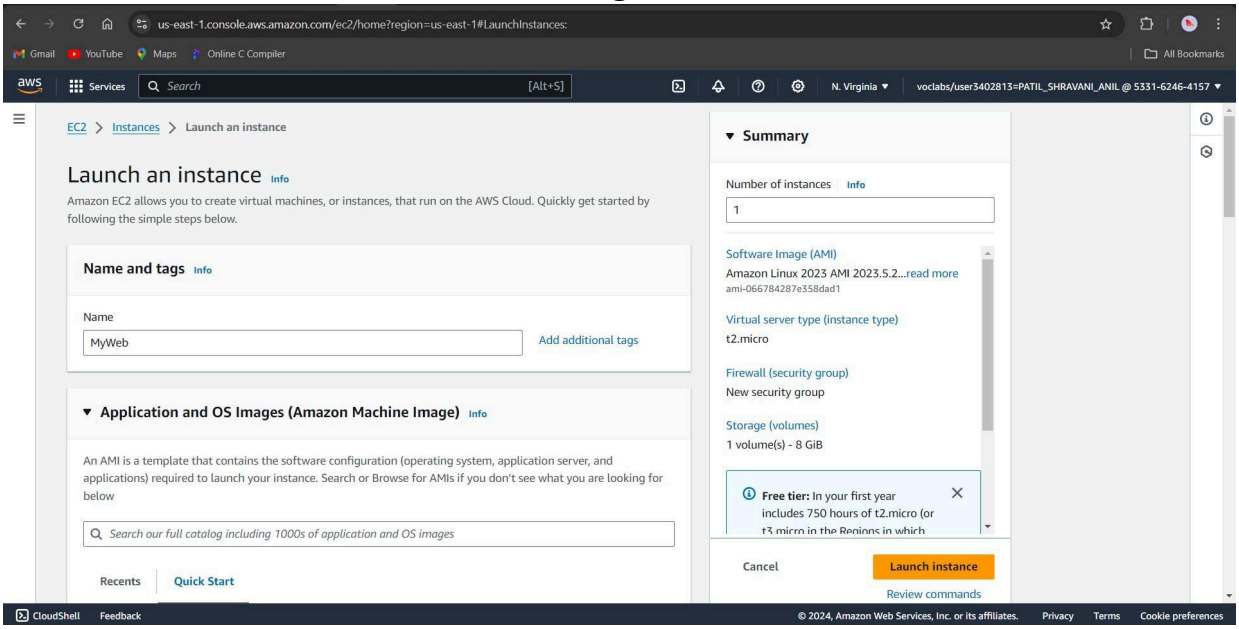
PCI DSS Compliance: Amazon EC2 supports the secure processing, storage, and transmission of credit card data, adhering to the Payment Card Industry Data Security Standard (PCI DSS). For further details and to request the AWS PCI Compliance Package, refer to the PCI DSS Level 1 guidelines.

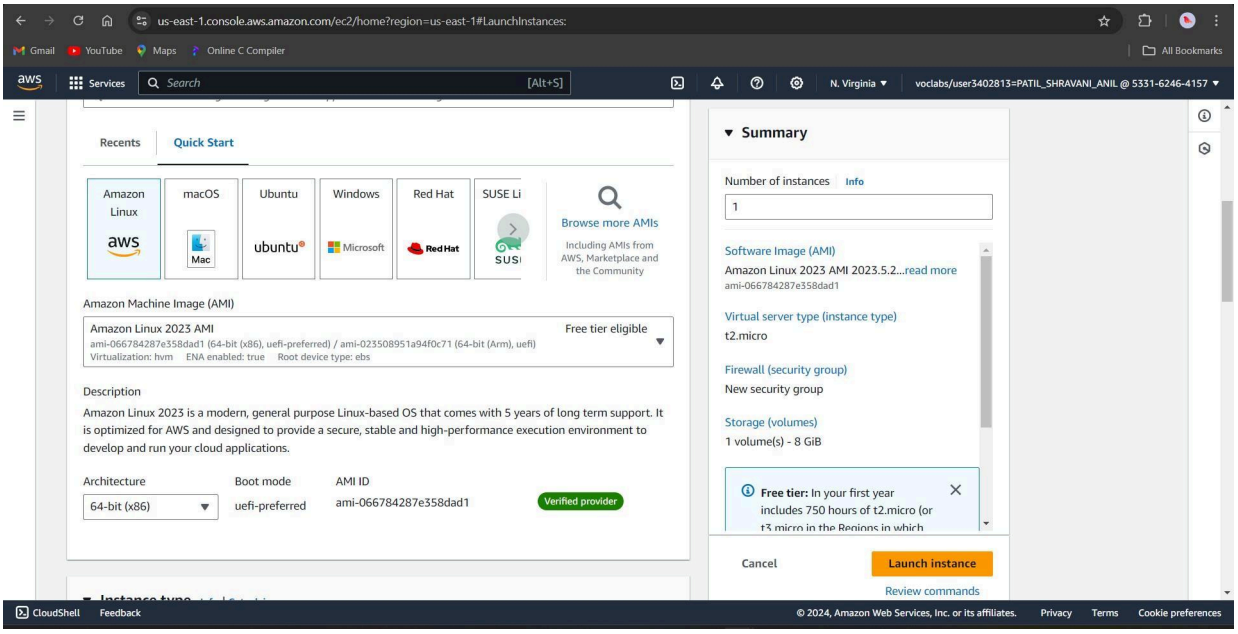
These features provide a robust and secure foundation for deploying, managing, and scaling applications in the cloud Implementation:-

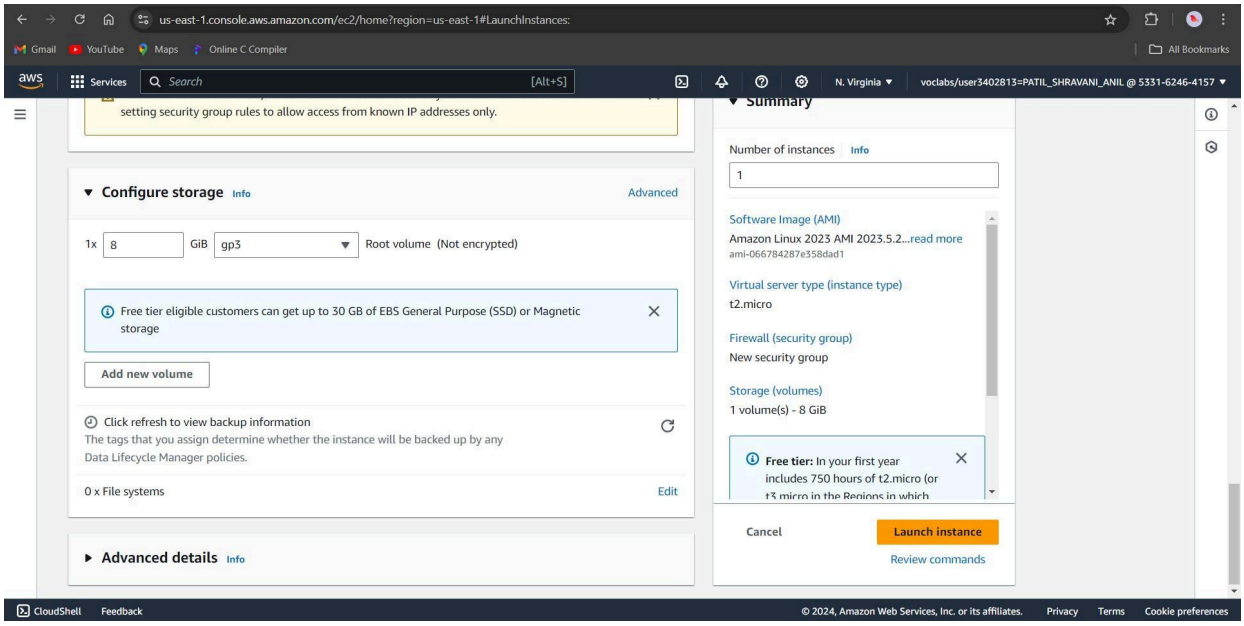
1. Open AWS Academy and select launch instance



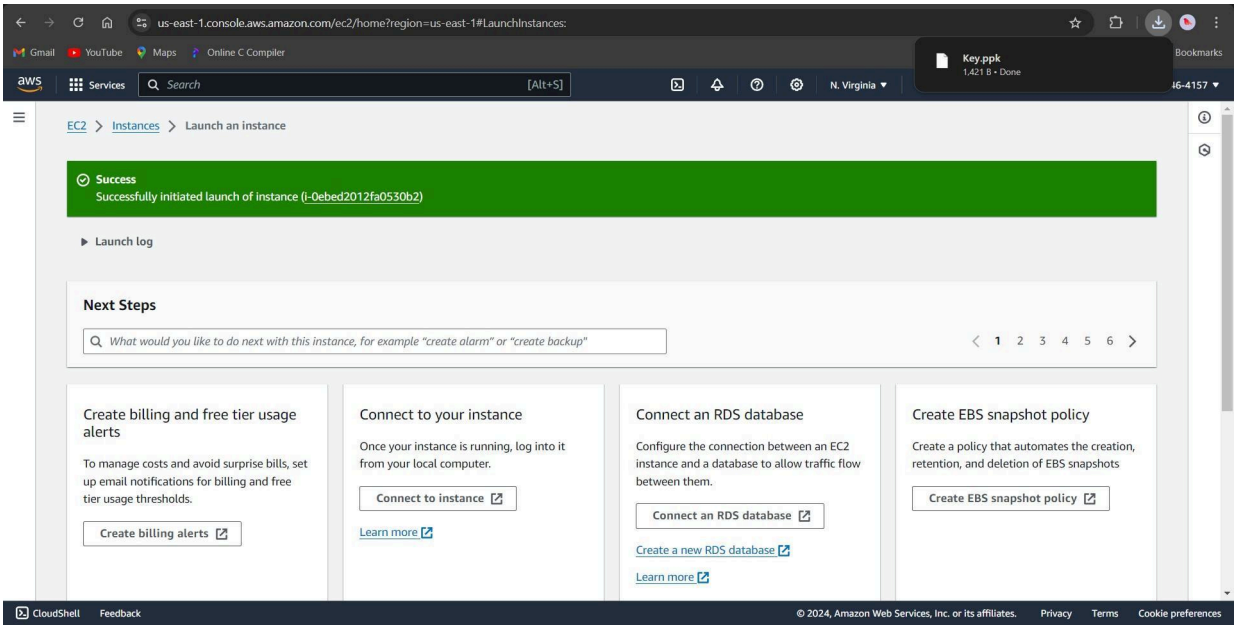
2.Enter name of the instance , add other configurations and click on launch instance .

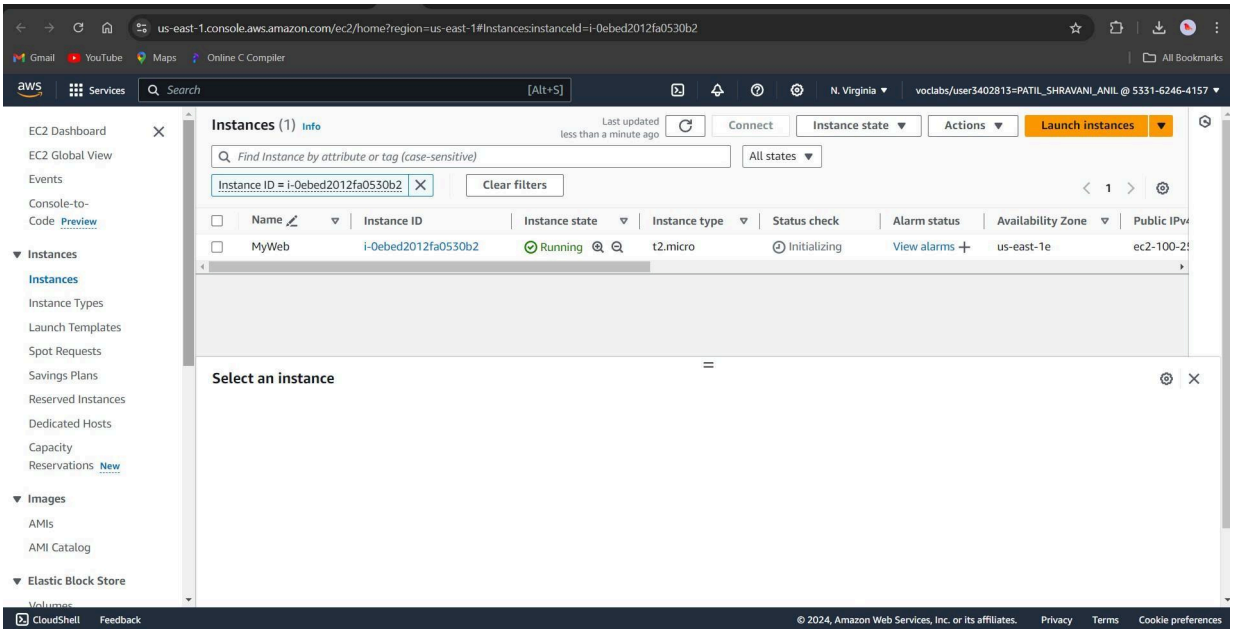



us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
Gmail
YouTube
Maps
Online C Compiler
All Bookmarks
aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3402813=PATIL_SHRAVANI_ANIL @ 5331-6246-4157
Recents
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE Li
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)
Amazon Linux 2023 AMI
ami-066784287e358dad1 (64-bit (x86), uefi-preferred) / ami-023508951a94f0c71 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs
Free tier eligible
Description
Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.
Architecture
64-bit (x86)
Boot mode
uefi-preferred
AMI ID
ami-066784287e358dad1
Verified provider
Summary
Number of instances Info
1
Software Image (AMI)
Amazon Linux 2023 AMI 2023.5.2...read more
ami-066784287e358dad1
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or
Cancel
Launch instance
Review commands
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
Gmail
YouTube
Maps
Online C Compiler
All Bookmarks
aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3402813=PATIL_SHRAVANI_ANIL @ 5331-6246-4157
setting security group rules to allow access from known IP addresses only.
Configure storage Info
Advanced
1x
8
GiB
gp3
Root volume (Not encrypted)
Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage
Add new volume
Click refresh to view backup information
The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.
0 x File systems
Edit
Advanced details Info
Summary
Number of instances Info
1
Software Image (AMI)
Amazon Linux 2023 AMI 2023.5.2...read more
ami-066784287e358dad1
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or
Cancel
Launch instance
Review commands
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

3. Instance successfully launched

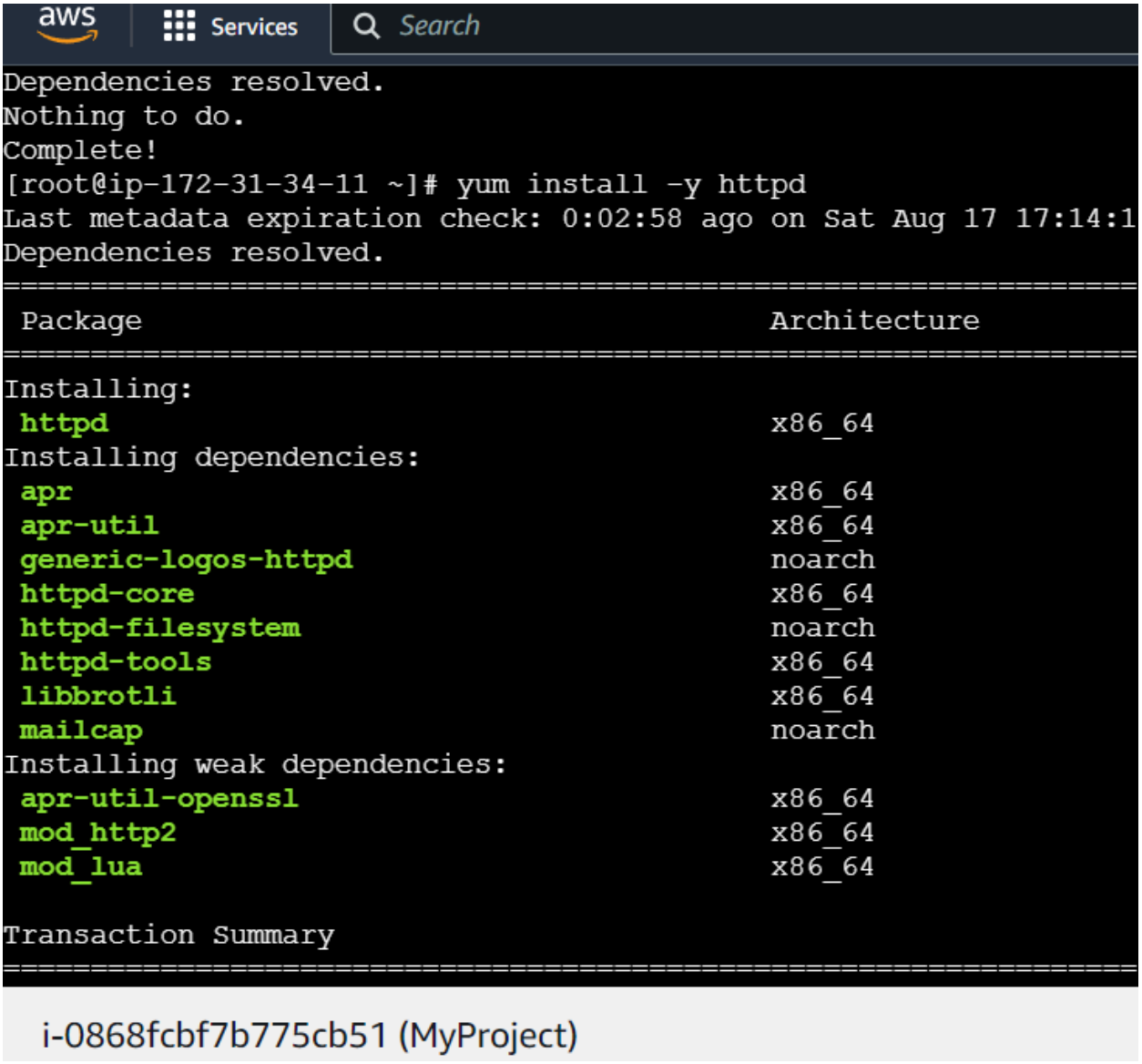




1. **Hosting a static website using EC2 instance**

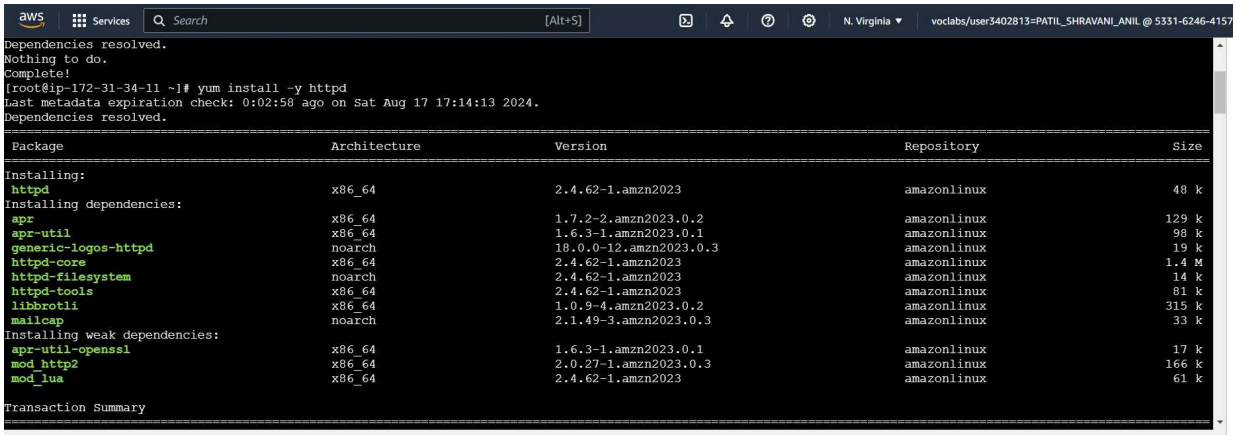
   sudo su - : allows you to execute commands with elevated privileges
   yum update -y - utilize the YUM package manager
   yum install -y httpd - is used to install the Apache HTTP Server (httpd) on a Linux system that uses the YUM package manager.

```
  ,     #_
  ~\_  ####_        Amazon Linux 2023
 ~~  \_#####\
 ~~     \###|
 ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
  ~~       V~' '->
   ~~~         /
     ~~._.   _/
        _/ _/
      _/m/'
Last login: Sat Aug 17 15:48:46 2024 from 18.206.107.27
[ec2-user@ip-172-31-52-166 ~]$ sudo su-
sudo: su-: command not found
[ec2-user@ip-172-31-52-166 ~]$ sudo su -
Last login: Sat Aug 17 15:49:22 UTC 2024 on pts/1
[root@ip-172-31-52-166 ~]# yum update -y
Last metadata expiration check: 1:24:30 ago on Sat Aug 17 15:44:42 2024.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-52-166 ~]# yum intall -y httpd
No such command: intall. Please use /usr/bin/yum --help
It could be a YUM plugin command, try: "yum install 'dnf-command(intall)'"
[root@ip-172-31-52-166 ~]# yum install -y httpd
Last metadata expiration check: 1:25:09 ago on Sat Aug 17 15:44:42 2024.
Package httpd-2.4.62-1.amzn2023.x86_64 is already installed.
Dependencies resolved.
```

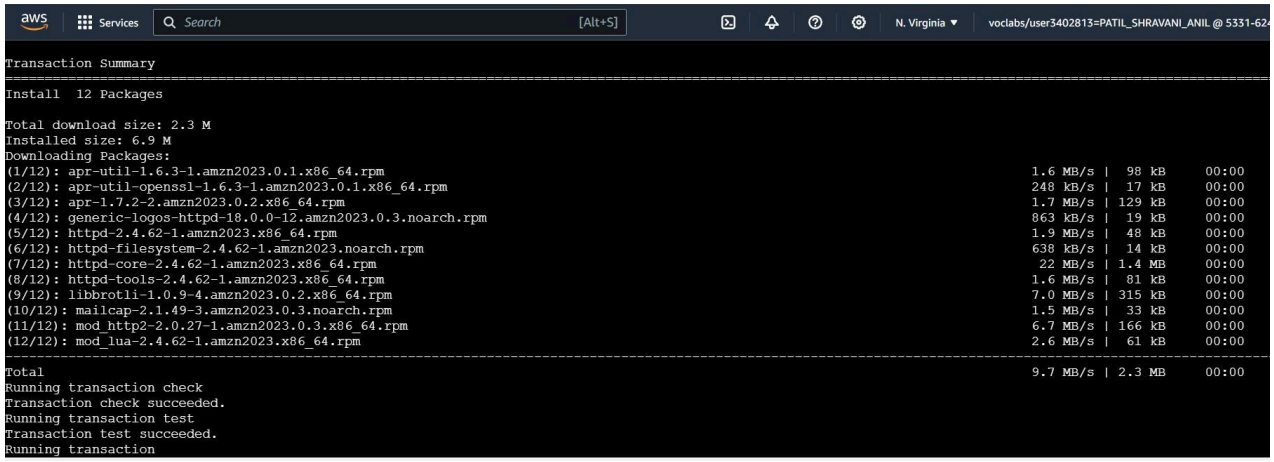


```
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-34-11 ~]# yum install -y httpd
Last metadata expiration check: 0:02:58 ago on Sat Aug 17 17:14:1
Dependencies resolved.
================================================================================
 Package                                          Architecture
================================================================================
Installing:
 httpd                                            x86_64
Installing dependencies:
 apr                                              x86_64
 apr-util                                         x86_64
 generic-logos-httpd                              noarch
 httpd-core                                       x86_64
 httpd-filesystem                                 noarch
 httpd-tools                                      x86_64
 libbrotli                                        x86_64
 mailcap                                          noarch
Installing weak dependencies:
 apr-util-openssl                                 x86_64
 mod_http2                                        x86_64
 mod_lua                                          x86_64

Transaction Summary
================================================================================
```

i-0868fcbf7b775cb51 (MyProject)



```
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-34-11 ~]# yum install -y httpd
Last metadata expiration check: 0:02:58 ago on Sat Aug 17 17:14:13 2024.
Dependencies resolved.
================================================================================
 Package                  Architecture  Version                    Repository     Size
================================================================================
Installing:
 httpd                    x86_64        2.4.62-1.amzn2023          amazonlinux    48 k
Installing dependencies:
 apr                      x86_64        1.7.2-2.amzn2023.0.2       amazonlinux   129 k
 apr-util                 x86_64        1.6.3-1.amzn2023.0.1       amazonlinux    98 k
 generic-logos-httpd      noarch        18.0.0-12.amzn2023.0.3     amazonlinux    19 k
 httpd-core               x86_64        2.4.62-1.amzn2023          amazonlinux   1.4 M
 httpd-filesystem         noarch        2.4.62-1.amzn2023          amazonlinux    14 k
 httpd-tools              x86_64        2.4.62-1.amzn2023          amazonlinux    81 k
 libbrotli                x86_64        1.0.9-4.amzn2023.0.2       amazonlinux   315 k
 mailcap                  noarch        2.1.49-3.amzn2023.0.3      amazonlinux    33 k
Installing weak dependencies:
 apr-util-openssl         x86_64        1.6.3-1.amzn2023.0.1       amazonlinux    17 k
 mod_http2                x86_64        2.0.27-1.amzn2023.0.3      amazonlinux   166 k
 mod_lua                  x86_64        2.4.62-1.amzn2023          amazonlinux    61 k

Transaction Summary
================================================================================
```

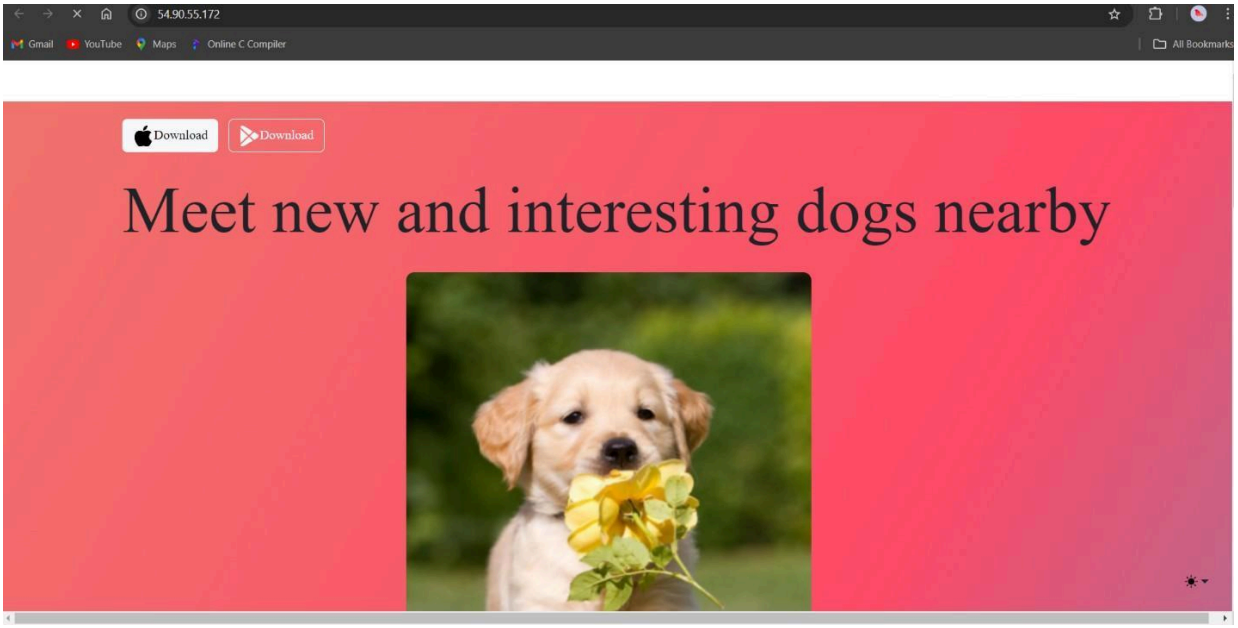


```
Transaction Summary
================================================================================
Install  12 Packages

Total download size: 2.3 M
Installed size: 6.9 M
Downloading Packages:
(1/12): apr-util-1.6.3-1.amzn2023.0.1.x86_64.rpm                1.6 MB/s |  98 kB     00:00
(2/12): apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64.rpm        248 kB/s |  17 kB     00:00
(3/12): apr-1.7.2-2.amzn2023.0.2.x86_64.rpm                     1.7 MB/s | 129 kB     00:00
(4/12): generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch.rpm   863 kB/s |  19 kB     00:00
(5/12): httpd-2.4.62-1.amzn2023.x86_64.rpm                      1.9 MB/s |  48 kB     00:00
(6/12): httpd-filesystem-2.4.62-1.amzn2023.noarch.rpm           638 kB/s |  14 kB     00:00
(7/12): httpd-core-2.4.62-1.amzn2023.x86_64.rpm                  22 MB/s | 1.4 MB     00:00
(8/12): httpd-tools-2.4.62-1.amzn2023.x86_64.rpm                1.6 MB/s |  81 kB     00:00
(9/12): libbrotli-1.0.9-4.amzn2023.0.2.x86_64.rpm               7.0 MB/s | 315 kB     00:00
(10/12): mailcap-2.1.49-3.amzn2023.0.3.noarch.rpm               1.5 MB/s |  33 kB     00:00
(11/12): mod_http2-2.0.27-1.amzn2023.0.3.x86_64.rpm             6.7 MB/s | 166 kB     00:00
(12/12): mod_lua-2.4.62-1.amzn2023.x86_64.rpm                   2.6 MB/s |  61 kB     00:00
--------------------------------------------------------------------------------
Total                                                           9.7 MB/s | 2.3 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
```

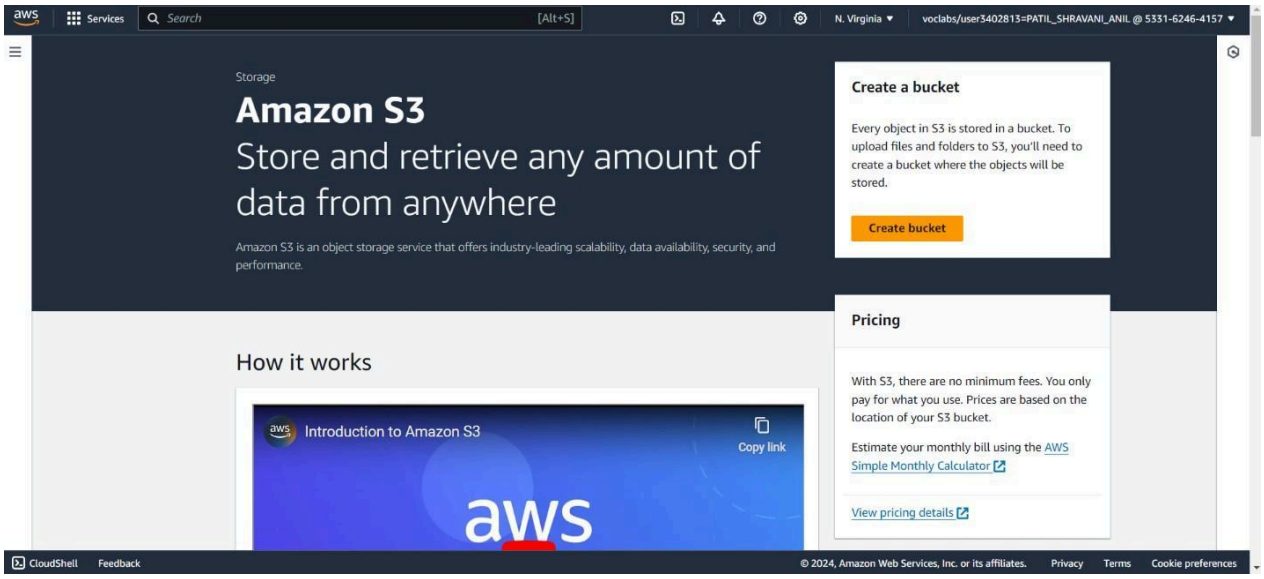
```
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Installing       : apr-1.7.2-2.amzn2023.0.2.x86_64                        1/12
  Installing       : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64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

```
  Verifying        : mod_lua-2.4.62-1.amzn2023.x86_64                      12/

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

Complete!
[root@ip-172-31-34-11 ~]# systemctl status httpd
○ httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:httpd.service(8)
[root@ip-172-31-34-11 ~]# mkdir aws_assg1
[root@ip-172-31-34-11 ~]# cd aws_assg1
[root@ip-172-31-34-11 aws_assg1]# ls -lrt
total 0
[root@ip-172-31-34-11 aws_assg1]# wget https://github.com/ShravaniAnilPatil/TinDog.git
--2024-08-17 17:21:08--  https://github.com/ShravaniAnilPatil/TinDog.git
Resolving github.com (github.com)... 140.82.112.3
Connecting to github.com (github.com)|140.82.112.3|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://github.com/ShravaniAnilPatil/TinDog [following]
--2024-08-17 17:21:08--  https://github.com/ShravaniAnilPatil/TinDog
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response... 200 OK
```

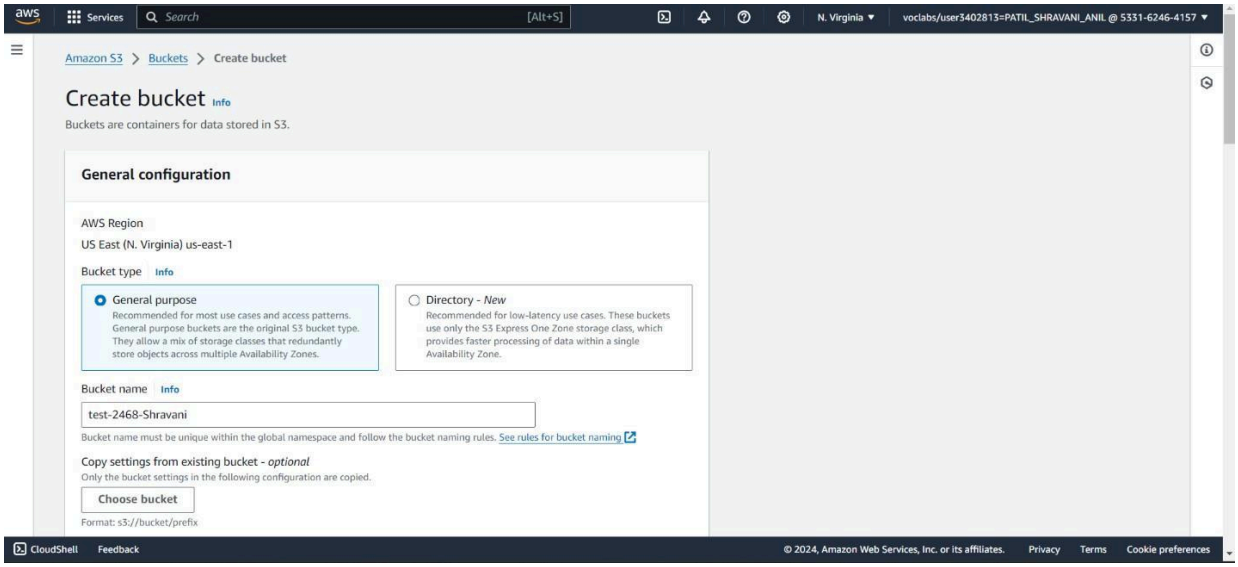
Website hosted



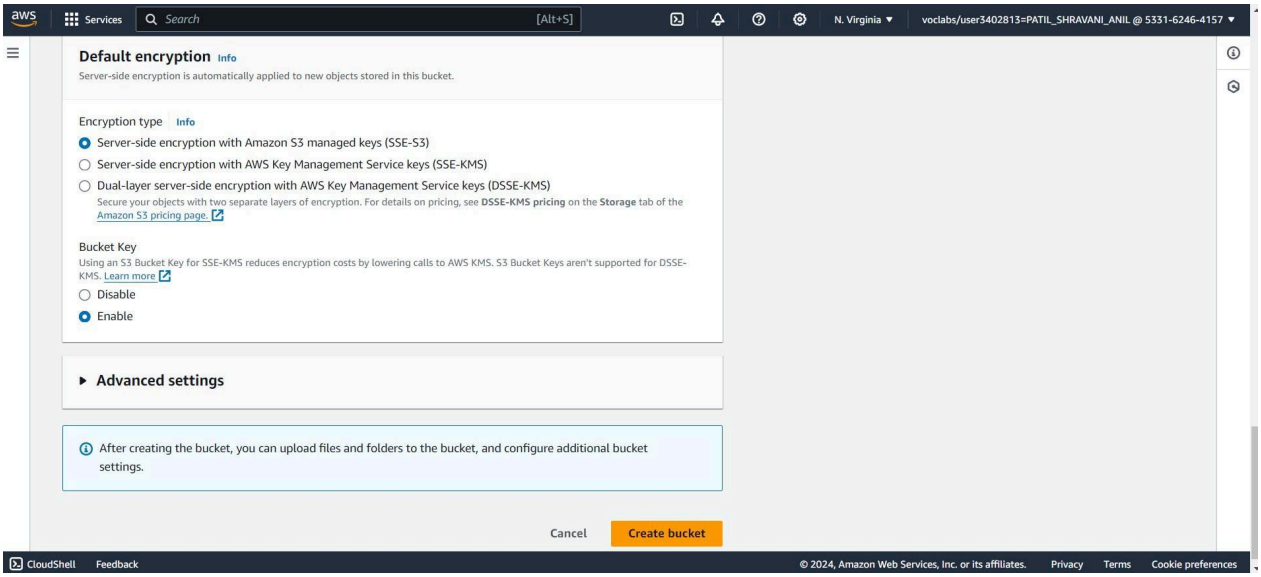
2. Hosting using S3 bucket



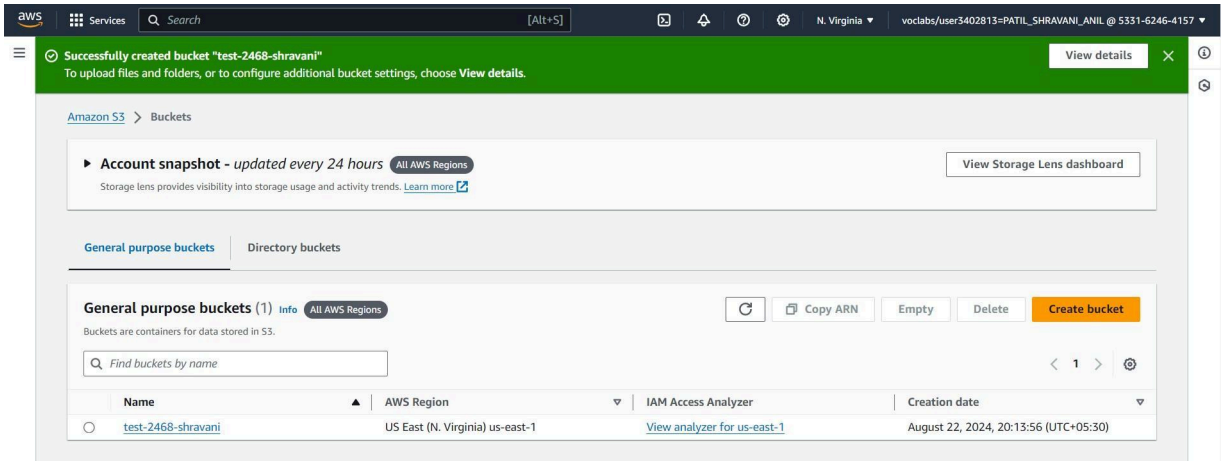
1. Visit S3 under the developer tools and create a Bucket.



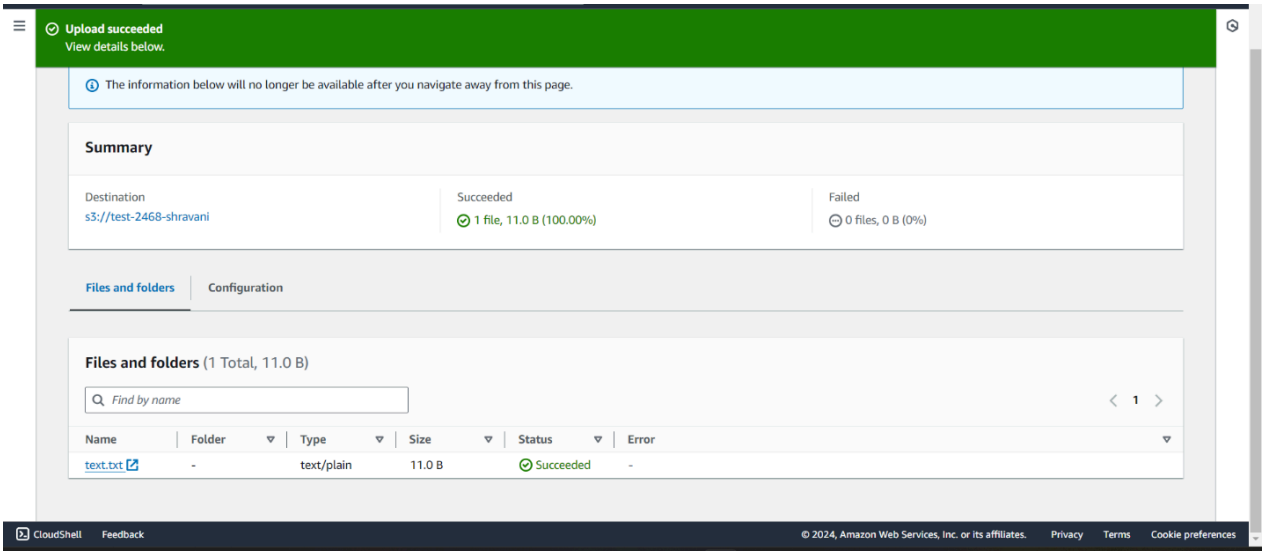
2. Add necessary configurations like encryption type and enable bucket key.

