

Advanced DevOps Experiment-1

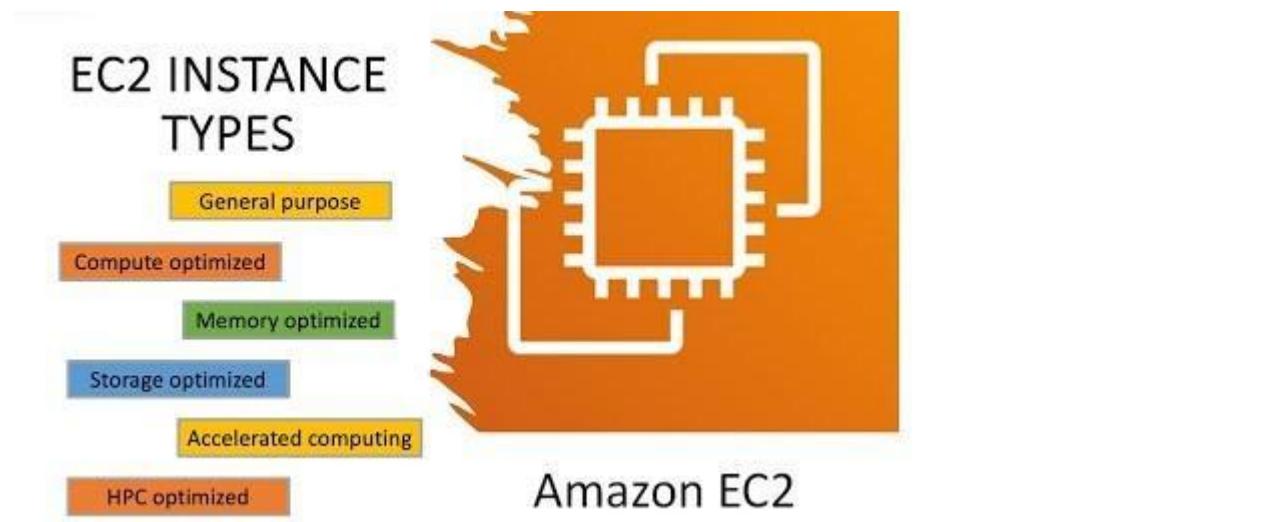
Aim: Using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

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

The screenshot shows the AWS EC2 Dashboard. On the left sidebar, under 'Instances', there are several options like Instances, Instance Types, Launch Templates, etc. In the main content area, there's a section titled 'Launch instance' with a large orange 'Launch instance' button. To the right of this, there's a 'Service health' section and an 'Explore AWS' sidebar with various promotional links.

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

The screenshot shows the 'Launch an instance' wizard. It has several tabs: 'Name and tags', 'Application and OS Images (Amazon Machine Image)', and 'Summary'. In the 'Name and tags' tab, the name 'MyWeb' is entered. In the 'Application and OS Images (Amazon Machine Image)' tab, an AMI is selected. The 'Summary' tab shows a summary of the configuration, including 1 instance, the selected AMI, the t2.micro instance type, and 1 volume(s) - 8 GiB. A tooltip for the 'Free tier' is visible, stating 'In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which you run it)'. At the bottom, there are 'Cancel', 'Launch instance', and 'Review commands' buttons.

The screenshot shows the AWS EC2 console with the URL us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances. The 'Amazon Machine Image (AMI)' section is selected, displaying the 'Amazon Linux 2023 AMI' (ami-066784287e358dad1). The instance type is set to 't2.micro'. A summary panel on the right indicates 1 instance will be launched, using the 'Amazon Linux 2023 AMI'. The 'Launch instance' button is highlighted.

The screenshot shows the continuation of the AWS EC2 launch process. The 'Configure storage' section is active, showing a single root volume of 8 GiB using gp3 storage. A note states that free-tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. The 'Advanced' tab is visible. The summary panel on the right remains the same, showing 1 instance launching with the specified AMI and storage.

3. Instance successfully launched

The screenshot shows the AWS EC2 Launch Instances page. A green success banner at the top states "Successfully initiated launch of instance (i-0ebed2012fa0530b2)". Below the banner, there is a "Launch log" link. The main area is titled "Next Steps" with a search bar. It lists several actions: "Create billing and free tier usage alerts", "Connect to your instance", "Connect an RDS database", and "Create EBS snapshot policy". Each action has a corresponding button or link. At the bottom, there are links for "CloudShell" and "Feedback".

The screenshot shows the AWS EC2 Instances (1) Info page. The left sidebar is expanded to show the "Instances" section, which includes "Instances Types", "Launch Templates", "Spot Requests", "Savings Plans", "Reserved Instances", "Dedicated Hosts", "Capacity", and "Reservations New". The main content area displays a table for the single instance "MyWeb". The table columns include Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4. The instance is listed with the details: Name: MyWeb, Instance ID: i-0ebed2012fa0530b2, Instance state: Running, Instance type: t2.micro, Status check: Initializing, Alarm status: View alarms +, Availability Zone: us-east-1e, and Public IPv4: ec2-100-2!. Below the table, a modal window titled "Select an instance" is open.

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.
```

aws | Services | Search

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
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-0868fcf7b775cb51 (MyProject)

aws | Services | Search [Alt+S] N. Virginia v ovlabs/user3402813=PATIL_SHRAVANI_ANIL @ 5331-6246-4157

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
httpd	x86_64	2.4.62-1.amzn2023	amazonlinux	48 k
Installing:				
httpd	x86_64	2.4.62-1.amzn2023	amazonlinux	129 k
Installing dependencies:				
apr	x86_64	1.7.2-2.amzn2023.0.2	amazonlinux	98 k
apr-util	x86_64	1.6.3-1.amzn2023.0.1	amazonlinux	19 k
generic-logos-httpd	noarch	18.0.0-12.amzn2023.0.3	amazonlinux	1.4 M
httpd-core	x86_64	2.4.62-1.amzn2023	amazonlinux	14 k
httpd-filesystem	noarch	2.4.62-1.amzn2023	amazonlinux	81 k
httpd-tools	x86_64	2.4.62-1.amzn2023	amazonlinux	315 k
libbrotli	x86_64	1.0.9-4.amzn2023.0.2	amazonlinux	33 k
mailcap	noarch	2.1.49-3.amzn2023.0.3	amazonlinux	
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

```

AWS | Services | Search [Alt+S] | N. Virginia | vclabs/user3402813=PATIL_SHRAVANI_ANIL @ 5331-62

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-httd-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_http-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 : 1/12
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_http-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-httd-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-httd-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 : mod_http-2.0.27-1.amzn2023.noarch 7/12
Verifying : mod_lua-2.4.62-1.amzn2023.x86_64 8/12
Verifying : libbrotli-1.0.9-4.amzn2023.0.2.x86_64 9/12

```

```

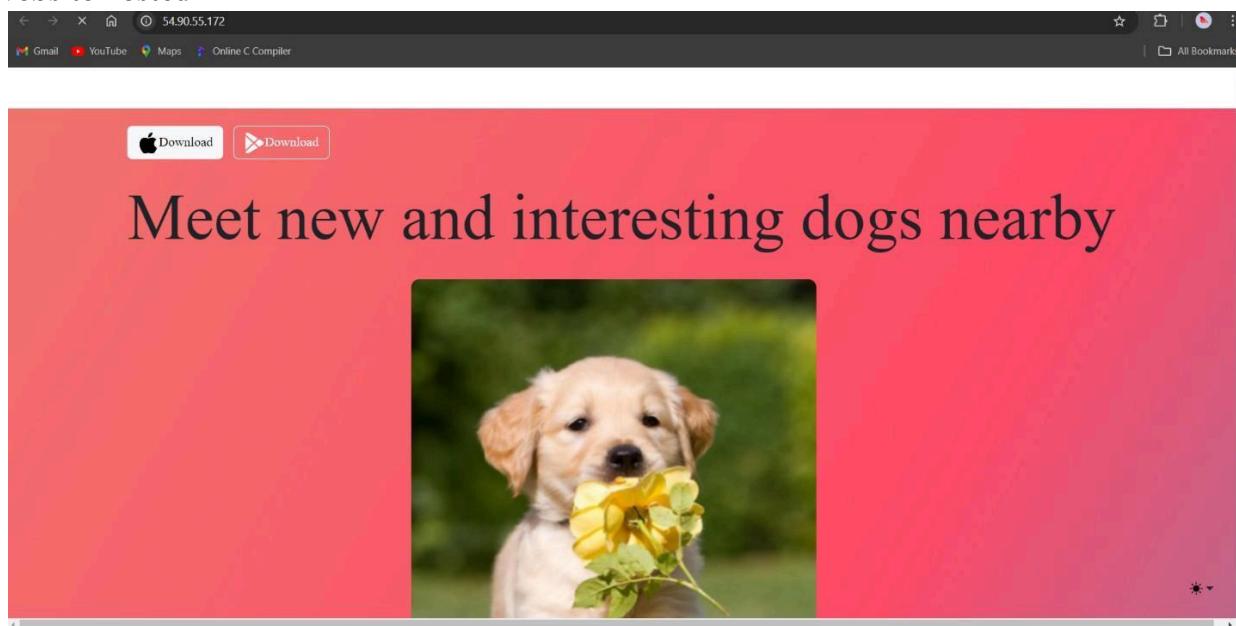
Verifying : mod_lua-2.4.62-1.amzn2023.x86_64 12