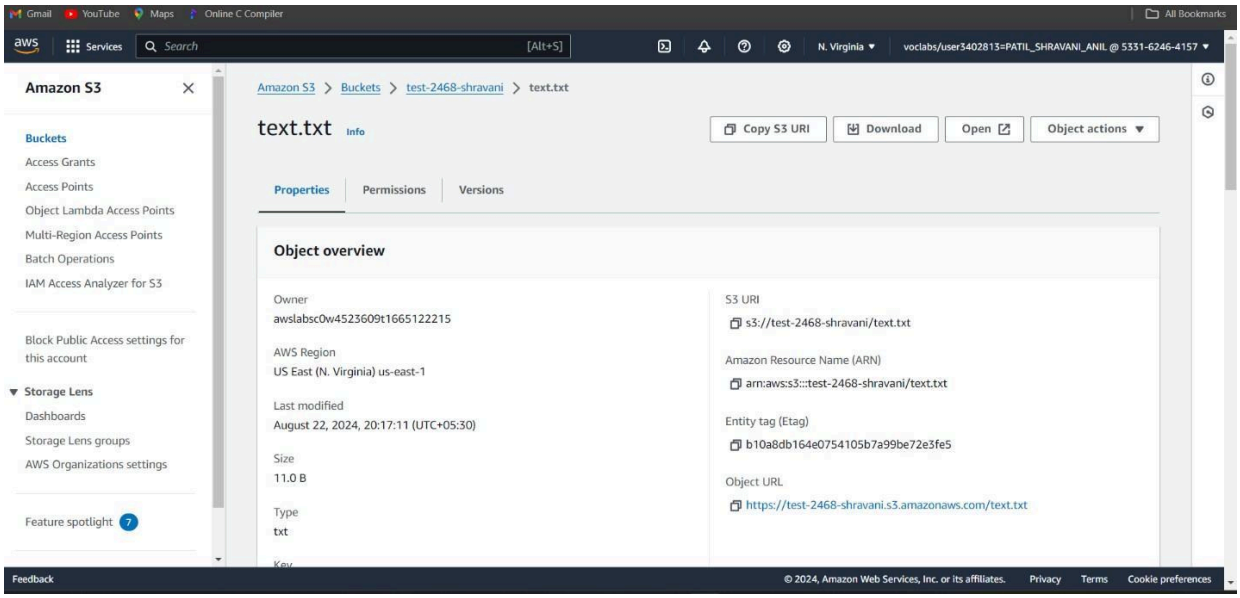


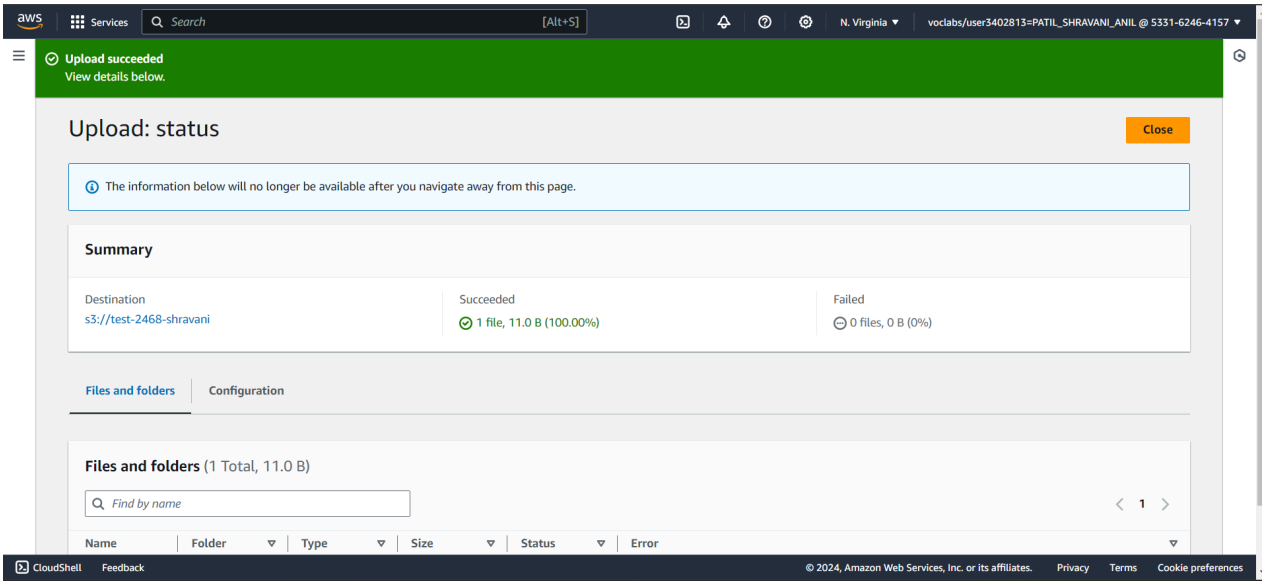
3. bucket successfully created



4.upload a file in the bucket



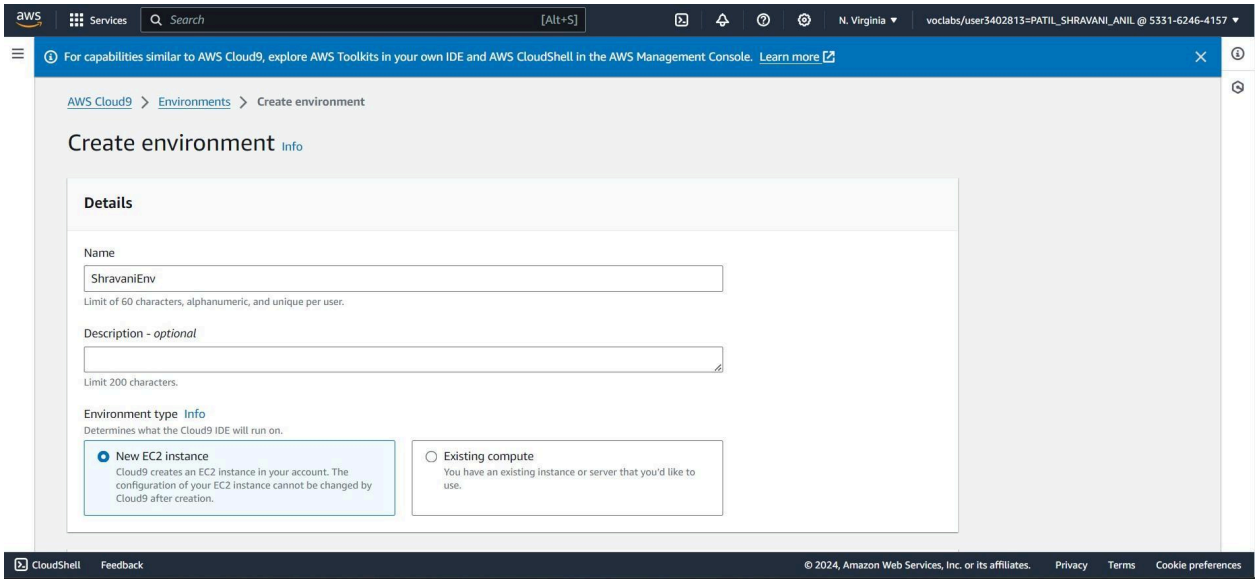


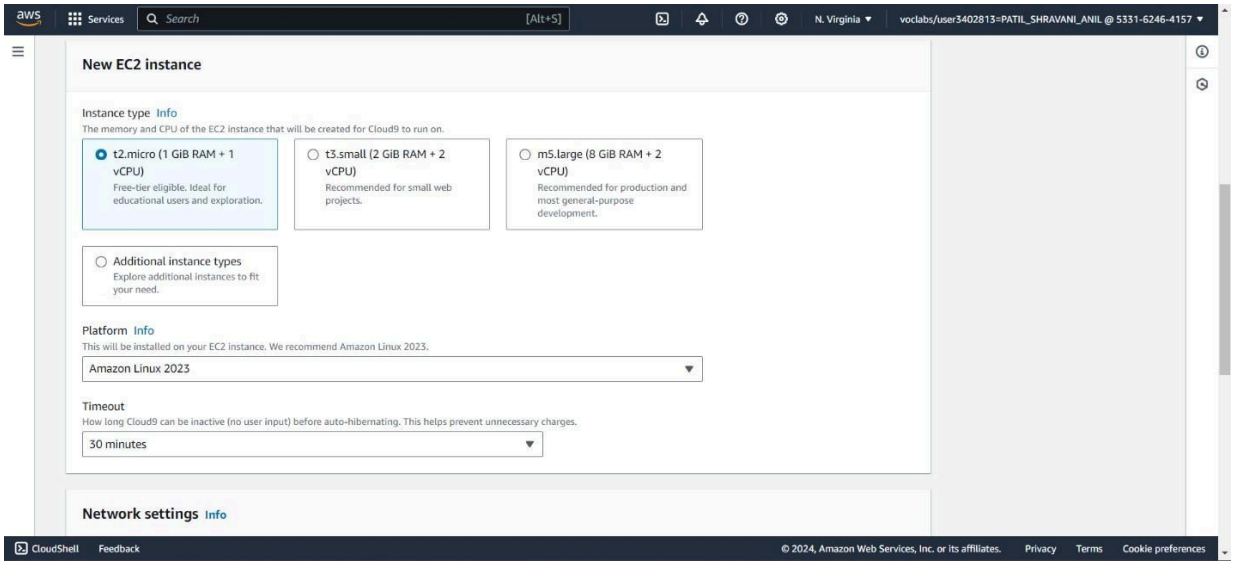


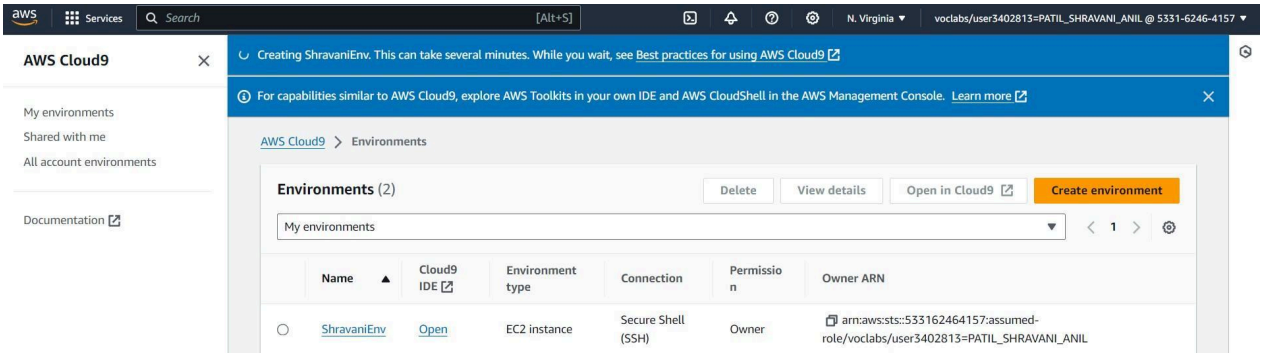
Website hosted - on the object url

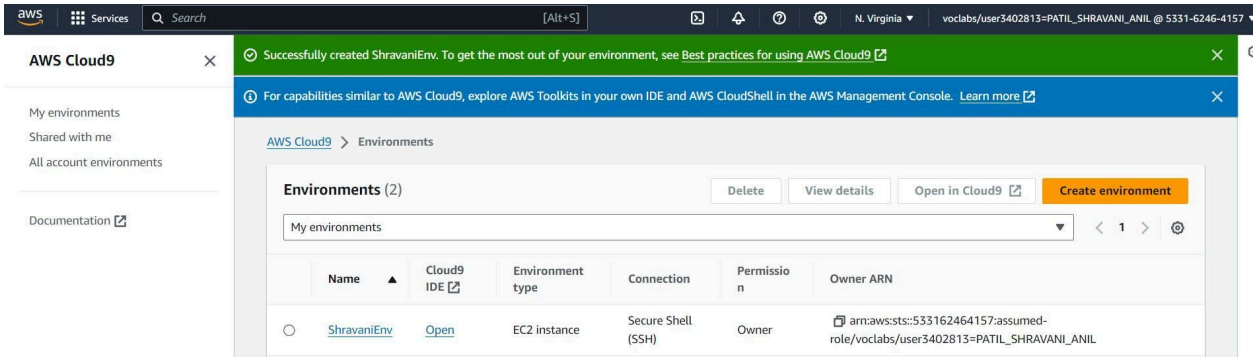

```
Hello World
```

3. Hosting using Cloud 9
create environment

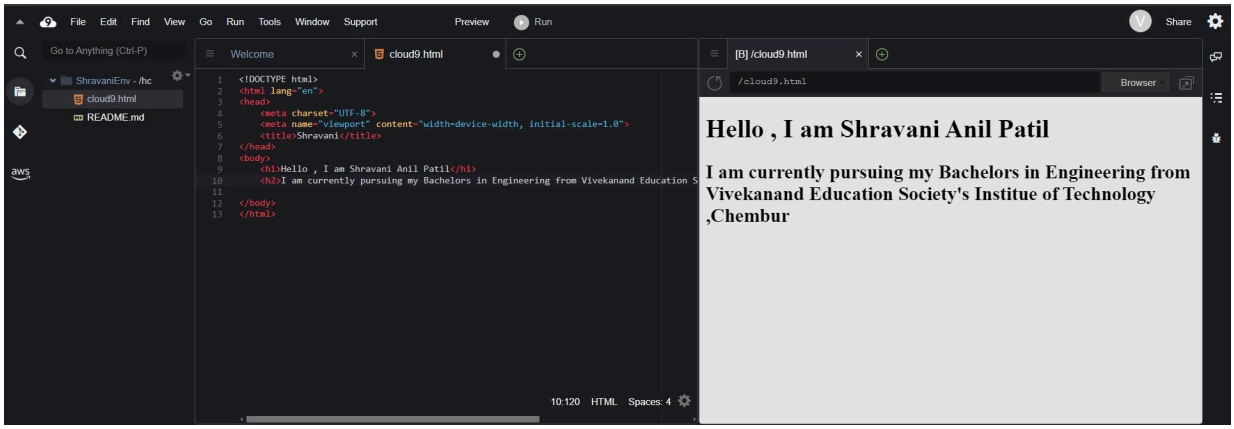








Website hosted



Advanced DevOps Experiment-2

Aim: Using AWS CodePipeline, deploy Sample Application on Elastic BeanStalk using AWS CodeDeploy.

Theory:-

Elastic Beanstalk simplifies the process of deploying and managing applications on the AWS cloud, allowing developers to focus on building applications without needing to worry about the underlying infrastructure. AWS offers over 100 services, which can sometimes make it challenging to know exactly how to configure and provision the right resources. Elastic Beanstalk removes this complexity by automatically managing the necessary AWS resources—such as Amazon EC2 instances, load balancing, and scaling—while still giving you control and flexibility over your setup.

Elastic Beanstalk supports a variety of programming languages including Go, Java, .NET, Node.js, PHP, Python, Ruby, and Docker platforms. This means that if your app is built in one of these languages or platforms, Elastic Beanstalk can handle its deployment. For Docker containers, you can even use your own custom configurations and dependencies, expanding the potential use cases beyond those natively supported languages.

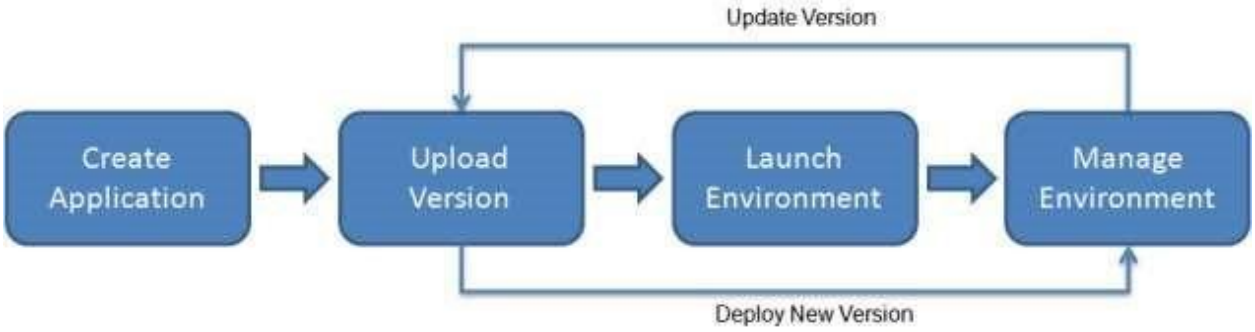
After your environment is set up, you can continue to manage and monitor it through the web interface or the CLI tools. You can scale your application by adjusting the number of EC2 instances or deploying new versions of your app with ease. The end result is a fully automated workflow that takes care of the technical heavy lifting, allowing you to deliver updates and maintain your app without diving deep into infrastructure management.

Here's a simplified breakdown of how Elastic Beanstalk works:

Create the application: Upload your code and define an application version.Elastic Beanstalk provisions resources: It automatically sets up the required AWS services and resources (EC2, load balancers, etc.).

Deploy and manage: You can monitor the application's performance, scale resources, and deploy new versions as needed.

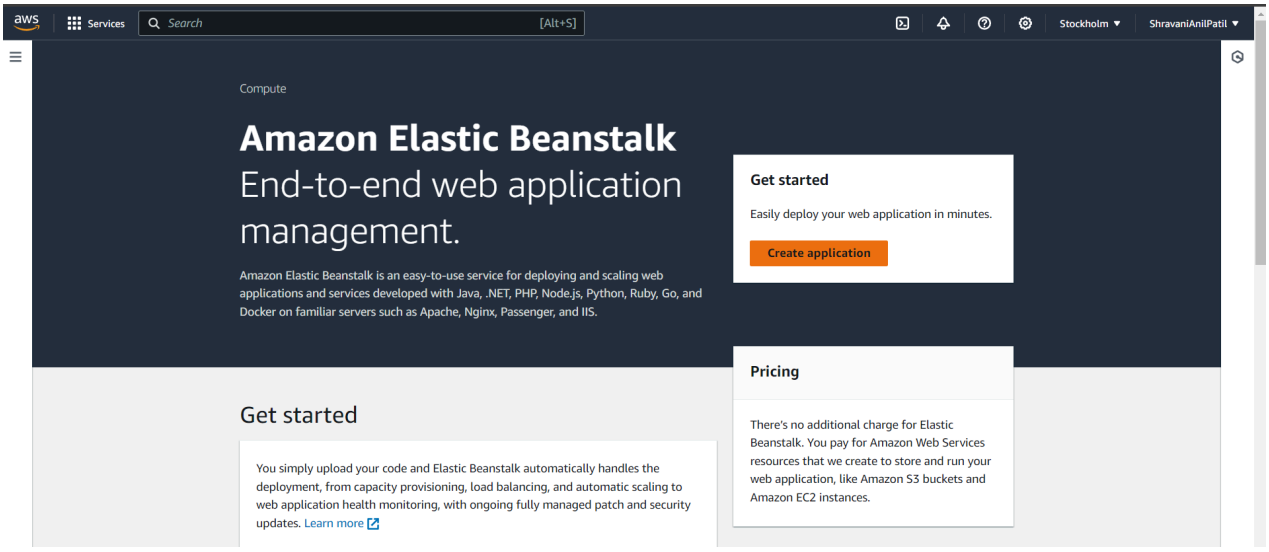
Elastic Beanstalk makes AWS cloud management simple, while keeping flexibility intact. It's like having a smart assistant for infrastructure that automates the complexities of cloud provisioning but still allows you to customize when needed.

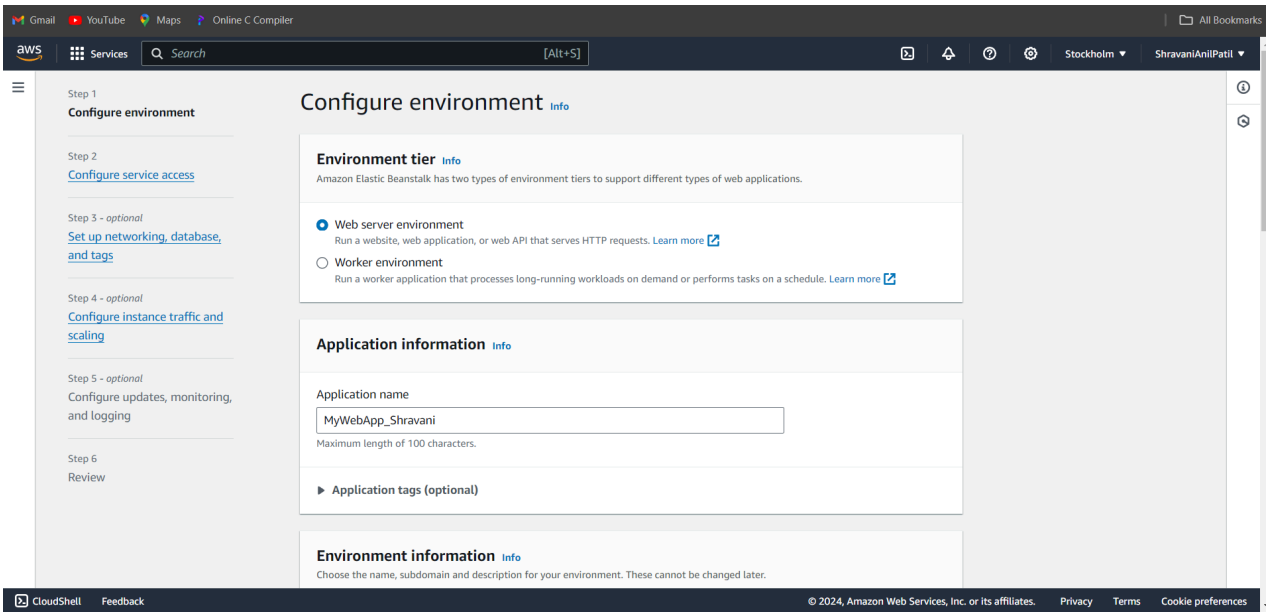


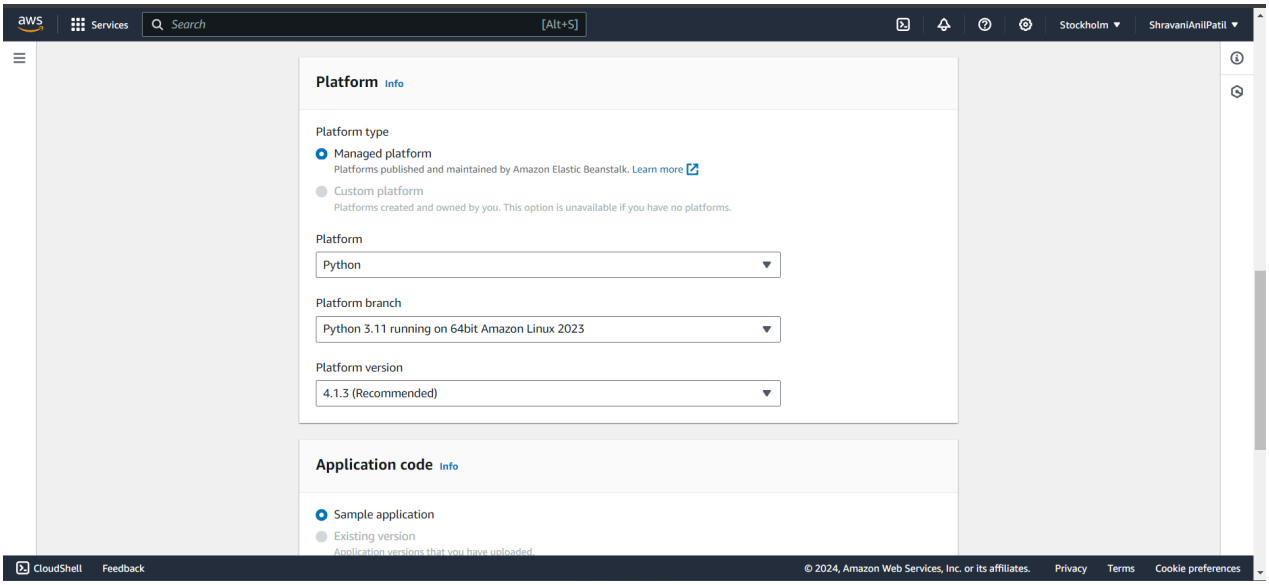
After you create and deploy your application, information about the application—including metrics, events, and environment status—is available through the Elastic Beanstalk console, APIs, or Command Line Interfaces, including the unified AWS CLI.

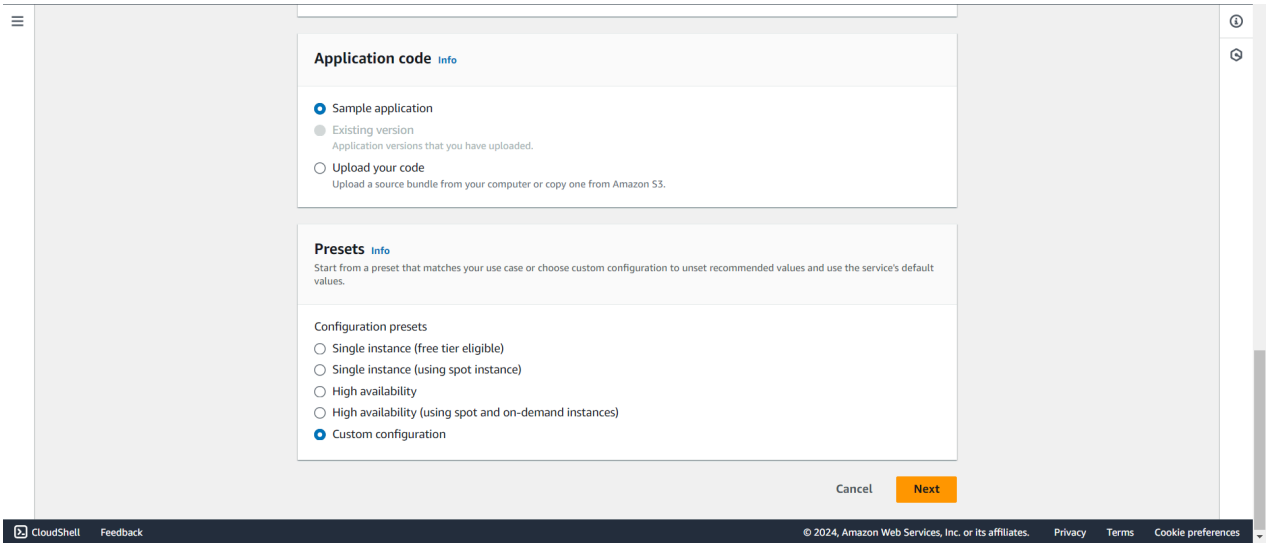
Implementation:-

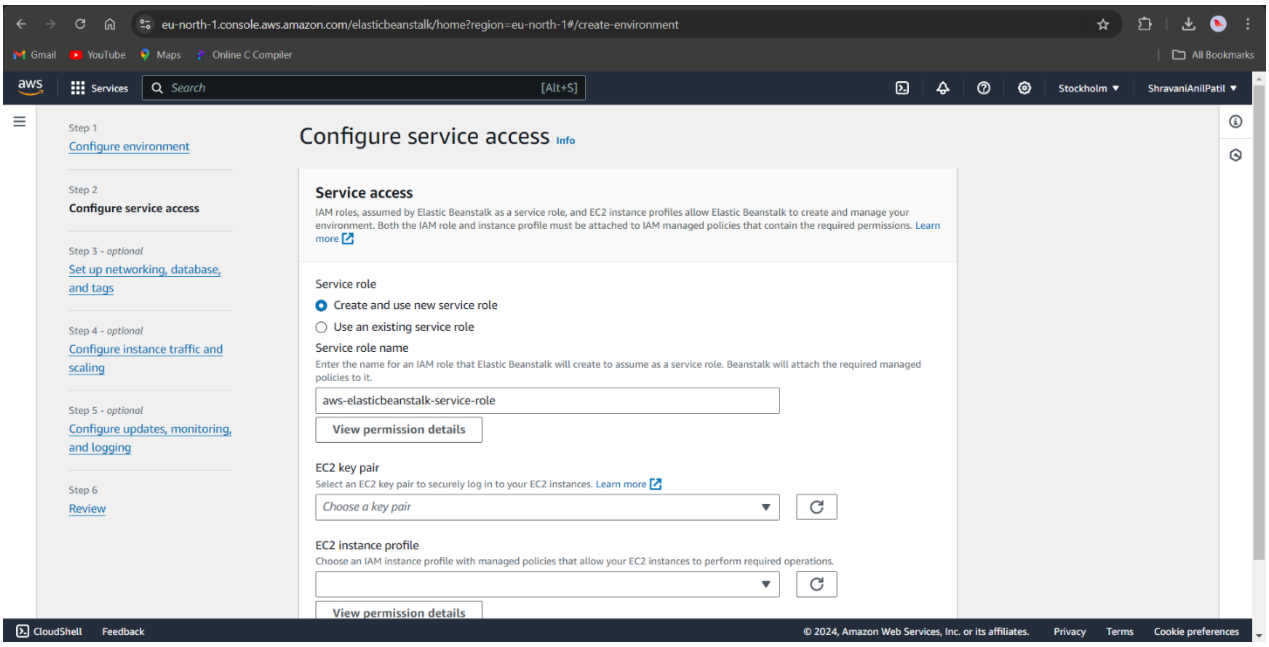
Deploying basic web page on Elastic Beanstalk

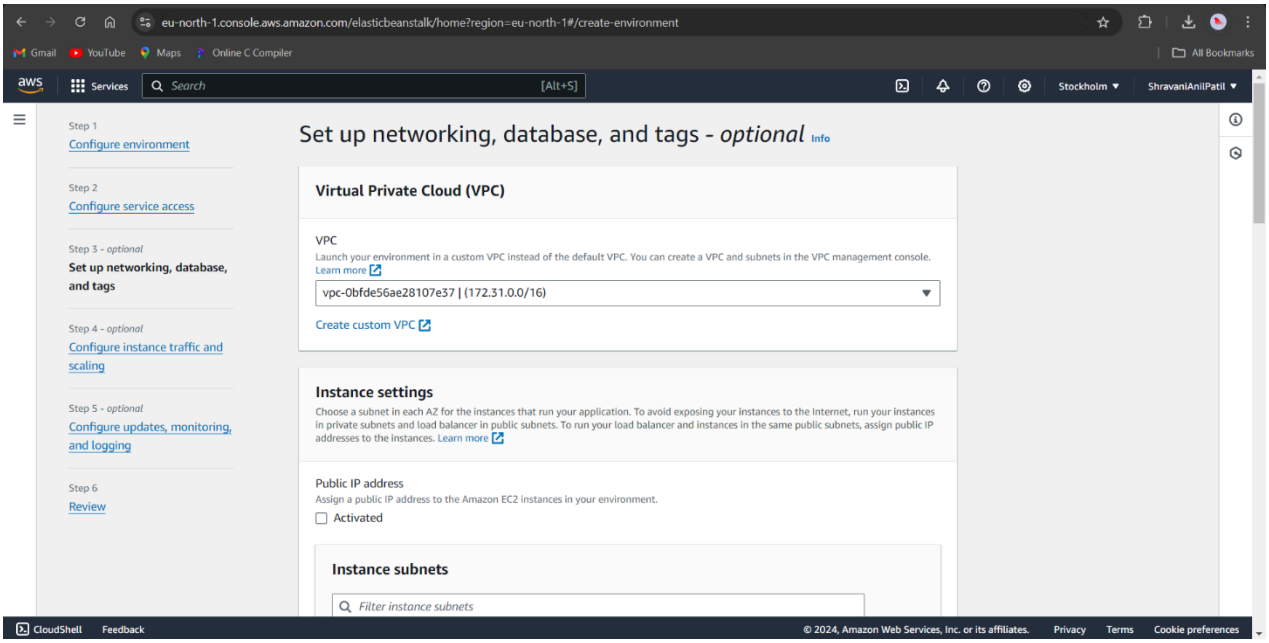


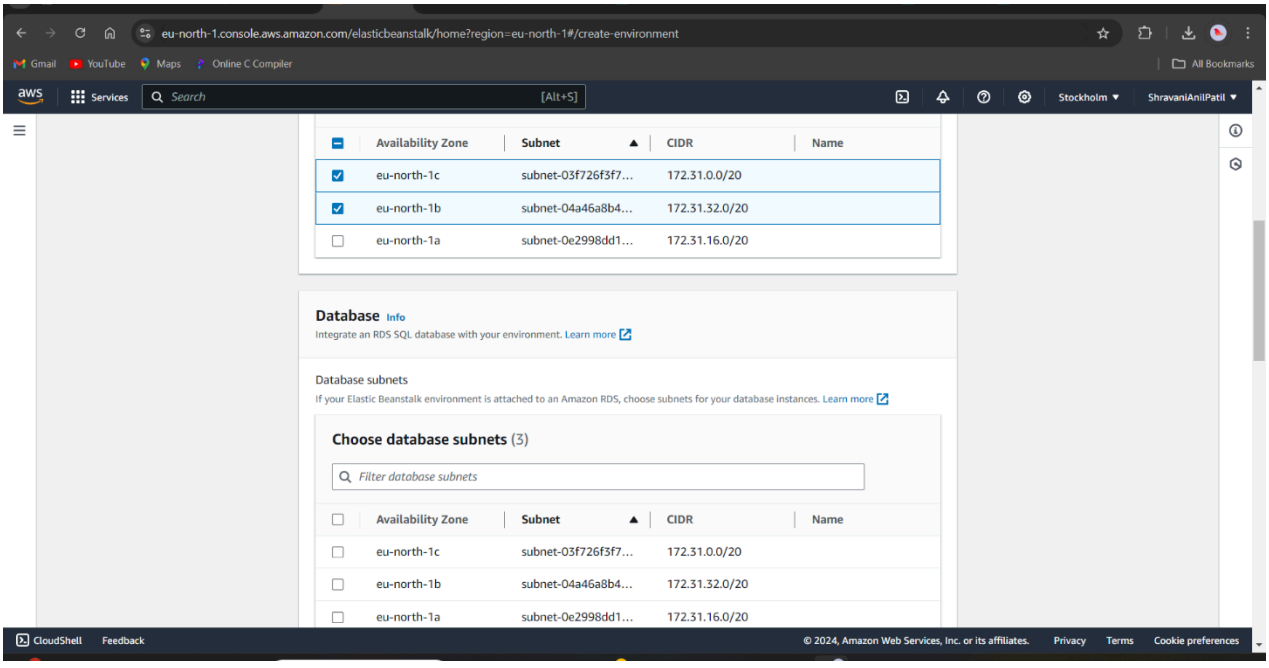


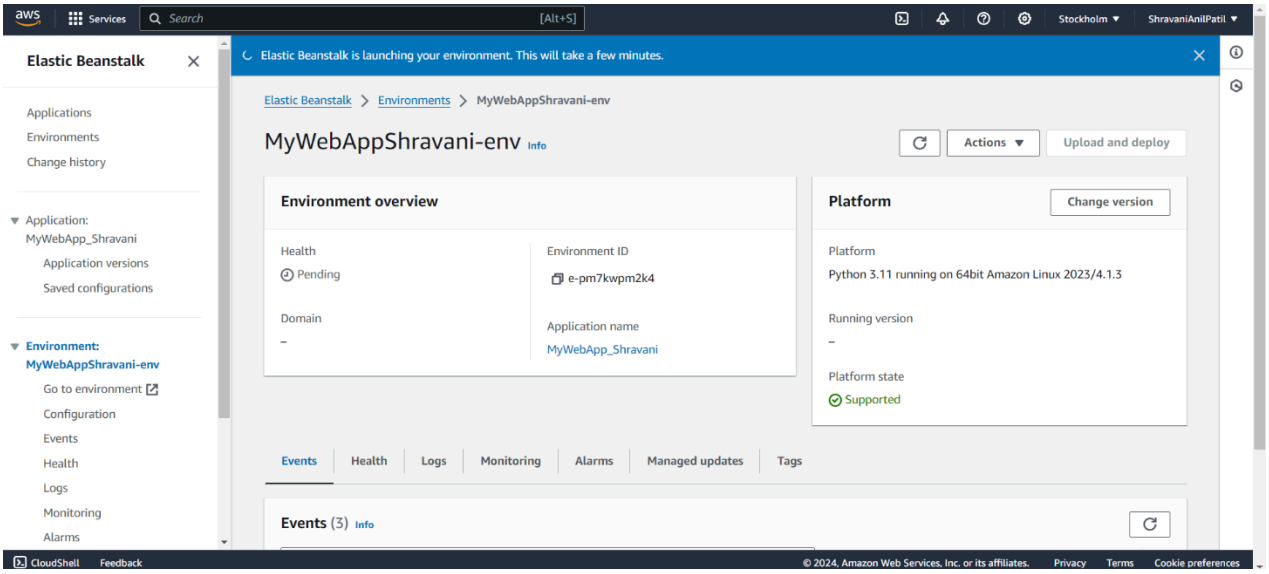

aws
Services
Search
[Alt+S]
Stockholm
ShravaniAnilPatil
Platform Info
Platform type
Managed platform
Platforms published and maintained by Amazon Elastic Beanstalk. Learn more
Custom platform
Platforms created and owned by you. This option is unavailable if you have no platforms.
Platform
Python
Platform 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
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

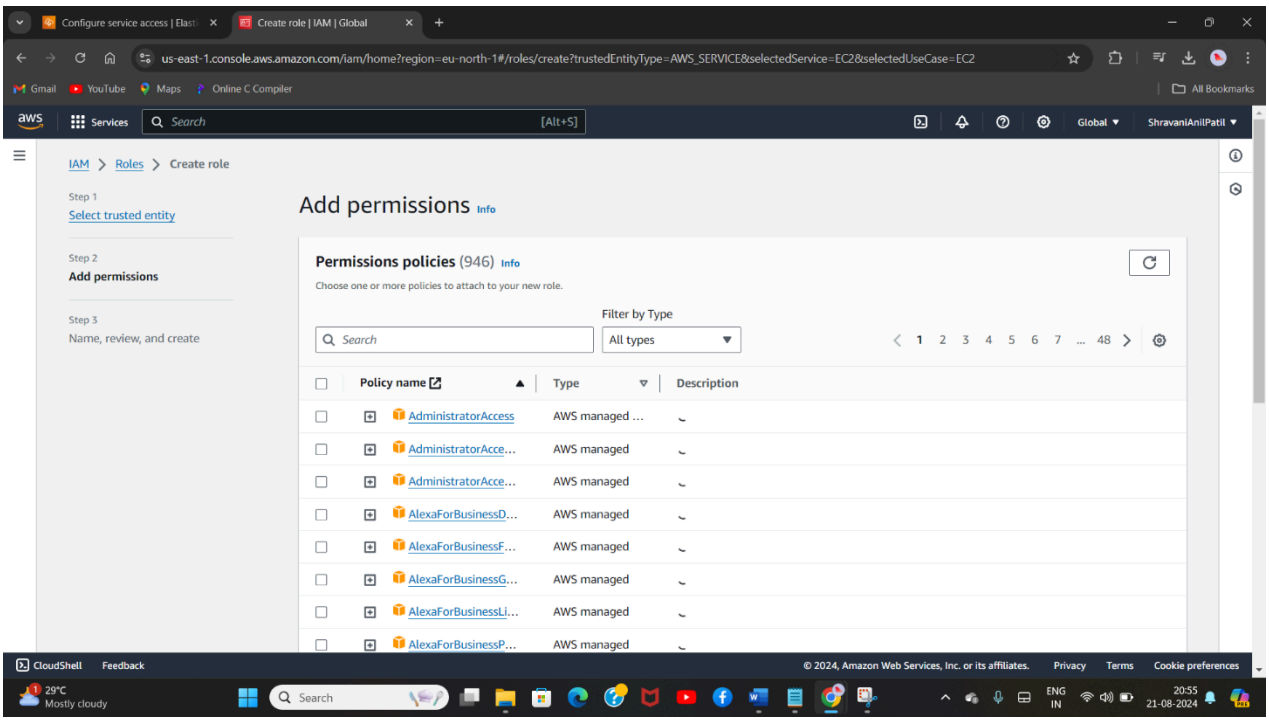

Application code Info
Sample application
Existing version
Application versions that you have uploaded.
Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.
Presets Info
Start from a preset that matches your use case or choose custom configuration to unset recommended 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
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

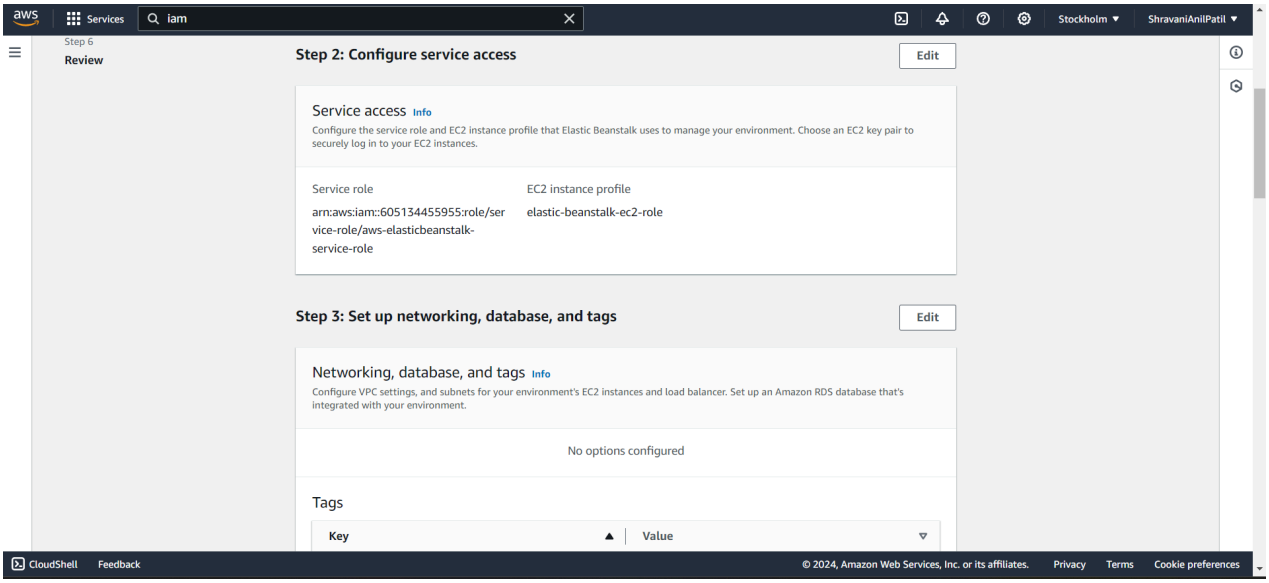

eu-north-1.console.aws.amazon.com/elasticbeanstalk/home?region=eu-north-1#/create-environment
Gmail
YouTube
Maps
Online C Compiler
All Bookmarks
aws
Services
Search
[Alt+S]
Stockholm
ShravaniAnilPatil
Step 1
Configure environment
Step 2
Configure service access
Step 3 - optional
Set up networking, database, and tags
Step 4 - optional
Configure instance traffic and scaling
Step 5 - optional
Configure updates, monitoring, and logging
Step 6
Review
Configure service access Info
Service access
IAM roles, assumed by Elastic Beanstalk as a service role, and EC2 instance profiles allow Elastic Beanstalk to create and manage your environment. Both the IAM role and instance profile must be attached to IAM managed policies that contain the required permissions. Learn more
Service role
Create and use new service role
Use an existing service role
Service role name
Enter the name for an IAM role that Elastic Beanstalk will create to assume as a service role. Beanstalk will attach the required managed policies to it.
aws-elasticbeanstalk-service-role
View permission details
EC2 key pair
Select an EC2 key pair to securely log in to your EC2 instances. Learn more
Choose a key pair
EC2 instance profile
Choose an IAM instance profile with managed policies that allow your EC2 instances to perform required operations.
View permission details
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

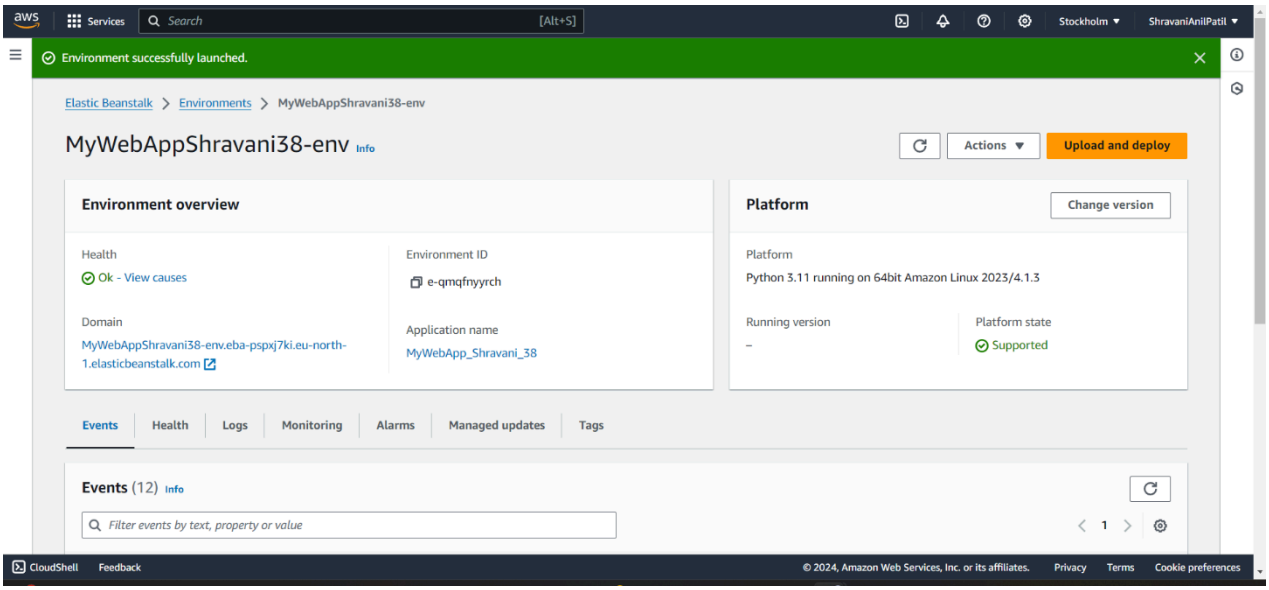

eu-north-1.console.aws.amazon.com/elasticbeanstalk/home?region=eu-north-1#/create-environment
Gmail YouTube Maps Online C Compiler
All Bookmarks
aws Services Search [Alt+S]
Stockholm ShravaniAnilPatil
Step 1
Configure environment
Step 2
Configure service access
Step 3 - optional
Set up networking, database, and tags
Step 4 - optional
Configure instance traffic and scaling
Step 5 - optional
Configure updates, monitoring, and logging
Step 6
Review
Set up networking, database, and tags - optional Info
Virtual Private Cloud (VPC)
VPC
Launch your environment in a custom VPC instead of the default VPC. You can create a VPC and subnets in the VPC management console.
Learn more
vpc-0bfde56ae28107e37 | (172.31.0.0/16)
Create custom VPC
Instance settings
Choose a subnet in each AZ for the instances that run your application. To avoid exposing your instances to the Internet, run your instances in private subnets and load balancer in public subnets. To run your load balancer and instances in the same public subnets, assign public IP addresses to the instances. Learn more
Public IP address
Assign a public IP address to the Amazon EC2 instances in your environment.
Activated
Instance subnets
Filter instance subnets
CloudShell Feedback
© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

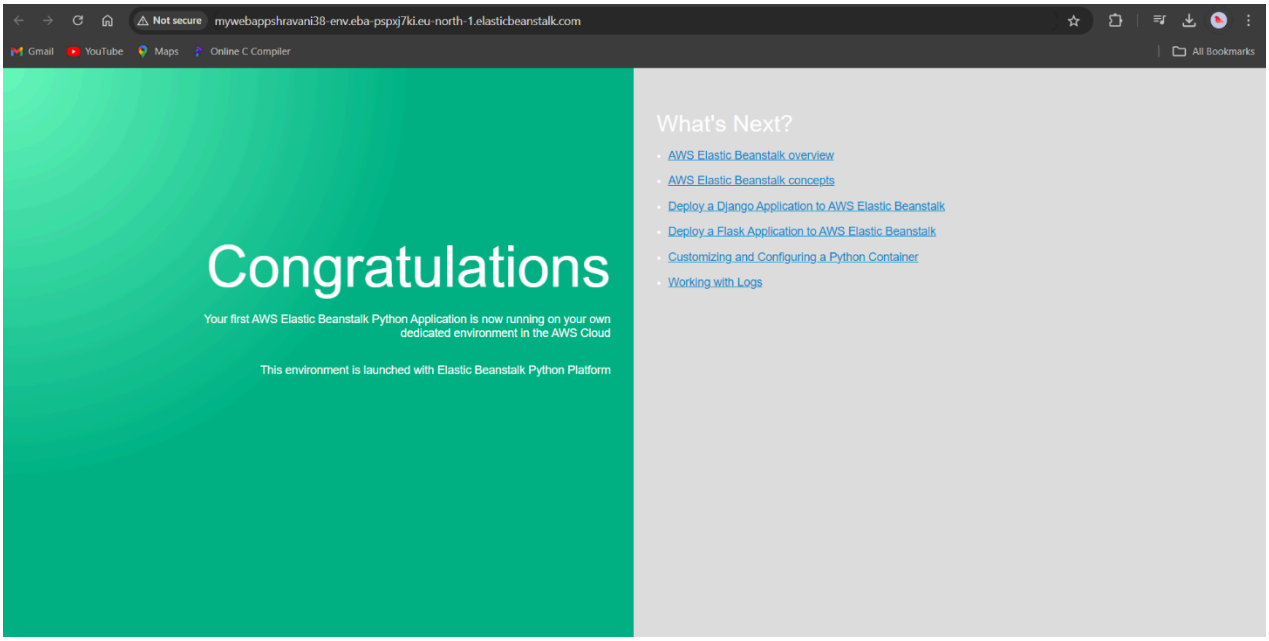

eu-north-1.console.aws.amazon.com/elasticbeanstalk/home?region=eu-north-1#/create-environment
Availability Zone Subnet CIDR Name
eu-north-1c subnet-03f726f3f7... 172.31.0.0/20
eu-north-1b subnet-04a46a8b4... 172.31.32.0/20
eu-north-1a subnet-0e2998dd1... 172.31.16.0/20
Database Info
Integrate an RDS SQL database with your environment. Learn more
Database subnets
If your Elastic Beanstalk environment is attached to an Amazon RDS, choose subnets for your database instances. Learn more
Choose database subnets (3)
Filter database subnets
Availability Zone Subnet CIDR Name
eu-north-1c subnet-03f726f3f7... 172.31.0.0/20
eu-north-1b subnet-04a46a8b4... 172.31.32.0/20
eu-north-1a subnet-0e2998dd1... 172.31.16.0/20
CloudShell Feedback
© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

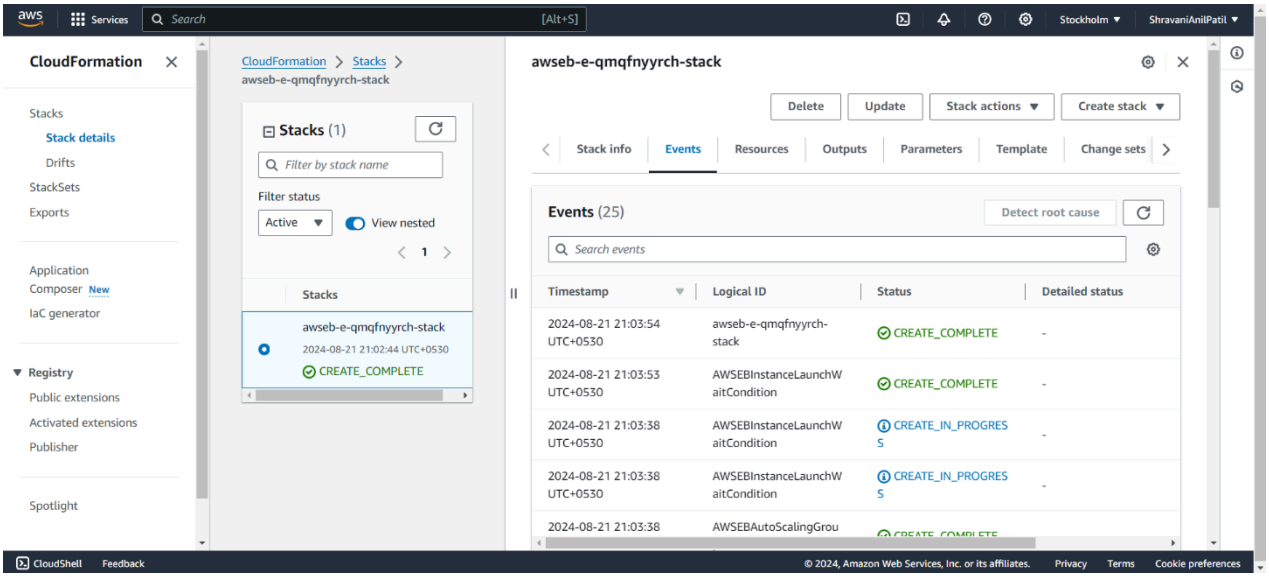

Elastic Beanstalk
Elastic Beanstalk is launching your environment. This will take a few minutes.
Elastic Beanstalk > Environments > MyWebAppShravani-env
MyWebAppShravani-env Info
Actions
Upload and deploy
Environment overview
Health
Pending
Environment ID
e-pm7kwpm2k4
Domain
-
Application name
MyWebApp_Shravani
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
Events (3) Info

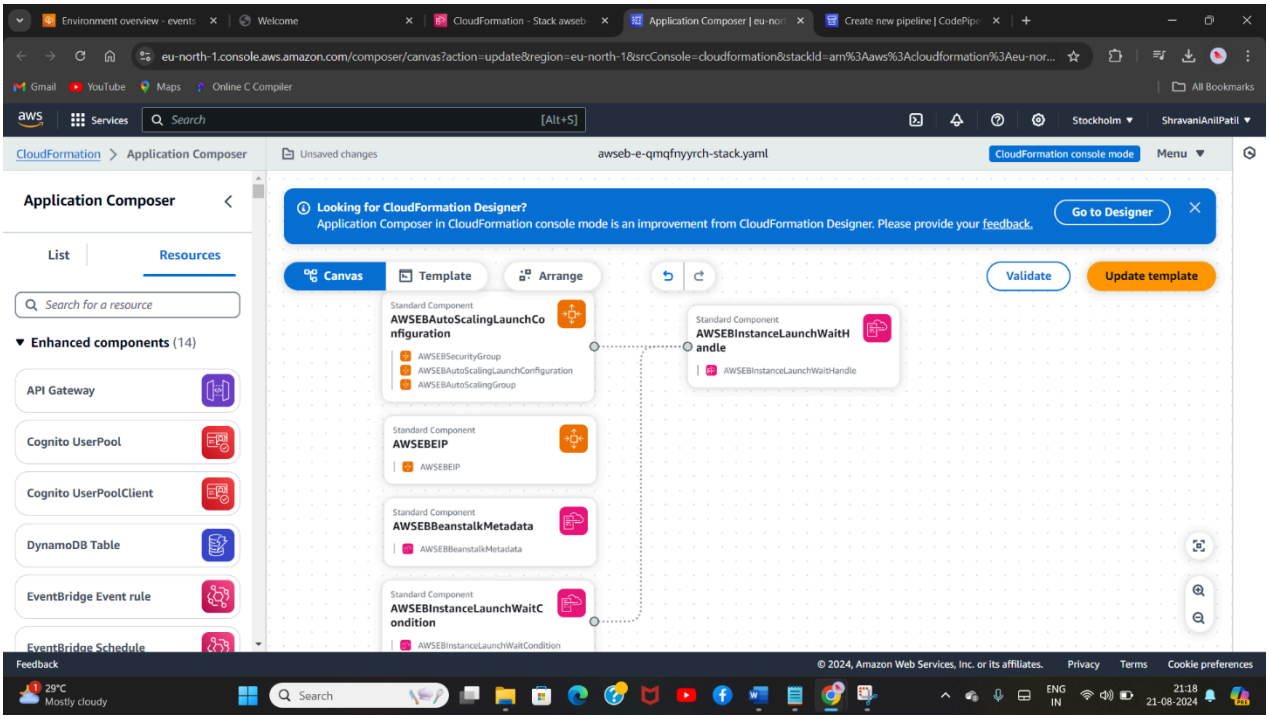

IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Add permissions Info
Permissions policies (946) Info
Choose one or more policies to attach to your new role.
Filter by Type
Search
All types
Policy name
Type
Description
AdministratorAccess
AWS managed ...
AdministratorAcce...
AWS managed
AlexaForBusinessD...
AlexaForBusinessF...
AlexaForBusinessG...
AlexaForBusinessLi...
AlexaForBusinessP...
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

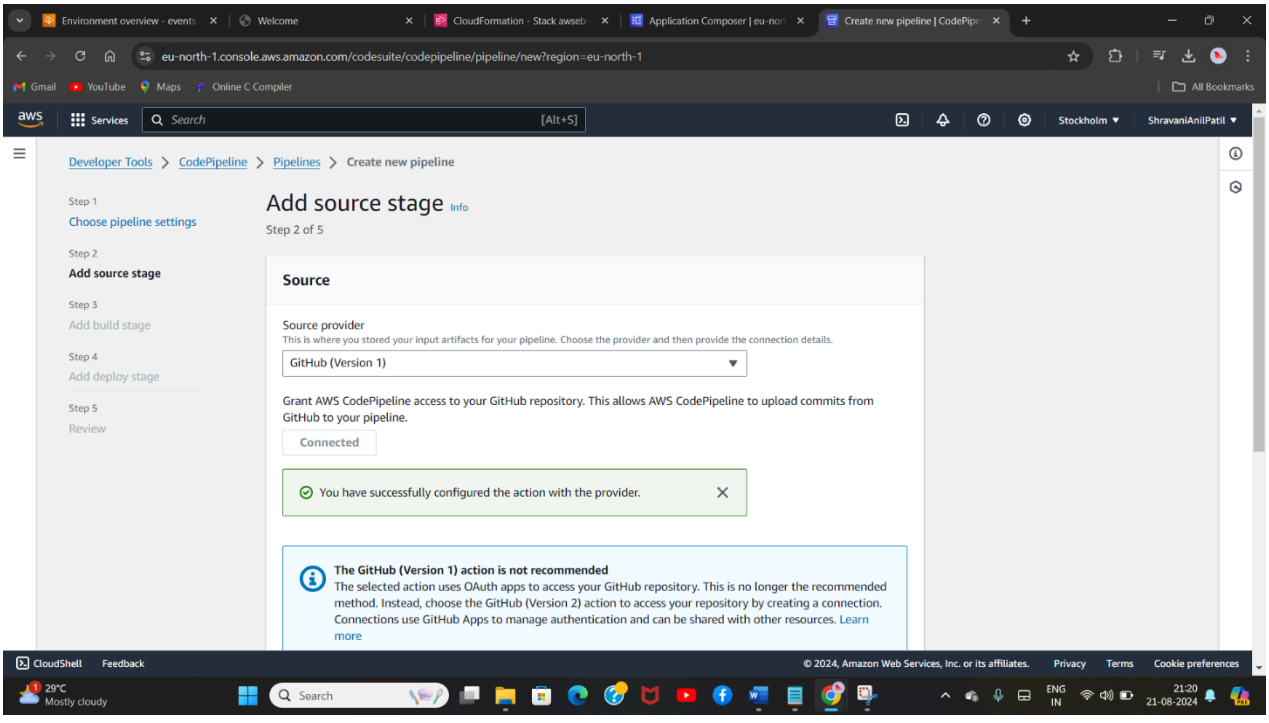

Services
iam
Stockholm
ShravaniAnilPatil
Step 6
Review
Step 2: Configure service access
Edit
Service access Info
Configure the service role and EC2 instance profile that Elastic Beanstalk uses to manage your environment. Choose an EC2 key pair to securely log in to your EC2 instances.
Service role
arn:aws:iam::605134455955:role/service-role/aws-elasticbeanstalk-service-role
EC2 instance profile
elastic-beanstalk-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
Key
Value
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

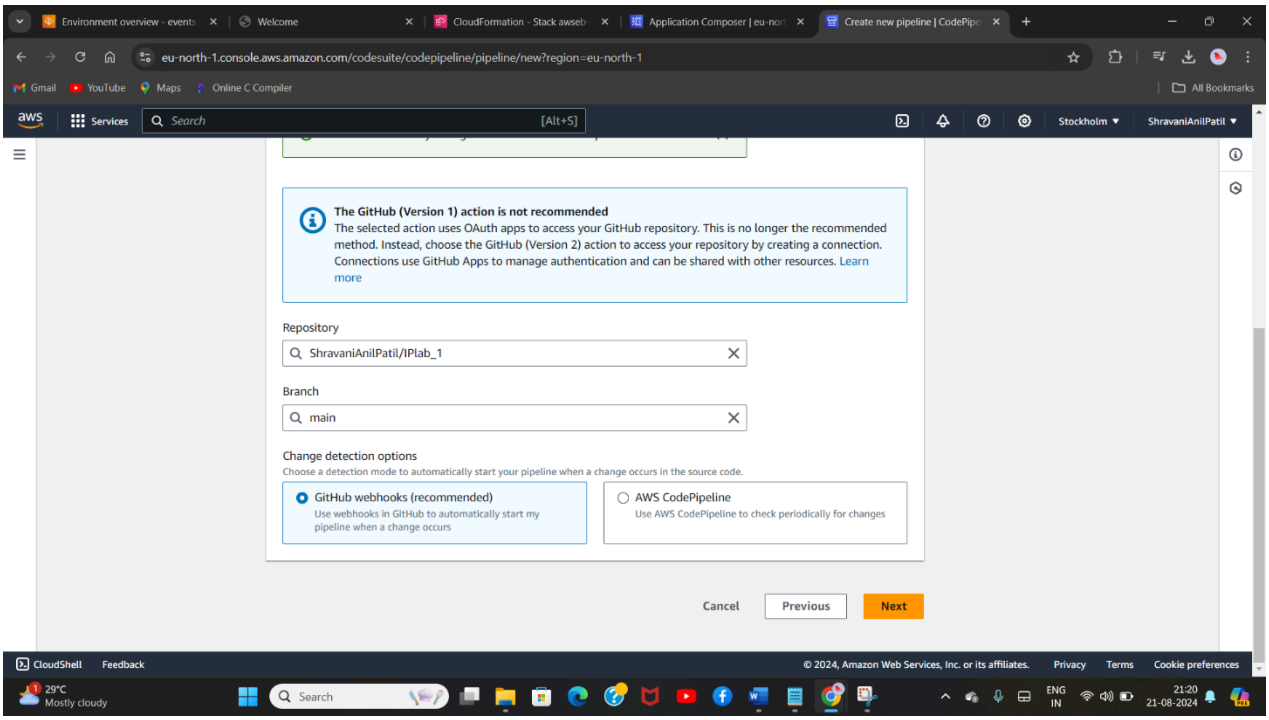

Services
Search
[Alt+S]
Stockholm
ShravaniAnilPatil
Environment successfully launched.
Elastic Beanstalk
Environments
MyWebAppShravani38-env
MyWebAppShravani38-env Info
Actions
Upload and deploy
Environment overview
Health
Ok - View causes
Environment ID
e-qmqfnyyrch
Domain
MyWebAppShravani38-env.eba-pspxj7ki.eu-north-1.elasticbeanstalk.com
Application name
MyWebApp_Shravani_38
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
Events (12) Info
Filter events by text, property or value
1
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

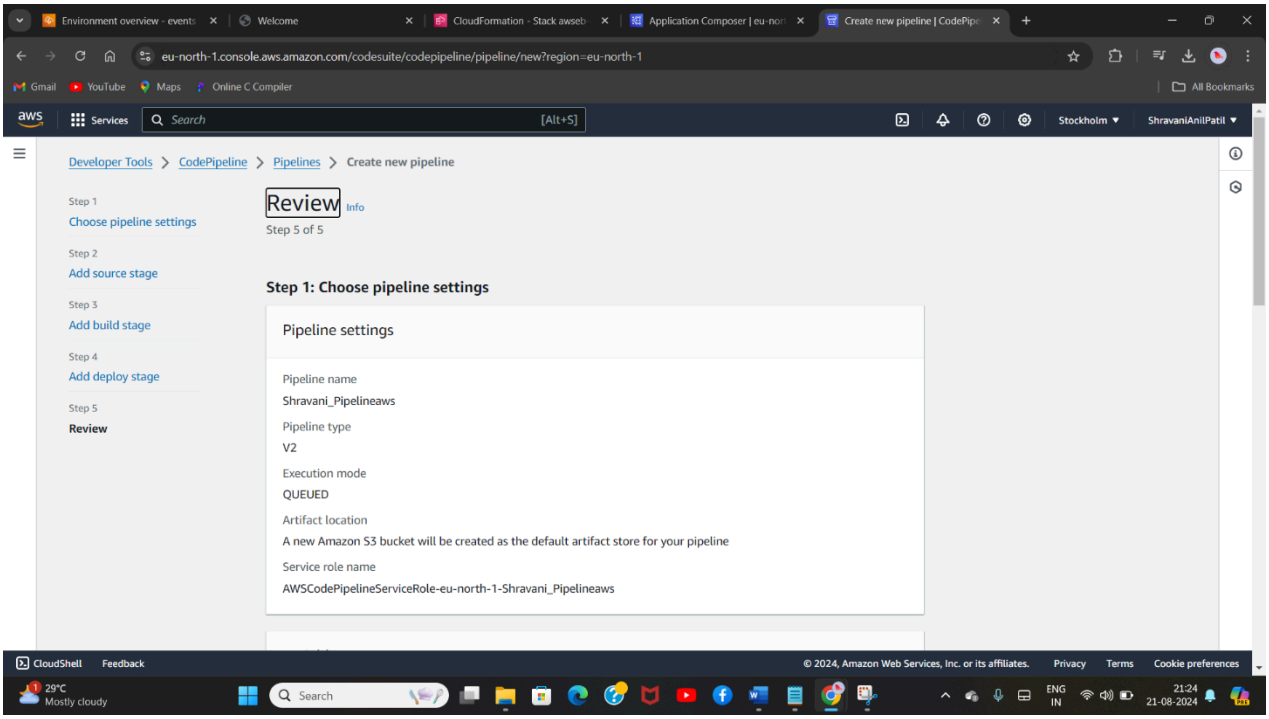

Not secure
mywebappshravani38-env.eba-pspxj7ki.eu-north-1.elasticbeanstalk.com
Gmail
YouTube
Maps
Online C Compiler
All Bookmarks
Congratulations
Your first AWS Elastic Beanstalk Python Application is now running on your own dedicated environment in the AWS Cloud
This environment is launched with Elastic Beanstalk Python Platform
What's Next?
AWS Elastic Beanstalk overview
AWS Elastic Beanstalk concepts
Deploy a Django Application to AWS Elastic Beanstalk
Deploy a Flask Application to AWS Elastic Beanstalk
Customizing and Configuring a Python Container
Working with Logs

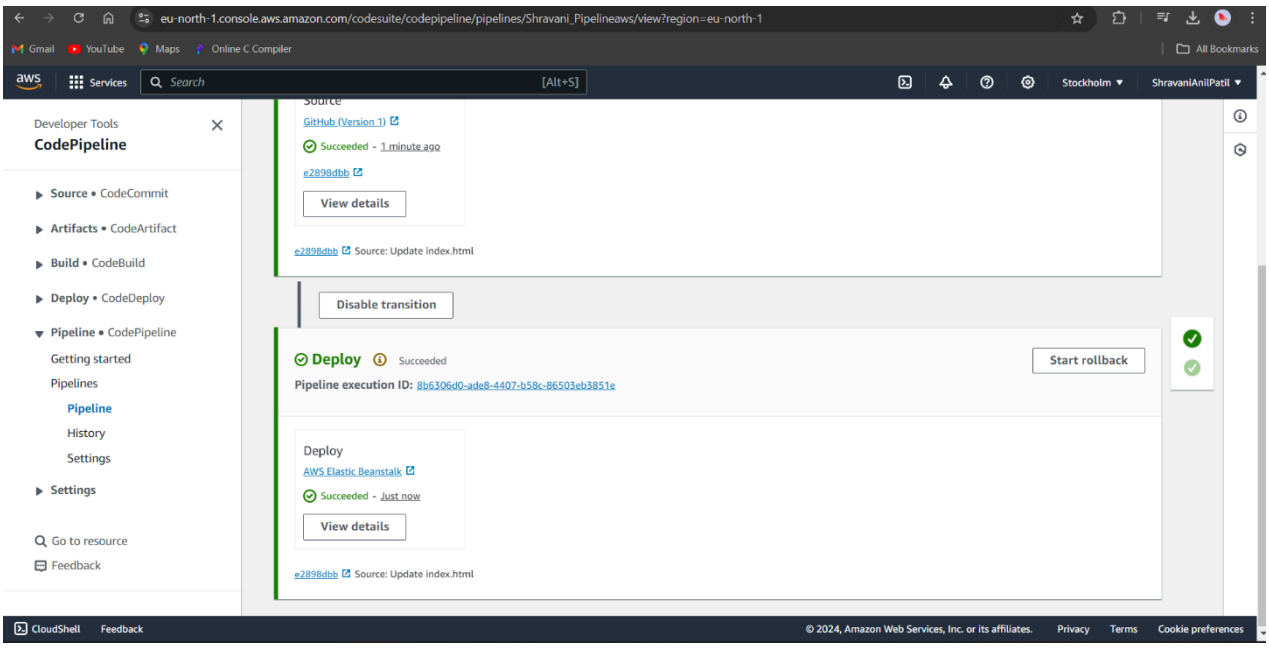

aws
Services
Search
[Alt+S]
Stockholm
ShravaniAnilPatil
CloudFormation
Stacks
Stack details
Drifts
StackSets
Exports
Application Composer New
IaC generator
Registry
Public extensions
Activated extensions
Publisher
Spotlight
CloudFormation > Stacks >
awseb-e-qmqfnyyrch-stack
Stacks (1)
Filter by stack name
Filter status
Active
View nested
Stacks
awseb-e-qmqfnyyrch-stack
2024-08-21 21:02:44 UTC+0530
CREATE_COMPLETE
awseb-e-qmqfnyyrch-stack
Delete
Update
Stack actions
Create stack
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Events (25)
Detect root cause
Search events
Timestamp
Logical ID
Status
Detailed status
2024-08-21 21:03:54 UTC+0530
awseb-e-qmqfnyyrch-stack
CREATE_COMPLETE
2024-08-21 21:03:53 UTC+0530
AWSEBInstanceLaunchWaitCondition
CREATE_COMPLETE
2024-08-21 21:03:38 UTC+0530
AWSEBInstanceLaunchWaitCondition
CREATE_IN_PROGRESS
2024-08-21 21:03:38 UTC+0530
AWSEBInstanceLaunchWaitCondition
CREATE_IN_PROGRESS
2024-08-21 21:03:38
AWSEBAutoScalingGrou
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

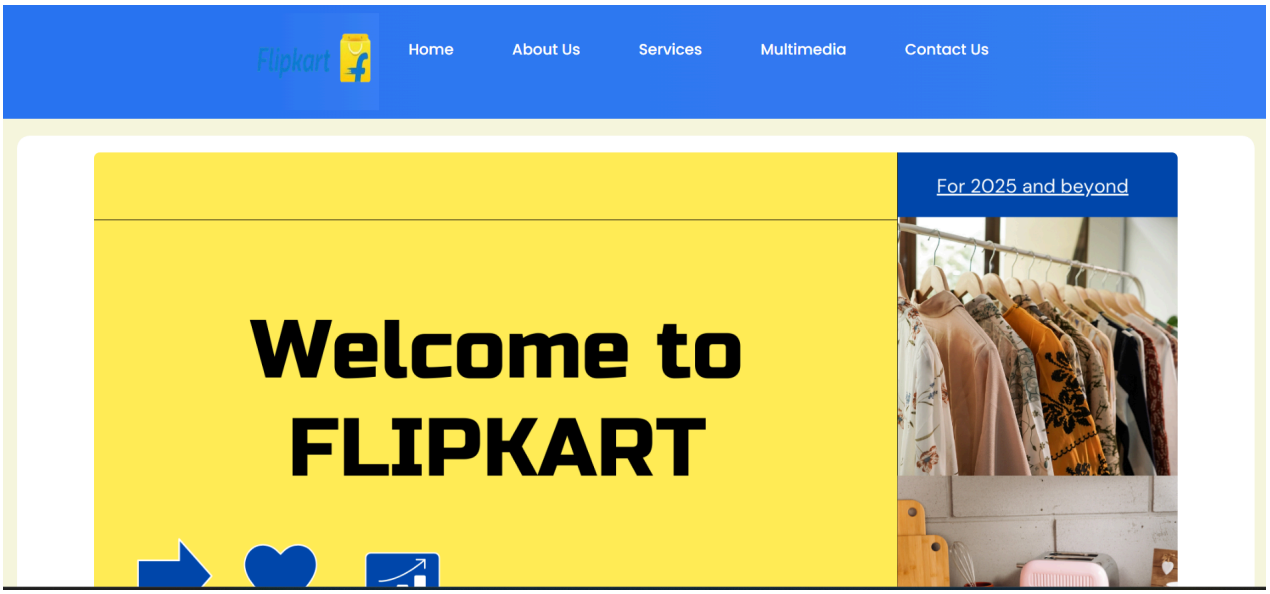

Application Composer
Looking for CloudFormation Designer?
Application Composer in CloudFormation console mode is an improvement from CloudFormation Designer. Please provide your feedback.
Go to Designer
Canvas
Template
Arrange
Validate
Update template
List
Resources
Search for a resource
Enhanced components (14)
API Gateway
Cognito UserPool
Cognito UserPoolClient
DynamoDB Table
EventBridge Event rule
AWSEBAutoScalingLaunchConfiguration
AWSEBInstanceLaunchWaitHandle
AWSEBEIP
AWSEBBeanstalkMetadata
AWSEBInstanceLaunchWaitCondition


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


Environment overview - events
Welcome
CloudFormation - Stack awseb
Application Composer | eu-nort
Create new pipeline | CodePipe
eu-north-1.console.aws.amazon.com/codesuite/codepipeline/pipeline/new?region=eu-north-1
Gmail
YouTube
Maps
Online C Compiler
All Bookmarks
aws
Services
Search
[Alt+S]
Stockholm
ShravaniAnilPatil
The GitHub (Version 1) action is not recommended
The selected action uses OAuth apps to access your GitHub repository. This is no longer the recommended method. Instead, choose the GitHub (Version 2) action to access your repository by creating a connection. Connections use GitHub Apps to manage authentication and can be shared with other resources. Learn more
Repository
ShravaniAnilPatil/IPlab_1
Branch
main
Change detection options
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.
GitHub webhooks (recommended)
Use webhooks in GitHub to automatically start my pipeline when a change occurs
AWS CodePipeline
Use AWS CodePipeline to check periodically for changes
Cancel
Previous
Next
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
29°C
Mostly cloudy
Search
ENG
IN
21:20
21-08-2024


eu-north-1.console.aws.amazon.com/codesuite/codepipeline/pipeline/new?region=eu-north-1
Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose pipeline settings
Step 2
Add source stage
Step 3
Add build stage
Step 4
Add deploy stage
Step 5
Review
Review Info
Step 5 of 5
Step 1: Choose pipeline settings
Pipeline settings
Pipeline name
Shravani_Pipelineaws
Pipeline type
V2
Execution mode
QUEUED
Artifact location
A new Amazon S3 bucket will be created as the default artifact store for your pipeline
Service role name
AWSCodePipelineServiceRole-eu-north-1-Shravani_Pipelineaws
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


eu-north-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/Shravani_Pipelineaws/view?region=eu-north-1
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
GitHub (Version 1)
Succeeded - 1 minute ago
e2898dbb
View details
e2898dbb Source: Update index.html
Disable transition
Deploy Succeeded
Start rollback
Pipeline execution ID: 8b6306d0-ade8-4407-b58c-86503eb3851e
Deploy
AWS Elastic Beanstalk
Succeeded - Just now
View details
e2898dbb Source: Update index.html
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Flipkart
Home
About Us
Services
Multimedia
Contact Us
Welcome to
FLIPKART
For 2025 and beyond

SHRAVANI ANIL PATIL **D15A-38**

Advanced DevOps Lab
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 resource availability 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

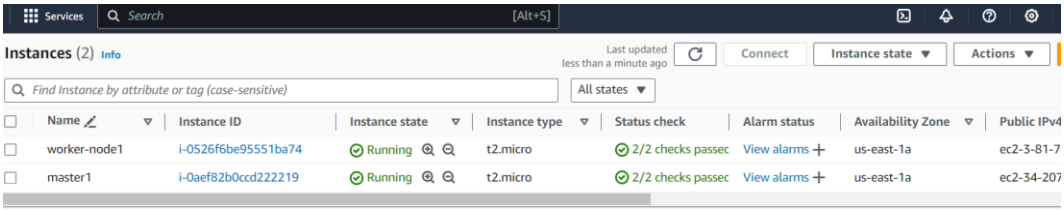
To install and set up a Kubernetes cluster, follow these general steps:

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.

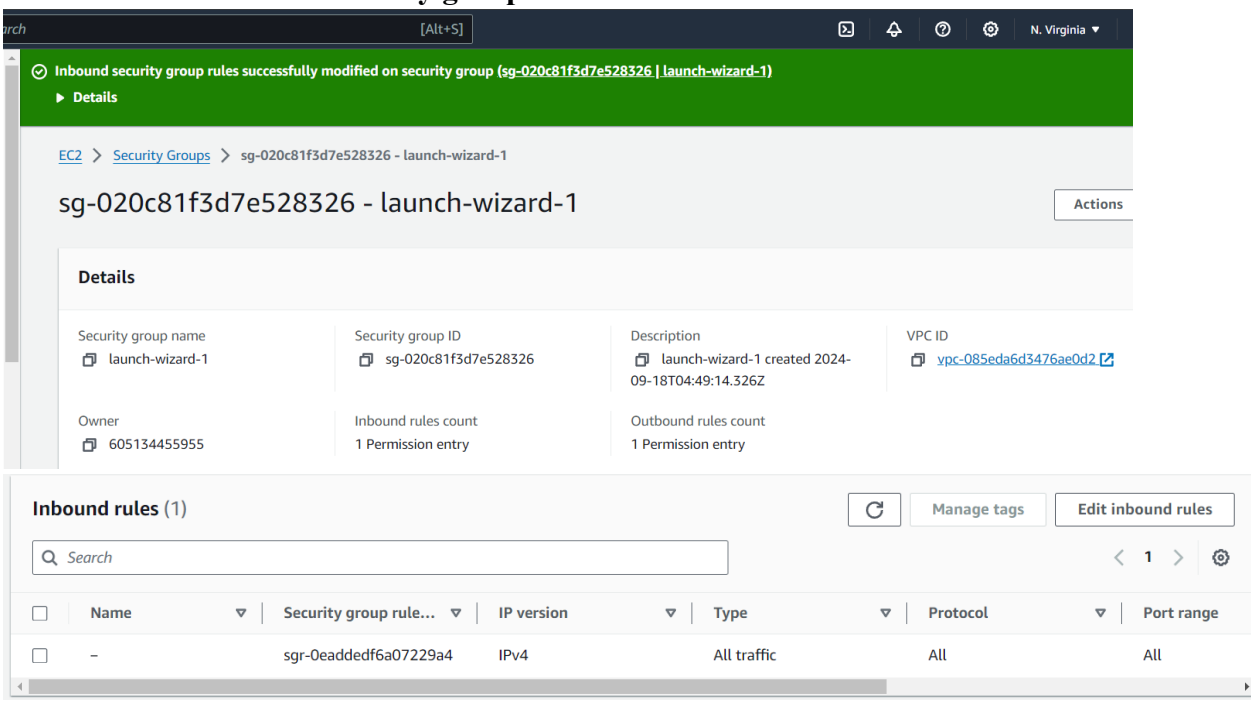
4. Join Worker Nodes: Once the Control Plane is set up, you can join worker nodes to the cluster using the token generated during the initialization.
5. Deploy Applications: After the cluster is up and running, you can deploy your applications using YAML configuration files that define the desired state of your Pods and Services..

Steps:

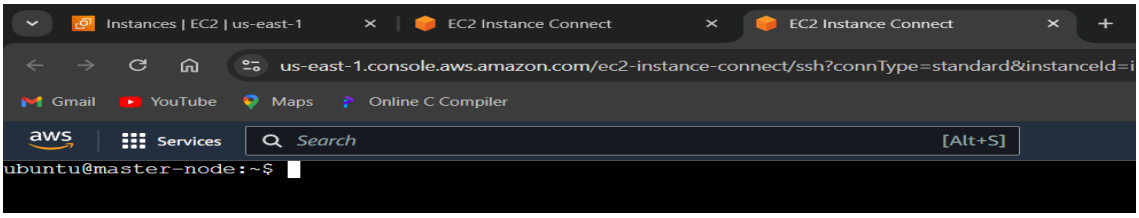
1.Creation of 2 EC2 Ubuntu Instances on AWS.



2.Edit inbound rules of security group 'launch-wizard-1' and set 'All Traffic'

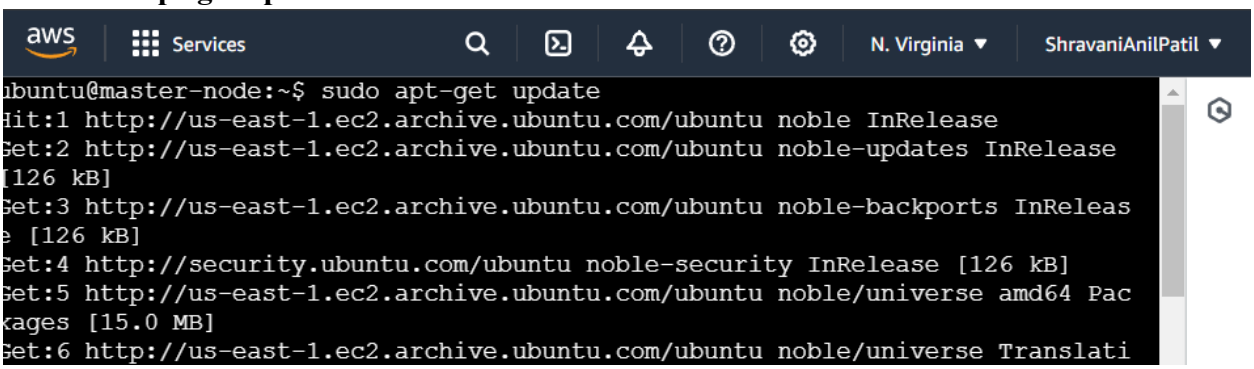


3. Set master and worker as hostname on respective servers

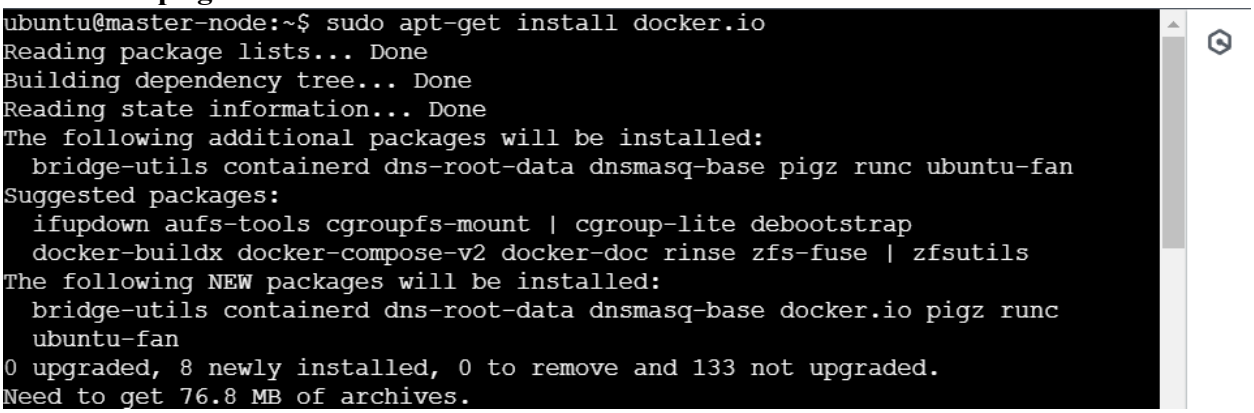


4.Installation of docker

4.1 - sudo apt-get update

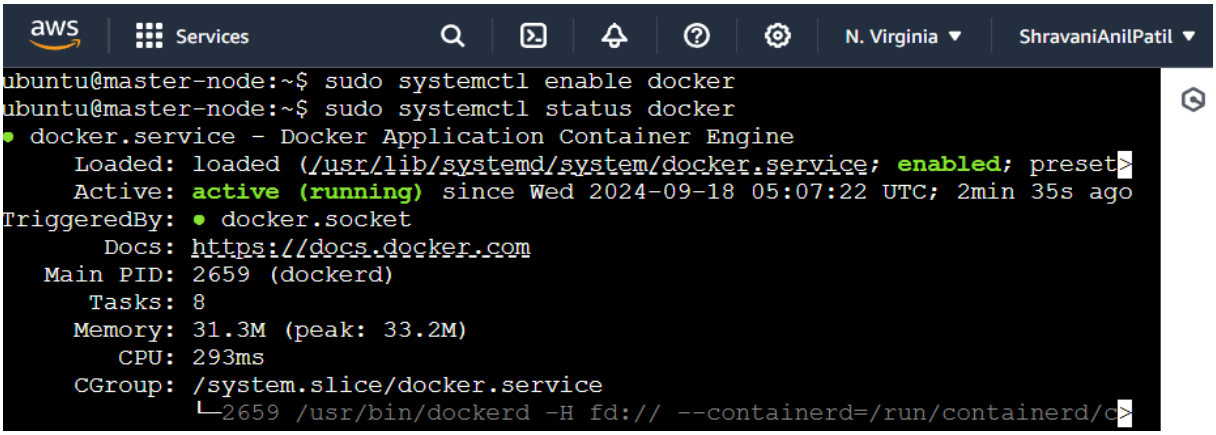


4.2 - sudo apt-get install docker.io



4.3 – sudo systmectl enable docker

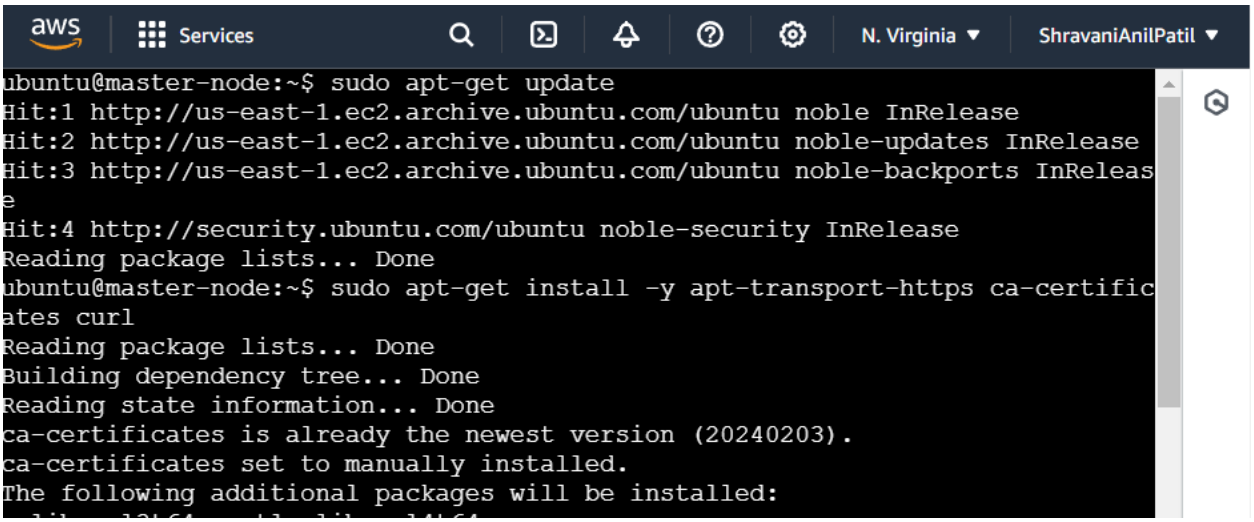
4.4 – sudo systemctl status docker



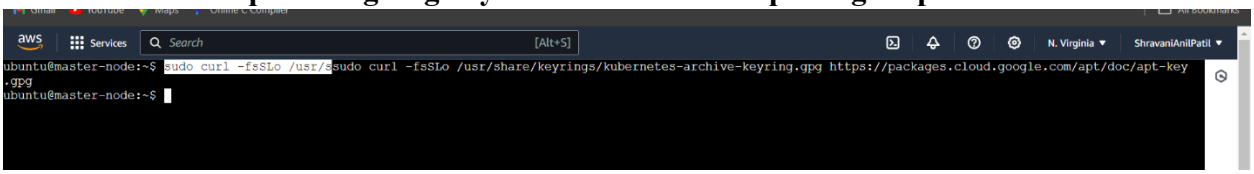
5- Installation of Kubernetes-

5.1 sudo apt-get update

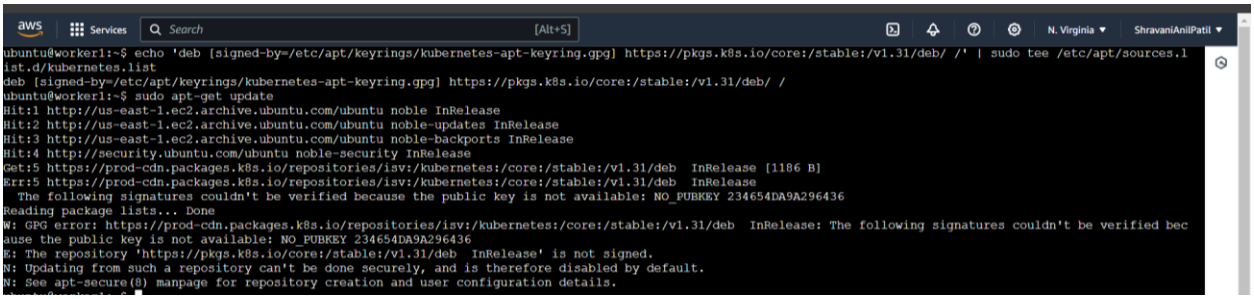
5.2 install ca certificate



5.3 Download the public signing key for the Kubernetes package repositories.

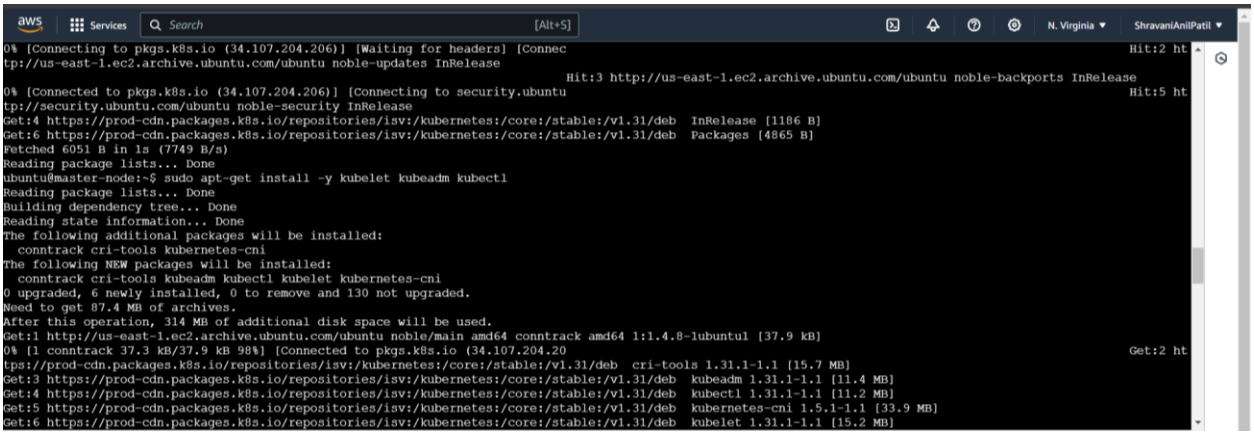


5.4 Add the appropriate Kubernetes apt repository

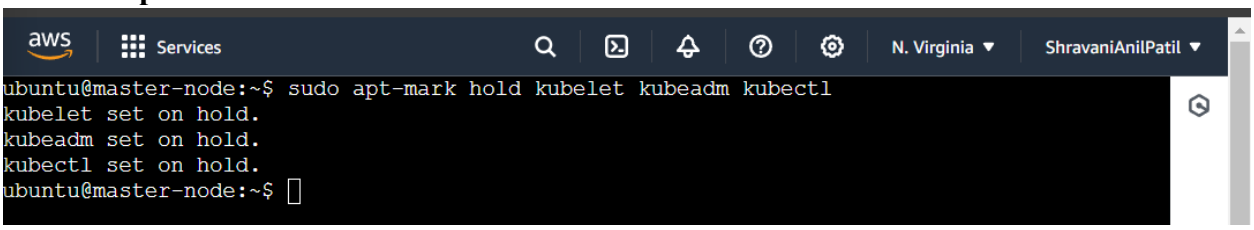


5.5 sudo apt-get update

5.6 sudo apt-get install -y kubelet kubeadm kubectl



5.7 sudo apt-mark hold kubelet kubeadm kubectl

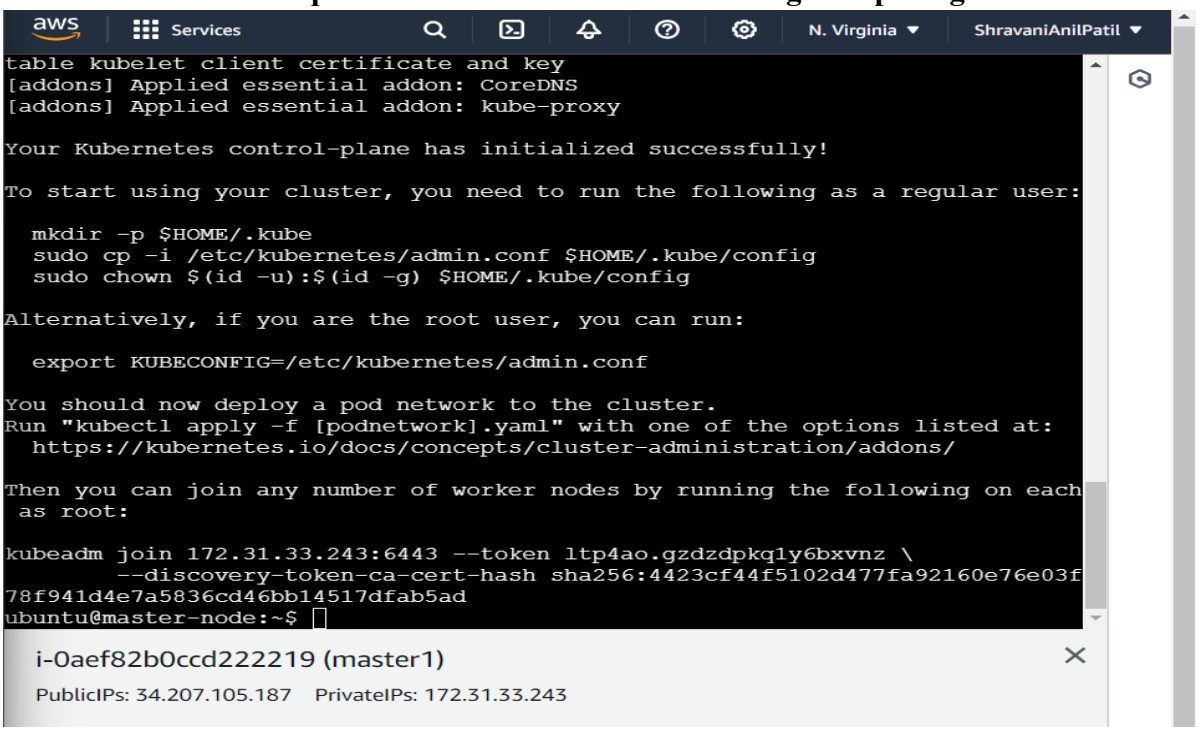


6. Kubernetes Deployment

6.1 sudo swapoff –a

6.2 Initialize Kubernetes on Master Node -

sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all



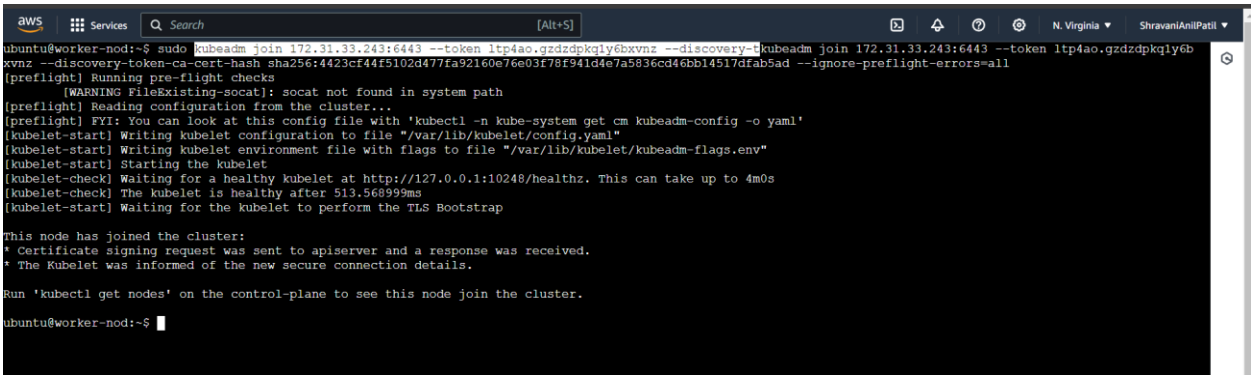
7.to create a directory for the cluster:

7.1mkdir -p $HOME/.kube

7.2sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

7.3sudo chown HOME/.kube/config

8. Deploy Pod Network to Cluster and Join Worker Node to Cluster



Verify that everything is running and communicating:

8.1kubectl get pods --all-namespaces

8.2kubectl get nodes

```
ubuntu@master-node:~$  kubectl get pods --all-namespaces
NAMESPACE      NAME                                  READY   STATUS             RESTARTS         AGE
kube-flannel   kube-flannel-ds-gx9xg                 1/1     Running            0                14m
kube-flannel   kube-flannel-ds-tl79d                 1/1     Running            0                6m2s
kube-system    coredns-7c65d6cfc9-nrns4              1/1     Running            0                23m
kube-system    coredns-7c65d6cfc9-pnh9p              1/1     Running            0                23m
kube-system    etcd-master-node                      1/1     Running            0                23m
kube-system    kube-apiserver-master-node            1/1     Running            0                23m
kube-system    kube-controller-manager-master-node   1/1     Running            0                23m
kube-system    kube-proxy-8rz72                      0/1     CrashLoopBackOff   7 (3m42s ago)    23m
kube-system    kube-proxy-w55hg                      0/1     CrashLoopBackOff   4 (29s ago)      6m2s
kube-system    kube-scheduler-master-node            1/1     Running            0                23m
ubuntu@master-node:~$  kubectl get nodes
NAME          STATUS   ROLES           AGE     VERSION
master-node   Ready    control-plane   23m     v1.31.1
worker-nod    Ready    <none>          6m22s   v1.31.1
ubuntu@master-node:~$
```

SHRAVANI ANIL PATIL **D15A-38**

Advanced DevOps Lab
Experiment:4

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

Overview of Kubernetes and Kubectl

What is Kubernetes?