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

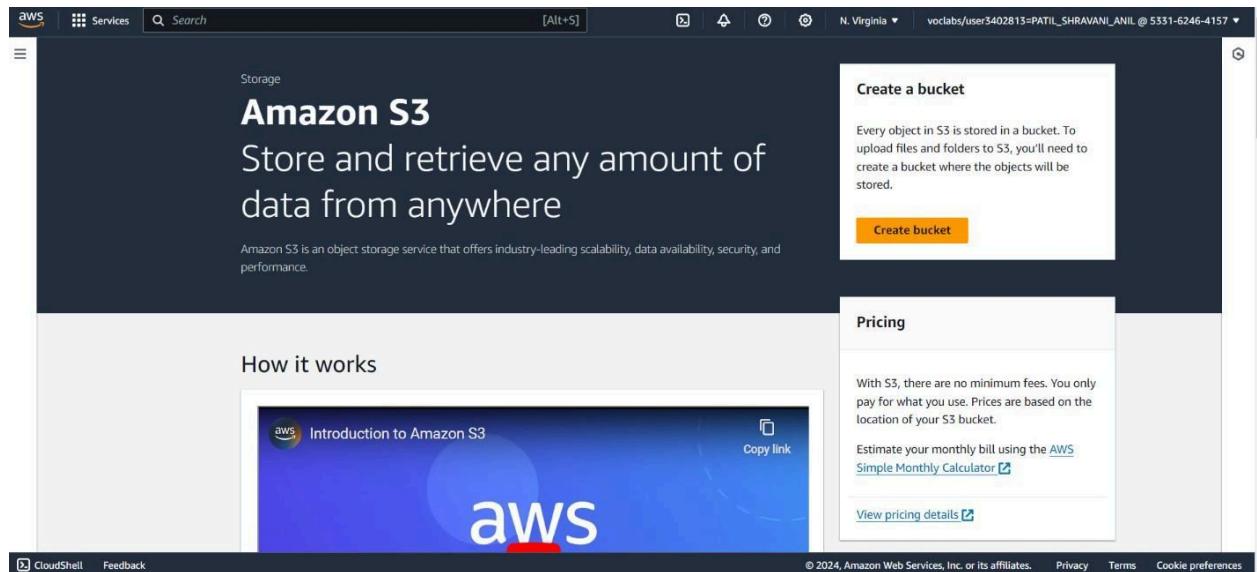
Complete!
[root@ip-172-31-34-11 ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:httpd.service(8)
[root@ip-172-31-34-11 ~]# mkdir aws assql
[root@ip-172-31-34-11 ~]# cd aws assql
[root@ip-172-31-34-11 aws_assql]# ls -lrt
total 0
[root@ip-172-31-34-11 aws_assql]# 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

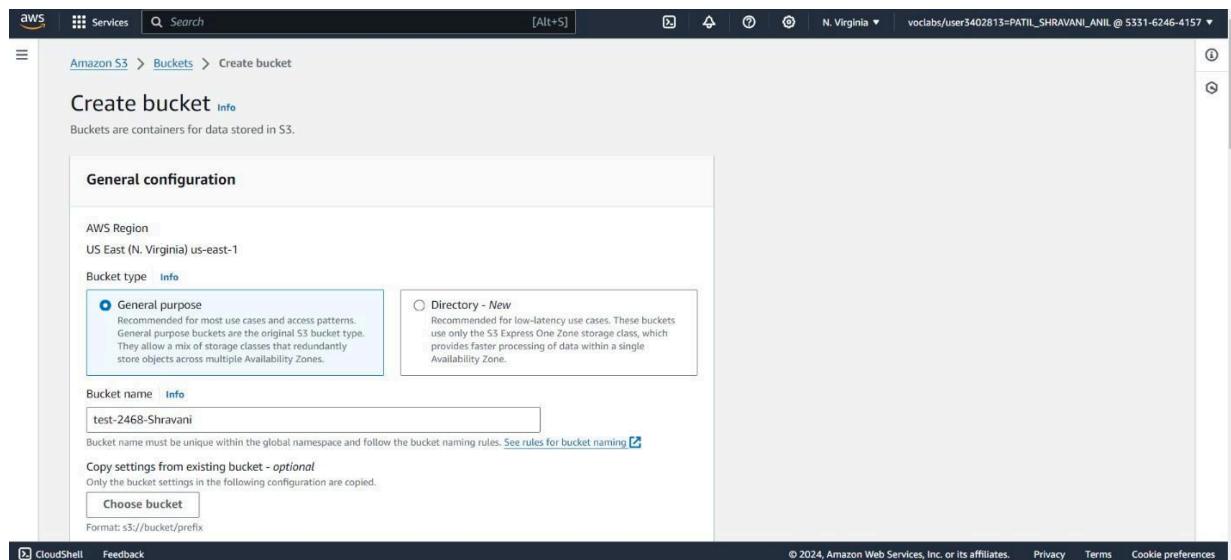


2. Hosting using S3 bucket



The screenshot shows the Amazon S3 homepage. On the right side, there is a prominent 'Create a bucket' button. Below it, a text box explains that every object in S3 is stored in a bucket and provides instructions for uploading files and folders. A smaller section titled 'Pricing' indicates that there are no minimum fees.

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



The screenshot shows the 'Create bucket' configuration page. Under 'General configuration', the 'Bucket name' field is set to 'test-2468-Shravani'. The 'Bucket type' dropdown is set to 'General purpose'. Other options like 'Directory - New' are also shown. The page includes sections for 'Copy settings from existing bucket - optional' and 'Choose bucket'.

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

The screenshot shows the 'Default encryption' section of the AWS S3 Bucket creation wizard. It includes options for 'Encryption type' (Server-side encryption with Amazon S3 managed keys (SSE-S3) is selected), 'Bucket Key' (Enable is selected), and a note about using SSE-KMS for cost reduction. A 'Create bucket' button is at the bottom right.

3. bucket successfully created

The screenshot shows the 'General purpose buckets' list in the AWS S3 console. It displays one bucket named 'test-2468-shravani' located in 'US East (N. Virginia)' (us-east-1). The bucket was created on August 22, 2024, at 20:13:56 (UTC+05:30). A 'Create bucket' button is visible at the top right of the table.

4.upload a file in the bucket

The screenshot shows the AWS S3 console interface. At the top, a green banner indicates "Upload succeeded" with the message "View details below." Below this, a note states "The information below will no longer be available after you navigate away from this page." The main area is titled "Summary" and shows the destination "s3://test-2468-shravani" with one succeeded file (1 file, 11.0 B) and zero failed files. A "Files and folders" tab is selected, displaying a table with one item: "text.txt" (text/plain, 11.0 B, Succeeded). The AWS navigation bar at the bottom includes CloudShell, Feedback, and links to Gmail, YouTube, Maps, Online C Compiler, and All Bookmarks.