Kubernetes, often referred to as K8s, is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. Originally developed by Google, it has become the industry standard for managing container workloads due to its flexibility and robust features.

Core Concepts of Kubernetes

Containers: These are lightweight, portable packages that include everything needed to run an application, ensuring consistency across different environments.

Pods: The smallest deployable units in Kubernetes, pods can contain one or more containers that share storage and network resources.

Nodes: A node is a worker machine in the Kubernetes cluster that runs at least one pod. Nodes can be either physical or virtual machines.

Clusters: A cluster comprises multiple nodes that run containerized applications. The control plane manages the cluster's state.

Services: Services provide stable endpoints for accessing pods and facilitate load balancing and service discovery.

Deployments: A deployment manages the lifecycle of pods, allowing users to specify the number of replicas and facilitating rolling updates and rollbacks.

Architecture of Kubernetes

Kubernetes follows a client-server architecture consisting of:

Control Plane: Manages the cluster and includes components like the API server (the front-end for the control plane), scheduler (assigns pods to nodes), controller manager (regulates cluster state), and etcd (a distributed key-value store for cluster data).

Worker Nodes: Each node runs components like kubelet (ensures containers are running), kube-proxy (manages network communication), and a container runtime (e.g., Docker).

Role of Kubectl in Kubernetes

What is Kubectl?

Kubectl is the command-line interface used to interact with the Kubernetes API server. It enables users to manage resources within a Kubernetes cluster effectively.

Configuration Files

Configuration files are essential for defining how resources should be created or modified within Kubernetes. Users can employ declarative configurations (using YAML/JSON files) or imperative commands directly in the terminal.

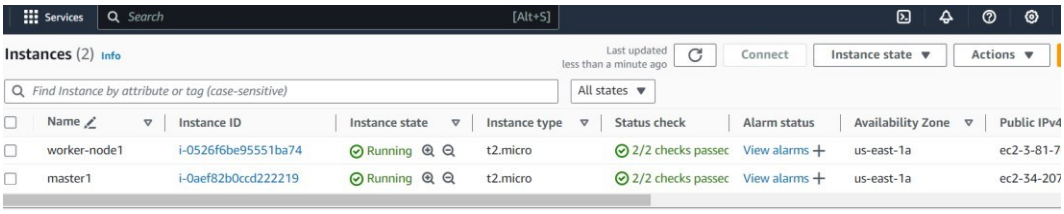
Deploying Applications on Kubernetes

Application Deployment Lifecycle

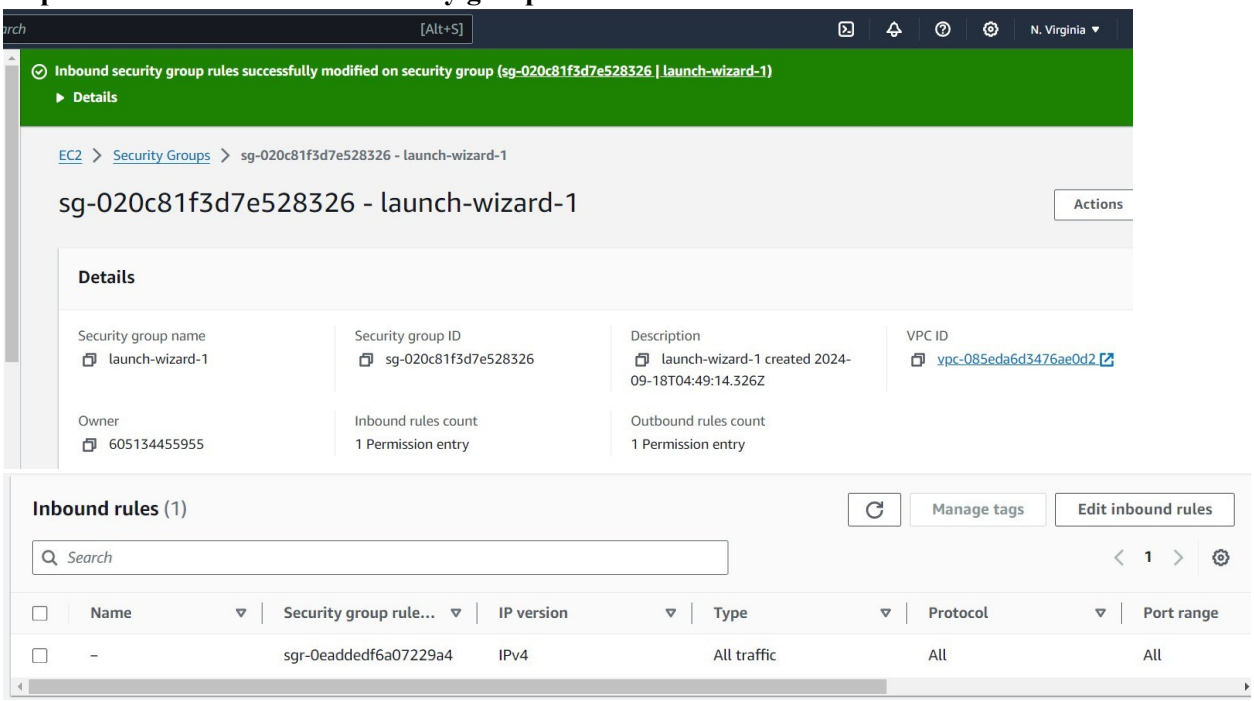
1. Define Application Requirements: Identify necessary resources such as CPU, memory, storage, etc.
2. Create Deployment Configurations: Write deployment manifests specifying container images, replicas for scaling, health checks, etc.
3. Deploying with Kubectl: Use kubectl commands like kubectl apply to deploy applications based on these configurations.
4. Monitoring and Scaling Applications: Monitor performance metrics and adjust deployments based on traffic demands.
5. Updating Applications: Modify deployment configurations for updates; Kubernetes supports rolling updates by default.
6. Rollback Capabilities: If an update causes issues, kubectl allows easy rollback to previous versions using commands like kubectl rollout undo.

Implementation

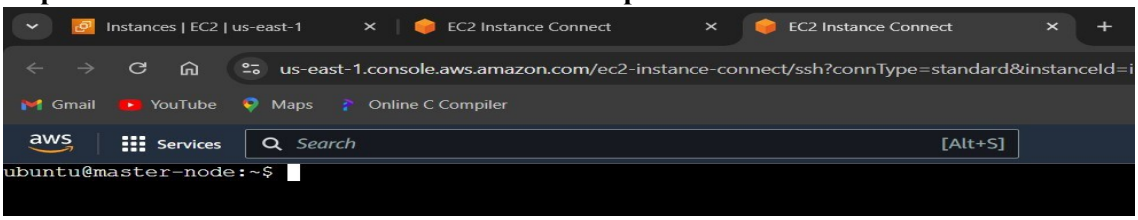
Step 1.Creation of 2 EC2 Ubuntu Instances on AWS.



Step 2.Edit inbound rules of security group 'launch-wizard-1' and set 'All Traffic'

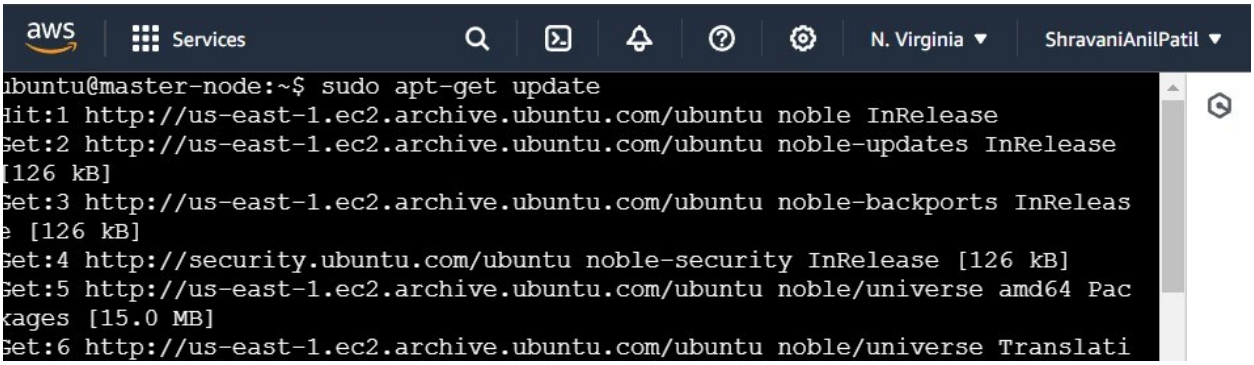


Step 3. Set master and worker as hostname on respective servers



Step 4.Installation of docker

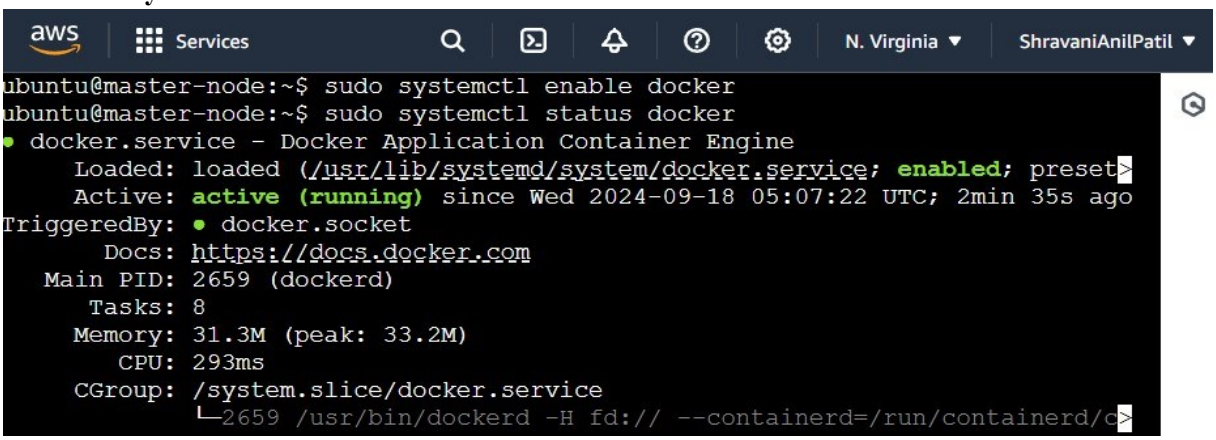
4.1 - sudo apt-get update



4.2 - sudo apt-get install docker.io

4.3 – sudo systmectl enable docker

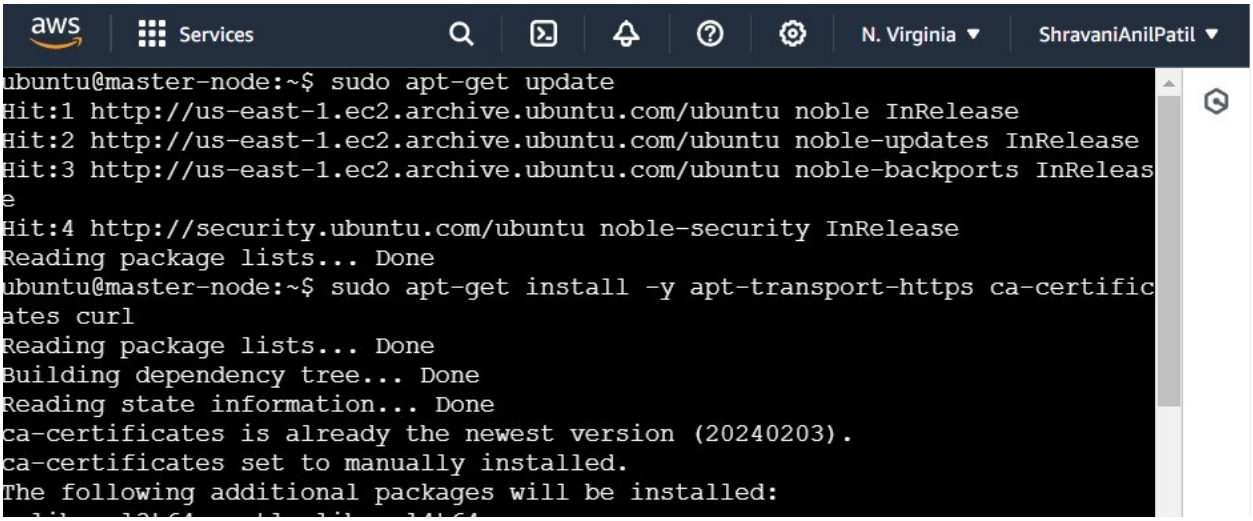
4.4 – sudo systemctl status docker



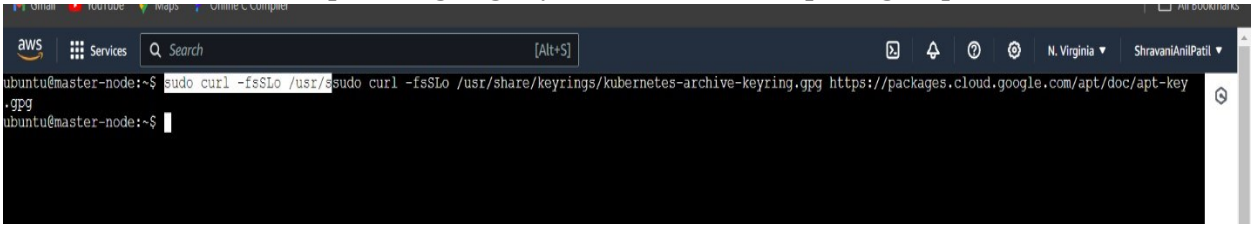
Step 5.Installation of Kubernetes-

5.1 sudo apt-get update

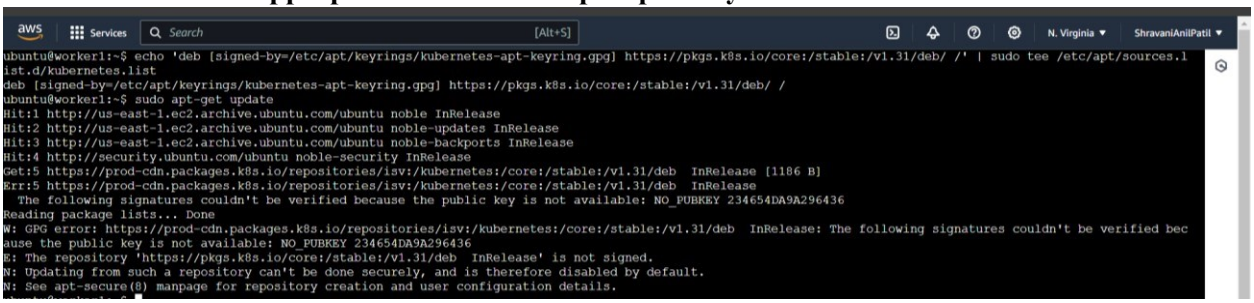
5.2 install ca certificate



5.3 Download the public signing key for the Kubernetes package repositories.

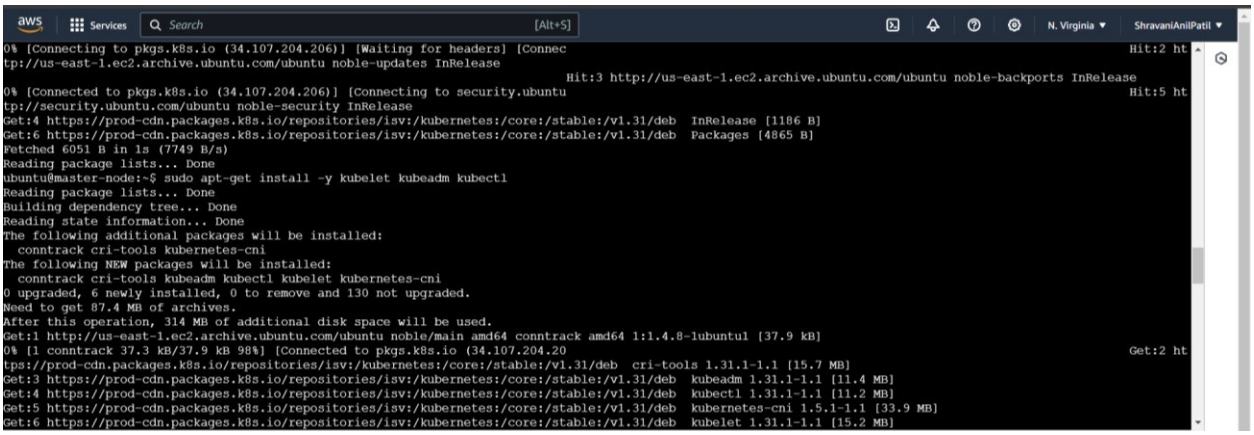


5.4 Add the appropriate Kubernetes apt repository

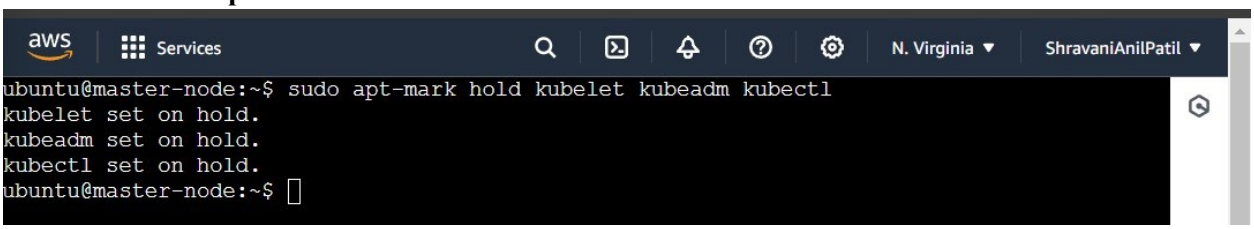


5.5 sudo apt-get update

5.6 sudo apt-get install -y kubelet kubeadm kubectl



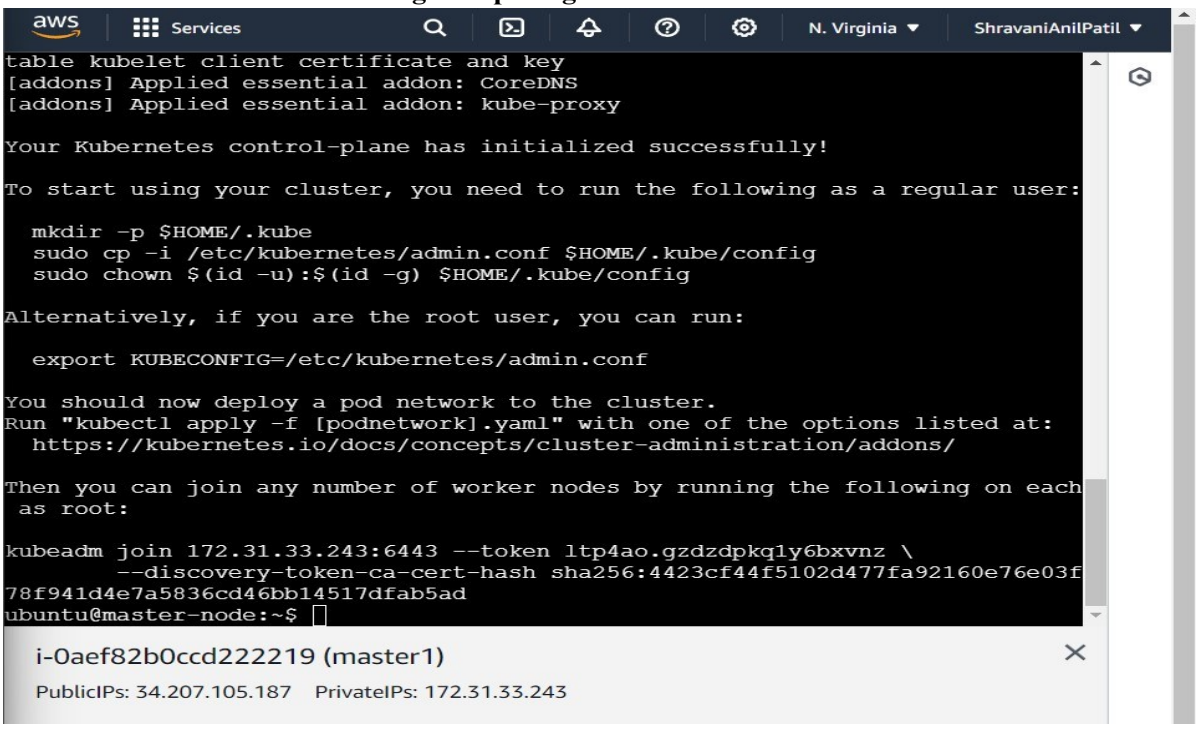
5.7 sudo apt-mark hold kubelet kubeadm kubectl



Step.6 Kubernetes Deployment

6.1 sudo swapoff –a

6.2 Initialize Kubernetes on Master Node - sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all



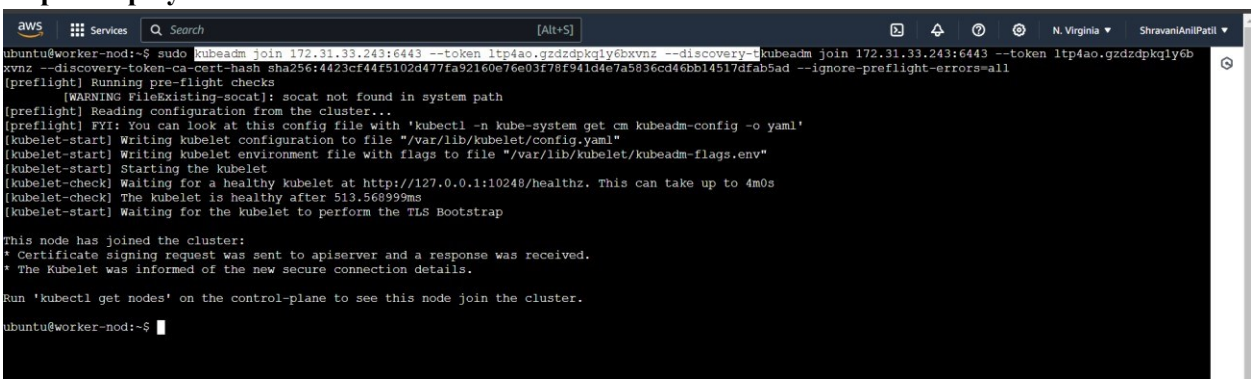
Step 7.to create a directory for the cluster:

7.1mkdir -p $HOME/.kube

7.2sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

7.3sudo chown HOME/.kube/config

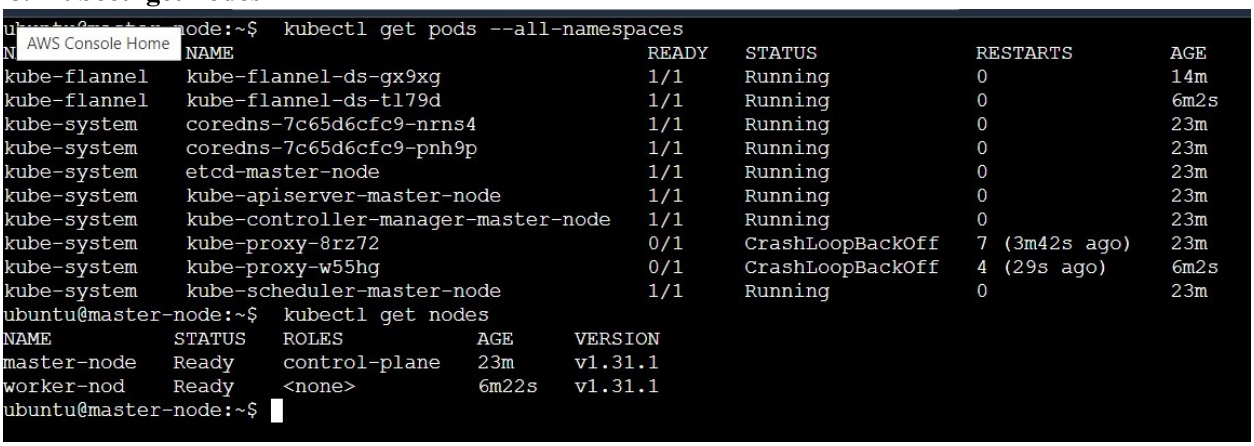
Step 8. Deploy Pod Network to Cluster and Join Worker Node to Cluster



Verify that everything is running and communicating:

8.1kubectl get pods --all-namespaces

8.2kubectl get nodes



Step 9. Create one file deploy.yaml

```
ubuntu@master-node:~$ sudo nano deploy.yaml
ubuntu@master-node:~$
```

```
ubuntu@master-node:~$ cat deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
ubuntu@master-node:~$
```

Step 10 : Create Deployment

```
aws | Services | Search [Alt+S]
ubuntu@master-node:~$ kubectl create -f deploy.yaml
deployment.apps/nginx-deployment created
ubuntu@master-node:~$
```

(EXTRA) – kubectl get namepaces

```
ubuntu@master-node:~$ kubectl get namespaces
NAME              STATUS   AGE
default           Active   9h
kube-flannel      Active   8h
kube-node-lease   Active   9h
kube-public       Active   9h
kube-system       Active   9h
ubuntu@master-node:~$
```

Step 11. After deployment verify the same:

```
ubuntu@master-node:~$ kubectl get deploy
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           3m13s
```

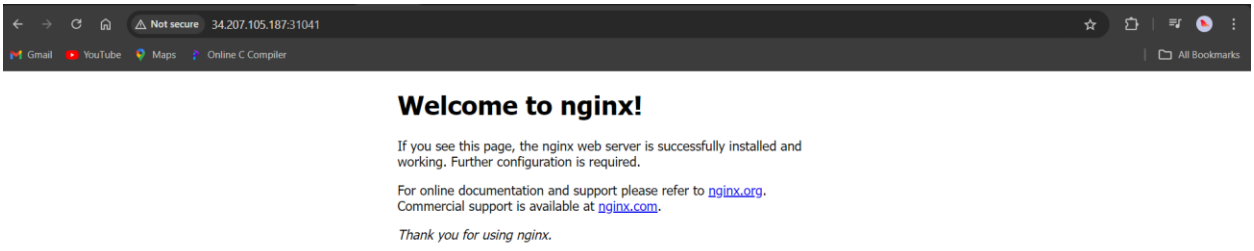
Step 12: Expose the Application: Create a service to expose the deployment.

```
ubuntu@master-node:~$ kubectl expose deployment.apps/nginx-deployment \
> --type="LoadBalancer"
service/nginx-deployment exposed
ubuntu@master-node:~$
```

Step 13:Verfiy the service

```
service/nginx-deployment exposed
ubuntu@master-node:~$ kubectl get svc
NAME               TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes         ClusterIP      10.96.0.1       <none>        443/TCP        4h43m
nginx-deployment   LoadBalancer   10.101.59.94    <pending>     80:31041/TCP   4m34s
ubuntu@master-node:~$
```

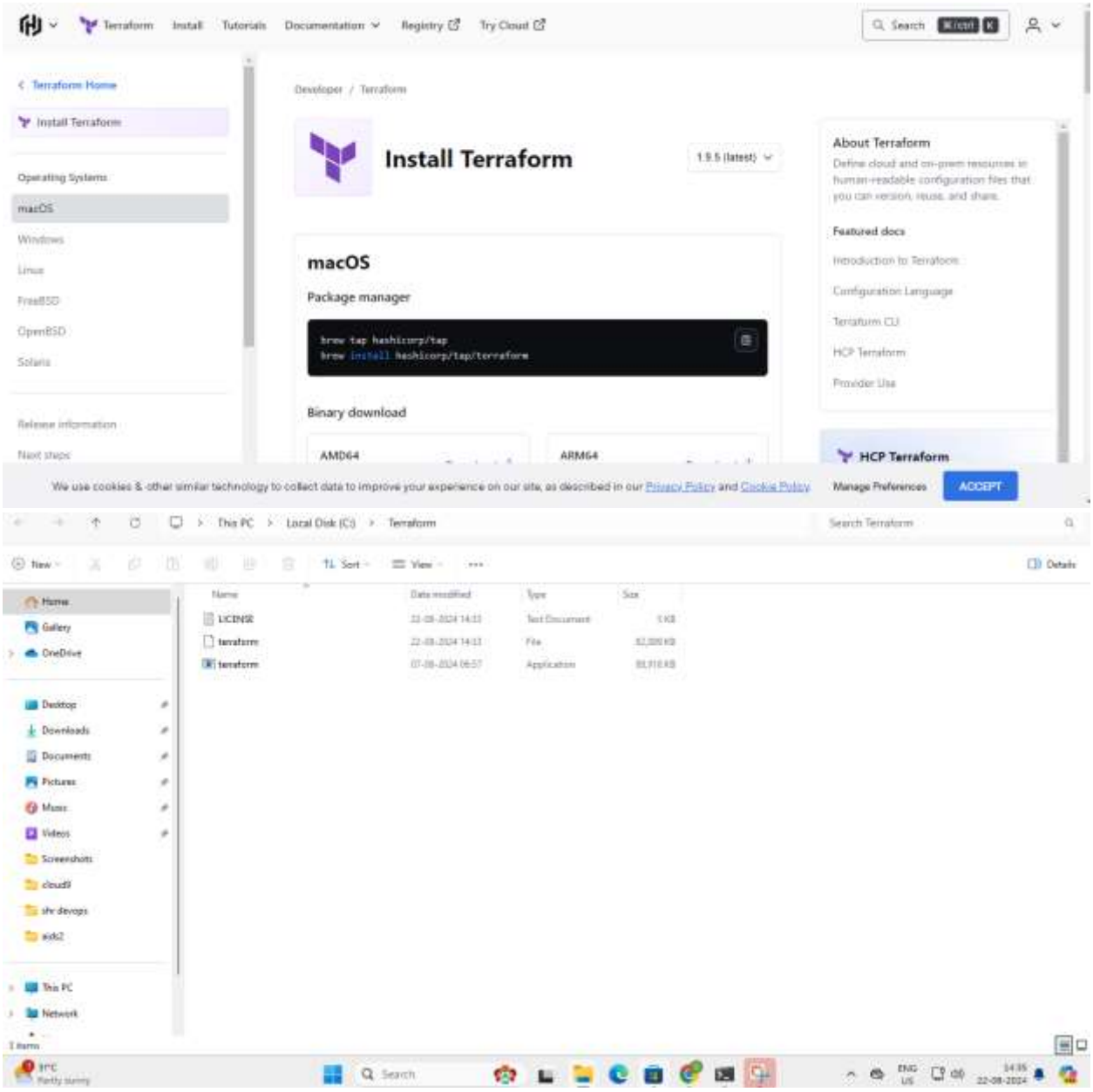
Step 14: Access the application

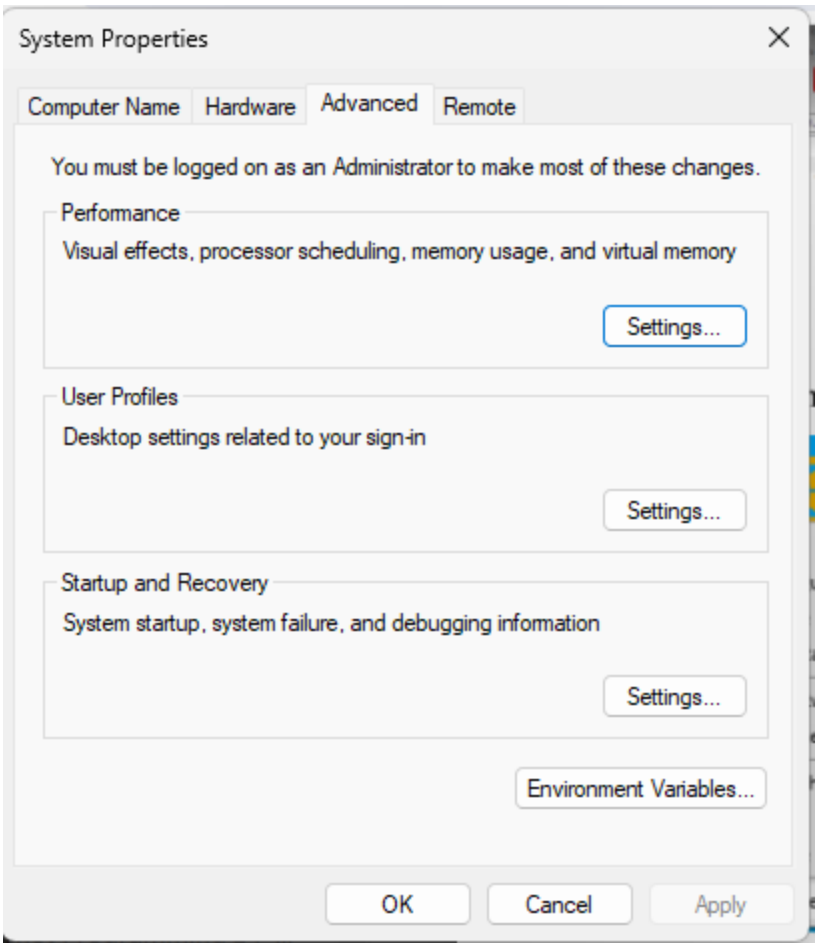


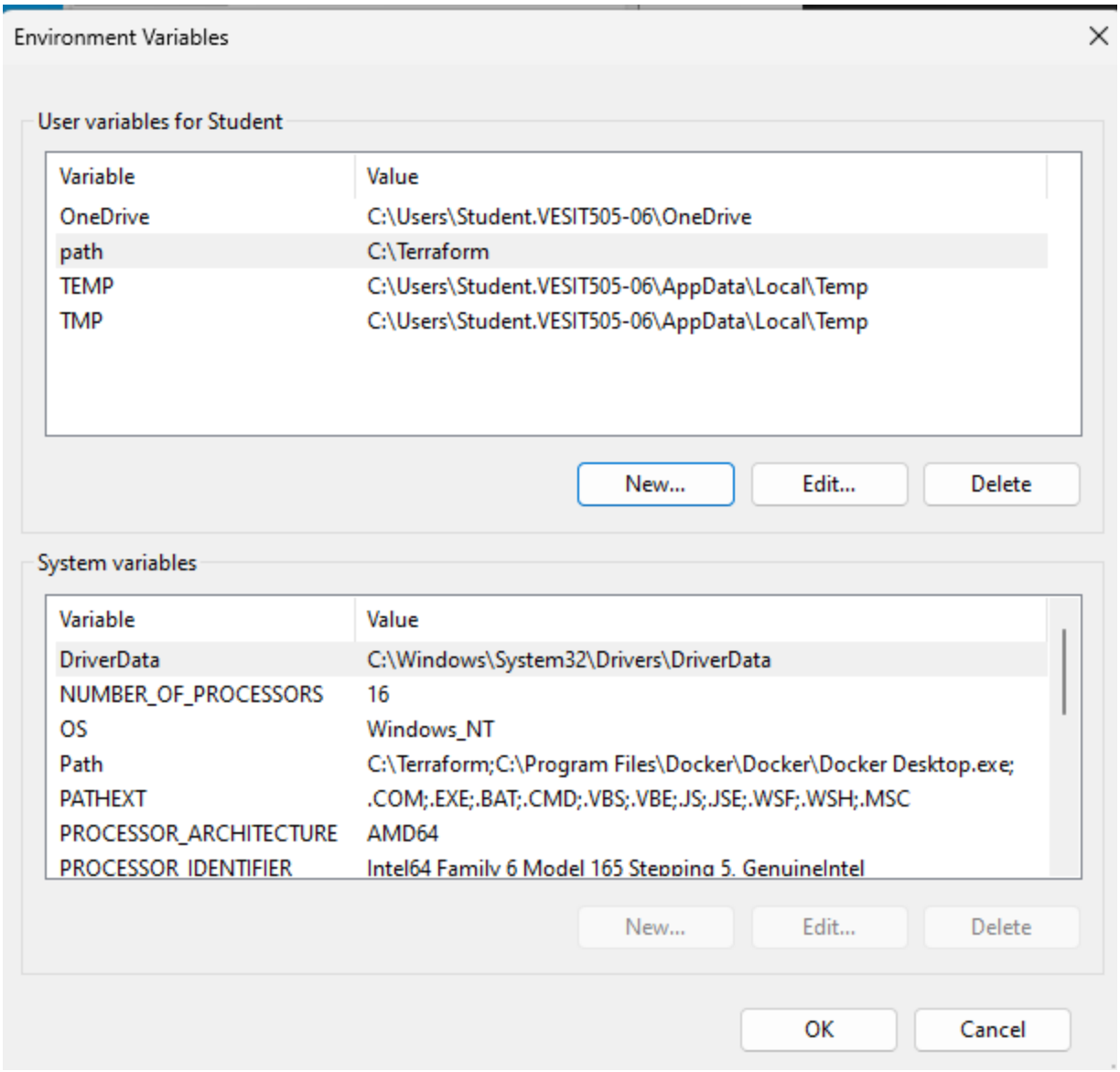
Shravani Anil Patil **D15A-38**

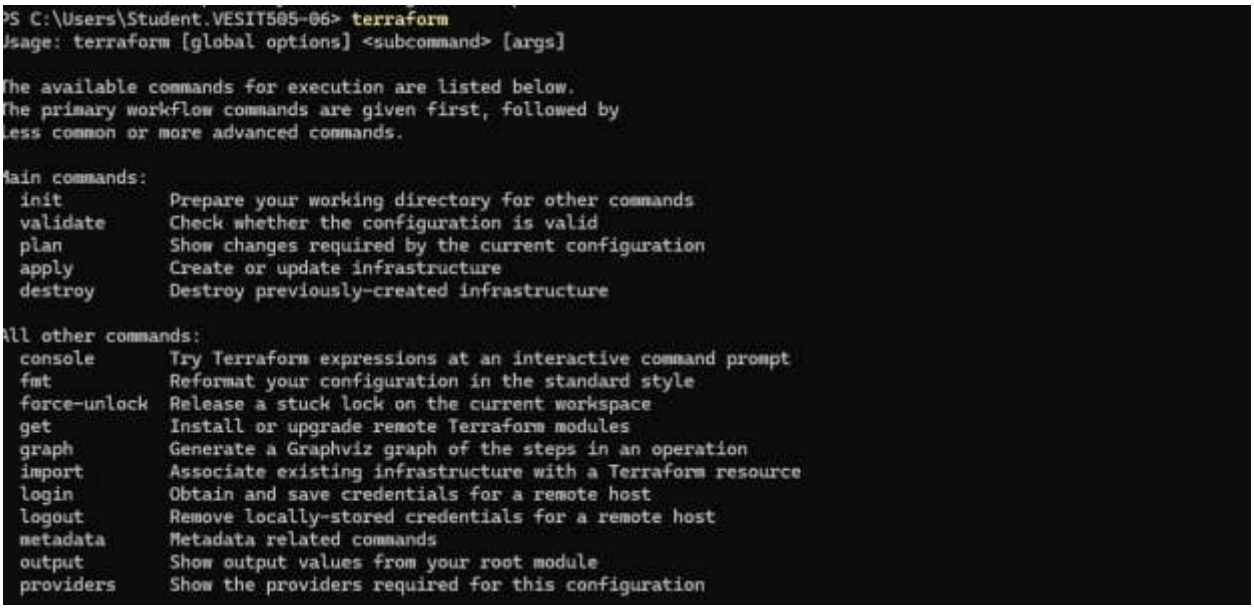
Experiment No 5 : Terraform

Aim : Installation and Configuration of Terraform in Window




System Properties
Computer Name
Hardware
Advanced
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 Student
Variable
Value
OneDrive
C:\Users\Student.VESIT505-06\OneDrive
path
C:\Terraform
TEMP
C:\Users\Student.VESIT505-06\AppData\Local\Temp
TMP
C:\Users\Student.VESIT505-06\AppData\Local\Temp
New...
Edit...
Delete
System variables
Variable
Value
DriverData
C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS
16
OS
Windows_NT
Path
C:\Terraform;C:\Program Files\Docker\Docker\Docker Desktop.exe;
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE
AMD64
PROCESSOR_IDENTIFIER
Intel64 Family 6 Model 165 Stepping 5, GenuineIntel
New...
Edit...
Delete
OK
Cancel


PS C:\Users\Student.VESIT505-06> 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

Shravani Anil Patil
D15A-38

EXPERIMENT NO. 6

Aim :To Build, change, and destroy AWS 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.

- **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 new desired 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.

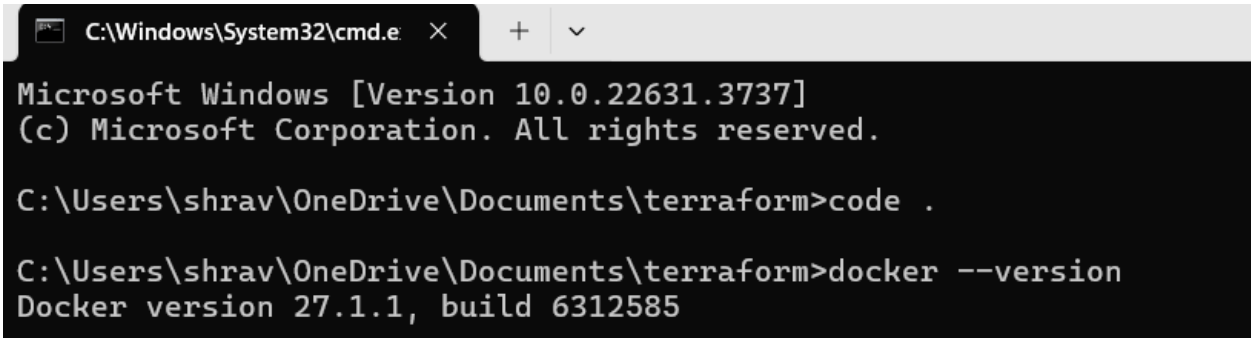
2. **Automation:** Manual processes are reduced, and infrastructure is provisioned, updated, and destroyed automatically based on code.
3. **Version Control:** Infrastructure configurations can be stored in version control systems (like Git), allowing teams to track changes, collaborate, and roll back 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.

Implementation :

Terraform and Docker -

Step 1 : check docker installation and version

```
C:\Users\shrav\OneDrive\Documents\terraform>docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

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

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx
  compose*    Docker Compose
  container   Manage containers
```



Step 2 : create docker.tf file and write following code for terraform and docker

Code -

```
terraform { required_providers {
    docker = {
      source = "kreuzwerker/docker" version =
      "~> 3.0.1"
   }
  }
}
provider "docker" {
host         = "npipe:////.//pipe//docker_engine"
}
resource "docker_image" "nginx" {
  name = "nginx:latest" keep_locally = false
} resource "docker_container" "nginx" {
  image   =   docker_image.nginx.image_id
  name = "tutorial" ports {
    internal = 80 external
    = 8000 }
}
```

credentials.txt | provider.tf | docker.tf | main.tf

docker > docker.tf > provider "docker"

```
terraform {
    required_providers {
docker = {
source = "kreuzwerker/docker"
version = "~> 3.0.1"
}
}
}
provider "docker" {
host    = "npipe:////.//pipe//docker_engine"
}
 resource "docker_image" "nginx" {
name    = "nginx:latest"
keep_locally = false
}
resource "docker_container" "nginx" {
image = docker_image.nginx.image_id
name = "tutorial"
ports {
    internal = 80
     external = 8000
     }
}

```

Step 3 : Type terraform init command to initialize terraform backend

```
shrav@LAPTOP-0MELEBGI MINGW64 ~/OneDrive/Documents/terraform/docker
$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "~> 3.0.1"...
- Installing kreuzwerker/docker v3.0.2...
- Installed kreuzwerker/docker v3.0.2 (self-signed, key ID BD080C4571C6104C)
Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

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

Step 4(EXTRA) : type terraform fmt and validate commands .
The two Terraform commands – terraform validate and terraform fmt – are used to maintain a clean, error-free, and well-structured Terraform codebase.

```
shrav@LAPTOP-0MELEBGI MINGW64 ~/OneDrive/Documents/terraform/docker
$ terraform fmt
docker.tf

shrav@LAPTOP-0MELEBGI MINGW64 ~/OneDrive/Documents/terraform/docker
$ terraform validate
Success! The configuration is valid.
```

Step 5 : Type Terraform plan command to create execution plan .

```
shrav@LAPTOP-0MELEBGI MINGW64 ~/OneDrive/Documents/terraform/docker
$ terraform plan

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

Terraform will perform the following actions:

  # docker_container.nginx will be created
  + resource "docker_container" "nginx" {
      + attach                                      = false
      + bridge                                      = (known after apply)
      + command                                     = (known after apply)
      + container_logs                              = (known after apply)
      + container_read_refresh_timeout_milliseconds = 15000
      + entrypoint                                  = (known after apply)
      + env                                         = (known after apply)
      + exit_code                                   = (known after apply)
      + hostname                                    = (known after apply)
      + id                                          = (known after apply)
      + image                                       = (known after apply)
      + init                                        = (known after apply)
      + ipc_mode                                    = (known after apply)
      + log_driver                                  = (known after apply)
      + logs                                        = false
      + must_run                                    = true
      + name                                        = "tutorial"
      + network_data                                = (known after apply)
      + read_only                                   = false
      + remove_volumes                              = true
      + restart                                     = "no"
      + rm                                          = false
      + runtime                                     = (known after apply)
      + security_opts                               = (known after apply)
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  PORTS                                  bash -

      + 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
      + wait                                        = false
      + wait_timeout                                = 60

      + healthcheck (known after apply)

      + labels (known after apply)

      + ports {
          + external = 8000
          + internal = 80
          + ip       = "0.0.0.0"
          + protocol = "tcp"
        }
    }

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

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

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.
```

Step 6 : Type terraform apply to apply changes .

```
docker_image.nginx: Still creating... [1m40s elapsed]
docker_image.nginx: Still creating... [1m50s elapsed]
docker_image.nginx: Still creating... [2m0s elapsed]
docker_image.nginx: Still creating... [2m10s elapsed]
docker_image.nginx: Still creating... [2m20s elapsed]
docker_image.nginx: Still creating... [2m30s elapsed]
docker_image.nginx: Still creating... [2m40s elapsed]
docker_image.nginx: Still creating... [2m50s elapsed]
docker_image.nginx: Still creating... [3m0s elapsed]
docker_image.nginx: Creation complete after 3m1s [id=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:latest]
docker_container.nginx: Creating...
docker_container.nginx: Creation complete after 8s [id=0b844cbfdc3ee0fca75bbf5a6577f7a7d824bd4867787cd570085b011ecdb82e]

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

shrav@LAPTOP-0MELEBGI MINGW64 ~/OneDrive/Documents/terraform/docker
$
```

Step 7 : Docker container after step 6 execution BEFORE -
AFTER -

```
shrav@LAPTOP-0MELEBGI MINGW64 ~/OneDrive/Documents/terraform/docker
$ docker container list
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                   NAMES
0b844cbfdc3e   5ef79149e0ec   "/docker-entrypoint.…"   2 minutes ago   Up 2 minutes   0.0.0.0:8000->80/tcp   tutorial

shrav@LAPTOP-0MELEBGI MINGW64 ~/OneDrive/Documents/terraform/docker
$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
nginx        latest    5ef79149e0ec   2 weeks ago   188MB
```

Step 8 (EXTRA) : Execution of change .

docker.tf

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "~> 3.0.1"
    }
  }
}

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

resource "docker_image" "nginx" {
  name         = "nginx:latest"
  keep_locally = false
}

resource "docker_container" "nginx" {
  image = docker_image.nginx.image_id
  name  = "tutorial"
  ports {
    internal = 80
    external = 8080
  }
}

```

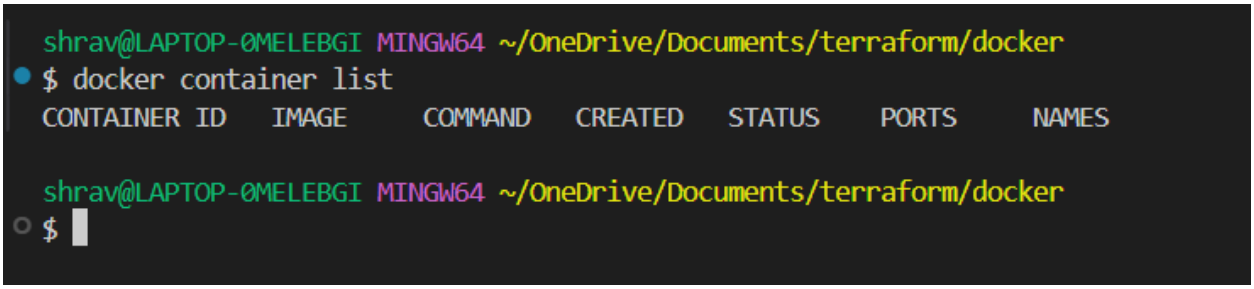
```
PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE   PORTS

          + tmpfs                                    = (known after apply)
          + tty                                      = (known after apply)
          + user                                     = (known after apply)
          + userns_mode                              = (known after apply)
          + wait                                     = (known after apply)
          + wait_timeout                             = (known after apply)
          + working_dir                              = (known after apply)
        } -> (known after apply)

      ~ ports {
          ~ external = 8000 -> 8080 # forces replacement
            # (3 unchanged attributes hidden)
        }
    }

Plan: 1 to add, 0 to change, 1 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_container.nginx: Destroying... [id=0b844cbfdc3ee0fca75bbf5a6577f7a7d824bd4867787cd570085b011ecdb82e]
docker_container.nginx: Destruction complete after 3s
docker_container.nginx: Creating...
```

Step 9 : terraform destroy to destroy infrastructure.

```
shrav@LAPTOP-0MELEBGI MINGW64 ~/OneDrive/Documents/terraform/docker
$ terraform destroy
docker_image.nginx: Refreshing state... [id=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:latest]
docker_container.nginx: Refreshing state... [id=fd0e4e24940ef385ecf3b680240ce727964686a2af4e4cfea5b7d70b4cd9a32d]

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.nginx will be destroyed
  - resource "docker_image" "nginx" {
      - id           = "sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:latest" -> null
      - image_id     = "sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03c" -> null
      - keep_locally = false -> null
      - name         = "nginx:latest" -> null
      - repo_digest  = "nginx@sha256:447a8665cc1dab95b1ca778e162215839ccbb9189104c79d7ec3a81e14577add" -> 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.
```

```
  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.nginx: Destroying... [id=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:latest]
  docker_image.nginx: Destruction complete after 2s

  Destroy complete! Resources: 1 destroyed.

  shrav@LAPTOP-0MELEBGI MINGW64 ~/OneDrive/Documents/terraform/docker
  $
```

Step 10 : Docker after destroy command.



Shravani Anil Patil **D15A-38**

Advanced Devops
Experiment No:07

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

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

What problems does SAST solve?
SAST takes place very early in the software development life cycle (SDLC) as it does not require a working application and can take place without code being executed. It helps developers identify vulnerabilities in the initial stages of development and quickly resolve issues without breaking builds or passing on vulnerabilities to the final release of the application.

SAST tools give developers real-time feedback as they code, helping them fix issues before they pass the code to the next phase of the SDLC. This prevents security-related issues from being considered an afterthought. SAST tools also provide graphical representations of the issues found, from source to sink. These help you navigate the code easier. Some tools point out the exact location of vulnerabilities and highlight the risky code. Tools can also provide in-depth guidance on how to fix issues and the best place in the code to fix them, without requiring deep security domain expertise. It's important to note that SAST tools must be run on the application on a regular basis, such as during daily/monthly builds, every time code is checked in, or during a code release.

Why is SAST important?
Developers dramatically outnumber security staff. It can be challenging for an organization to find the resources to perform code reviews on even a fraction of its applications. A key strength of SAST tools is the ability to analyze 100% of the codebase. Additionally, they are much faster than manual secure code reviews performed by humans. These tools can scan millions of lines of code in a matter of minutes. SAST tools automatically identify critical vulnerabilities—such as buffer overflows, SQL injection, cross-site scripting, and others—with high confidence. Thus, integrating static analysis into the SDLC can yield dramatic results in the overall quality of the code developed.

What are the key steps to run SAST effectively?

There are six simple steps needed to perform SAST efficiently in organizations that have a very large number of applications built with different languages, frameworks, and platforms.
1. Finalize the tool. Select a static analysis tool that can perform code reviews of applications written in the programming languages you use. The tool should also be able to comprehend the underlying framework used by your software.
2. Create the scanning infrastructure, and deploy the tool. This step involves handling the licensing requirements, setting up access control and authorization, and procuring the resources required (e.g., servers and databases) to deploy the tool.
3. Customize the tool. Fine-tune the tool to suit the needs of the organization. For example, you might configure it to reduce false positives or find additional security vulnerabilities by writing new rules or updating existing ones. Integrate the tool into the build environment, create dashboards for tracking scan results, and build custom reports.
4. Prioritize and onboard applications. Once the tool is ready, onboard your applications. If you have a large number of applications, prioritize the high-risk applications to scan first. Eventually, all your applications should be onboarded and scanned regularly, with application scans synced with release cycles, daily or monthly builds, or code check-ins.
5. Analyze scan results. This step involves triaging the results of the scan to remove false positives. Once the set of issues is finalized, they should be tracked and provided to the deployment teams for proper and timely remediation.
6. Provide governance and training. Proper governance ensures that your development teams are employing the scanning tools properly. The software security touchpoints should be present within the SDLC. SAST should be incorporated as part of your application development and deployment process.

Integrating Jenkins with SonarQube:

Windows installation
Step 1 Install JDK 1.8
Step 2 download and install jenkins
installing-the-default-jre-jdk
Step 1 Install JDK 1.8
sudo apt-get install openjdk-8-jre
sudo apt install default-jre /

Open SSH

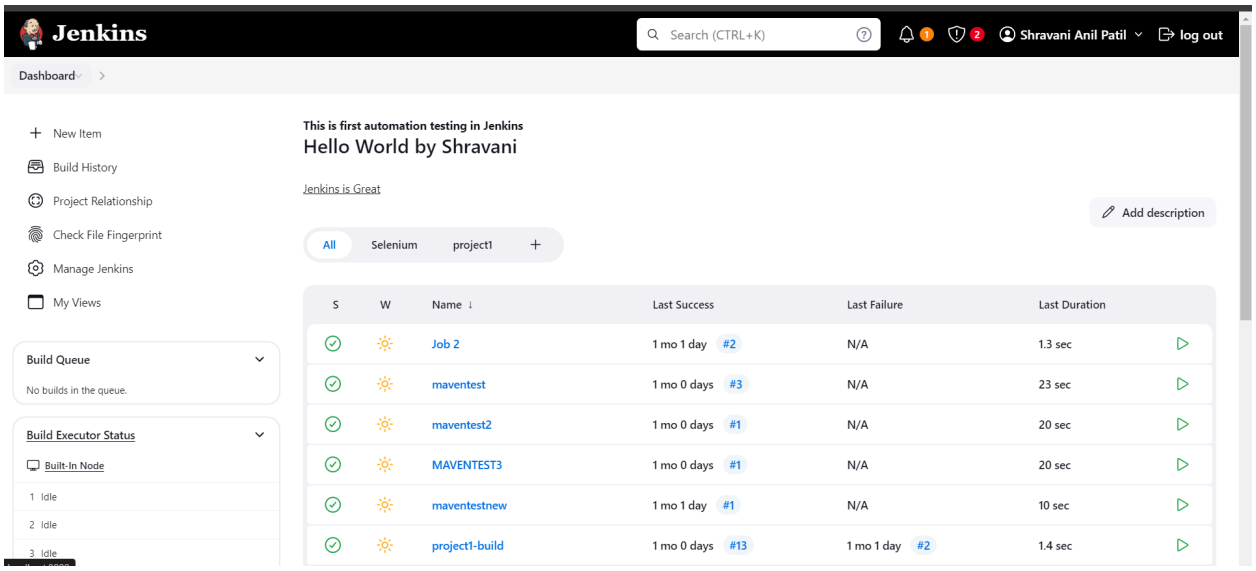
Prerequisites:
● Jenkins installed
● Docker Installed (for SonarQube)

(sudo apt-get install docker-ce=5:20.10.15~3-0~ubuntu-jammy docker-ce-cli=5:20.10.15~3-0~ubuntu-jammy containerd.io docker-compose-plugin)

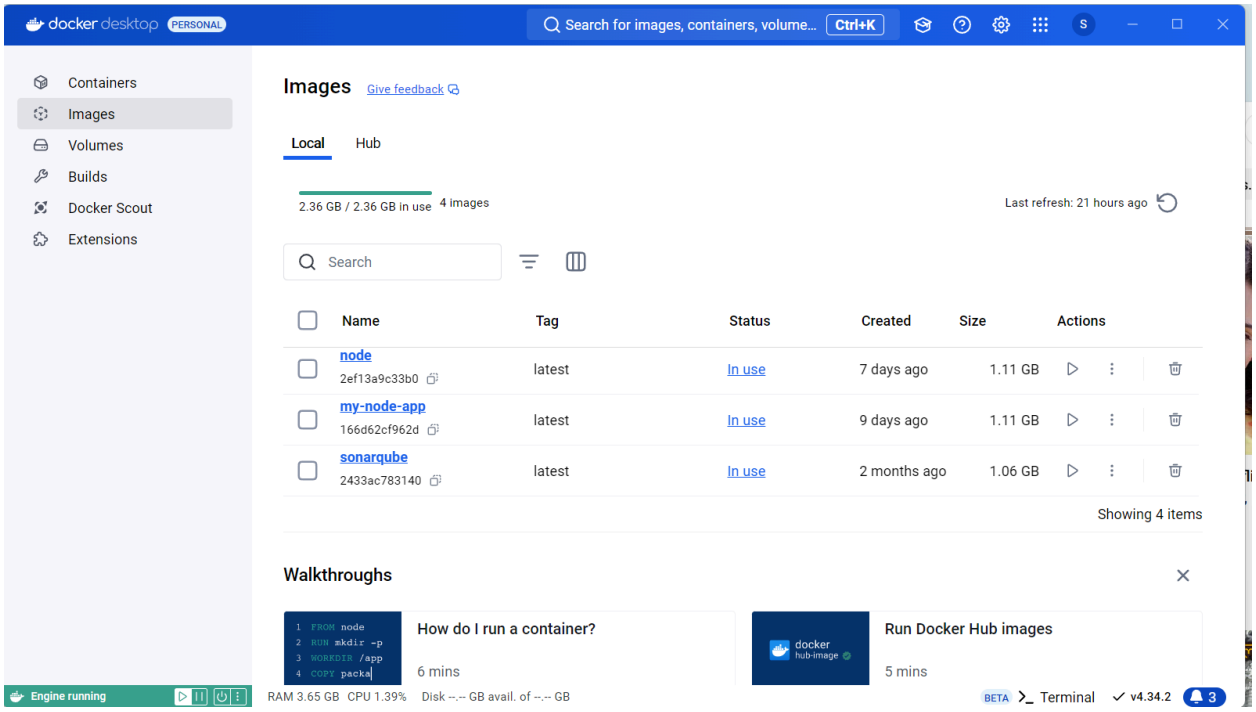
- SonarQube Docker Image

Steps to integrate Jenkins with SonarQube

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

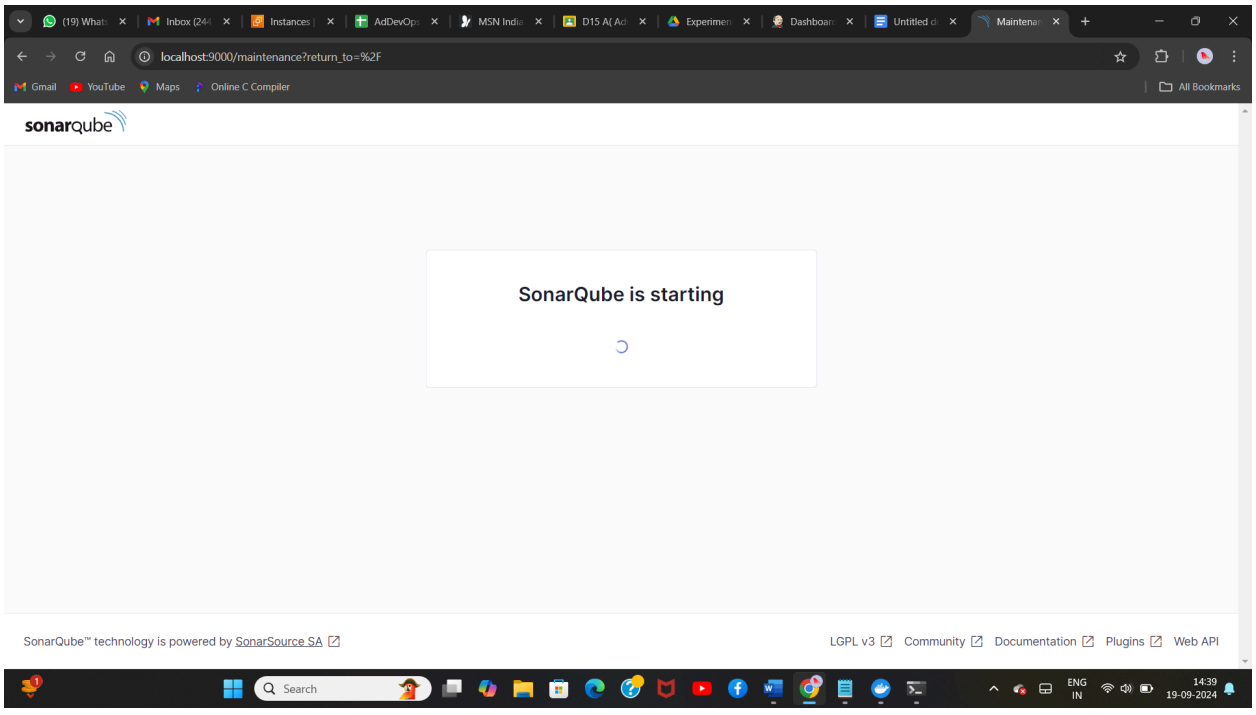


2. Run SonarQube in a Docker container using this command
Ensure Docker engine is running



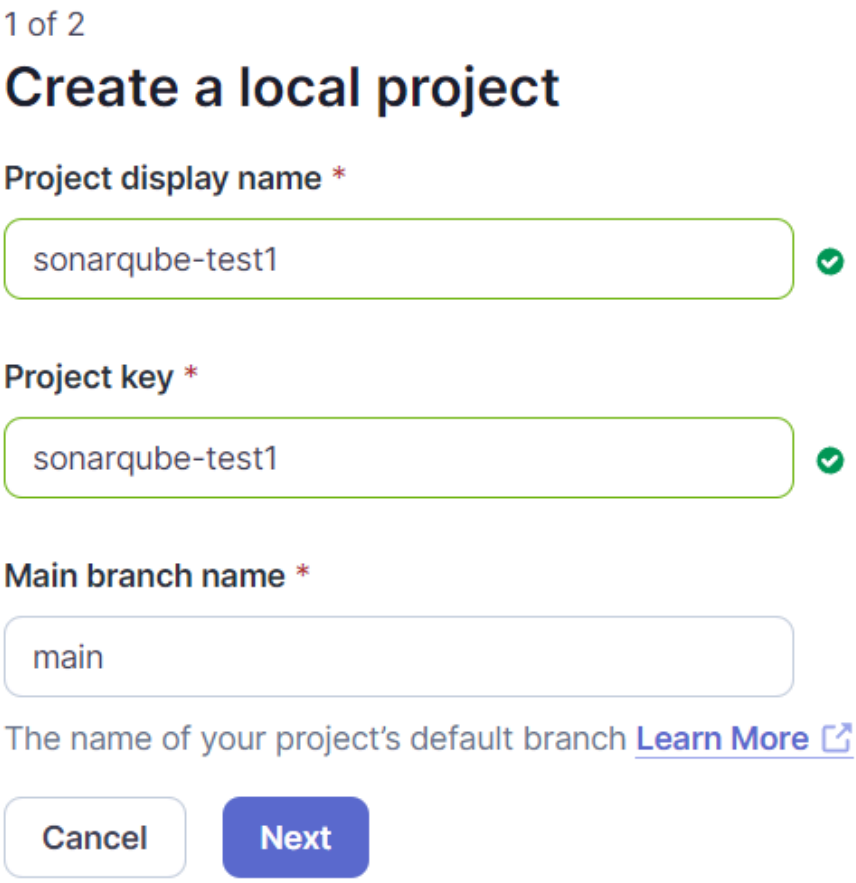
```
Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

C:\Users\shrav>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
e4d231de58da6871a773bbd5b77ac510bb10d0ff003c24d82542d1f5cbce4ea1

C:\Users\shrav>
```

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

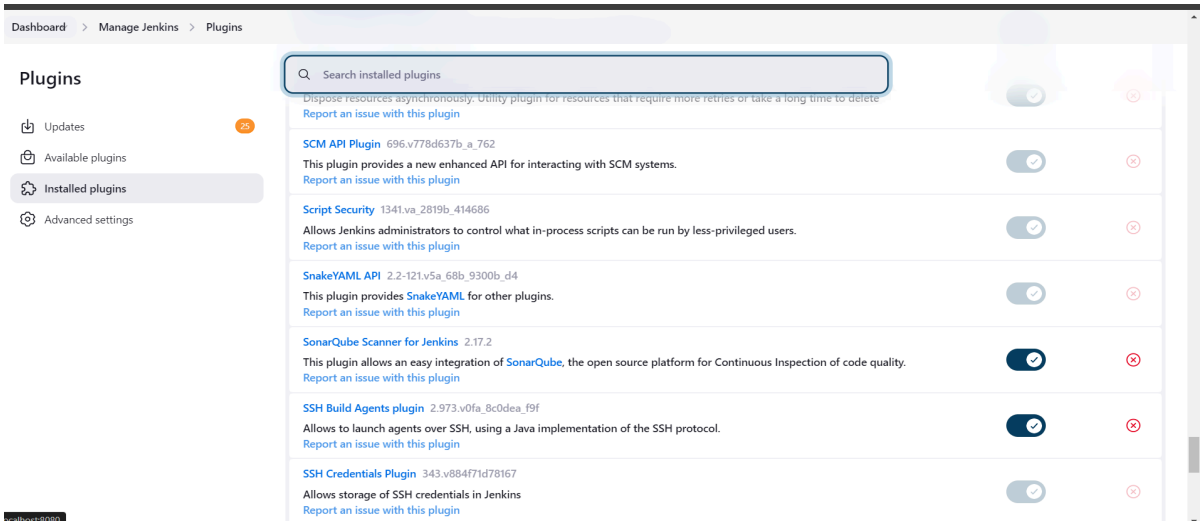


4. Login to SonarQube using username admin and password admin.

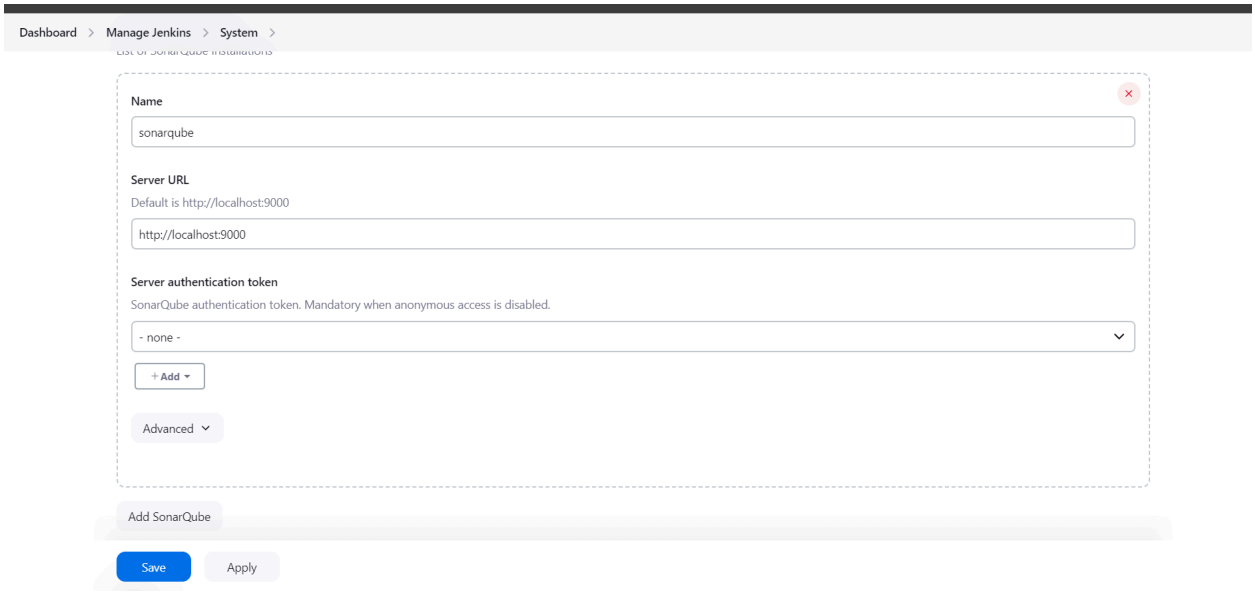
5. Create a manual project in SonarQube1 with the name sonarqube1



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

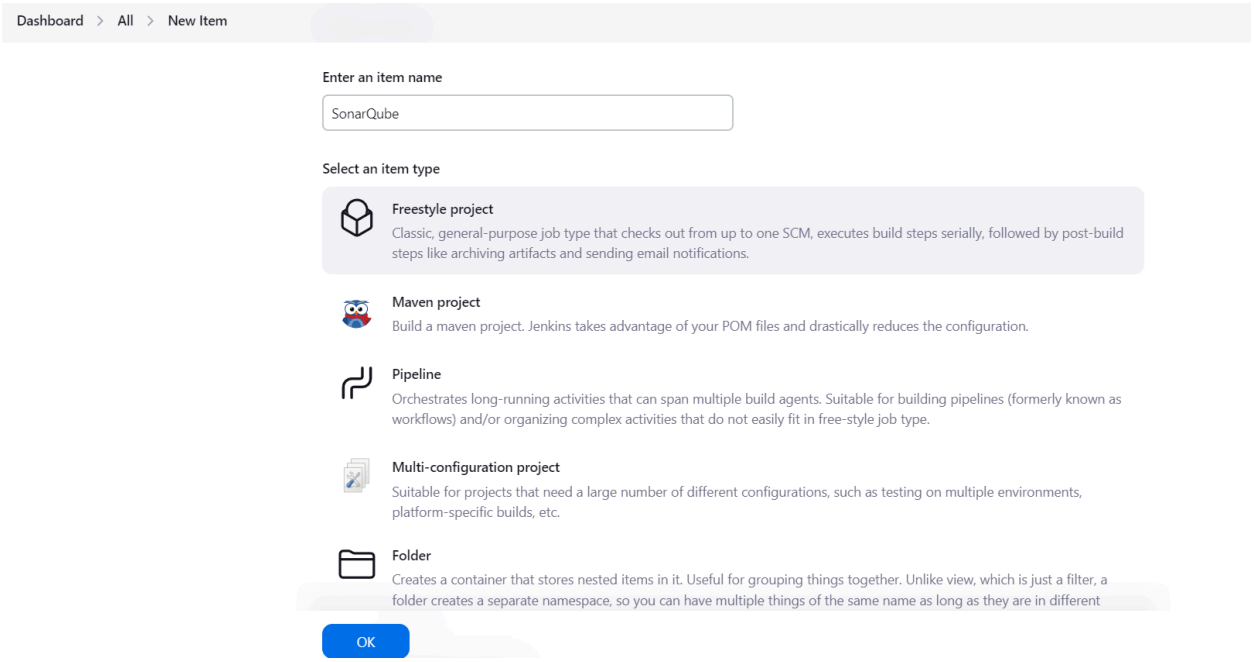


7. Under Jenkins ‘Configure System’, look for SonarQube Servers and enter the details. Enter the Server Authentication token if needed.



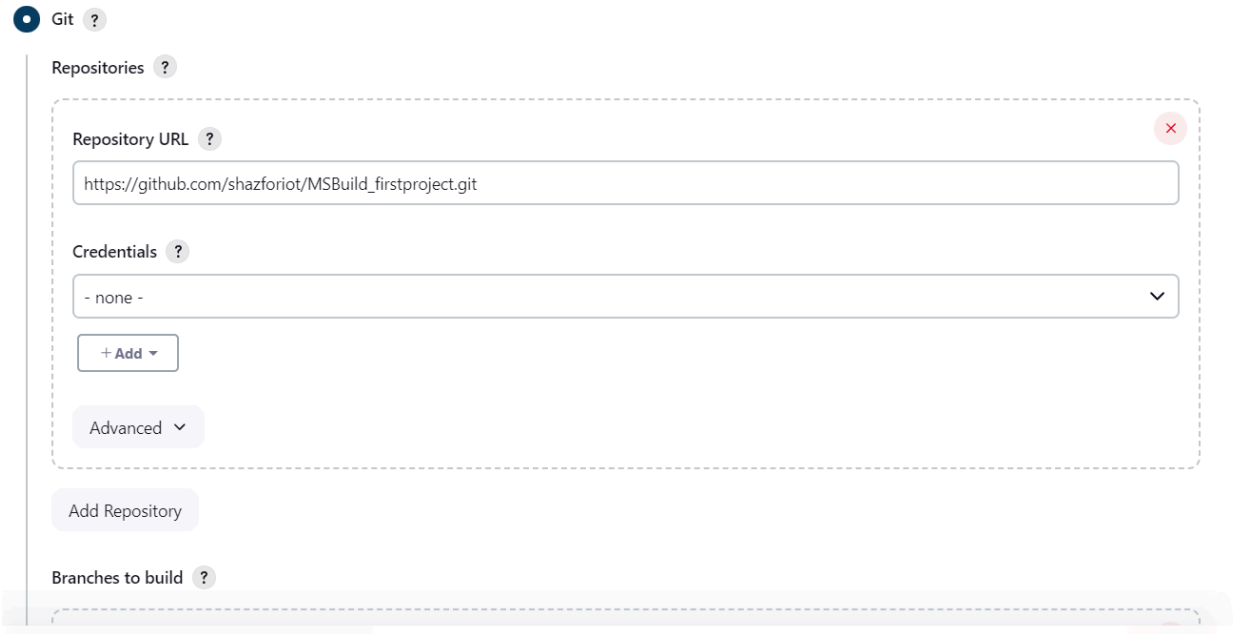
8. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.

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

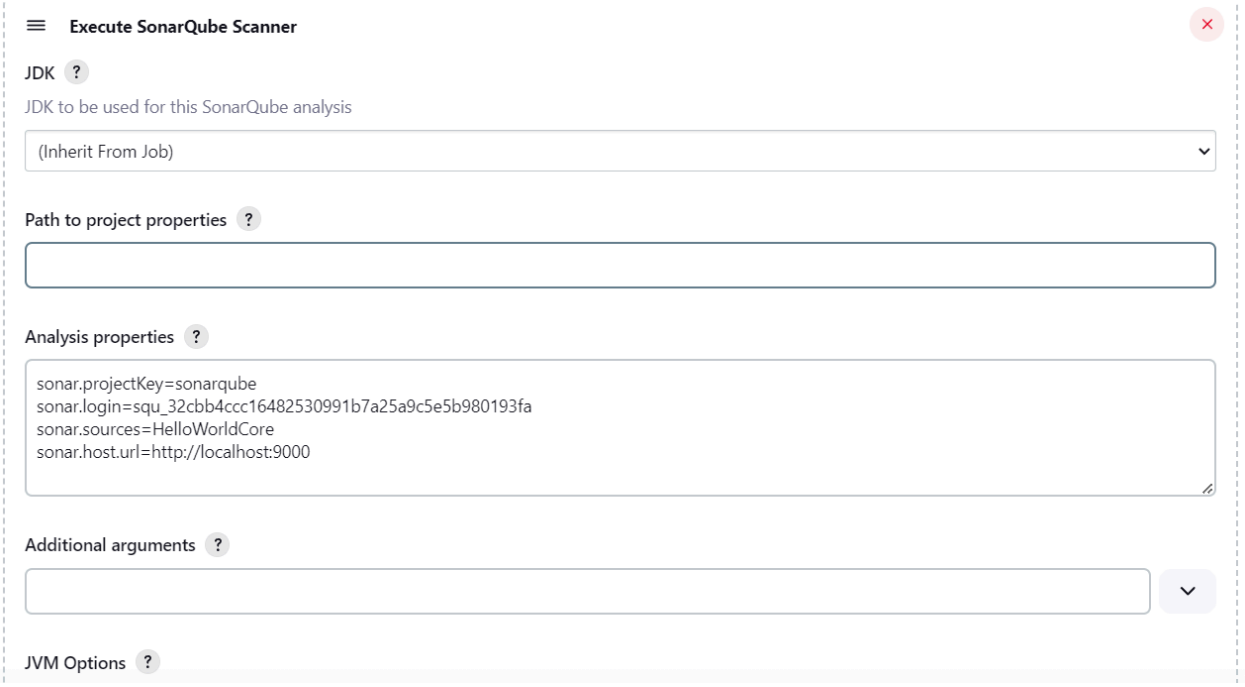


10. Choose this GitHub repository in Source Code Management.

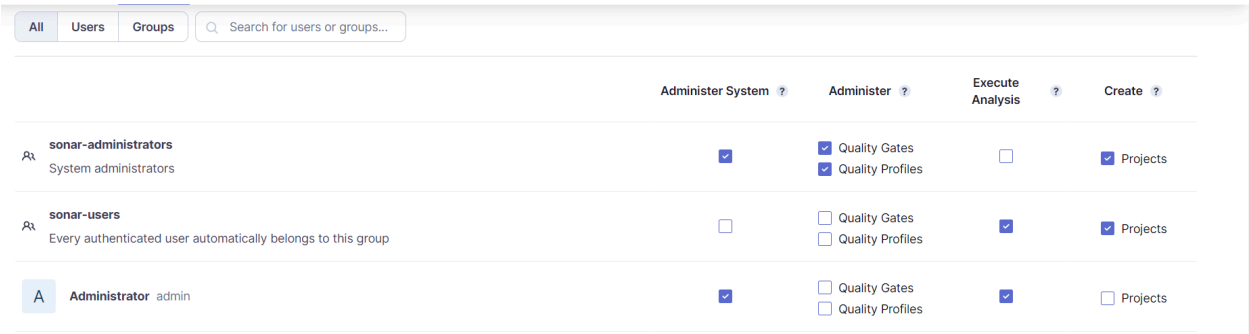
https://github.com/shazforiot/MSBuild_firstproject.git



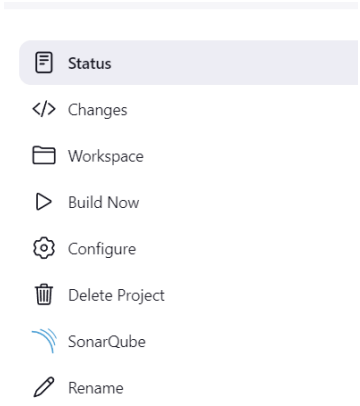
11. Under Build-> Execute SonarQube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.



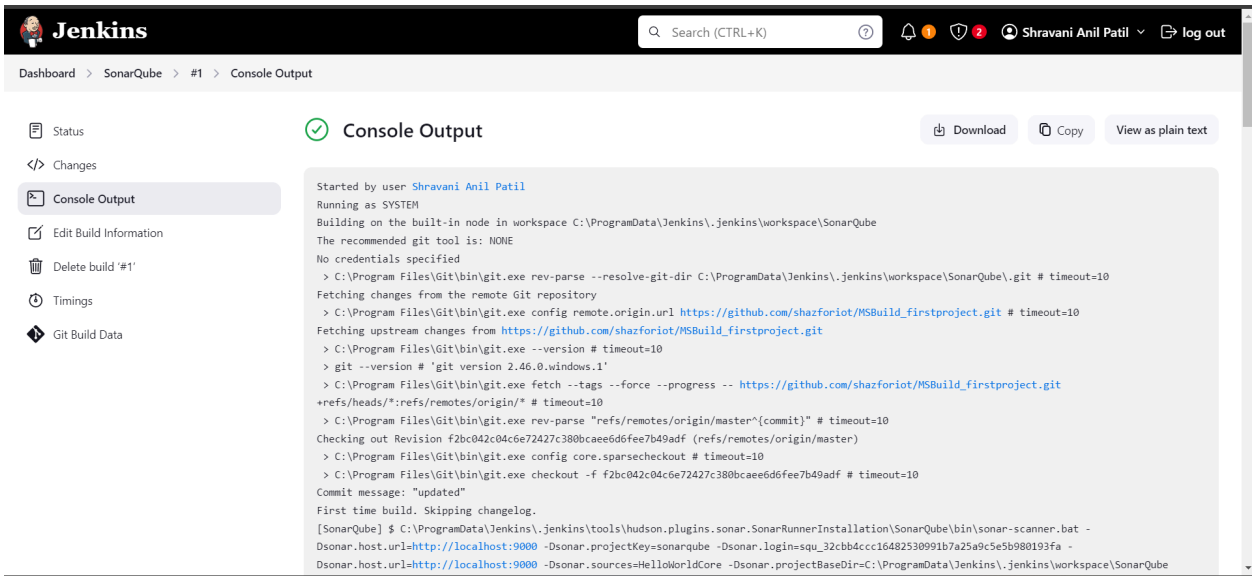
12. Go to http://localhost:9000/admin/permissions and allow Execute Permissions to the Admin user.

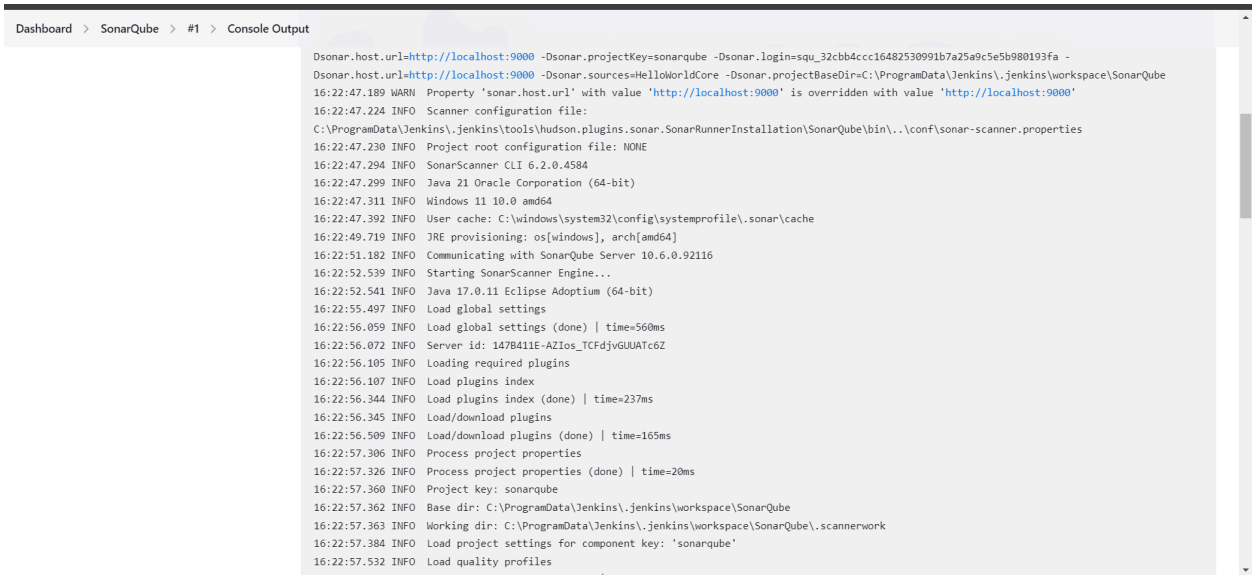


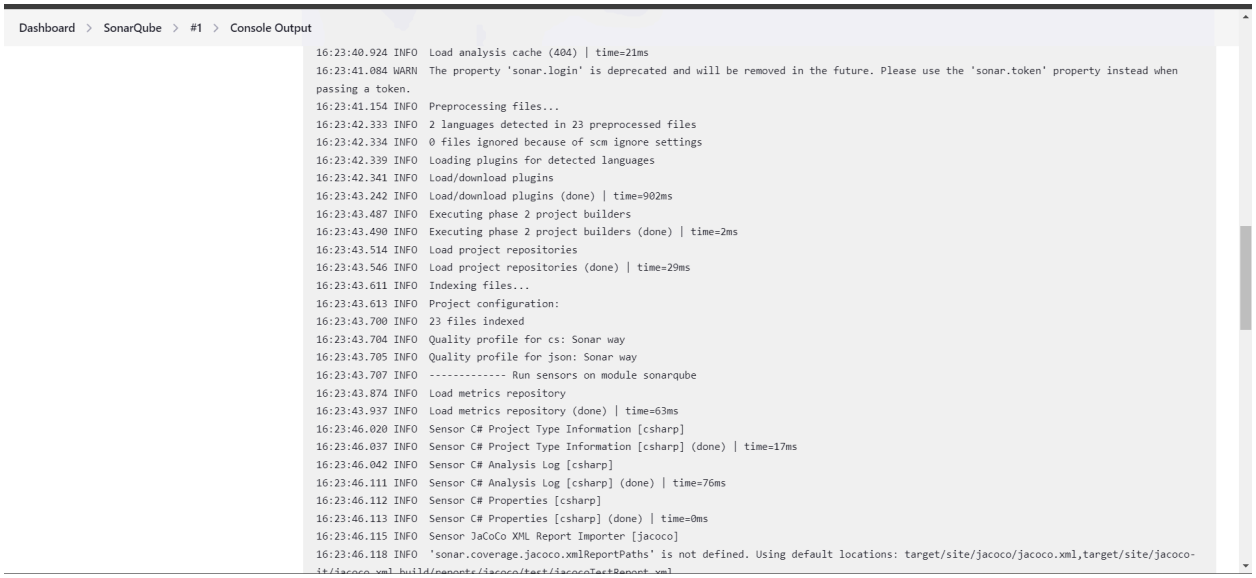
13. Run The Build.



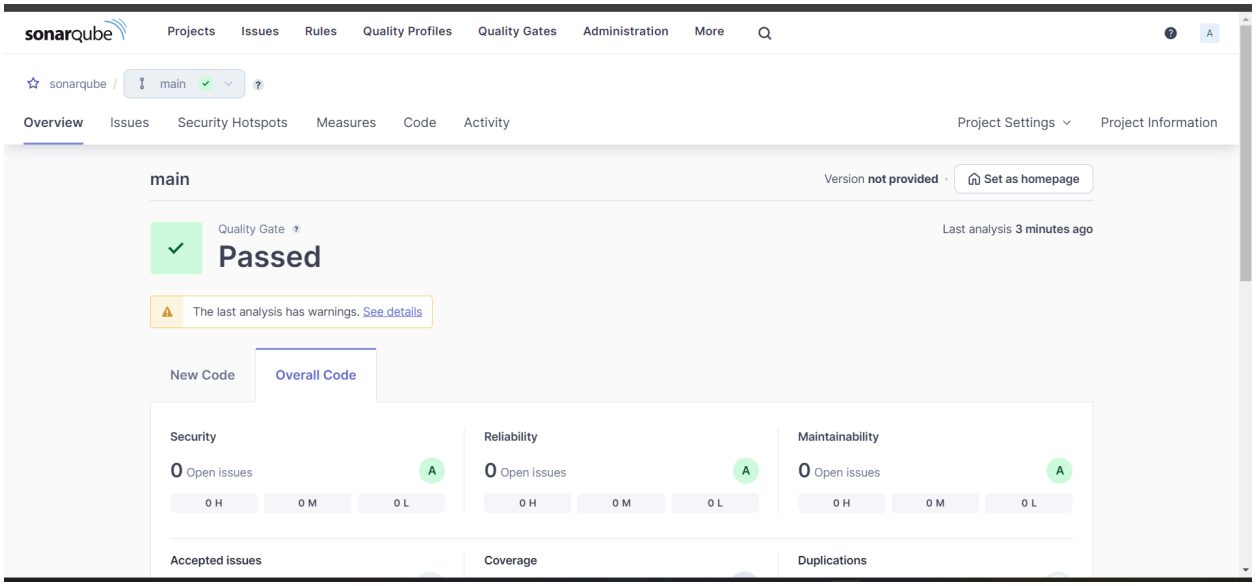
14. Console Output







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


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
More
sonarqube / main
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
main
Version not provided
Set as homepage
Quality Gate
Passed
Last analysis 3 minutes ago
The last analysis has warnings. See details
New Code
Overall Code
Security
0 Open issues
A
0 H
0 M
0 L
Reliability
0 Open issues
A
0 H
0 M
0 L
Maintainability
0 Open issues
A
0 H
0 M
0 L
Accepted issues
Coverage
Duplications

Shravani Anil Patil **D15A-38**

Advanced Devops
Experiment 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) :SAST is a methodology for testing an application's source code to identify security vulnerabilities before the code is compiled. This type of testing, also referred to as white-box testing, helps improve application security by finding weaknesses early in development.

Problems SAST Solves

- **Early Detection**: SAST finds vulnerabilities early in the Software Development Life Cycle (SDLC), allowing developers to fix issues without affecting builds or passing vulnerabilities to the final release.
- **Real-Time Feedback**: Developers receive immediate feedback during coding, helping them address security issues before moving to the next stage of development.
- **Graphical Representations**: SAST tools often provide visual aids to help developers navigate the code and identify the exact location of vulnerabilities, offering suggestions for fixes.
- **Regular Scanning**: SAST tools can be configured to scan code regularly, such as during daily builds, code check-ins, or before releases.

Importance of SAST

- **Resource Efficiency**: With a larger number of developers than security experts, SAST allows full codebase analysis quickly and efficiently, without relying on manual code reviews.
- **Speed**: SAST tools can analyze millions of lines of code within minutes, detecting critical vulnerabilities such as buffer overflows, SQL injection, and cross-site scripting (XSS) with high accuracy.

CI/CD Pipeline

A Continuous Integration/Continuous Delivery (CI/CD) pipeline is a sequence of automated tasks designed to build, test, and deploy new software versions rapidly and consistently. It plays a crucial role in DevOps practices, ensuring fast and reliable software releases.

SonarQube

SonarQube is an open-source platform from SonarSource that performs continuous code quality inspections through static code analysis. It identifies bugs, code smells, security vulnerabilities, and code duplications in a wide range of programming languages. SonarQube is extendable with plugins and integrates seamlessly into CI/CD pipelines.

Benefits of SonarQube

Sustainability: By reducing complexity and vulnerabilities, SonarQube extends the lifespan of applications and helps maintain cleaner code.

Increased Productivity: SonarQube minimizes maintenance costs and risks, resulting in fewer code changes and a more stable codebase.

Quality Code: Ensures code quality checks are integrated into the development process.

Error Detection: Automatically identifies coding errors and alerts developers to resolve them before moving to production.

Consistency: Helps maintain consistent code quality by detecting and reporting violations of coding standards.

Business Scaling: SonarQube supports scaling as the business grows without any restrictions.