Summary

Destination	Succeeded	Failed
s3://test-2468-shravani	1 file, 11.0 B (100.00%)	0 files, 0 B (0%)

Files and folders (1 Total, 11.0 B)

Name	Folder	Type	Size	Status	Error
text.txt	-	text/plain	11.0 B	Succeeded	-

Amazon S3

Buckets

- Access Grants
- Access Points
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

- Dashboards
- Storage Lens groups
- AWS Organizations settings

Feature spotlight ?

text.txt Info

Properties **Permissions** **Versions**

Object overview

Owner	s3://test-2468-shravani/text.txt
AWS Region	arn:aws:s3:::test-2468-shravani/text.txt
Last modified	Entity tag (Etag)
August 22, 2024, 20:17:11 (UTC+05:30)	b10a8db164e0754105b7a99be72e3fe5
Size	Object URL
11.0 B	https://test-2468-shravani.s3.amazonaws.com/text.txt
Type	
txt	
Key	

aws Services Search [Alt+S] N. Virginia v vocabs/user3402813=PATIL_SHRAVANI_ANIL @ 5331-6246-4157

Upload succeeded
View details below.

Upload: status

Close

The information below will no longer be available after you navigate away from this page.

Summary

Destination	Succeeded	Failed
s3://test-2468-shravani	1 file, 11.0 B (100.00%)	0 files, 0 B (0%)

Files and folders Configuration

Files and folders (1 Total, 11.0 B)

Find by name < 1 >

Name	Folder	Type	Size	Status	Error

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Website hosted - on the object url

Hello World

3. Hosting using Cloud 9 create environment

The screenshot shows the 'Create environment' page in the AWS Management Console. The 'Details' section is active, displaying fields for 'Name' (ShravaniEnv) and 'Description - optional'. Under 'Environment type', the 'New EC2 instance' option is selected, with a note explaining it creates a new EC2 instance in the user's account. There is also an 'Existing compute' option for using an existing instance or server.

The screenshot shows the 'New EC2 instance' configuration page. In the 'Instance type' section, the 't2.micro (1 GiB RAM + 1 vCPU)' option is selected, described as free-tier eligible and ideal for educational users. Other options shown include 't3.small (2 GiB RAM + 2 vCPU)' and 'm5.large (8 GiB RAM + 2 vCPU)'. In the 'Platform' section, 'Amazon Linux 2023' is chosen. Under 'Network settings', a timeout of '30 minutes' is set. The bottom of the page includes standard AWS navigation links like CloudShell and Feedback.

The screenshot shows the AWS Cloud9 interface. A modal window at the top indicates 'Creating ShravaniEnv. This can take several minutes. While you wait, see Best practices for using AWS Cloud9'. Below it, another message says 'For capabilities similar to AWS Cloud9, explore AWS Toolkits in your own IDE and AWS CloudShell in the AWS Management Console. Learn more'. The main 'Environments' page lists one environment: 'ShravaniEnv' (Status: Open, Type: EC2 instance, Connection: Secure Shell (SSH), Owner: Owner, ARN: arn:aws:sts::533162464157:assumed-role/voclabs/user3402813=PATIL_SHRAVANI_ANIL). The sidebar on the left includes links for 'My environments', 'Shared with me', 'All account environments', and 'Documentation'.

This screenshot is identical to the one above, showing the successful creation of the 'ShravaniEnv' environment. The modal message now says 'Successfully created ShravaniEnv. To get the most out of your environment, see Best practices for using AWS Cloud9'. The rest of the interface and environment details remain the same.

Website hosted

The screenshot shows the AWS Cloud9 IDE interface. On the left, the file structure shows a folder named 'ShravaniEnv - /nc' containing 'cloud9.html' and 'README.md'. The main editor window displays the content of 'cloud9.html':