Implementation:

Prerequisites

1. Jenkins installed on your machine.
2. Docker installed to run SonarQube.
3. SonarQube installed via Docker

1. Set Up Jenkins

- Open Jenkins Dashboard on localhost:8080 or your configured port

- Install the necessary plugins:
- SonarQube Scanner Plugin

2. Run SonarQube in Docker

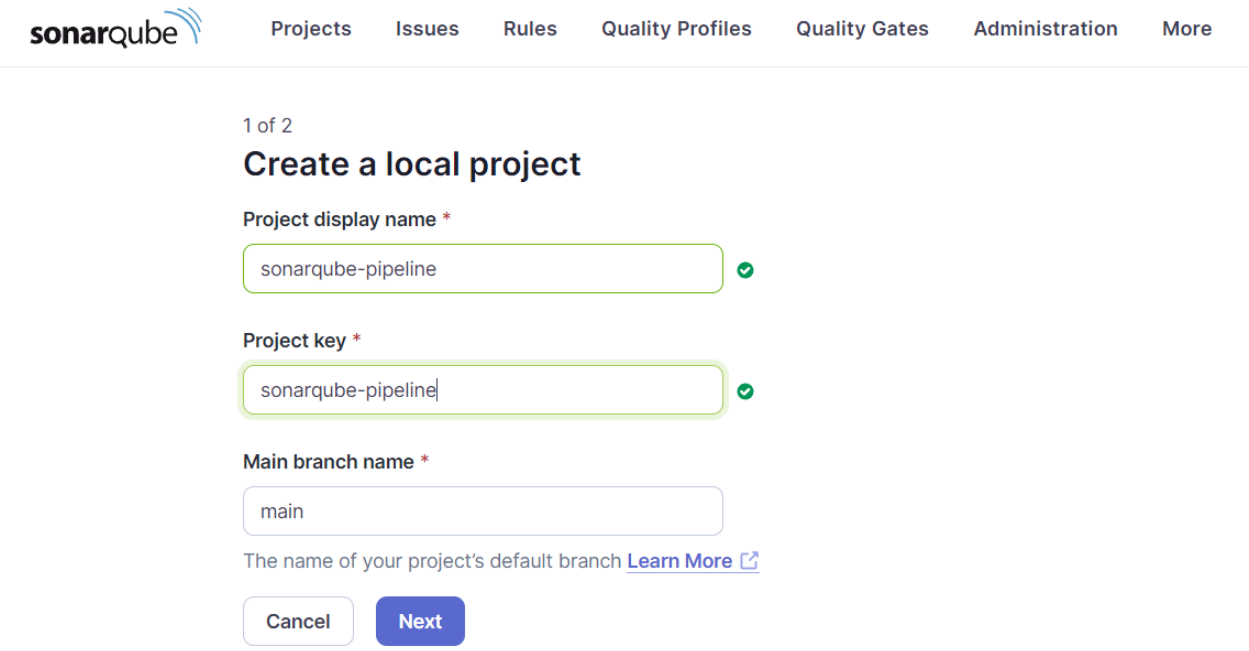
Run the following command to start SonarQube in a Docker container: command

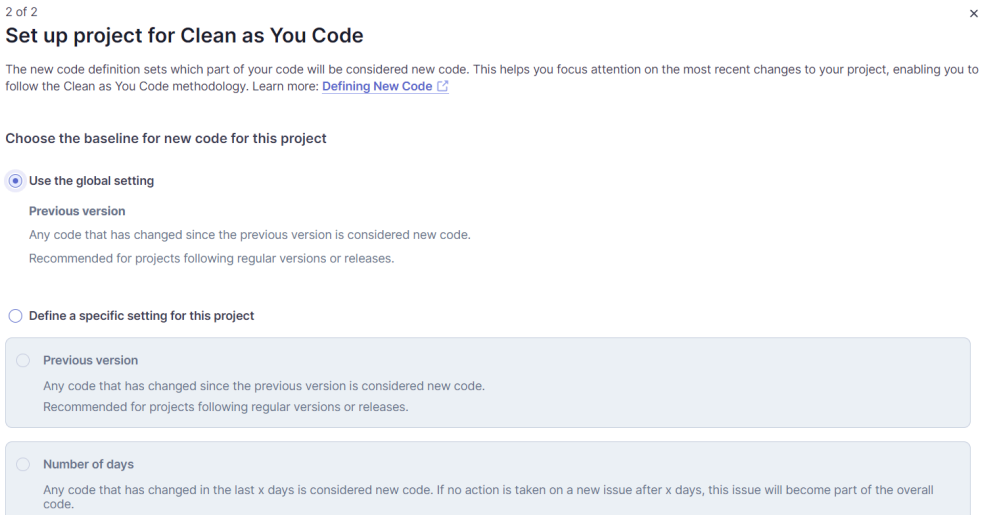
:

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

- Check SonarQube status at http://localhost:9000.
- Login with your credentials:

Step 1: Log in to sonarqube portal and create a local project.



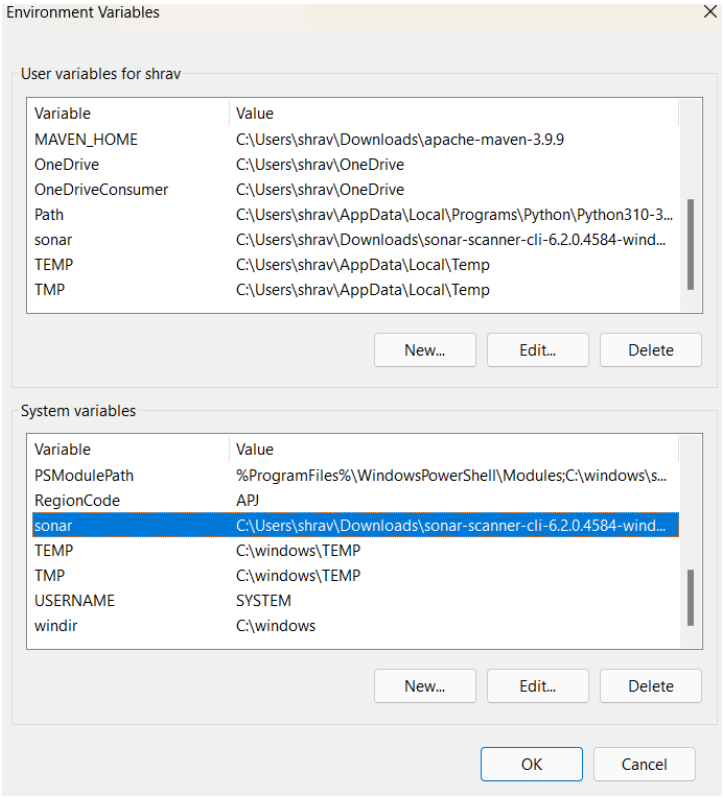


Step 2: Go to [download_sonarscanner]() to download sonar scanner

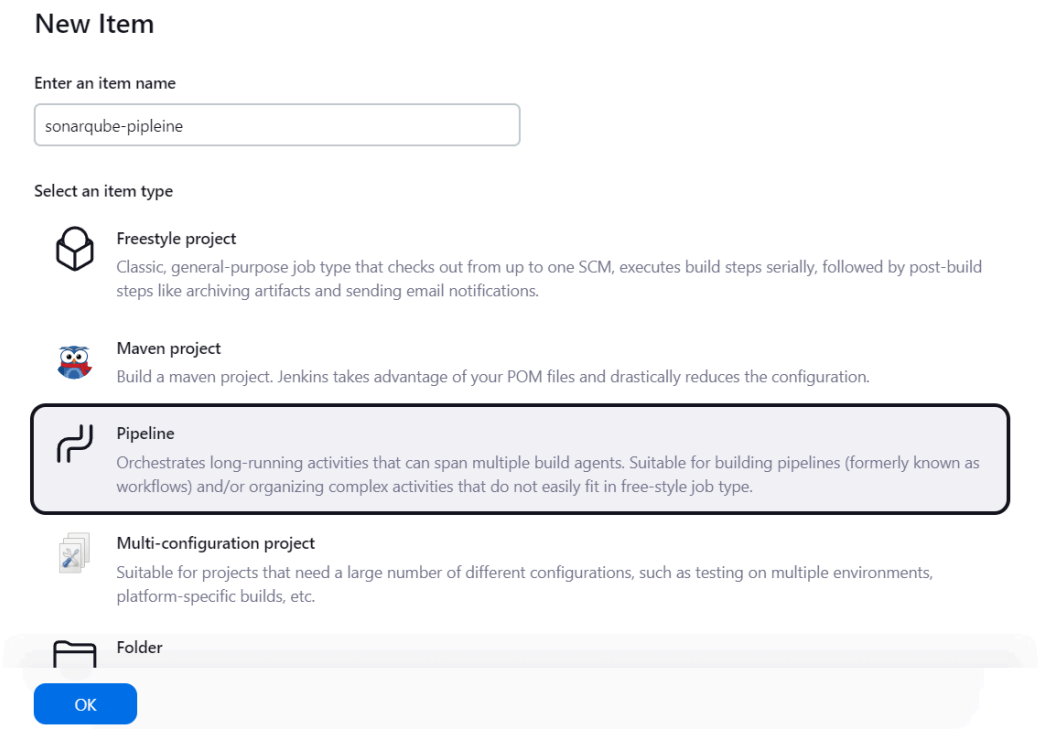
After the download is complete, extract the file and copy the path to bin folder

Go to environment variables, system variables and click on path

Add a new path, paste the path copied earlier.

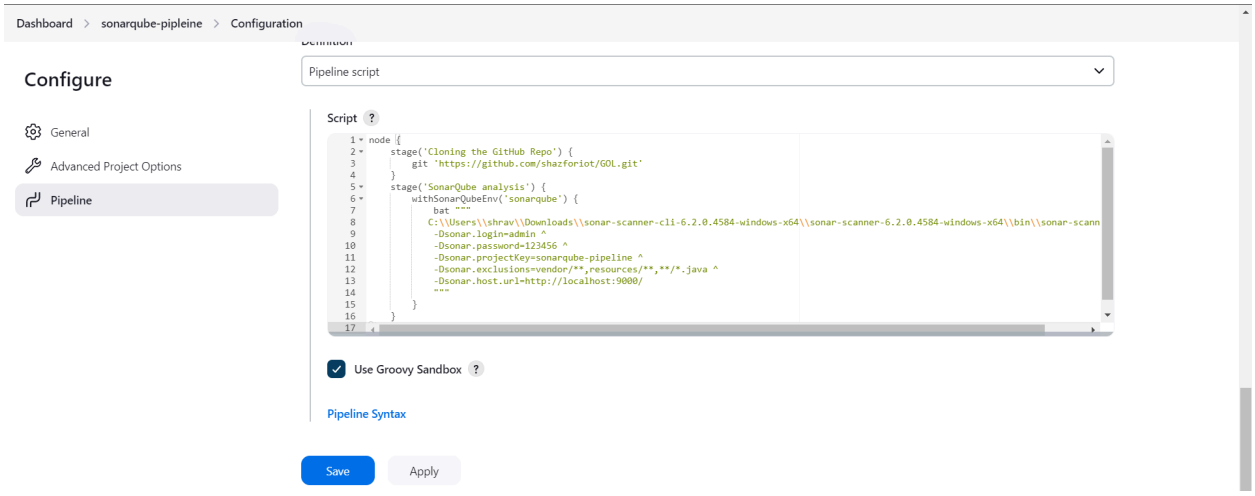


Step 3: Create a New Item in Jenkins, choose Pipeline.

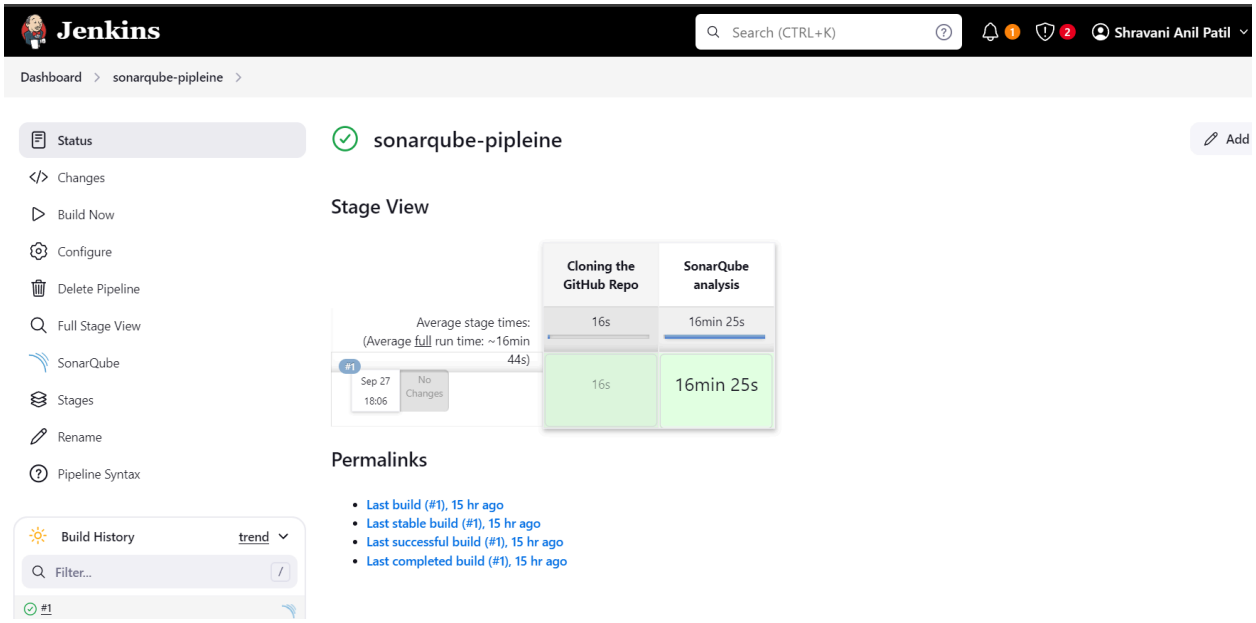


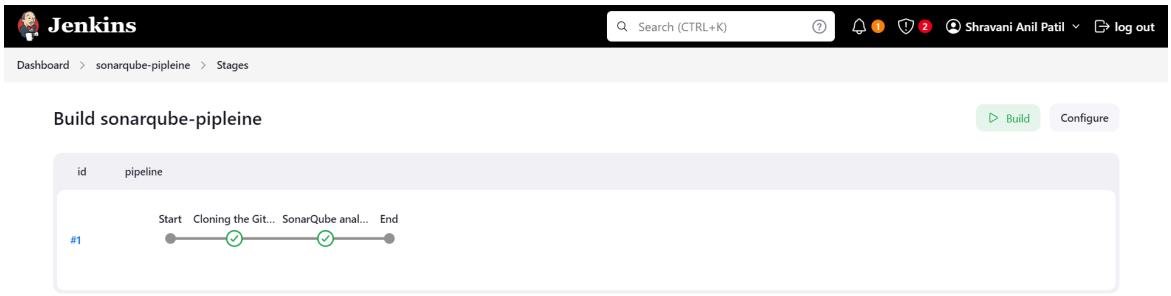
Add pipeline script :

```
node {
    stage('Cloning the GitHub Repo') {
        git 'https://github.com/shazforiot/GOL.git'
    }
    stage('SonarQube analysis') {
        withSonarQubeEnv('sonarqube') {
            bat """

C:\\Users\\shrav\\Downloads\\sonar-scanner-cli-6.2.0.4584-windows-x64\\sonar-scanner-6.2.0.4584-windows-x64\\bin\\sonar-scanner.bat ^
            -Dsonar.login=admin ^
            -Dsonar.password=123456 ^
            -Dsonar.projectKey=sonarqube-pipeline ^
            -Dsonar.exclusions=vendor/**,resources/**,**/*.java ^
            -Dsonar.host.url=http://localhost:9000/
            """
        }
    }
}
```

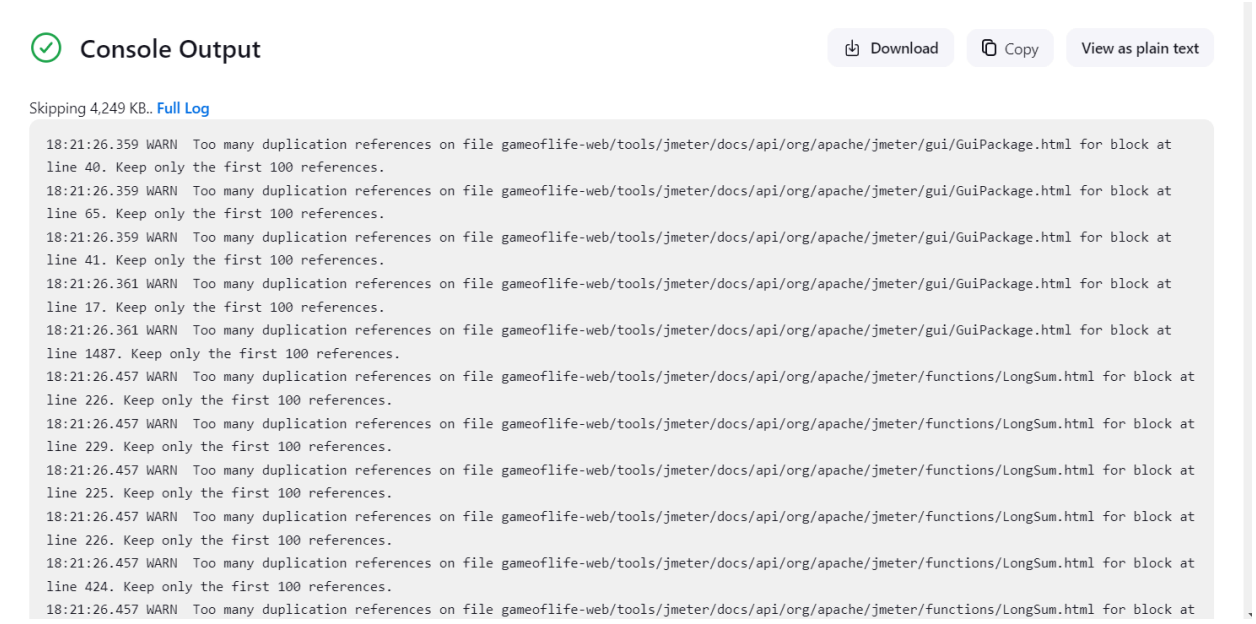


Step 4: Save the pipeline and build it.





Console output:

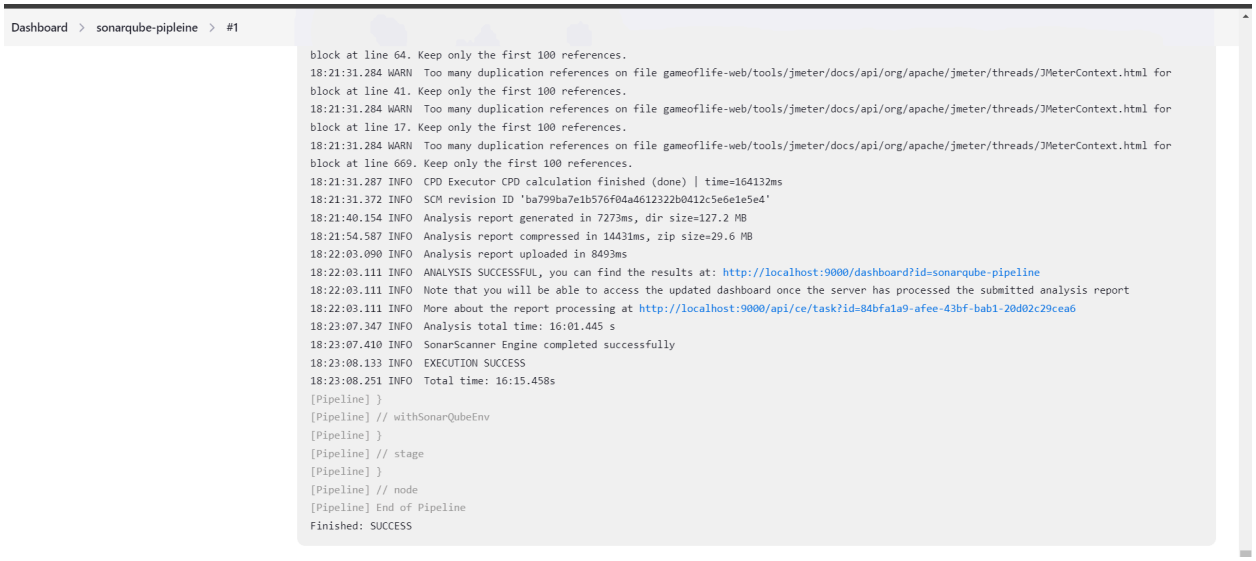


Console Output

Download Copy View as plain text

Skipping 4,249 KB.. Full Log

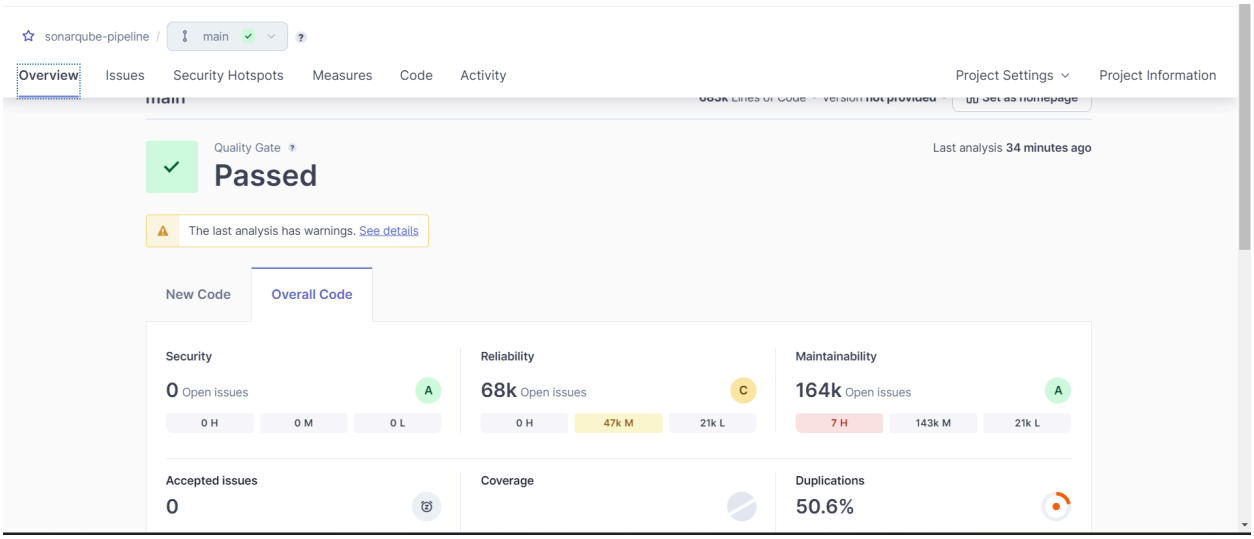
```
18:21:26.359 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/GuiPackage.html for block at
line 40. Keep only the first 100 references.
18:21:26.359 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/GuiPackage.html for block at
line 65. Keep only the first 100 references.
18:21:26.359 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/GuiPackage.html for block at
line 41. Keep only the first 100 references.
18:21:26.361 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/GuiPackage.html for block at
line 17. Keep only the first 100 references.
18:21:26.361 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/GuiPackage.html for block at
line 1487. Keep only the first 100 references.
18:21:26.457 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/functions/LongSum.html for block at
line 226. Keep only the first 100 references.
18:21:26.457 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/functions/LongSum.html for block at
line 229. Keep only the first 100 references.
18:21:26.457 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/functions/LongSum.html for block at
line 225. Keep only the first 100 references.
18:21:26.457 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/functions/LongSum.html for block at
line 226. Keep only the first 100 references.
18:21:26.457 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/functions/LongSum.html for block at
line 424. Keep only the first 100 references.
18:21:26.457 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/functions/LongSum.html for block at
```



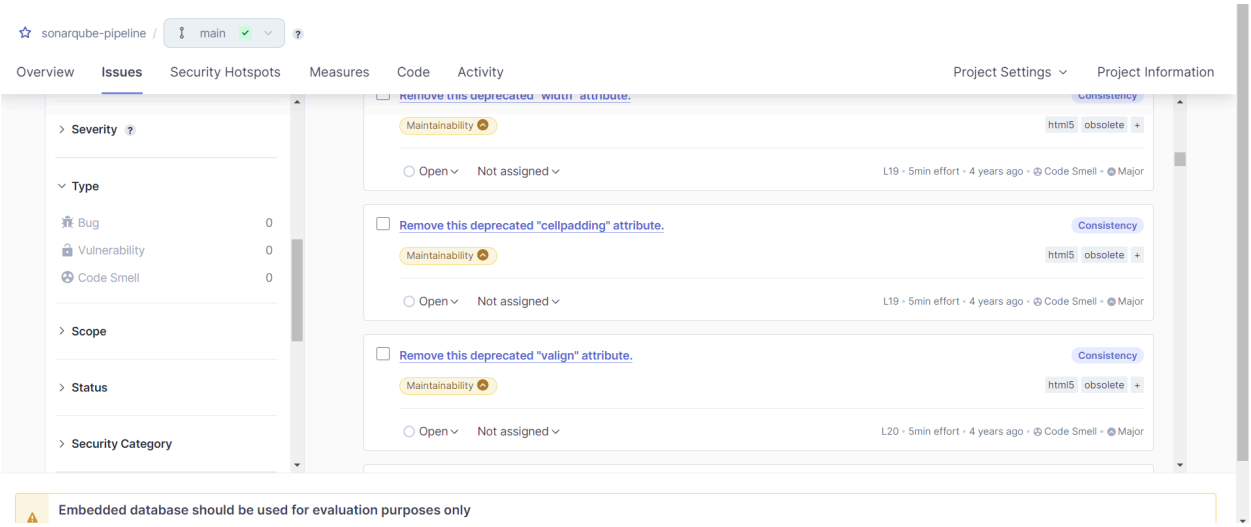
Dashboard > sonarqube-pipleine > #1

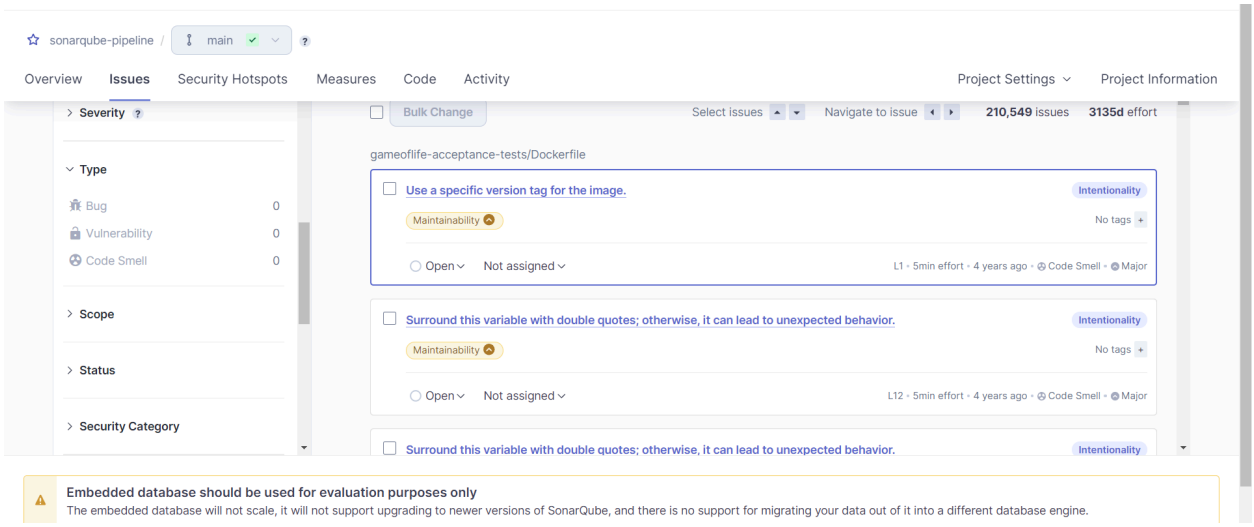
```
block at line 64. Keep only the first 100 references.
18:21:31.284 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/threads/JMeterContext.html for
block at line 41. Keep only the first 100 references.
18:21:31.284 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/threads/JMeterContext.html for
block at line 17. Keep only the first 100 references.
18:21:31.284 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/threads/JMeterContext.html for
block at line 669. Keep only the first 100 references.
18:21:31.287 INFO  CPD Executor CPD calculation finished (done) | time=164132ms
18:21:31.372 INFO  SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
18:21:40.154 INFO  Analysis report generated in 7273ms, dir size=127.2 MB
18:21:54.587 INFO  Analysis report compressed in 14431ms, zip size=29.6 MB
18:22:03.090 INFO  Analysis report uploaded in 8493ms
18:22:03.111 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube-pipeline
18:22:03.111 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
18:22:03.111 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=84bfa1a9-afee-43bf-bab1-20d02c29cea6
18:23:07.347 INFO  Analysis total time: 16:01.445 s
18:23:07.410 INFO  SonarScanner Engine completed successfully
18:23:08.133 INFO  EXECUTION SUCCESS
18:23:08.251 INFO  Total time: 16:15.458s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

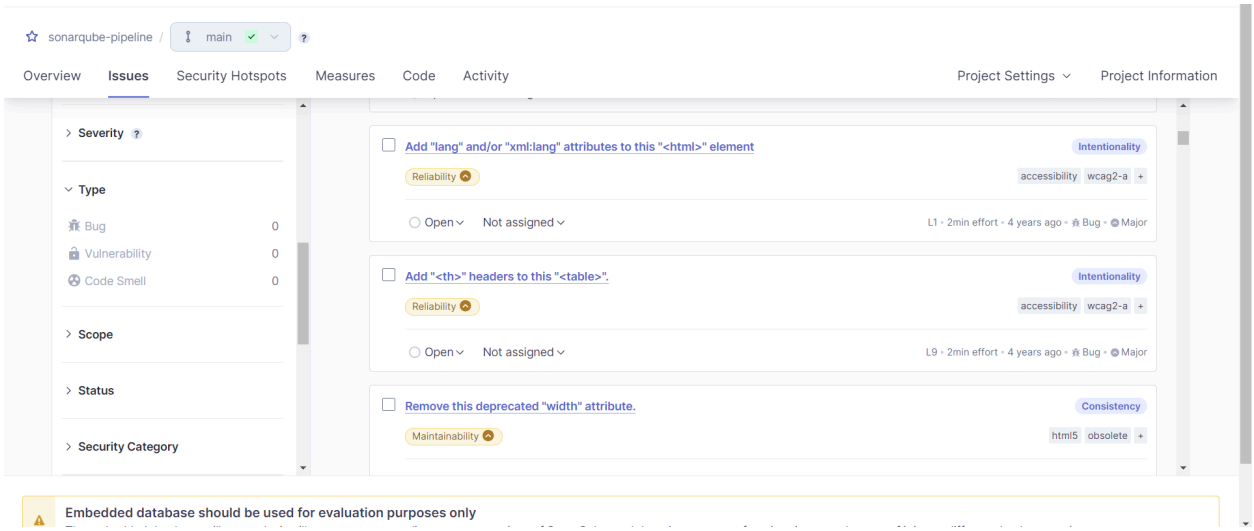
Step 5: After that, check the project in SonarQube

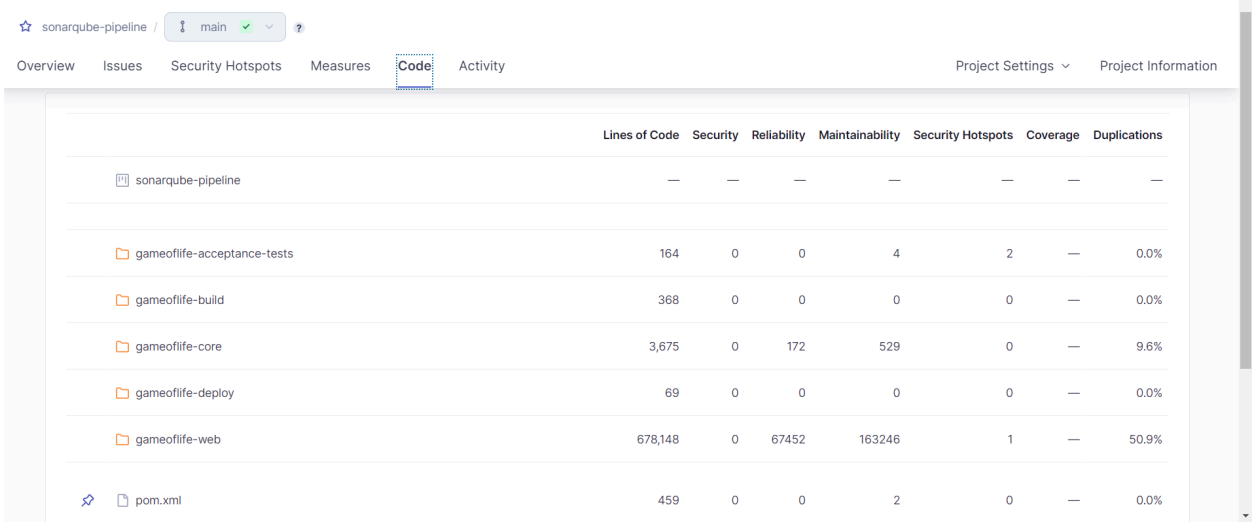


Under different tabs, check all different issues with the code

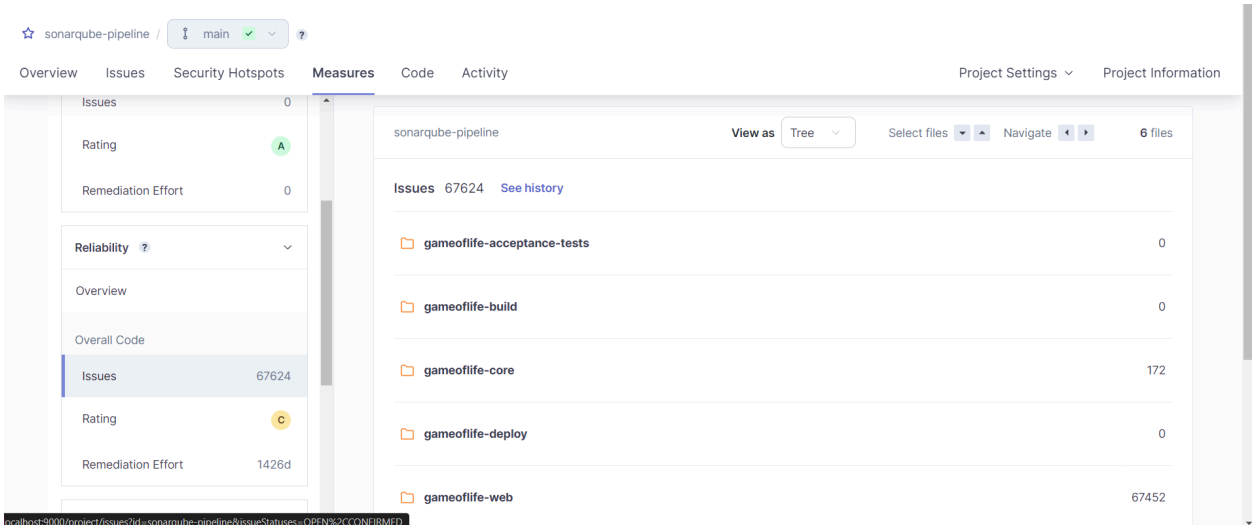


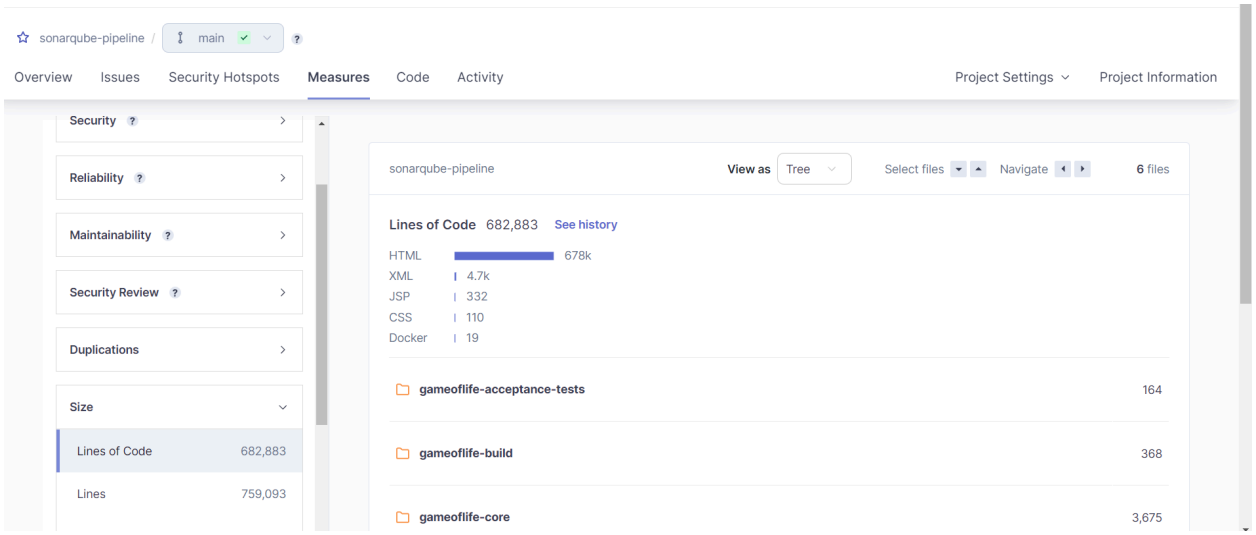


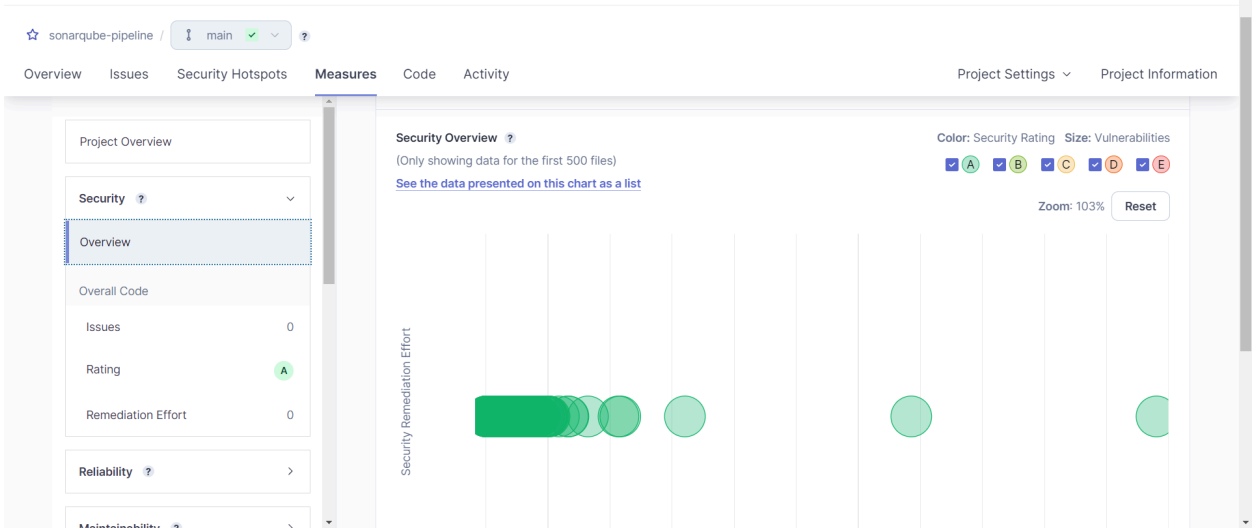




	Lines of Code	Security	Reliability	Maintainability	Security Hotspots	Coverage	Duplications
sonarqube-pipeline	—	—	—	—	—	—	—
gameoflife-acceptance-tests	164	0	0	4	2	—	0.0%
gameoflife-build	368	0	0	0	0	—	0.0%
gameoflife-core	3,675	0	172	529	0	—	9.6%
gameoflife-deploy	69	0	0	0	0	—	0.0%
gameoflife-web	678,148	0	67452	163246	1	—	50.9%
pom.xml	459	0	0	2	0	—	0.0%


sonarqube-pipeline / main
Overview Issues Security Hotspots Measures Code Activity
Project Settings Project Information
Issues 0
Rating A
Remediation Effort 0
Reliability
Overview
Overall Code
Issues 67624
Rating C
Remediation Effort 1426d
sonarqube-pipeline
View as Tree
Select files Navigate
6 files
Issues 67624 See history
gameoflife-acceptance-tests 0
gameoflife-build 0
gameoflife-core 172
gameoflife-deploy 0
gameoflife-web 67452


sonarqube-pipeline / main
Overview Issues Security Hotspots Measures Code Activity
Project Settings Project Information
Security
Reliability
Maintainability
Security Review
Duplications
Size
Lines of Code 682,883
Lines 759,093
sonarqube-pipeline
View as Tree
Select files Navigate
6 files
Lines of Code 682,883 See history
HTML 678k
XML 4.7k
JSP 332
CSS 110
Docker 19
gameoflife-acceptance-tests 164
gameoflife-build 368
gameoflife-core 3,675


sonarqube-pipeline / main
Overview Issues Security Hotspots Measures Code Activity
Project Settings Project Information
Project Overview
Security
Overview
Overall Code
Issues 0
Rating A
Remediation Effort 0
Reliability
Security Overview
(Only showing data for the first 500 files)
See the data presented on this chart as a list
Color: Security Rating Size: Vulnerabilities
A B C D E
Zoom: 103% Reset
Security Remediation Effort

Shravani Anil Patil **D15A-38**

Advanced DevOps Experiment-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

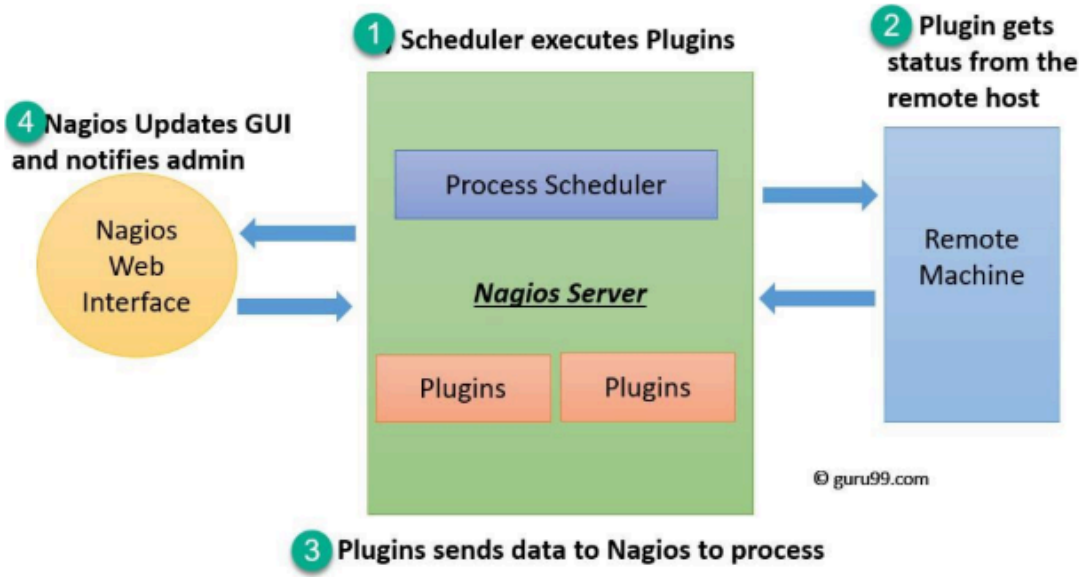
Features of Nagios

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.

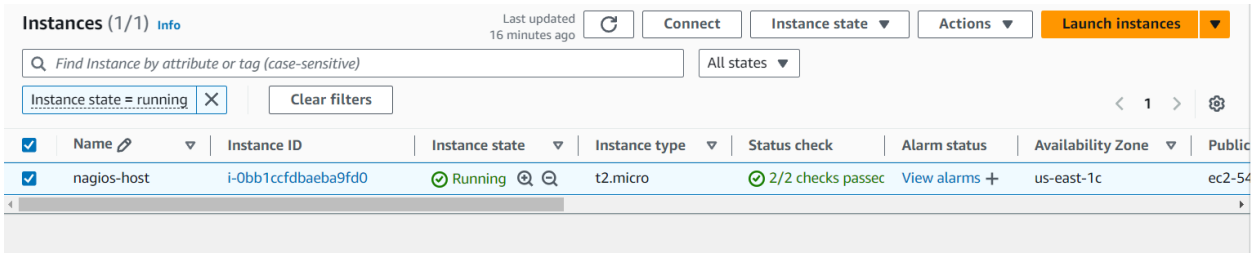


1. The scheduler is a component of the server part of Nagios. It sends a signal to execute the plugins at the remote host.
2. The plugin gets the status from the remote host
3. The plugin sends the data to the process scheduler
4. The process scheduler updates the GUI and notifications are sent to admins.

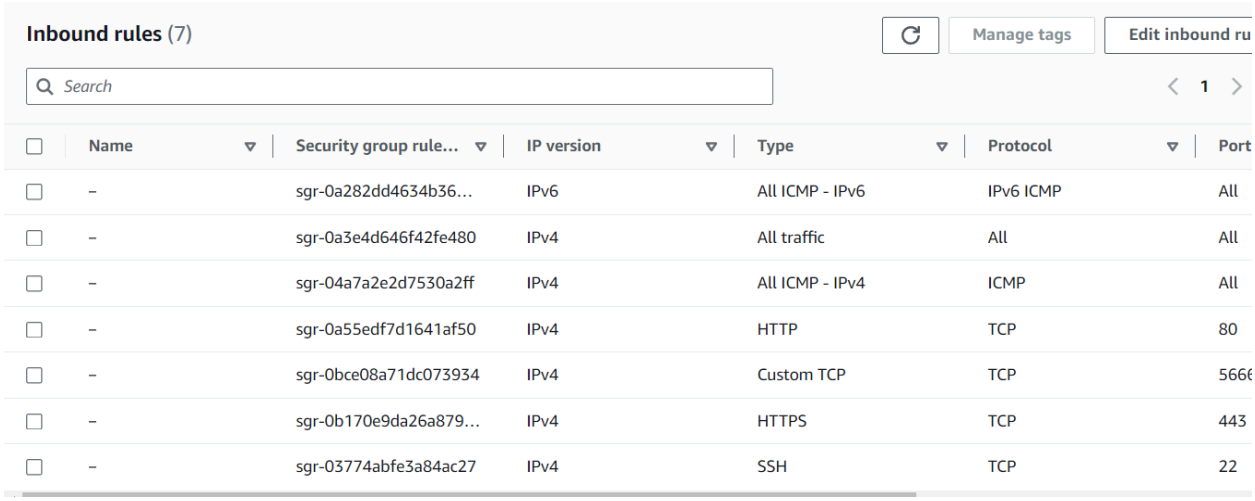
Installation of Nagios
Prerequisites: AWS Free Tier

Steps:

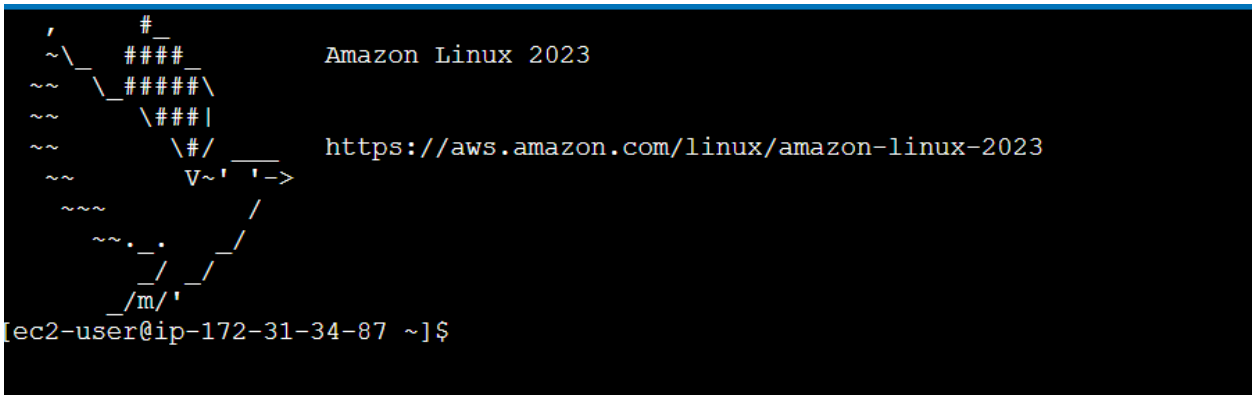
1. Create an Amazon Linux EC2 Instance in AWS and name it - nagios-host



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



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

```
  libtiff-4.4.0-4.amzn2023.0.18.x86_64                libtiff-devel-4.4.0-4.amzn2023.0.18.x86_64
  libwebp-1.2.4-1.amzn2023.0.6.x86_64                 libwebp-devel-1.2.4-1.amzn2023.0.6.x86_64
  libxcb-1.13.1-7.amzn2023.0.2.x86_64                 libxcb-devel-1.13.1-7.amzn2023.0.2.x86_64
  libxml2-devel-2.10.4-1.amzn2023.0.6.x86_64          pcre2-devel-10.40-1.amzn2023.0.3.x86_64
  pcre2-utf16-10.40-1.amzn2023.0.3.x86_64             pcre2-utf32-10.40-1.amzn2023.0.3.x86_64
  pixman-0.40.0-3.amzn2023.0.3.x86_64                 sysprof-capture-devel-3.40.1-2.amzn2023.0.2.x86_64
  xml-common-0.6.3-56.amzn2023.0.2.noarch             xorg-x11-proto-devel-2021.4-1.amzn2023.0.2.noarch
  xz-devel-5.2.5-9.amzn2023.0.2.x86_64                zlib-devel-1.2.11-33.amzn2023.0.5.x86_64

Complete!
[ec2-user@ip-172-31-34-87 ~]$
```

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

```
[ec2-user@ip-172-31-34-87 ~]$ sudo useradd nagios
sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password fails the dictionary check - it is too simplistic/systematic
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-34-87 ~]$
```

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-172-31-34-87 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-34-87 ~]$ sudo usermod -aG nagcmd nagios
sudo usermod -aG nagcmd apache
[ec2-user@ip-172-31-34-87 ~]$
```

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

9. Use wget to download the source zip files.
Wget http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.0.8.tar.gz

```
nagios-4.4.6.tar.gz                 100%[=============================================================================>]  10.81M  12.0MB/s

2024-10-02 16:58:34 (12.0 MB/s) - 'nagios-4.4.6.tar.gz' saved [11333414/11333414]

--2024-10-02 16:58:34--  https://nagios-plugins.org/download/nagios-plugins-2.3.3.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|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2782610 (2.7M) [application/x-gzip]
Saving to: 'nagios-plugins-2.3.3.tar.gz'

nagios-plugins-2.3.3.tar.gz         100%[=============================================================================>]   2.65M  6.08MB/s

2024-10-02 16:58:35 (6.08 MB/s) - 'nagios-plugins-2.3.3.tar.gz' saved [2782610/2782610]

[ec2-user@ip-172-31-34-87 downloads]$
```

10. Use tar to unzip and change to that directory.
tar zxvf nagios-4.0.8.tar.gz

```
[ec2-user@ip-172-31-34-87 downloads]$ tar zxvf nagios-4.4.6.tar.gz
cd nagios-4.4.6
nagios-4.4.6/
nagios-4.4.6/.gitignore
nagios-4.4.6/.travis.yml
nagios-4.4.6/CONTRIBUTING.md
nagios-4.4.6/Changelog
nagios-4.4.6/INSTALLING
nagios-4.4.6/LEGAL
nagios-4.4.6/LICENSE
nagios-4.4.6/Makefile.in
nagios-4.4.6/README.md
nagios-4.4.6/THANKS
nagios-4.4.6/UPGRADING
nagios-4.4.6/aclocal.m4
nagios-4.4.6/autoconf-macros/
nagios-4.4.6/autoconf-macros/.gitignore
nagios-4.4.6/autoconf-macros/CHANGELOG.md
```

```
nagios-4.4.6/xdata/xodtemplate.c
nagios-4.4.6/xdata/xodtemplate.h
nagios-4.4.6/xdata/xpddefault.c
nagios-4.4.6/xdata/xpddefault.h
nagios-4.4.6/xdata/xrddefault.c
nagios-4.4.6/xdata/xrddefault.h
nagios-4.4.6/xdata/xsddefault.c
nagios-4.4.6/xdata/xsddefault.h
[ec2-user@ip-172-31-34-87 nagios-4.4.6]$
```

11. Run the configuration script with the same group name you previously created.
./configure --with-command-group=nagcmd

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


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

[ec2-user@ip-172-31-34-87 nagios-4.4.6]$
```

12. Compile the source code.
make all

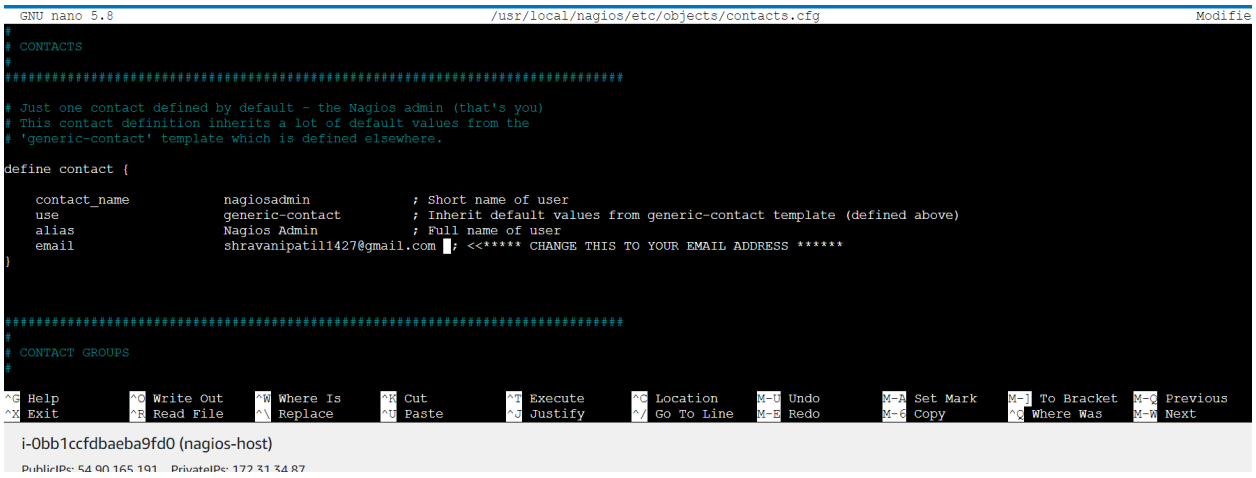
```
*** Support Notes *****************************************

If you have questions about configuring or running Nagios,
please make sure that you:

     - Look at the sample config files
     - Read the documentation on the Nagios Library at:
          https://library.nagios.com

before you post a question to one of the mailing lists.
Also make sure to include pertinent information that could
help others help you.  This might include:

     - What version of Nagios you are using
     - What version of the plugins you are using
     - Relevant snippets from your config files
     - Relevant error messages from the Nagios log file

For more information on obtaining support for Nagios, visit:

       https://support.nagios.com

************************************************************

Enjoy.

[ec2-user@ip-172-31-34-87 nagios-4.4.6]$
```

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-172-31-34-87 nagios-4.4.6]$ sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.4.6/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.4.6/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.4.6/cgi'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.4.6/cgi'
```

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

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

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

*** External command directory configured ***

[ec2-user@ip-172-31-34-87 nagios-4.4.6]$
```

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



15. Configure the web interface.
sudo make install-webconf

```
[ec2-user@ip-172-31-34-87 nagios-4.4.6]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

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

[ec2-user@ip-172-31-34-87 nagios-4.4.6]$
```

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

```
[ec2-user@ip-172-31-34-87 nagios-4.4.6]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-34-87 nagios-4.4.6]$
```

17. Restart Apache
sudo service httpd restart
18. Go back to the downloads folder and unzip the plugins zip file.
cd ~/downloads
tar zxvf nagios-plugins-2.3.3.tar.gz
cd nagios-plugins-2.3.3

```
[ec2-user@ip-172-31-34-87 nagios-4.4.6]$ sudo systemctl restart httpd
[ec2-user@ip-172-31-34-87 nagios-4.4.6]$ cd ~/downloads
tar zxvf nagios-plugins-2.3.3.tar.gz
cd nagios-plugins-2.3.3
nagios-plugins-2.3.3/
nagios-plugins-2.3.3/perlmods/
nagios-plugins-2.3.3/perlmods/Config-Tiny-2.14.tar.gz
nagios-plugins-2.3.3/perlmods/parent-0.226.tar.gz
nagios-plugins-2.3.3/perlmods/Test-Simple-0.98.tar.gz
nagios-plugins-2.3.3/perlmods/Makefile.in
nagios-plugins-2.3.3/perlmods/version-0.9903.tar.gz
nagios-plugins-2.3.3/perlmods/Makefile.am
nagios-plugins-2.3.3/perlmods/Module-Runtime-0.013.tar.gz
nagios-plugins-2.3.3/perlmods/Module-Metadata-1.000014.tar.gz
nagios-plugins-2.3.3/perlmods/Params-Validate-1.08.tar.gz
nagios-plugins-2.3.3/perlmods/Class-Accessor-0.34.tar.gz
nagios-plugins-2.3.3/perlmods/Try-Tiny-0.18.tar.gz
```

```
nagios-plugins-2.3.3/plugins-scripts/check_mailq.pl
nagios-plugins-2.3.3/plugins-scripts/check_wave.pl
nagios-plugins-2.3.3/plugins-scripts/check_ircd.pl
nagios-plugins-2.3.3/plugins-scripts/utils.sh.in
nagios-plugins-2.3.3/plugins-scripts/check_ifstatus.pl
nagios-plugins-2.3.3/plugins-scripts/check_sensors.sh
nagios-plugins-2.3.3/pkg/
nagios-plugins-2.3.3/pkg/fedora/
nagios-plugins-2.3.3/pkg/fedora/requires
nagios-plugins-2.3.3/pkg/solaris/
nagios-plugins-2.3.3/pkg/solaris/preinstall
nagios-plugins-2.3.3/pkg/solaris/solpkg
nagios-plugins-2.3.3/pkg/solaris/pkginfo.in
nagios-plugins-2.3.3/pkg/solaris/pkginfo
nagios-plugins-2.3.3/pkg/redhat/
nagios-plugins-2.3.3/pkg/redhat/requires
[ec2-user@ip-172-31-34-87 nagios-plugins-2.3.3]$
```

19. Compile and install plugins

cd nagios-plugins-2.3.3

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

make

sudo make install

```
  done; \
  for file in Makevars; do \
    rm -f /usr/local/nagios/share/gettext/po/$file; \
  done; \
else \
  : ; \
fi
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/po'
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.3.3'
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.3.3'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3'
[ec2-user@ip-172-31-34-87 nagios-plugins-2.3.3]$
```

```
[ec2-user@ip-172-31-34-87 nagios-plugins-2.3.3]$ sudo chkconfig --add nagios
sudo chkconfig nagios on
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
sudo systemctl start nagios
error reading information on service nagios: No such file or directory
Note: Forwarding request to 'systemctl enable nagios.service'.
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.

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

```
Checking for circular paths...
        Checked 1 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
[ec2-user@ip-172-31-34-87 nagios-plugins-2.3.3]$
```

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.

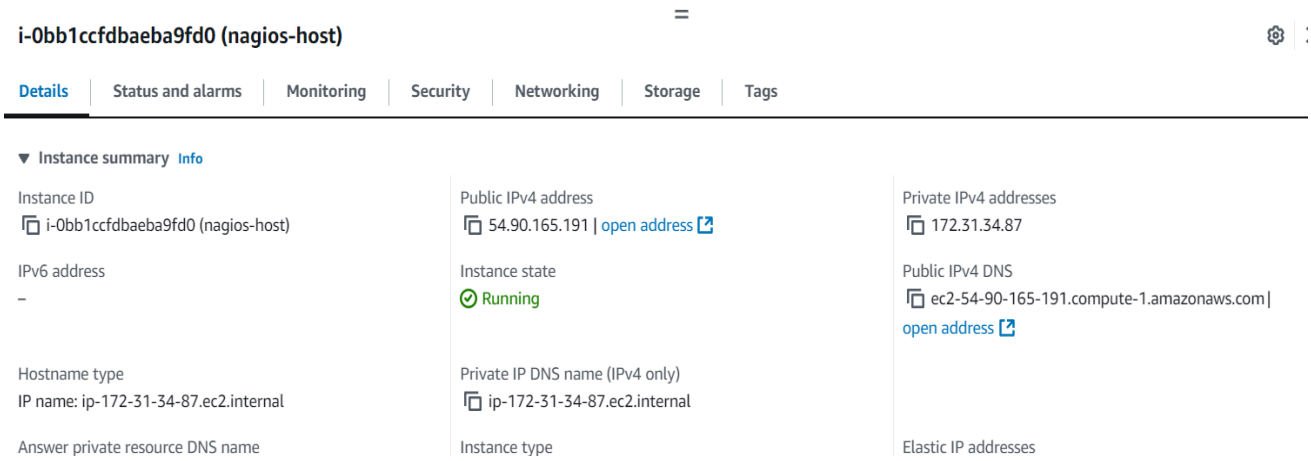
sudo service nagios start

21. Check the status of Nagios

sudo systemctl status nagios

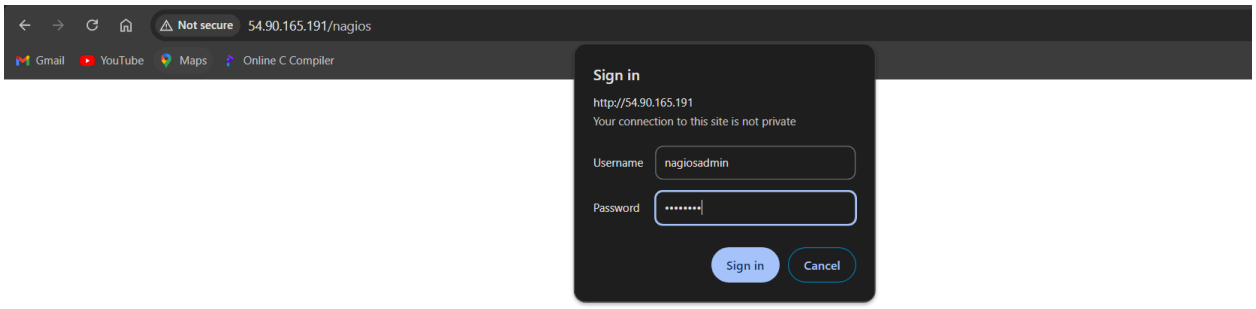
```
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Wed 2024-10-02 17:07:23 UTC; 50s ago
       Docs: https://www.nagios.org/documentation
    Process: 67754 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 67755 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 67756 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 2.1M
        CPU: 27ms
     CGroup: /system.slice/nagios.service
             ├─67756 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─67757 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─67758 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─67759 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─67760 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─67761 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 02 17:07:23 ip-172-31-34-87.ec2.internal nagios[67756]: qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successfully initialized
Oct 02 17:07:23 ip-172-31-34-87.ec2.internal nagios[67756]: qh: core query handler registered
Oct 02 17:07:23 ip-172-31-34-87.ec2.internal nagios[67756]: qh: echo service query handler registered
Oct 02 17:07:23 ip-172-31-34-87.ec2.internal nagios[67756]: qh: help for the query handler registered
Oct 02 17:07:23 ip-172-31-34-87.ec2.internal nagios[67756]: wproc: Successfully registered manager as @wproc with query handler
Oct 02 17:07:23 ip-172-31-34-87.ec2.internal nagios[67756]: wproc: Registry request: name=Core Worker 67760;pid=67760
Oct 02 17:07:23 ip-172-31-34-87.ec2.internal nagios[67756]: wproc: Registry request: name=Core Worker 67759;pid=67759
Oct 02 17:07:23 ip-172-31-34-87.ec2.internal nagios[67756]: wproc: Registry request: name=Core Worker 67758;pid=67758
lines 1-26
```

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

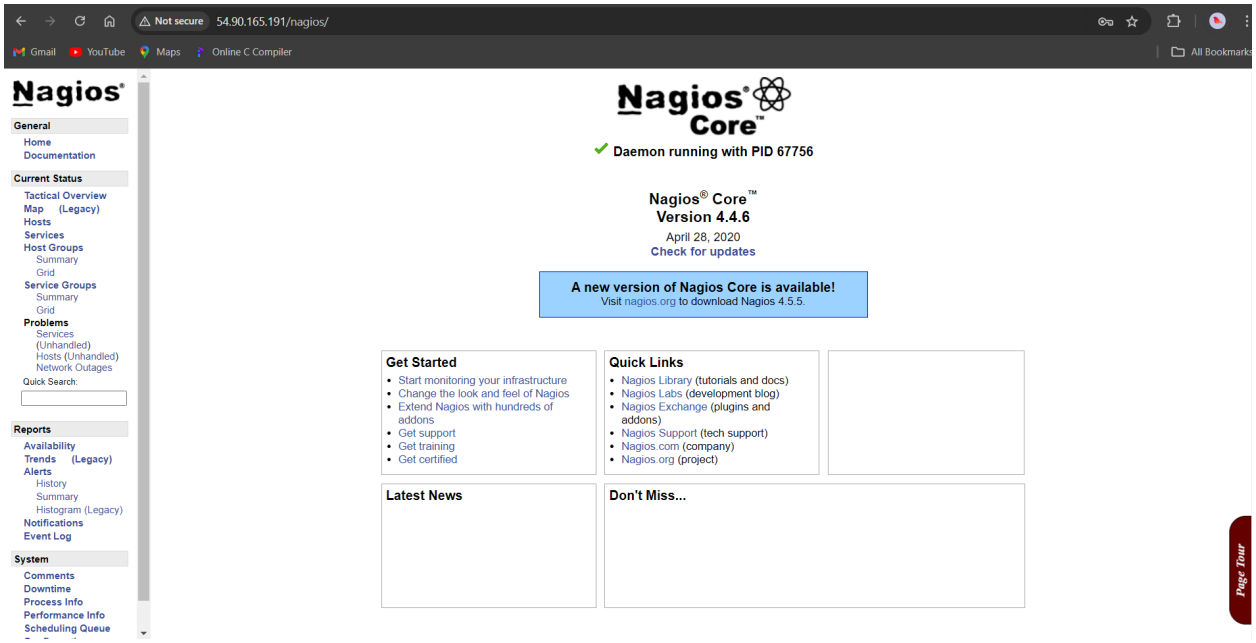


23. Open up your browser and look for http://<your_public_ip_address>/nagios

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



24. After entering the correct credentials, you will see this page.



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

Shravani Anil Patil **D15A-38**

Advanced DevOps Lab
Experiment 10

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

Theory:

Nagios is a comprehensive monitoring and alerting platform designed to keep track of IT infrastructure, networks, and applications. It provides real-time monitoring, alerting, and reporting capabilities to ensure the health and performance of critical systems.

Key Components of Nagios

1.Nagios Core: The open-source foundation of the Nagios monitoring system. It provides the basic framework for monitoring and alerting.

2.Nagios XI: A commercial version of Nagios that offers advanced features, a more user-friendly interface, and additional support options.

3.Nagios Log Server: A tool for centralized log management, allowing you to view, analyze, and archive logs from various sources.

4.Nagios Network Analyzer: Provides detailed insights into network traffic and bandwidth usage.

5.Nagios Fusion: Centralizes monitoring data from multiple Nagios instances, providing a unified view of the entire networks.

How Nagios Works

1.Configuration: Administrators define what to monitor and how to monitor it using configuration files.

2.Plugins: Nagios uses plugins to gather information about the status of various services and hosts. These plugins can be custom scripts or pre-built ones.

3.Scheduling: Nagios schedules regular checks of the defined services and hosts using the configured plugins.

4.Alerting: If a check indicates a problem, Nagios triggers an alert. Alerts can be configured to escalate if not acknowledged within a certain timeframe.

5. Log Management: Centralizing and analyzing logs from various sources to detect issues and ensure compliance.

Implementation :

Prerequisites

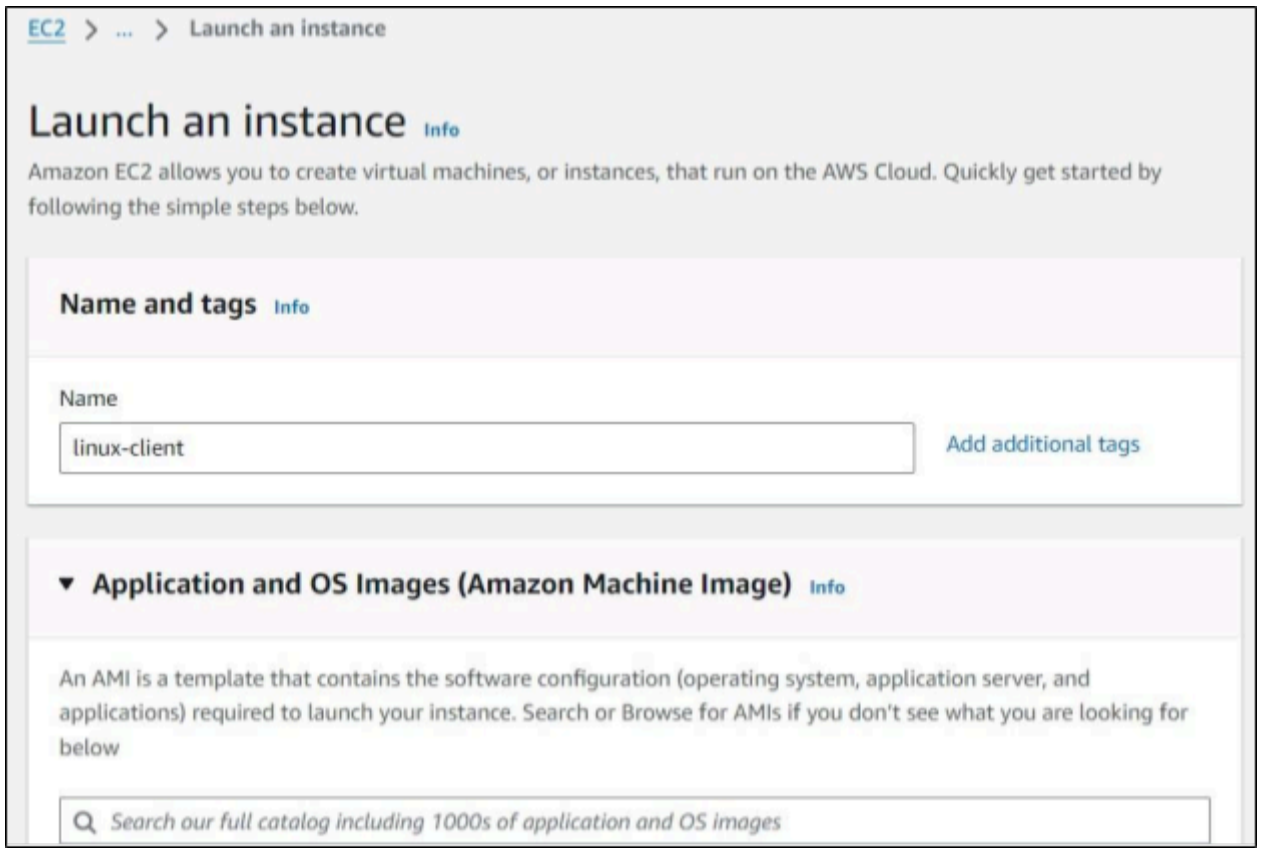
- AWS Free Tier
- Nagios Server running on an Amazon Linux Machine

1. **Confirm Nagios is Running on the Server**
   - sudo systemctl status nagios
   - Proceed if you see that Nagios is active and running.

```
Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-42-50 nagios-plugins-2.3.3]$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Mon 2024-10-07 16:28:45 UTC; 38s ago
       Docs: https://www.nagios.org/documentation
    Process: 69362 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
    Process: 69363 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (c
   Main PID: 69364 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 2.1M
        CPU: 22ms
     CGroup: /system.slice/nagios.service
             ├─69364 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─69365 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─69366 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─69367 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─69368 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─69369 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
```

2. Create an Ubuntu 20.04 Server EC2 Instance

- Name it linux-client.
- Use the same security group as the Nagios Host



3. Verify Nagios Process on the Server

Commands

- - ps -ef | grep nagios

```
[ec2-user@ip-172-31-42-50 nagios-plugins-2.3.3]$ ps -ef | grep nagios
nagios     69364       1  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios -d
nagios     69365   69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios --
nagios     69366   69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios --
nagios     69367   69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios --
nagios     69368   69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios --
nagios     69369   69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios -d
ec2-user   70969    2909  0 16:55 pts/0    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-42-50 nagios-plugins-2.3.3]$
```

4. Become Root User and Create Directories

sudo su

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

```
[ec2-user@ip-172-31-42-50 nagios-plugins-2.3.3]$ sudo su
mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-42-50 nagios-plugins-2.3.3]#
```

5. Copy Sample Configuration File

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

```
[root@ip-172-31-42-50 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/
[root@ip-172-31-42-50 ec2-user]# nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
[root@ip-172-31-42-50 ec2-user]#
```

6. Edit the Configuration File

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

- Change hostname to linuxserver everywhere in 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                   linuxserver
    address                 127.0.0.1
}

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

7. Update Nagios Configuration

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

- Add the following line: cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/
- Change hostgroup_name under hostgroup to linux-servers1

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

# 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
}



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

8. Verify Configuration Files

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

```
[root@ip-172-31-42-50 ec2-user]# sudo /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 global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-42-50 ec2-user]#
```

9. **Restart Nagios Service**
   - sudo systemctl restart nagios

10. SSH into the Client Machine

● Use SSH or EC2 Instance Connect to access the linux-client.

11. Update Package Index and Install Required Packages

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

12. Edit NRPE Configuration File Commands -

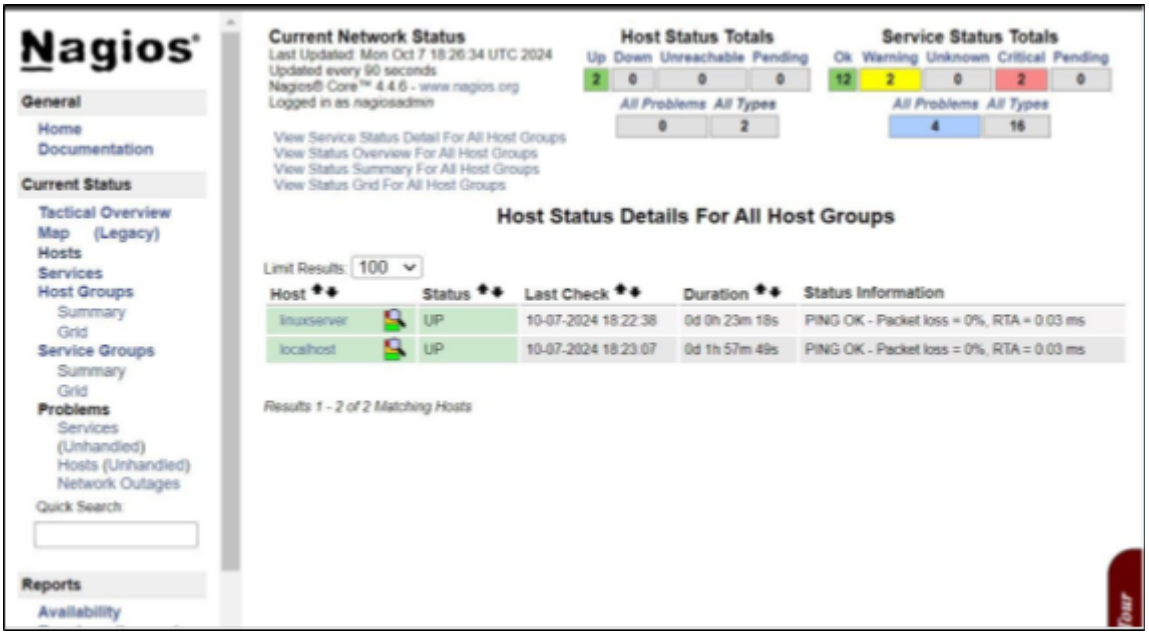
sudo nano /etc/nagios/nrpe.cfg

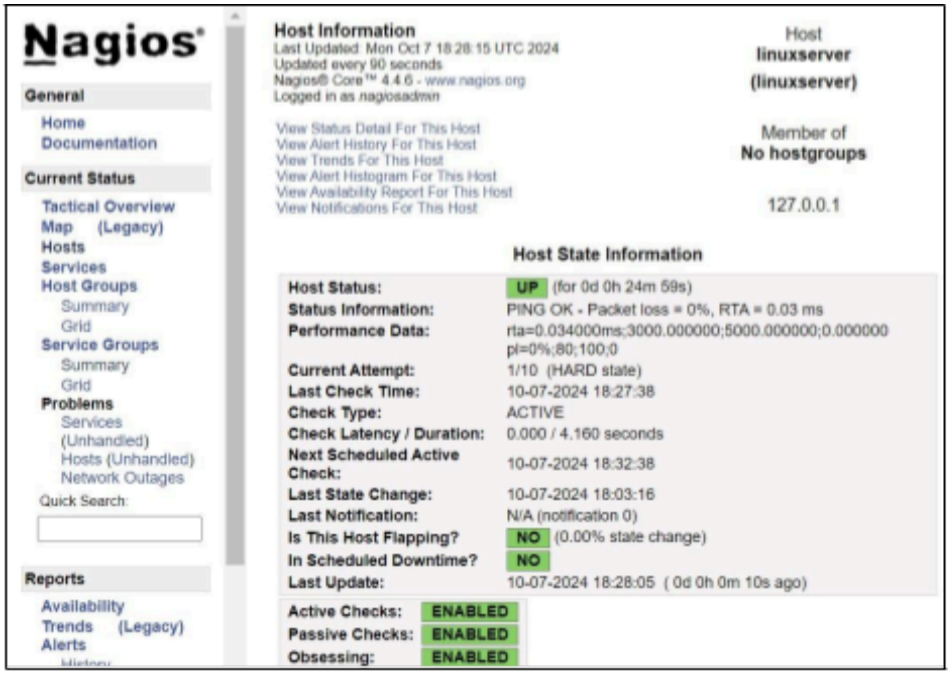
● Add your Nagios host IP address under allowed_hosts: allowed_hosts=

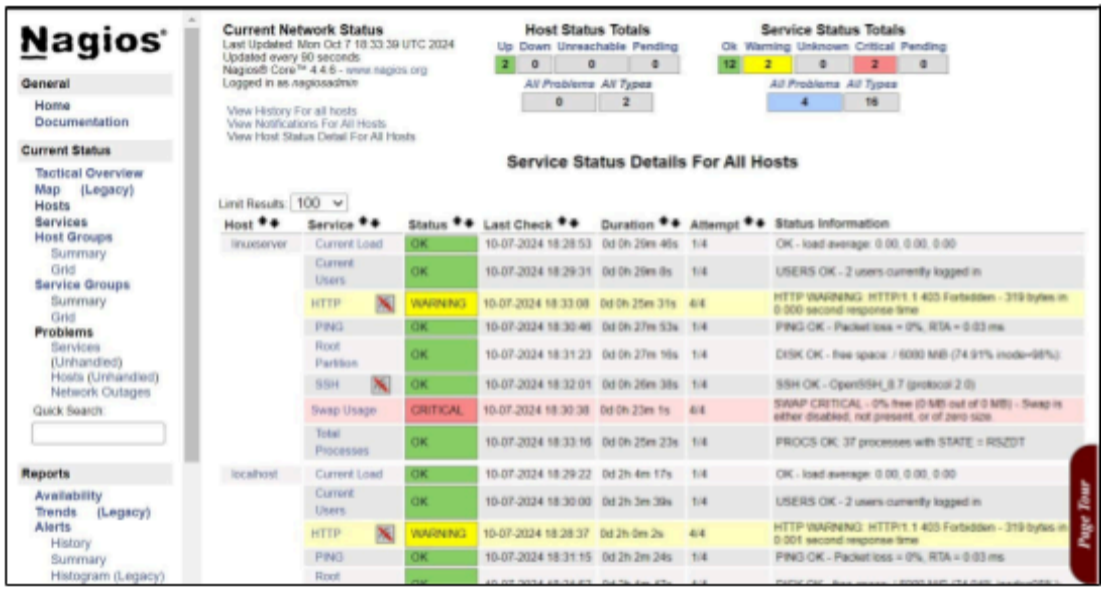
```
# supported.
#
# Note: The daemon only does rudimentary checking of the client's IP
# address.  I would highly recommend adding entries in your /etc/hosts.allow
# file to allow only the specified host to connect to the port
# you are running this daemon on.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

allowed_hosts=127.0.0.1,3.81.151.142



# COMMAND ARGUMENT PROCESSING
# This option determines whether or not the NRPE daemon will allow clients
# to specify arguments to commands that are executed.  This option only works
# if the daemon was configured with the --enable-command-args configure script
# option.
#
```

14. Check Nagios Dashboard

- Open your browser and navigate to http:///nagios.
- Log in with nagiosadmin and the password you set earlier.
- You should see the new host linuxserver added.
- Click on Hosts to see the host details.
- Click on Services to see all services and ports being monitored







Conclusion:

To perform port, service, and Windows/Linux server monitoring using Nagios, configure the necessary plugins and agents, define the monitoring parameters in the configuration files, and set up alerting mechanisms to ensure timely notifications of any issues. This comprehensive approach ensures robust monitoring and quick response to potential problems, maintaining the health and performance of your IT infrastructure.

Shravani Anil Patil **D15A-38**

Advanced DevOps Lab
Experiment 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 run these functions. AWS Lambda is a fully managed service that takes care of all the infrastructure for you. And so "serverless" doesn't mean that there are no servers involved: it just means that the servers, the operating systems, the network layer and the rest of the infrastructure have already been taken care of so that you can focus on writing application code.

Features of AWS Lambda

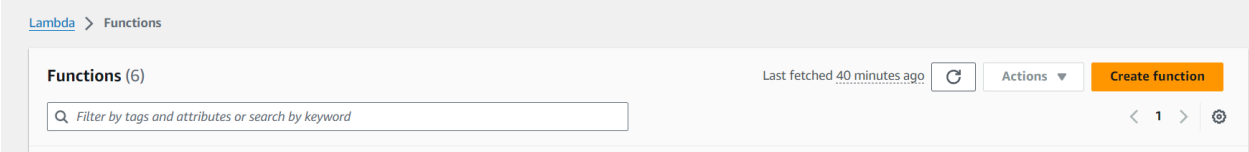
● AWS Lambda easily scales the infrastructure without any additional configuration. It reduces the operational work involved.
● It offers multiple options like AWS S3, CloudWatch, DynamoDB, API Gateway, Kinesis, CodeCommit, and many more to trigger an event.
● You don't need to invest upfront. You pay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

Packaging Functions

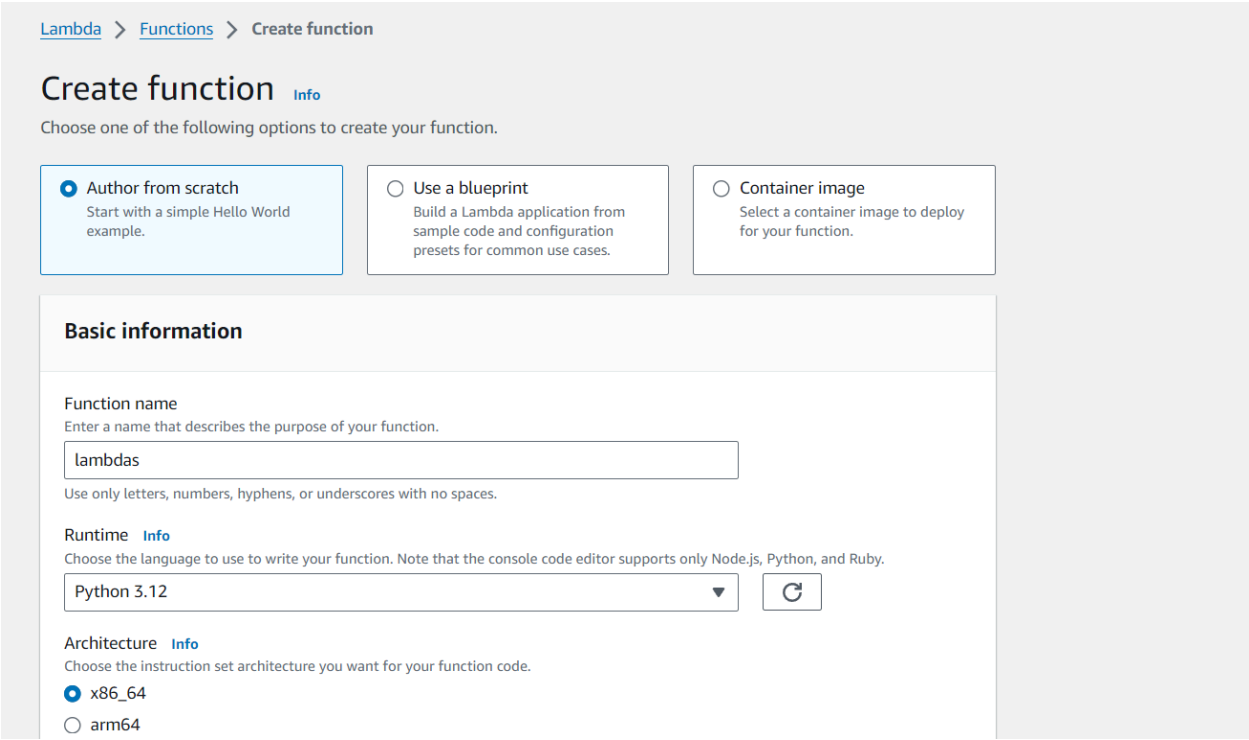
Lambda functions need to be packaged and sent to AWS. This is usually a process of compressing the function and all its dependencies and uploading it to an S3 bucket. And letting AWS know that you want to use this package when a specific event takes place. To help us with this process we use the Serverless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

Steps to create an AWS Lambda function

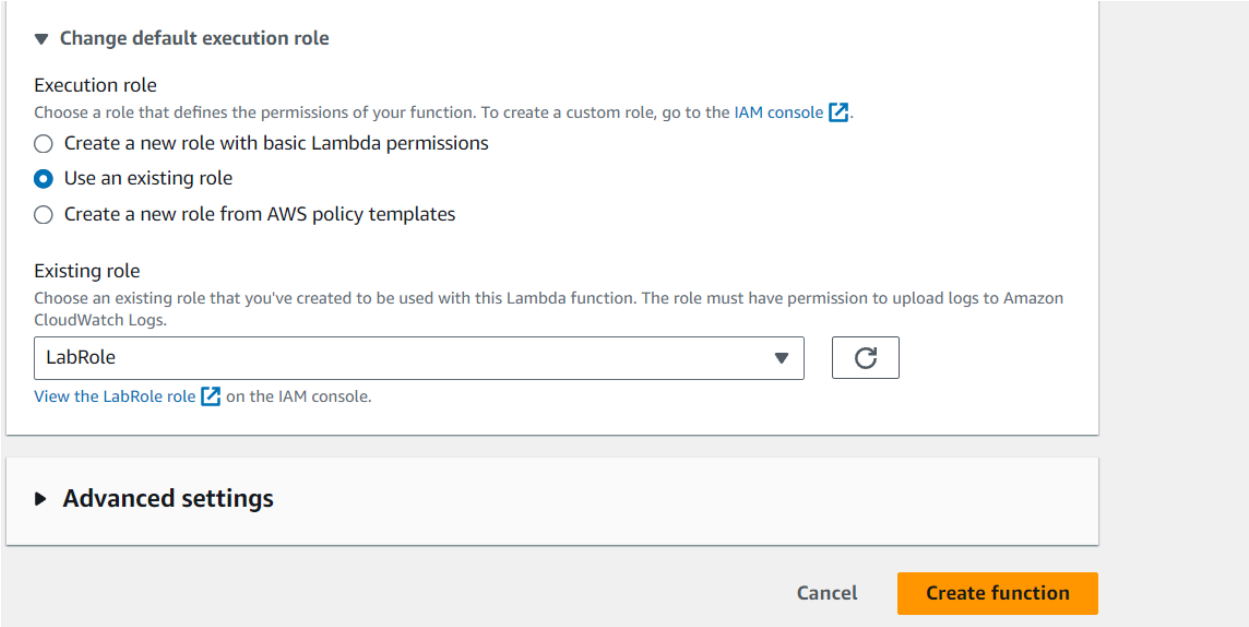
1. Open up the Lambda Console and click on the Create button.



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.

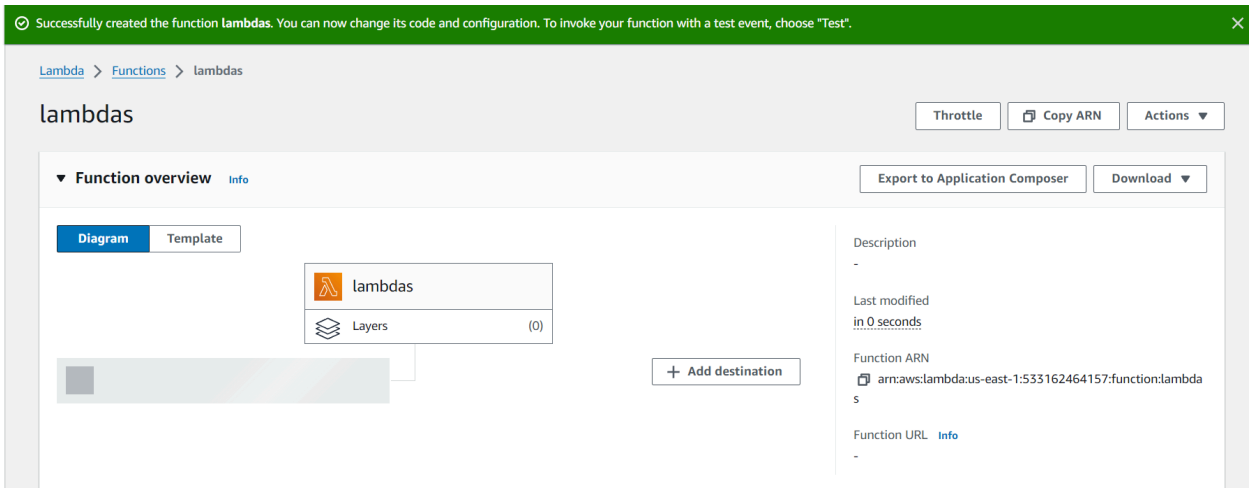


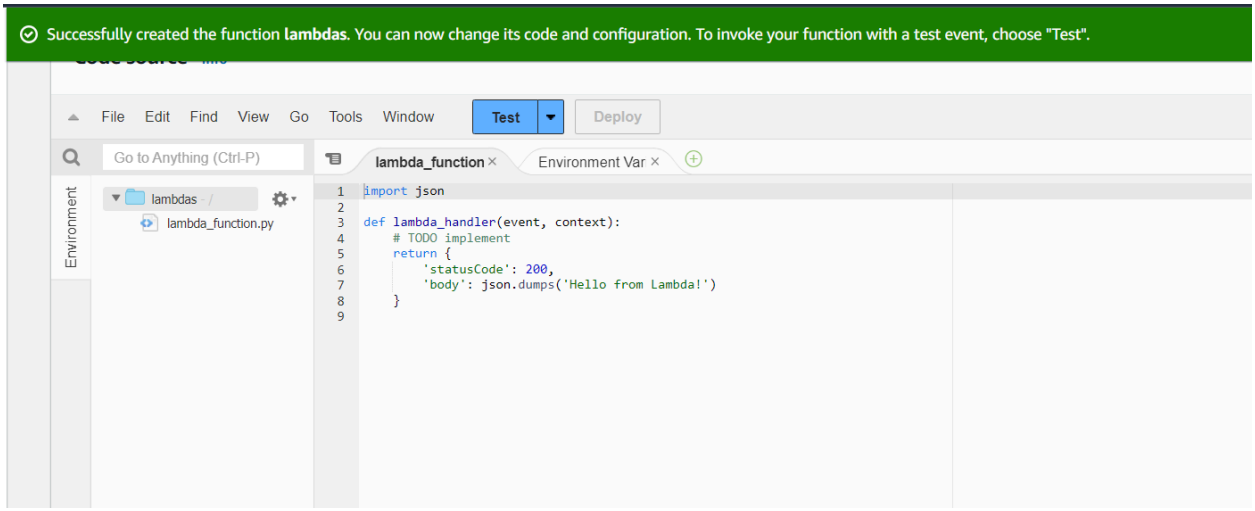
After that, choose an existing role or create a new role with basic Lambda permissions if you don't have an existing one.



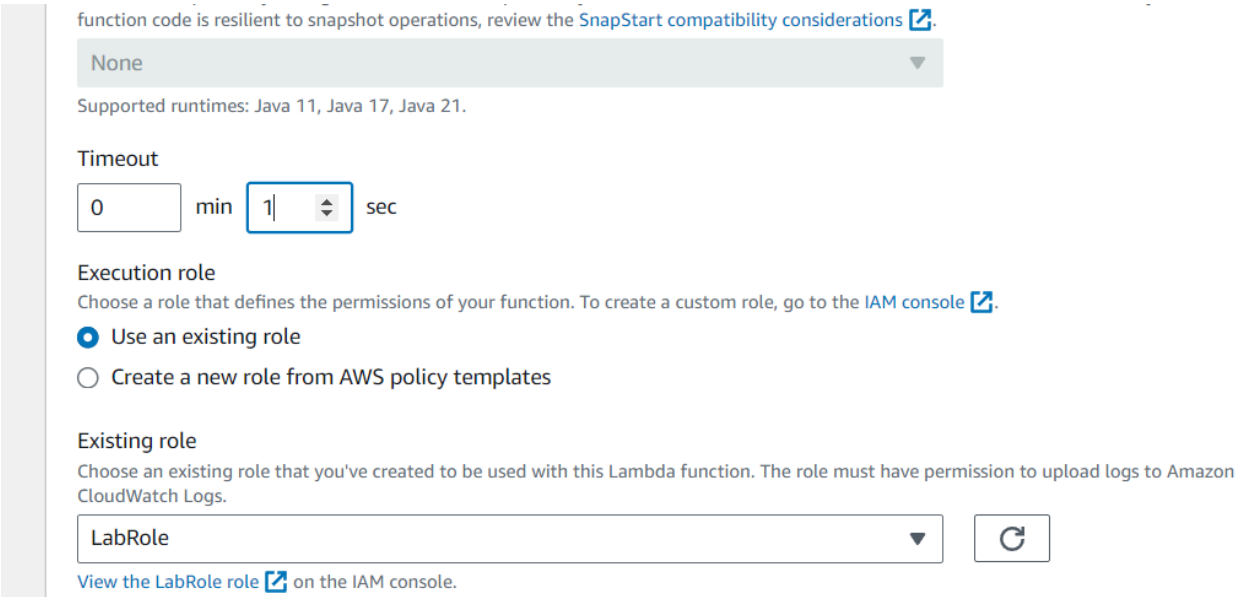
Click on the Create button.