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Shravani</title>
</head>
<body>
<h1>Hello , I am Shravani Anil Patil</h1>
<h2>I am currently pursuing my Bachelors in Engineering from Vivekanand Education ,Chembur</h2>
</body>
</html>
```

To the right, the browser preview window shows the rendered HTML output:

Hello , I am Shravani Anil Patil

I am currently pursuing my Bachelors in Engineering from Vivekanand Education Society's Institute of Technology ,Chembur

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.

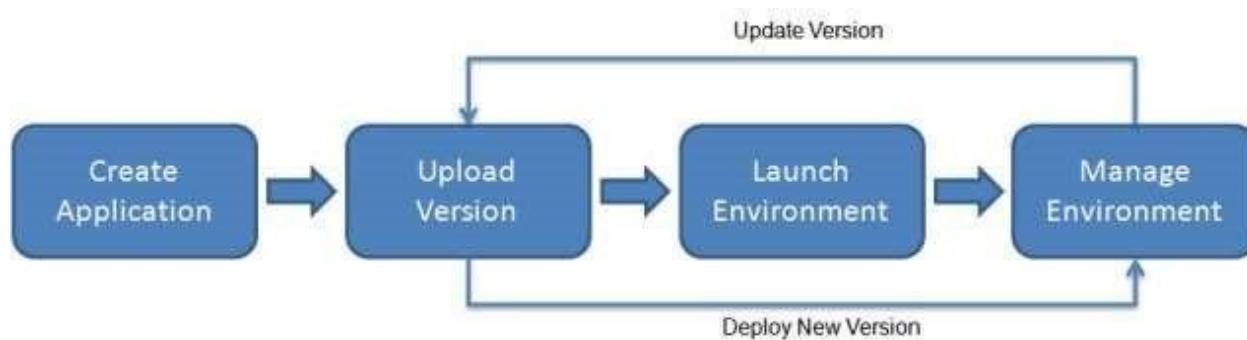
After your environment is set up, you can continue to manage and monitor it through the web interface or the CLI tools. You can scale your application by adjusting the number of EC2 instances or deploying new versions of your app with ease. The end result is a fully automated workflow that takes care 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

The screenshot shows the Amazon Elastic Beanstalk landing page. At the top, there's a navigation bar with the AWS logo, a 'Services' dropdown, a search bar, and account information for 'ShravaniAnilPatil'. Below the header, the page title 'Amazon Elastic Beanstalk' and subtitle 'End-to-end web application management.' are displayed. A brief description explains that Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications. To the right, there are two call-to-action boxes: 'Get started' (with a 'Create application' button) and 'Pricing' (describing no additional charge). On the left, a 'Get started' section provides instructions for uploading code.

The screenshot shows the 'Configure environment' step of the Elastic Beanstalk wizard. On the left, a sidebar lists steps: 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), and Step 6 (Review). The main content area is titled 'Configure environment' and contains three sections: 'Environment tier' (selected as 'Web server environment'), 'Application information' (with an application name of 'MyWebApp_Shrawan'), and 'Environment information' (with a note about不可更改). The bottom of the page includes standard AWS footer links like CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

The screenshot shows the AWS Elastic Beanstalk Platform configuration interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, a search bar, and user information for 'ShravaniAnilPatil'. Below the navigation is a sidebar with three horizontal bars. The main content area has two sections: 'Platform' and 'Application code'.
Platform Section:

- Platform type:** A radio button is selected for 'Managed platform', which is described as 'Platforms published and maintained by Amazon Elastic Beanstalk'. A link 'Learn more' is provided.
- Platform:** A dropdown menu is set to 'Python'.
- Platform branch:** A dropdown menu is set to 'Python 3.11 running on 64bit Amazon Linux 2023'.
- Platform version:** A dropdown menu is set to '4.1.3 (Recommended)'.

Application code Section:

- Application code type:** A radio button is selected for 'Sample application'.
- Existing version:** A link 'Application versions that you have uploaded' is shown.

At the bottom of the page, there are links for 'CloudShell', 'Feedback', and 'Cookie preferences', along with copyright information: '© 2024, Amazon Web Services, Inc. or its affiliates.' and links for 'Privacy', 'Terms', and '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**

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](#)

The screenshot shows the 'Set up networking, database, and tags - optional' step of the AWS Elastic Beanstalk environment creation wizard. The left sidebar lists steps 1 through 6. Step 3 is currently selected. The main content area is divided into sections: 'Virtual Private Cloud (VPC)', 'Instance settings', and 'Instance subnets'. In the VPC section, a dropdown menu shows 'vpc-0bfd56ae28107e37 | (172.31.0.0/16)'. Below it is a 'Create custom VPC' button. The Instance settings section contains a note about assigning public IP addresses to instances in private subnets. The Instance subnets section shows three availability zones: eu-north-1c, eu-north-1b, and eu-north-1a, each with its corresponding subnet and CIDR range.

Availability Zone	Subnet	CIDR
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

This screenshot shows the 'Database subnets' section of the same wizard step. It includes a note about integrating an RDS SQL database and a 'Choose database subnets' table. The table lists the same three availability zones as the instance subnets table above, with checkboxes next to each row.

Availability Zone	Subnet	CIDR
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

The screenshot displays two parallel sessions on an AWS interface.

Elastic Beanstalk Environment Overview:

- Left Sidebar:** Applications, Environments, Change history; Application: MyWebApp_Shrawani, Application versions, Saved configurations; Environment: MyWebAppShravani-env, Go to environment, Configuration, Events, Health, Logs, Monitoring, Alarms.
- Central Content:** Elastic Beanstalk is launching your environment. This will take a few minutes. Environment ID: e-pm7kwpm2k4, Application name: MyWebApp_Shrawani. Platform: Python 3.11 running on 64bit Amazon Linux 2023/4.1.3, Running version: -, Platform state: Supported.
- Bottom Navigation:** Events (3), Health, Logs, Monitoring, Alarms, Managed updates, Tags.

IAM Role Creation:

- Left Sidebar:** Step 1: Select trusted entity, Step 2: Add permissions, Step 3: Name, review, and create.
- Central Content:** Add permissions. Permissions policies (946). Choose one or more policies to attach to your new role. Filter by Type: All types. Policies listed:
 - AdministratorAccess
 - AdministratorAccess
 - AdministratorAccess
 - AlexaForBusiness...
 - AlexaForBusinessF...
 - AlexaForBusinessG...
 - AlexaForBusinessL...
 - AlexaForBusinessP...
- Bottom Navigation:** CloudShell, Feedback.

The screenshot shows two stacked screenshots of the AWS Elastic Beanstalk console.

Top Screenshot: Step 2: Configure service access. It shows a table with one row: Service role (arn:aws:iam::605134455955:role/service-role/aws-elasticbeanstalk-service-role) and EC2 instance profile (elastic-beanstalk-ec2-role). A note says: "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."

Bottom Screenshot: Step 3: Set up networking, database, and tags. It shows a table with one row: Networking, database, and tags (No options configured). A note says: "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."

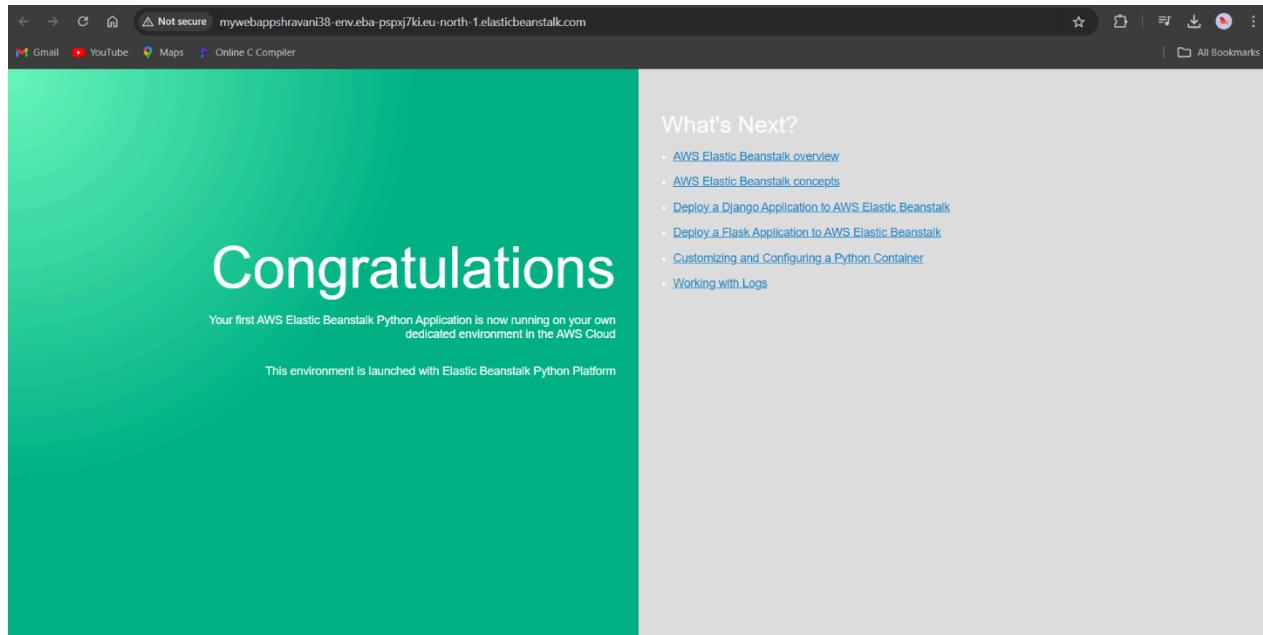
Environment Overview:

- Events:** Shows 12 events.
- Health:** Status is Ok - View causes.
- Logs:** Shows a search bar: "Filter events by text, property or value".
- Monitoring:** Not visible in the screenshot.
- Alarms:** Not visible in the screenshot.
- Managed updates:** Not visible in the screenshot.
- Tags:** A table with columns Key and Value.

Platform:

- Platform:** Python 3.11 running on 64bit Amazon Linux 2023/4.1.3
- Running version:** -
- Platform state:** Supported

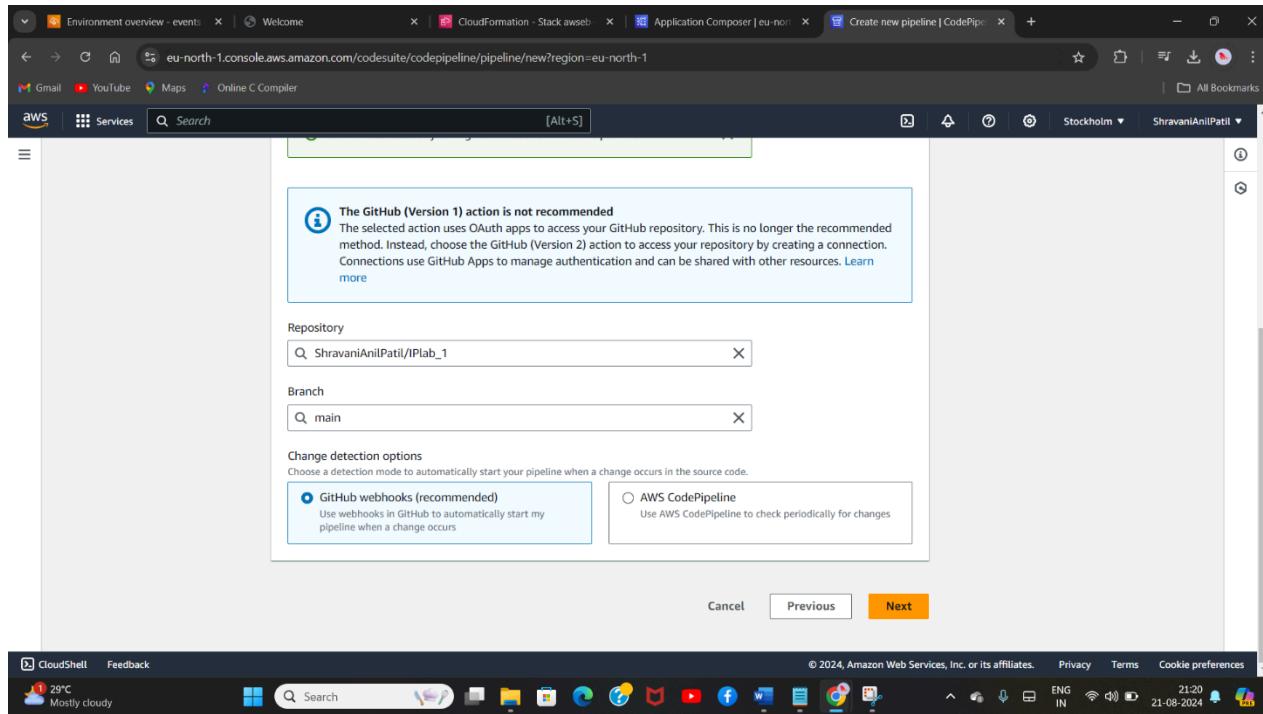
Actions: Includes a "Upload and deploy" button.



Timestamp	Logical ID	Status	Detailed status
2024-08-21 21:03:54 UTC+0530	awseb-e-qmqfnyyrcb-stack	CREATE_COMPLETE	-
2024-08-21 21:03:53 UTC+0530	AWSEBInstanceLaunchWailCondition	CREATE_COMPLETE	-
2024-08-21 21:03:58 UTC+0530	AWSEBInstanceLaunchWailCondition	CREATE_IN_PROGRESS	-
2024-08-21 21:03:58 UTC+0530	AWSEBInstanceLaunchWailCondition	CREATE_IN_PROGRESS	-
2024-08-21 21:03:58 UTC+0530	AWSEBAutoScalingGroup	CREATE_COMPLETE	-

The screenshot shows the AWS Application Composer interface in CloudFormation console mode. The template file is named `awseb-e-qmqfnyrch-stack.yaml`. The canvas displays several standard components: `AWSEBAutoScalingLaunchConfiguration`, `AWSEBInstanceLaunchHandle`, `AWSEBEIP`, `AWSEBBBeanstalkMetadata`, and `AWSEBInstanceLaunchWaitCondition`. A dashed line connects the `AWSEBAutoScalingLaunchConfiguration` component to the `AWSEBInstanceLaunchHandle` component. The left sidebar lists various enhanced components like API Gateway, Cognito UserPool, and DynamoDB Table.

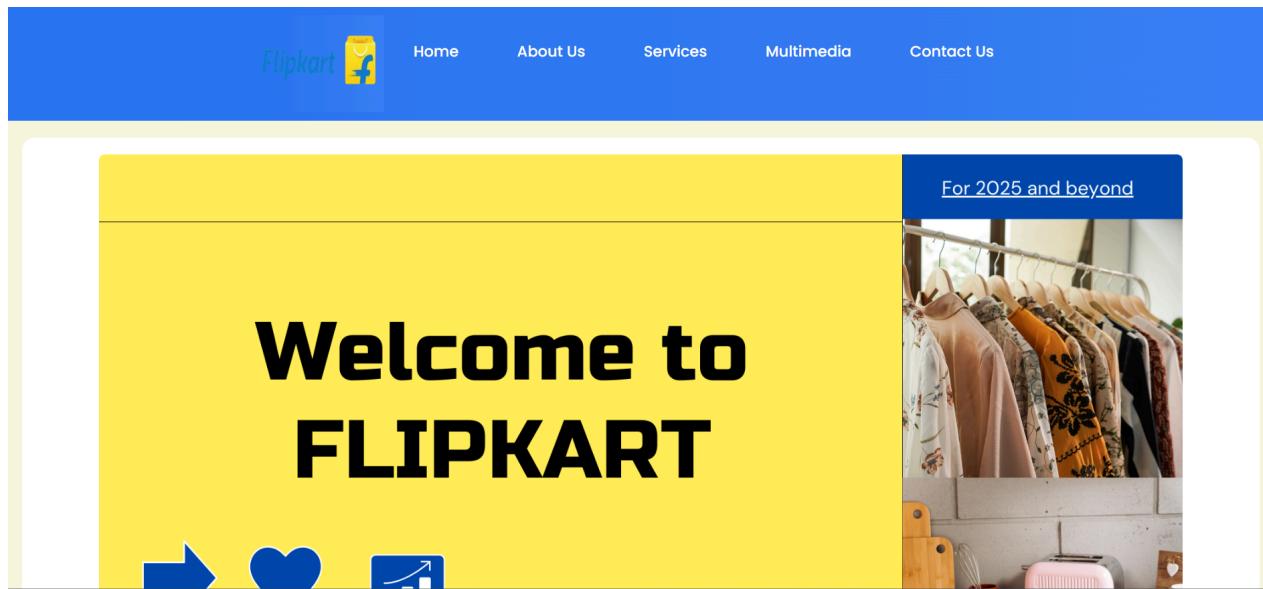
The screenshot shows the AWS CodePipeline 'Create new pipeline' wizard at Step 2 of 5, titled 'Add source stage'. It is configured to use GitHub (Version 1) as the source provider, which is connected. A note indicates that the GitHub (Version 1) action is no longer recommended. The pipeline has five steps: Step 1 (Choose pipeline settings), Step 2 (Add source stage), Step 3 (Add build stage), Step 4 (Add deploy stage), and Step 5 (Review).



The screenshot displays two browser windows related to AWS CodePipeline.

Top Window: Shows the "Create new pipeline" process at Step 1: Choose pipeline settings. The pipeline name is "Shravani_Pipelineaws", type is "V2", mode is "QUEUED", artifact location is "A new Amazon S3 bucket will be created as the default artifact store for your pipeline", and service role name is "AWSCodePipelineServiceRole-eu-north-1-Shravani_Pipelineaws".

Bottom Window: Shows the "Shravani_Pipelineaws" pipeline details. The "Source" stage has a GitHub step (version 1) that succeeded 1 minute ago. The "Deploy" stage uses AWS Elastic Beanstalk and has also succeeded just now. Pipeline execution ID is 8b6306d0-ade8-4407-b58c-86503eb3851e.



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.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
worker-node1	i-0526f6be95551ba74	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	ec2-3-81-7
master1	i-0aef82b0cccd222219	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	ec2-34-207

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

Security group name	Security group ID	Description	VPC ID
launch-wizard-1	sg-020c81f3d7e528326	launch-wizard-1 created 2024-09-18T04:49:14.326Z	vpc-085eda6d3476ae0d2

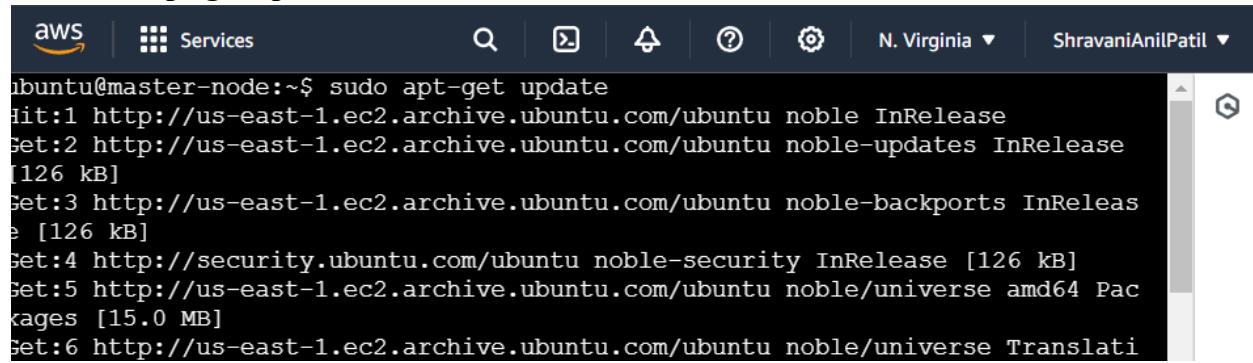
Owner	Inbound rules count	Outbound rules count
605134455955	1 Permission entry	1 Permission entry

Inbound rules (1)	
Name	Type
-	All traffic

3. Set master and worker as hostname on respective servers

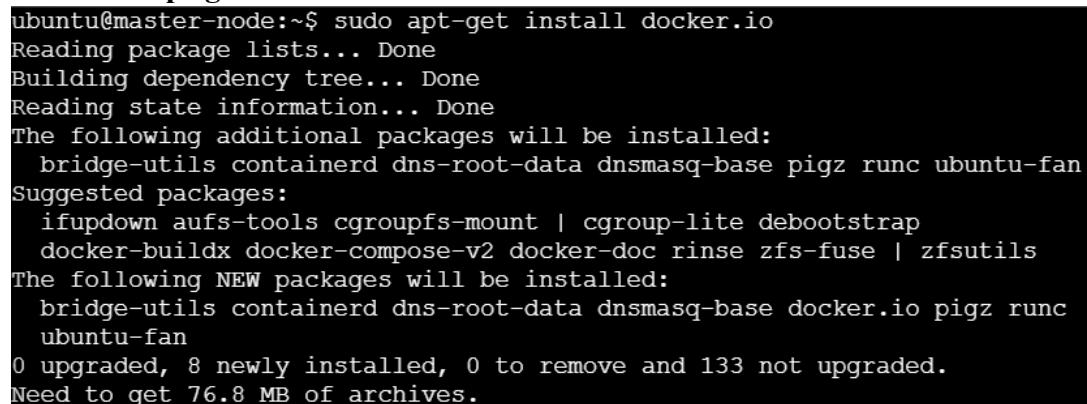
4. Installation of docker

4.1 - sudo apt-get update