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

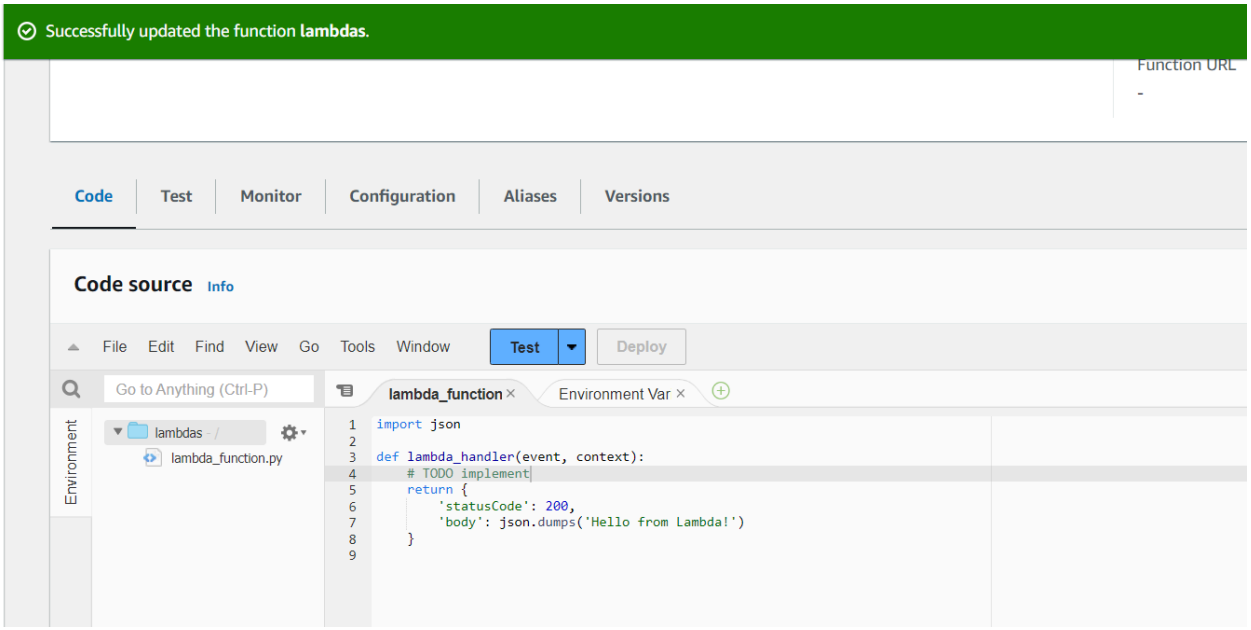




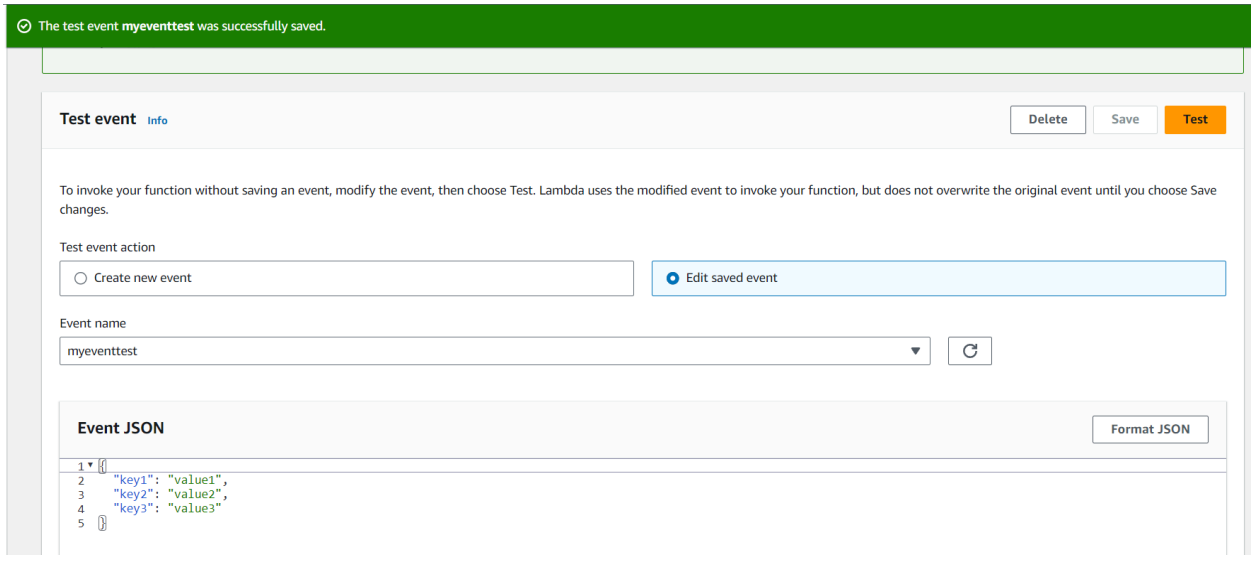
4. To change the configuration, open up the Configuration tab and under General Configuration, choose Edit.
Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.



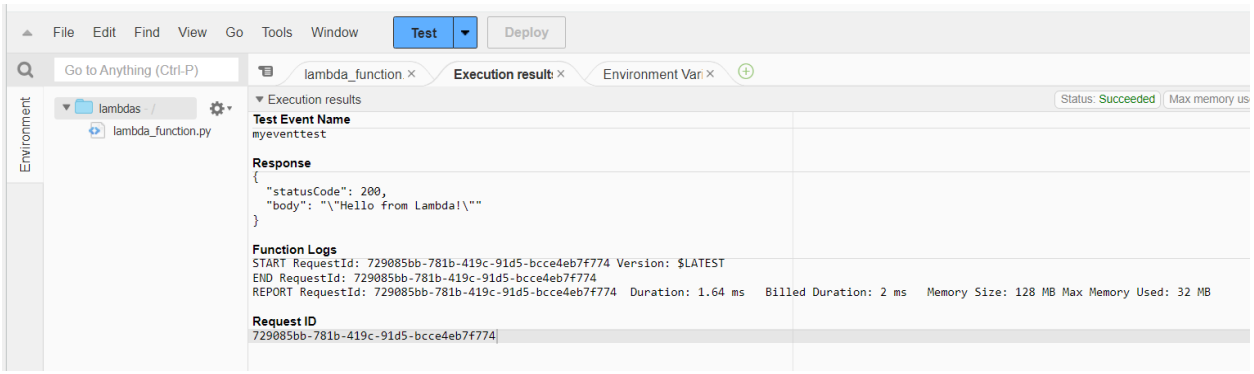
5. You can make changes to your function inside the code editor. You can also upload a zip file of your function or upload one from an S3 bucket if needed.
Press Ctrl + S to save the file and click Deploy to deploy the changes.



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, AWS Lambda offers an excellent opportunity for students to implement event-driven applications without the hassle of managing server infrastructure. By using Lambda, students can focus on developing their coding skills while creating scalable and cost-effective applications.

For instance, students can create projects that automatically process data when a file is uploaded to an Amazon S3 bucket, such as image processing or data analysis, reinforcing their understanding of real-time data handling. The straightforward workflow of defining functions, configuring triggers, and deploying applications allows students to quickly bring their ideas to life.

Additionally, with support for popular programming languages like Python, Java, and Node.js, students can choose the language they are most comfortable with or explore new ones. This hands-on experience with AWS Lambda not only enhances their technical skills but also prepares them for modern software development practices in a cloud-based environment. Overall, AWS Lambda is a valuable tool for students looking to build innovative, efficient, and scalable applications in their learning journey.

Shravani Anil Patil **D15A-38**

Advanced DevOps Lab
Experiment 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.

AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers. It automatically scales your application by running code in response to events. One of the most common use cases for Lambda is integrating it with Amazon S3, a scalable object storage service.

When an object is added to or modified in an S3 bucket, you can configure an S3 event notification to trigger a Lambda function. This event-driven architecture simplifies processing data in real time and reduces the need for manual intervention.

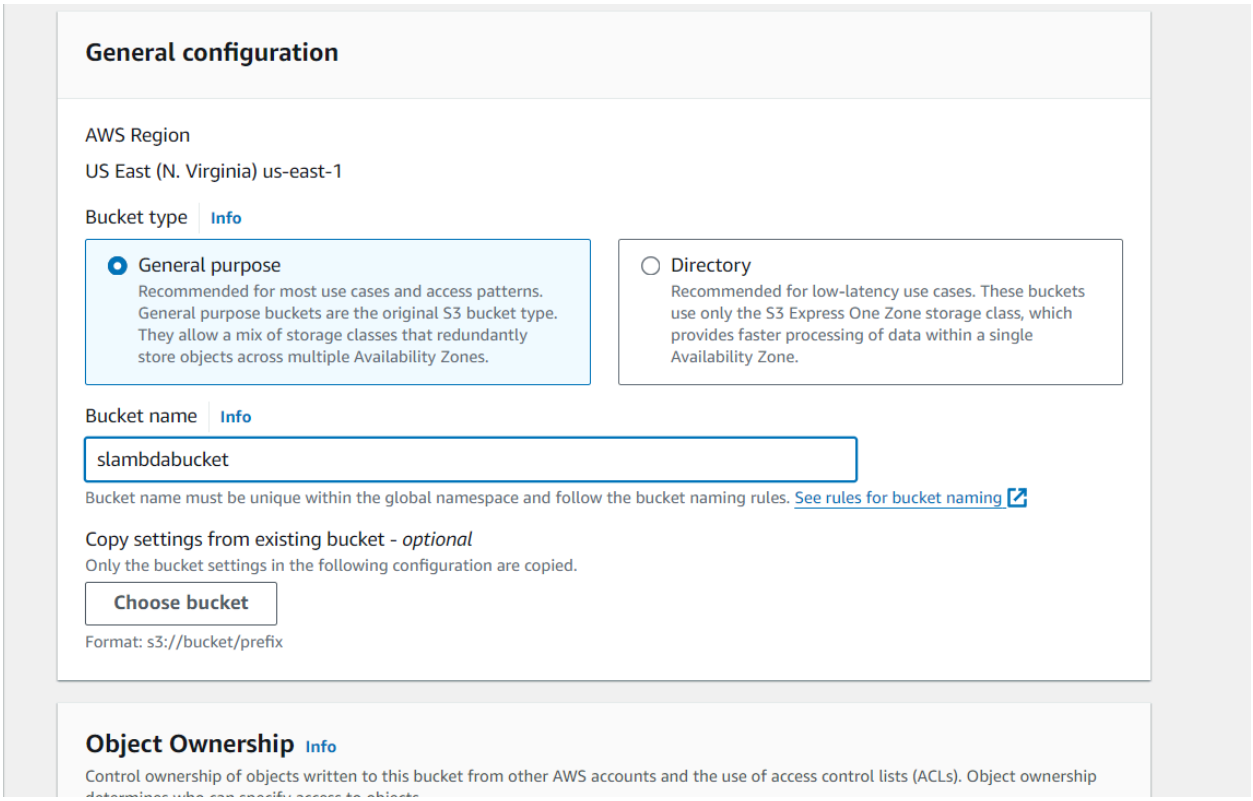
Key Benefits

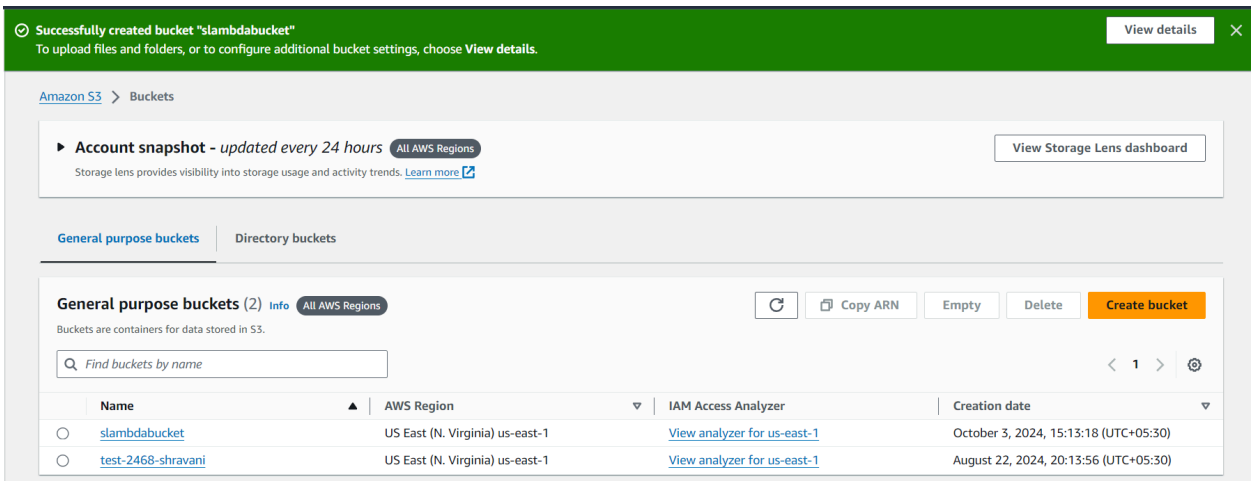
1. **Serverless Architecture**:
    - No need to manage servers or infrastructure. You focus solely on writing code.
    - AWS handles scaling automatically based on the number of events.
2. **Cost Efficiency**:
    - Pay only for the compute time you consume. You are charged based on the number of requests and the duration of execution.
3. **Real-time Processing**:
    - Automatically process files as they are uploaded, allowing for instant reactions to events (e.g., generating thumbnails, analyzing data).
4. **Ease of Integration**:
    - Lambda integrates easily with other AWS services, enabling seamless workflows and data processing pipelines.
5. **Event-Driven Processing**:
    - Respond to changes in data, such as new uploads, deletions, or updates, without the need for polling or scheduled tasks.

Workflow:

1. Create an S3 Bucket:

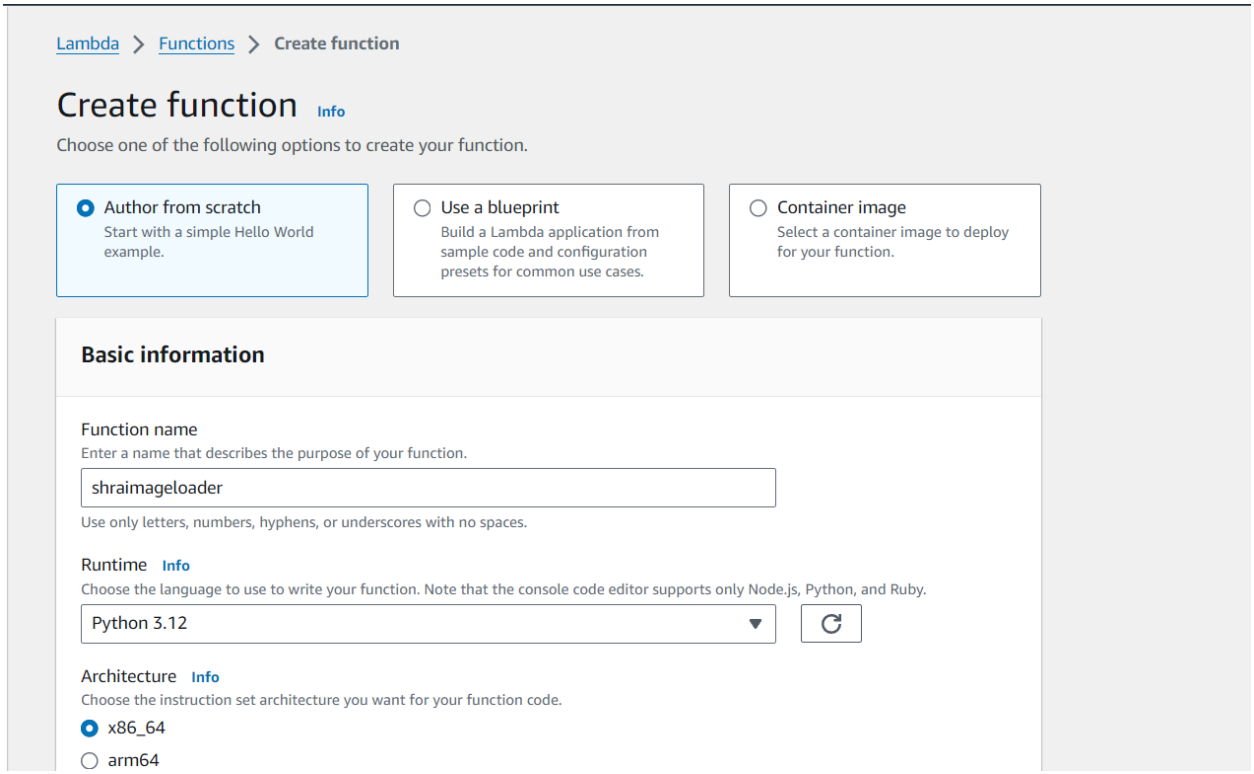
○ First, create an S3 bucket that will store the objects. This bucket will act as the trigger source for the Lambda function.

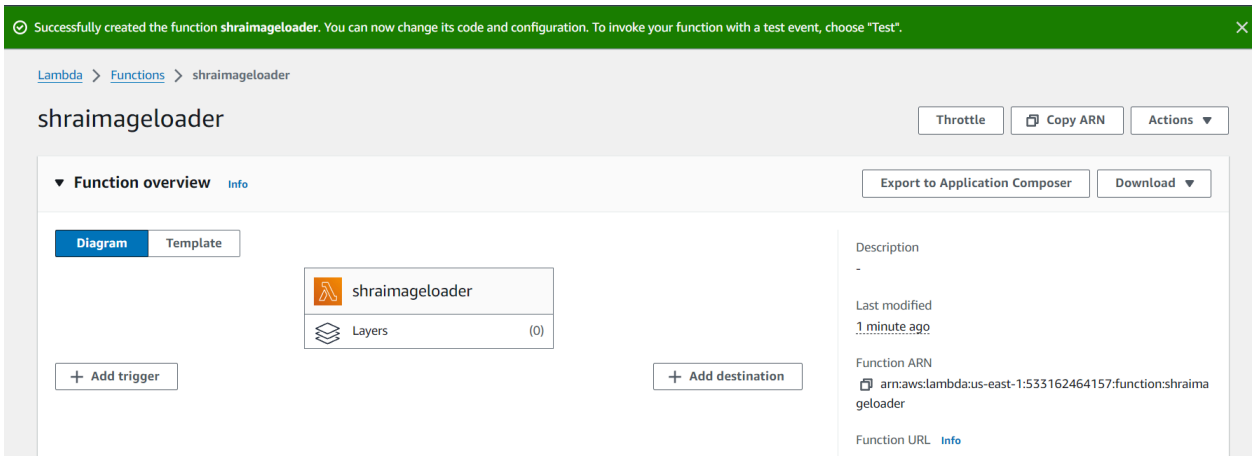




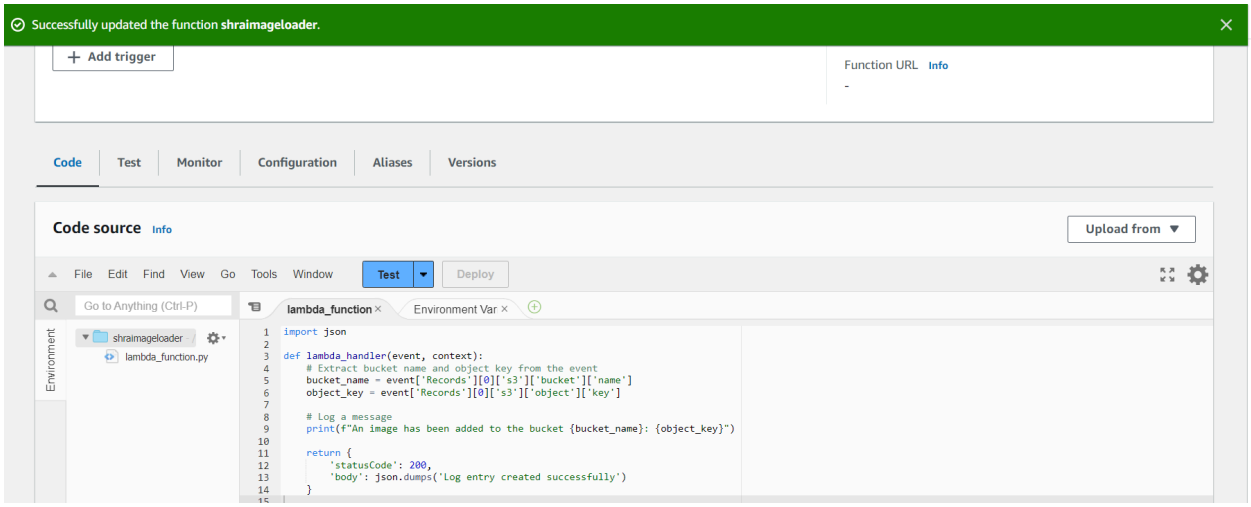
2. Create the Lambda Function:

○ 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.**

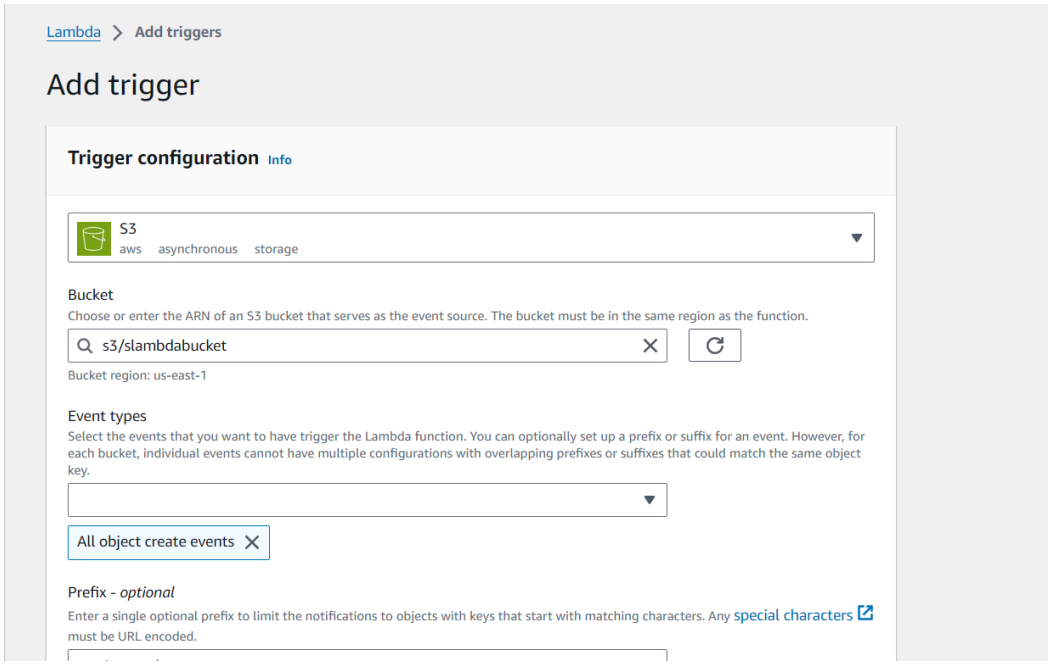


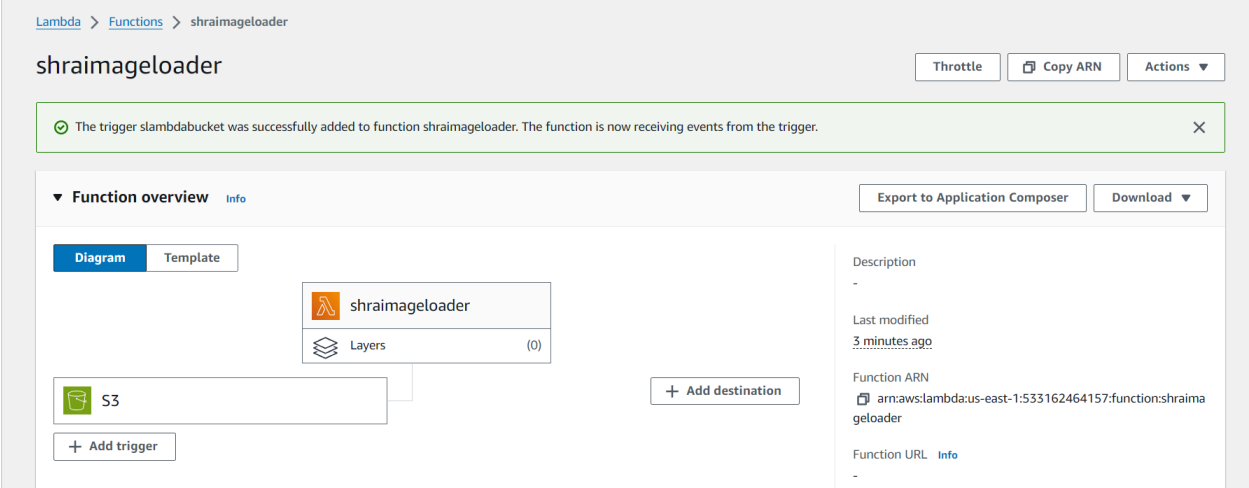
3. Set Up Permissions:

○ 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:

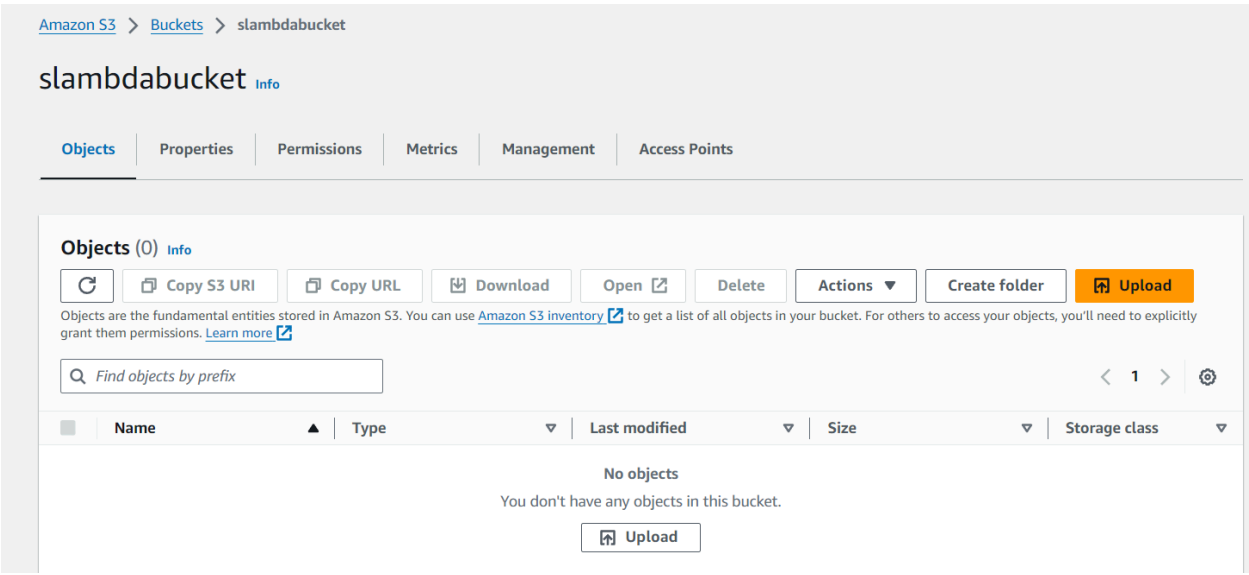
○ Link the S3 bucket to the Lambda function by setting up a trigger. Specify that the function should be triggered when an object is created in the bucket (e.g., when an image is uploaded).



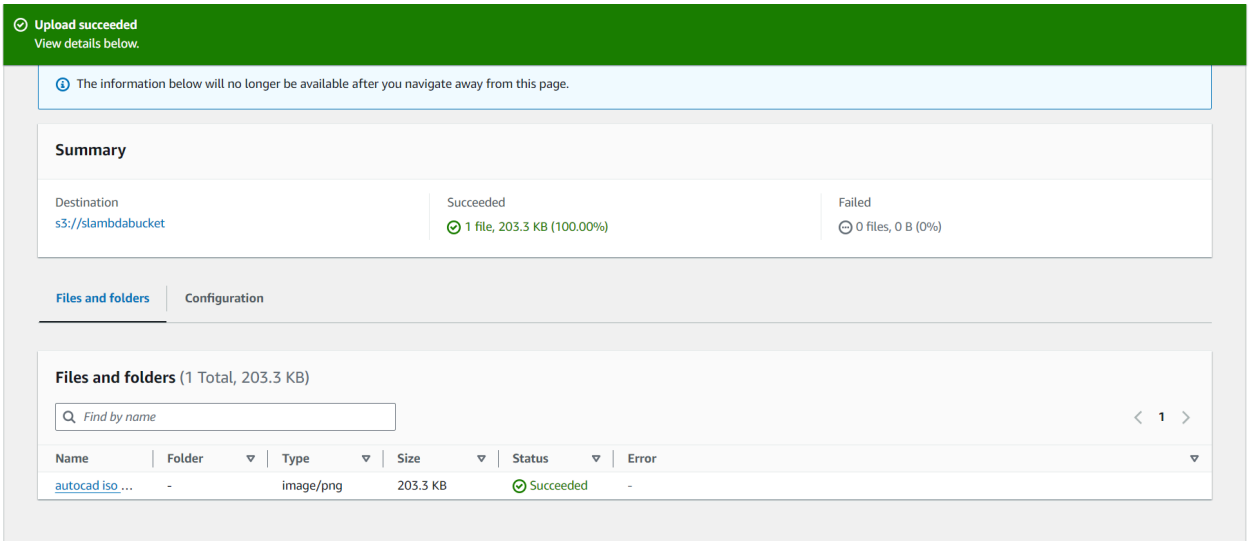


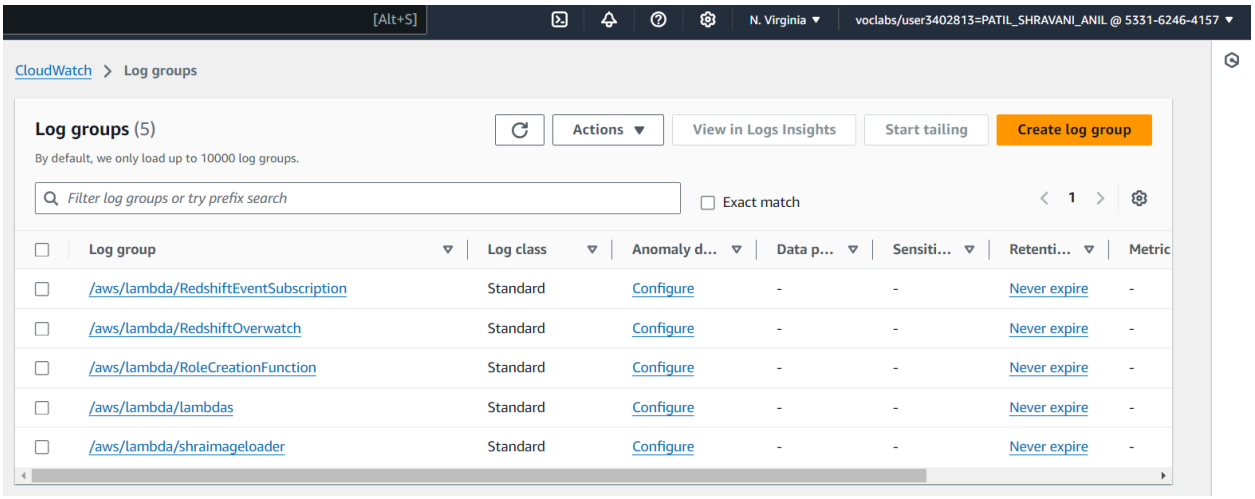
5. Test the Setup:

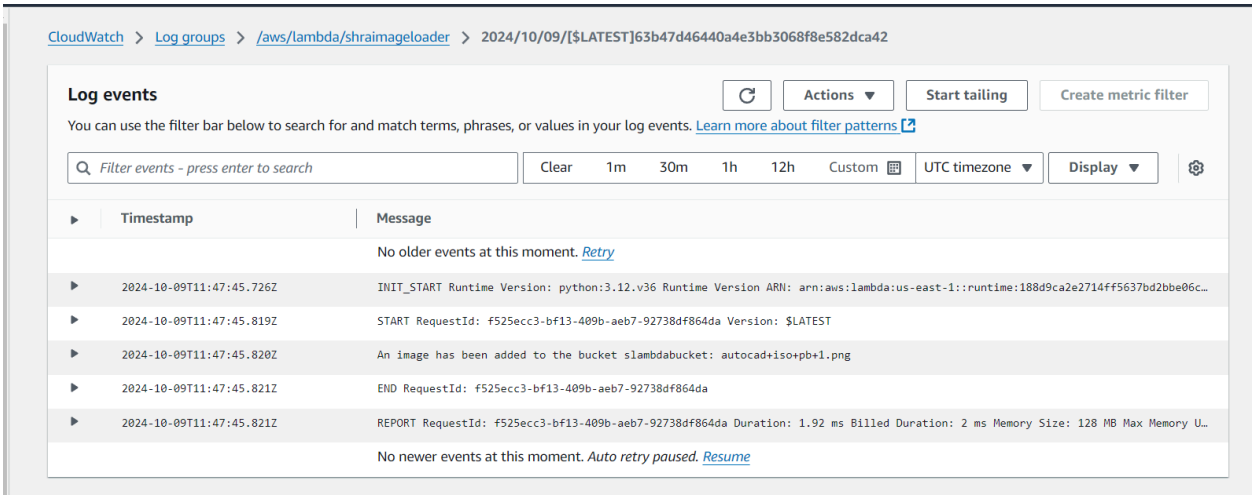
○ Upload an object (e.g., an image) to the S3 bucket to test the trigger. The Lambda function should execute and log the message "An Image has been added" in AWS CloudWatch Logs.



Outcomes:


Upload succeeded
View details below.
The information below will no longer be available after you navigate away from this page.
Summary
Destination
s3://slambdabucket
Succeeded
1 file, 203.3 KB (100.00%)
Failed
0 files, 0 B (0%)
Files and folders
Configuration
Files and folders (1 Total, 203.3 KB)
Find by name
Name
Folder
Type
Size
Status
Error
autocad iso ...
-
image/png
203.3 KB
Succeeded
-


[Alt+S]
N. Virginia
voclabs/user3402813=PATIL_SHRAVANI_ANIL @ 5331-6246-4157
CloudWatch
Log groups
Log groups (5)
By default, we only load up to 10000 log groups.
Actions
View in Logs Insights
Start tailing
Create log group
Filter log groups or try prefix search
Exact match
Log group
Log class
Anomaly d...
Data p...
Sensiti...
Retenti...
Metric
/aws/lambda/RedshiftEventSubscription
Standard
Configure
Never expire
/aws/lambda/RedshiftOverwatch
Standard
Configure
Never expire
/aws/lambda/RoleCreationFunction
Standard
Configure
Never expire
/aws/lambda/lambdas
Standard
Configure
Never expire
/aws/lambda/shraimageloader
Standard
Configure
Never expire


CloudWatch
Log groups
/aws/lambda/shraimageloader
2024/10/09/[$LATEST]63b47d46440a4e3bb3068f8e582dca42
Log events
Actions
Start tailing
Create metric filter
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns
Filter events - press enter to search
Clear
1m
30m
1h
12h
Custom
UTC timezone
Display
Timestamp
Message
No older events at this moment. Retry
2024-10-09T11:47:45.726Z
INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:188d9ca2e2714ff5637bd2bbe06c...
2024-10-09T11:47:45.819Z
START RequestId: f525ecc3-bf13-409b-aeb7-92738df864da Version: $LATEST
2024-10-09T11:47:45.820Z
An image has been added to the bucket slambdabucket: autocad+iso+pb+1.png
2024-10-09T11:47:45.821Z
END RequestId: f525ecc3-bf13-409b-aeb7-92738df864da
2024-10-09T11:47:45.821Z
REPORT RequestId: f525ecc3-bf13-409b-aeb7-92738df864da Duration: 1.92 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory U...
No newer events at this moment. Auto retry paused. Resume

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.

Shravani Patil
D15A - 38

04/08

27/09/24 Advance Devops - Assignment no: 01

Q.1. Use S3 bucket and host video streaming

→ Amazon S3 bucket with Amazon cloudfront is used to host on demand videos (video content is stored on server eg movies can watch any time) Eg Amazon cloudfront is web service that speeds up distribution of static & dynamic web content to users. cloudfront delivers contents through worldwide network of data centers called edge collections cloudfront used the cache for the latest lowest latency. If content is already in edge collection it will deliver immediately otherwise it will retrieve from the origin (S3 bucket). In our case, media package HTTP server.

steps for host video streaming using S3 bucket:-

i. Create an S3 bucket
ii. Upload a video to the S3 bucket.
iii. Create a cloudfront origin access identity.
iv. Create a cloudfront distribution
v. Access the the video through cloudfront distribution
vi. Configure your cloudfront distribution to use your custom domain name.

vii. Access S3 video through cloudfront distribution with the custom domain name.

viii. Optional - view data requests received by your cloudfront distribution.

ix. clean up.

Q.4. What is Nagios? explain how nagios is used in E-services?

→ Nagios is an open source monitoring platform/ tool designed to oversee systems, networks & infrastructure. It helps organisation identify and resolve IT infrastructure problems before they impact critical business processes.

Nagios uses in e-services:

→ They are publically available services such as HTTP, STP, SMTP etc. These services are network accessible. Services like web services, email services while private services need intermediary agents for monitoring.
Nagios used plugins to monitor e-services many of which come pre-installed & additional plugins can be fetched, found online or developed by users. To monitor a service, a host must first be defined in Nagios configuration. Nagios provide alert if services failed to respond...

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

→ Overview:

BMW is a leading automotive manufacturer known for its luxury vehicles, while Hotstar is a popular streaming platform in India, offering a variety of content, including movies, TV shows and live sports. Both companies have utilised AWS to drive the innovation, improve customer experience & optimise their operations.

BMW's use of AWS

1. Connected vehicles and Data Analytics

BMW has been at the forefront of integrating technology into their vehicles. By leveraging AWS they can collect and analyse vast amounts data from their connected cars. The data includes

- vehicle performance data: Monitoring engine performance, fuel efficiency & maintenance needs.
- Driver behaviour: Understanding how drivers interact with their vehicles which can lead to personalised services and safety features.

Benefits:

- Predictive Maintainence: AWS enables BMW to use machine learning algorithms to predict when a vehicle needs servicing, reducing downtime & improving customer features.
- Enhanced Customer insights: By analyzing driven patterns, BMW can tailor marketing strategies & develop features that resonate with these customers.

2. Scalability & cost management
BMW utilizes AWS's scalable infrastructure to handle varying workloads, especially during product launches or events.
- Cost efficiency - BMW can scale resources up or down based on demand, ensuring they only pay for what they use.
- Global reach - AWS's global infrastructure allows BMW to deploy applications closer to their customers, reducing latency & improving service delivery.

Hotstar use of AWS

1. Content Delivery & streaming services

Hotstar relies heavily on AWS to manage its massive content library & deliver high

deliver high quality streaming experiences to millions of users

- AWS cloudfront - Hotstar uses AWS's content delivery Network (CDN) to deliver content quickly eg efficiently, ensuring minimal buffering eg downtime during peak times such as major sports events.

Benefits:

- Scalability: During events like IPL, when there is a massive traffic, AWS allows Hotstar to scale resources dynamically to handle these spikes without compromising performance.
- Global Content Reach: AWS enables Hotstar to distribute content across multiple regions, ensuring that users worldwide can access their services seamlessly.

Q: Data Analytics for User Engagement
Hotstar leverages AWS data analytics tools to gather insights about user behaviour, content preferences eg viewing patterns.

- AWS redshift eg Athena - These services help Hotstar analyse large datasets to improve content recommendations eg user engagement strategies.

Benefits:

- Personalised content Recommendations: By analysing viewing habits can suggest relevant content to users, enhancing their viewing experience.
- Targeted Advertising: Insights gathered from user data enable platform to serve more targeted ads, increasing ads revenue & improving user satisfaction

Challenges & solutions:

While both BMW & Globoplay have seen significant benefits from using AWS, they also face challenges.

1. Data security & compliance - protecting user data & adhering to regulations is paramount especially in the automotive & streaming industries.
   - Solutions - Both companies utilise AWS robust security features like encryption, identity & access management (IAM) & compliance certifications
2. Cost Management - As the usage scales, managing cost can become challenging.
   Solutions: Implementing AWS cost explorer & using AWS Budgets helps both companies

monitor & optimize their cloud expenditure.

Conclusion..

Therefore, By leveraging AWS's scalability, data analytics & global infrastructure both companies are well structured to meet the evolving need of their customers and stay ahead of the competition.

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

→ What is Kubernetes?

Kubernetes is an open source container orchestration platform designed to automate the deployment, scaling and management of containerized applications. Originally developed by Google, it has become a standard for managing cloud-native applications & micro services architecture.

Advantages:

i. Automated Deployment & scaling: facilitates easy deployment and scaling of app^ns based on demand.

ii. self Healing: Automatically restarts or replaces failed containers.

iii. Load balancing - distributes traffic across containers for optimal resource use.

Disadvantages:

i. Complexity - steep learning curve & operational challenges, especially for beginners.
ii. Resource overhead - can require significant computational resources.
iii. Network challenges - configuring networking can be complex.
iv. Frequent updates - rapid evolution can lead to compatibility issues.

How Adidas uses Kubernetes

i. Microservices Architecture:
Adidas has adopted a microservices architecture to enable agility & faster delivery of features. Kubernetes allows Adidas to manage these microservices effectively.

ii. Scalability & Performance:
During high traffic events, Adidas can use Kubernetes to scale its applications automatically based on user demand. For example, during Black Friday sales, Kubernetes can help ensure that the e-commerce platform remains responsive, handling large volumes of simultaneous transactions without downtime.

Shivani Patil
D15A-38

05/05/22

Advance DevOps
Assignment no: 02

1. Create a REST API with serverless framework

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

1. Serverless framework: A powerfull tool that simplifies deployment of serverless applications across various cloud providers such as AWS, Azure & Google cloud.
2. Serverless Architecture: This design model allows developers to build applications without worrying about underlying infrastructure, enabling focus on code & business logic.
3. Rest API: Representational State Transfer is architectural style for designing network applications.

Steps for creating RESTAPI for serverless framework:

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

2. create a Node.js serverless project.
A directory is created for your project where you initialise a serverless service (project). This service will host all your lambda functions, configurations & cloud resources. Using the command 'serverless create' you set up a template for AWS Node.js microservices that will eventually deploy to AWS lambda.

3. Project structure:
The project scaffold creates essential files like handler.js which contains code for lambda function.

4. create a REST API Resource
To in the serverless.yml file you define function that handles HTTP POST requests.

5. Deploy the service:
With the S/S deploy commands, serverless framework packages your applications uploads necessary resources to AWS & set up the infrastructure.

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

7. Storing data in Dynamo DB
To store submitted candidate data, you integrate AWS Dynamo DB as a database.

8. Adding more functionalities like 'list all candidates, get candidates by ID.

9. AWS IAM permissions:
You need to ensure that serverless framework is given right permissions to interact with AWS resources like Dynamo DB.

10. Monitoring & maintenance
After deployment serverless framework provides service information like deployed endpoints API key, log streams.

Q.2. Case study on sonarqube.
- Create your own project in sonarqube for testing project publicly
- Use sonarcloud to analyse your github code.
- Install sonarlint in your Java intellije IDE or eclipse IDE. On eclipse IDE and analyse your java code.
- Analyse python project with sonarqube.
- Analyse nodejs project with sonarqube.

→ Solution

- create the sonarqube profile for testing project quality.
- open intellij setting, find tools > sonarlint, ruby & select to open connection wizard.
- enter 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 on sonarqube connection only.

- for sonarcloud only select organisation that you want to connect
- sonarqube & sonarcloud can push notifications 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 sonarcloud.
- choose the correct project from sonarqube.

☐ Analyse the project (Python project)

- Trigger an analysis by going to code analyse code > sonarlint

Analyse Node.js project

- Make sure your Node.js project is properly configured with sonar-project.properties file or equivalent for the analysis to run.

Q3 At large organisation your centralised 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 & managing services in your organisation, allowing them to efficiently deploy services in compliance with your organisations policies. Terraform cloud can also integrate with tooling systems like service now to automatically generate new infrastructure requests.

→ self serve infrastructure model with terraform modules:

At a large organisation, implementation 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

organisational standards eg test practises.

key aspects of this self-service model includes:

a. Standardisation through terraform modules
by creating and utilising terraform modules
both - cloud organisations can codify their
infrastructure deployment eg management
standards. These modules serve as reusable
packages of terraform configurations. that
encapsulate common patterns eg best practises.

b. Efficient deployment. product teams can
leverage these standardised modules to
rapidly deploy services without needing to
reinvent the wheel or wait for the
centralised operation team to handle every
request.

c. Compliance
By using predefined modules, teams ensure
their deployments comply with the organisations
established practises eg security guidelines.

d. Automation:
The use of terraform modules promotes automation
reducing manual interventions eg potential
human errors in infrastructure management.

e. version control: with modules stored in version control systems like git, teams can track changes, collaborate on improvements & maintain a history of infrastructure configurations

Integration with ticketing system:
Terraform cloud offers integration capabilities that further enhance the self service model.

a. Automate Infrastructure Requests: Automatically generates new infrastructure requests. This automation streamlines the process of submitting & tracking infrastructure changes.

b. Centralized Management - By centralized infrastructure management through Terraform cloud organizations & maintain better control over who can request & approve infrastructure changes.

c. Governance - The integration with ticketing systems allows for better governance of infrastructure requests, ensuring that all changes go through proper approval processes before deployment.

Collaborative Infrastructure Management:

a. Team Based performance permissions
b. state & Run history.
c. sensitive information protection
d. Module registry.

By implementing these features & practices large organizations can effectively leverage Terraform to build a robust, scalable & compliant infrastructure management system that supports both centralized control & decentralized autonomy.