```
ubuntu@master-node:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translati
```

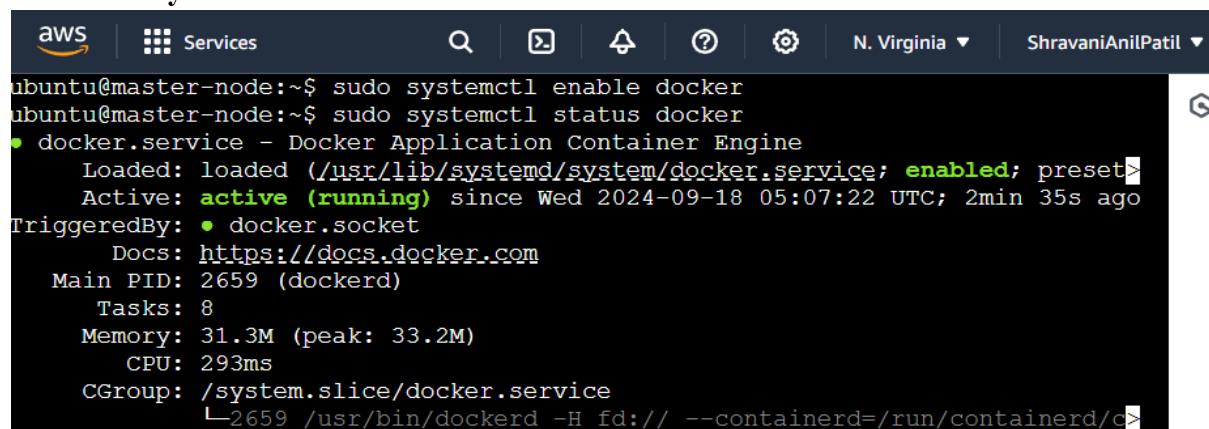
4.2 - sudo apt-get install docker.io



```
ubuntu@master-node:~$ sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap
  docker-buildx docker-compose-v2 docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc
  ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 133 not upgraded.
Need to get 76.8 MB of archives.
```

4.3 – sudo systemctl enable docker

4.4 – sudo systemctl status docker



```
ubuntu@master-node:~$ sudo systemctl enable docker
ubuntu@master-node:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset>
   Active: active (running) since Wed 2024-09-18 05:07:22 UTC; 2min 35s ago
     TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
      Main PID: 2659 (dockerd)
        Tasks: 8
       Memory: 31.3M (peak: 33.2M)
         CPU: 293ms
      CGroup: /system.slice/docker.service
              └─2659 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/c
```

5- Installation of Kubernetes-

5.1 sudo apt-get update

5.2 install ca certificate

```
aws Services 🔎 🚨 ⓘ N. Virginia ⓘ ShravaniAnil  
ubuntu@master-node:~$ sudo apt-get update  
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease  
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease  
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease  
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease  
Reading package lists... Done  
ubuntu@master-node:~$ sudo apt-get install -y apt-transport-https ca-certificates curl  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
ca-certificates is already the newest version (20240203).  
ca-certificates set to manually installed.  
The following additional packages will be installed:  
libcurl3 libcurl3-nss
```

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

A screenshot of a terminal window titled "Ubuntu" with the following content:

```
ubuntu@master-node:~$ sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg
```

The terminal shows the command being run, which adds the GPG key for the Kubernetes archive repository.

5.4 Add the appropriate Kubernetes apt repository

```
aws Services Search [Alt+S] N. Virginia ShravaniAnilPatil

ubuntu@worker1:~$ echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb /" | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb /
ubuntu@worker1:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/istv/kubernetes:/core:/stable:/v1.31/deb InRelease [1186 B]
Err:5 https://prod-cdn.packages.k8s.io/repositories/istv/kubernetes:/core:/stable:/v1.31/deb InRelease
      The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 234654DA9A296436
Reading package lists... Done
W: GPG error: https://prod-cdn.packages.k8s.io/repositories/istv/kubernetes:/core:/stable:/v1.31/deb InRelease: The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 234654DA9A296436
E: The repository 'https://pkgs.k8s.io/core:/stable:/v1.31/deb InRelease' is not signed.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
```

5.5 sudo apt-get update

5.6 sudo apt-get install -y kubelet kubeadm kubectl

```

aws | Services | Search [Alt+S] | N. Virginia | ShravaniAnilPatil
0% [Connecting to pkgs.k8s.io (34.107.204.206)] [Waiting for headers] [Connec
tp://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease Hit:2 ht
0% [Connected to pkgs.k8s.io (34.107.204.206)] [Connecting to security.ubuntu
tp://security.ubuntu.com/ubuntu noble-security InRelease Hit:5 ht
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb InRelease [11186 B]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb Packages [4865 B]
Fetched 6051 B in 1s (7749 B/s)
Reading package lists... Done
ubuntu@master-node:~$ sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 130 not upgraded.
Need to get 97.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
0% [conntrack 37.3 kB/37.9 kB 98%] [Connected to pkgs.k8s.io (34.107.204.20
Get:2 ht
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb cri-tools 1.31.1-1.1 [15.7 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb kubeadm 1.31.1-1.1 [11.4 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb kubernetes-cni 1.5.1-1.1 [33.9 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb kubelet 1.31.1-1.1 [15.2 MB]

```

5.7 sudo apt-mark hold kubelet kubeadm kubectl

```

aws | Services | Search | N. Virginia | ShravaniAnilPatil
ubuntu@master-node:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@master-node:~$ 

```

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

```

aws | Services | Search | N. Virginia | ShravaniAnilPatil
table kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

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

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each
as root:

kubeadm join 172.31.33.243:6443 --token ltp4ao.gzdzdpkqly6bxvnz \
  --discovery-token-ca-cert-hash sha256:4423cf44f5102d477fa92160e76e03f
78f941d4e7a5836cd46bb14517dfab5ad
ubuntu@master-node:~$ 

```

i-Oaef82b0cccd222219 (master1)

PublicIPs: 34.207.105.187 PrivateIPs: 172.31.33.243

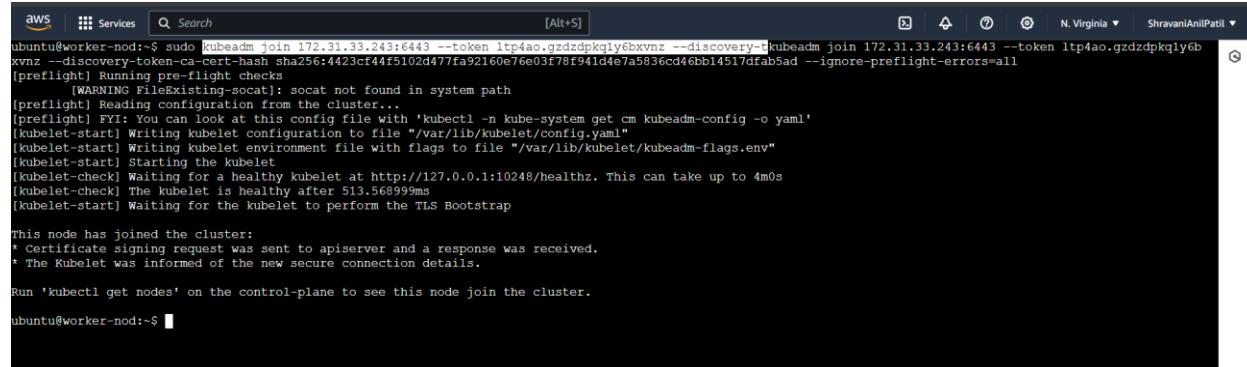
7.to create a directory for the cluster:

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



```
ubuntu@worker-node:~$ sudo kubeadm join 172.31.33.243:6443 --token ltp4ao.gzdzdpkqly6bxvnz --discovery-token-ca-cert-hash sha256:4423cf44f5102d47fa921e0e76e03f78f941d4e7a5836cd46bb14517dfab5ad --ignore-preflight-errors=all
[preflight] Running pre-flight checks
[WARNING Fileexisting-socat]: socat not found in system path
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 513.56899ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

ubuntu@worker-node:~$
```

Verify that everything is running and communicating:

8.1kubectl get pods --all-namespaces

8.2kubectl get nodes



NAME	READY	STATUS	RESTARTS	AGE
kube-flannel kube-flannel-ds-gx9xg	1/1	Running	0	14m
kube-flannel kube-flannel-ds-t179d	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

NAME	STATUS	ROLES	AGE	VERSION
master-node	Ready	control-plane	23m	v1.31.1
worker-node	Ready	<none>	6m22s	v1.31.1

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.

The screenshot shows the AWS EC2 Instances page with two instances listed:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
worker-node1	i-0526f6be95551ba74	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	ec2-3-81-7
master1	i-0aef82b0cc222219	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	ec2-34-207

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

The screenshot shows the AWS Security Groups page for the security group 'sg-020c81f3d7e528326 - launch-wizard-1'. The 'Inbound rules (1)' section displays a single rule:

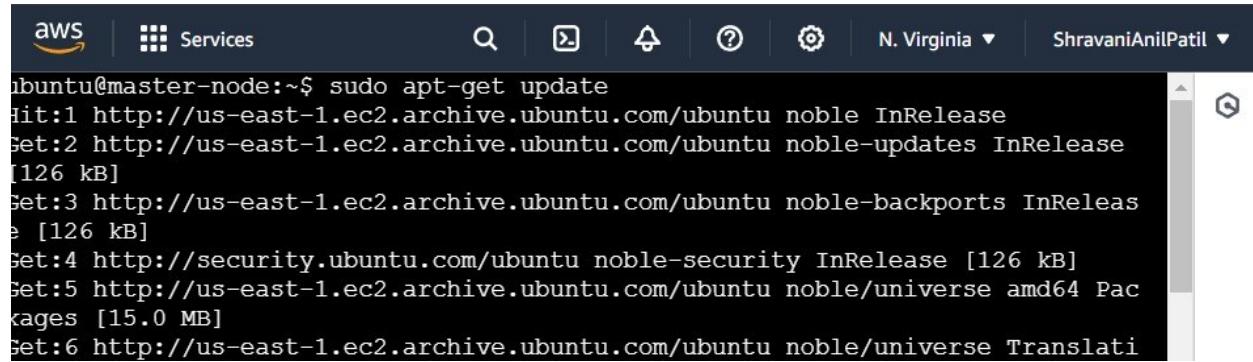
Name	Security group rule...	IP version	Type	Protocol	Port range
-	sgr-0eaddedf6a07229a4	IPv4	All traffic	All	All

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

The screenshot shows a terminal session on the master1 EC2 instance. The command 'hostnamectl set-hostname master1' is being run, changing the host name from 'ubuntu' to 'master1'.

Step 4. Installation of docker

4.1 - sudo apt-get update



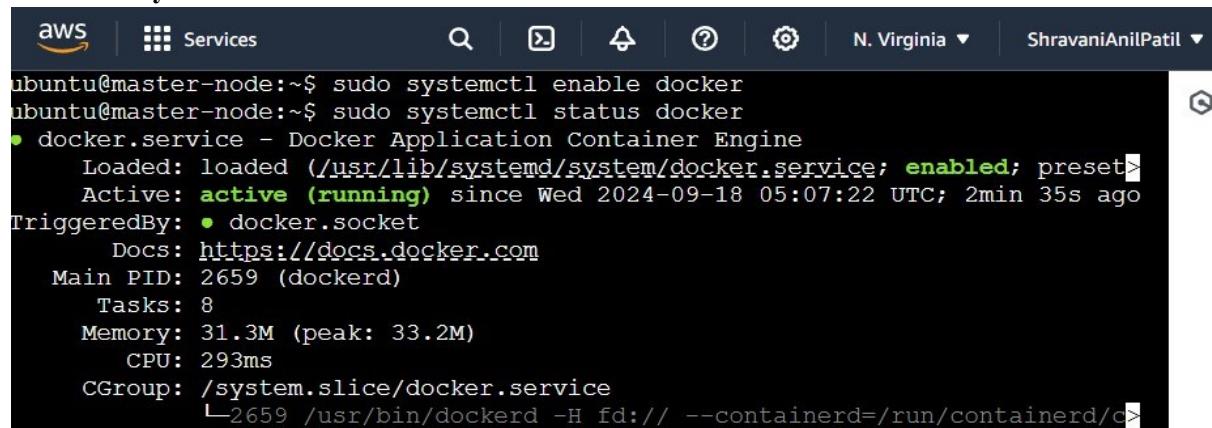
```
ubuntu@master-node:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translati
```

4.2 - sudo apt-get install docker.io

```
ubuntu@master-node:~$ sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap
  docker-buildx docker-compose-v2 docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc
  ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 133 not upgraded.
Need to get 76.8 MB of archives.
```

4.3 – sudo systemctl enable docker

4.4 – sudo systemctl status docker



```
ubuntu@master-node:~$ sudo systemctl enable docker
ubuntu@master-node:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset>
   Active: active (running) since Wed 2024-09-18 05:07:22 UTC; 2min 35s ago
     TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
      Main PID: 2659 (dockerd)
        Tasks: 8
       Memory: 31.3M (peak: 33.2M)
         CPU: 293ms
        CGroup: /system.slice/docker.service
                  └─2659 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/c
```

Step 5. Installation of Kubernetes-

5.1 sudo apt-get update

5.2 install ca certificate

```
aws Services 🔎 🚨 🛡 N. Virginia ▾ ShravaniAnil

ubuntu@master-node:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
ubuntu@master-node:~$ sudo apt-get install -y apt-transport-https ca-certificates curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
The following additional packages will be installed:
  libcurl4 libcurl4-openssl-dev libcurl4-openssl-dev:i386 libcurl4-openssl-dev:amd64
```

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

A screenshot of a terminal window titled "Ubuntu" in the AWS Lambda interface. The terminal shows the following command being run:
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg
The terminal prompt is "ubuntu@master-node:~\$".

5.4 Add the appropriate Kubernetes apt repository

5.5 sudo apt-get update

5.6 sudo apt-get install -y kubelet kubeadm kubectl

```

aws | Services | Search [Alt+S] | N. Virginia | ShravaniAnilPatil
0% [Connecting to pkgs.k8s.io (34.107.204.206)] [Waiting for headers] [Connec
tp://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease Hit:2 ht
0% [Connected to pkgs.k8s.io (34.107.204.206)] [Connecting to security.ubuntu
tp://security.ubuntu.com/ubuntu noble-security InRelease Hit:5 ht
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb InRelease [11186 B]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb Packages [4865 B]
Fetched 6051 B in 1s (7749 B/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 130 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
0% [conntrack 37.3 kB/37.9 kB 98%] [Connected to pkgs.k8s.io (34.107.204.206)]
Get:2 ht
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb cri-tools 1.31.1-1.1 [15.7 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb kubeadm 1.31.1-1.1 [11.4 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb kubectl 1.31.1-1.1 [11.2 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb kubernetes-cni 1.5.1-1.1 [33.9 MB]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb kubelet 1.31.1-1.1 [15.2 MB]

```

5.7 sudo apt-mark hold kubelet kubeadm kubectl

```

aws | Services | Search | N. Virginia | ShravaniAnilPatil
ubuntu@master-node:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@master-node:~$ 

```

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

```

aws | Services | Search | N. Virginia | ShravaniAnilPatil
table kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

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

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each
as root:

kubeadm join 172.31.33.243:6443 --token ltp4ao.gzdzdpkqly6bxvnz \
  --discovery-token-ca-cert-hash sha256:4423cf44f5102d477fa92160e76e03f
78f941d4e7a5836cd46bb14517dfab5ad
ubuntu@master-node:~$ 

```

i-Oaef82b0cc222219 (master1)

PublicIPs: 34.207.105.187 PrivateIPs: 172.31.33.243

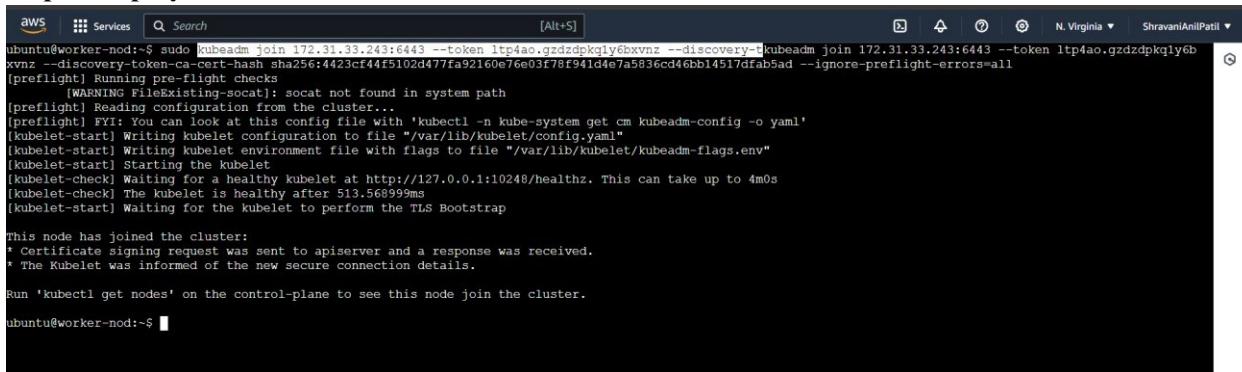
Step 7.to create a directory for the cluster:

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



```
aws Services Search [Alt+S]
ubuntu@worker-node:~$ sudo kubeadm join 172.31.33.243:6443 --token ltp4ao.gzdzdpkqly6xvnz --discovery-token-ca-cert-hash sha256:4423cf44f5102d477fa92160e76e03f78f941d4e7a5836cd46bb14517dfab5ad --ignore-preflight-errors=all
[preflight] Running pre-flight checks
[WARNING Fileexisting-socat]: socat not found in system path
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 513.56899ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

ubuntu@worker-node:~$
```

Verify that everything is running and communicating:

8.1kubectl get pods --all-namespaces

8.2kubectl get nodes



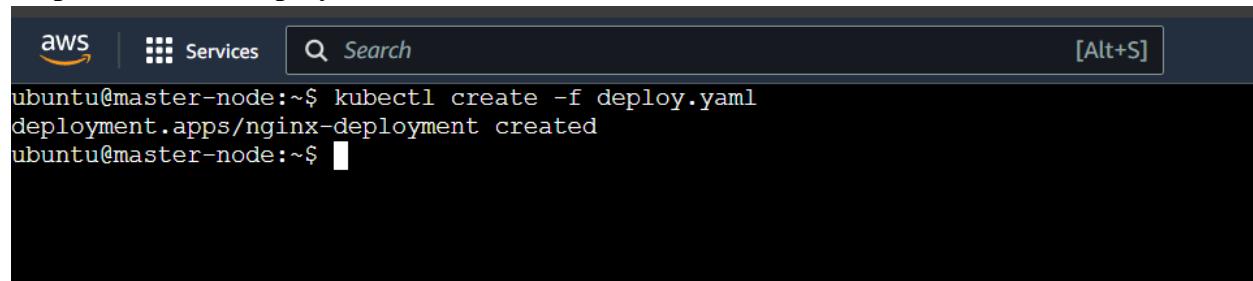
```
ubuntu@master-node:~$ kubectl get pods --all-namespaces
NAME                               READY   STATUS    RESTARTS   AGE
kube-flannel   kube-flannel-ds-gx9xg   1/1     Running   0          14m
kube-flannel   kube-flannel-ds-t179d   1/1     Running   0          6m2s
kube-system    coredns-7c65d6cf9-nrns4  1/1     Running   0          23m
kube-system    coredns-7c65d6cf9-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-node  Ready   <none>    6m22s   v1.31.1
ubuntu@master-node:~$
```

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



The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with the AWS logo, a 'Services' button, a search bar containing 'Search', and a keyboard shortcut '[Alt+S]'. Below the navigation bar, the main area displays a terminal window. The terminal shows the command 'kubectl create -f deploy.yaml' being run, followed by the output 'deployment.apps/nginx-deployment created'.

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

(EXTRA) – kubectl get namespaces

```
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: Verify the service

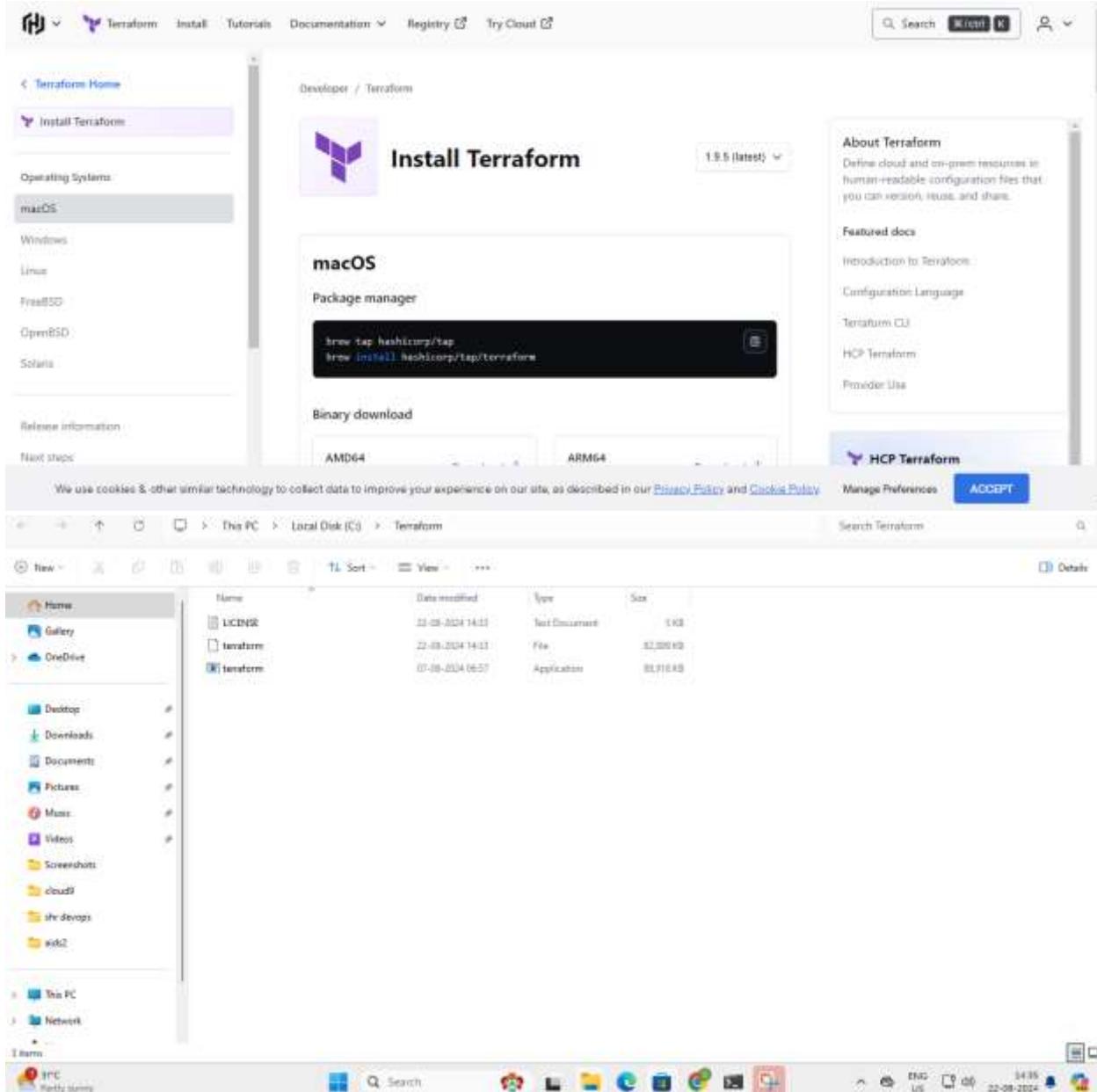
```
service/nginx-deployment exposed
ubuntu@master-node:~$ kubectl get svc
NAME            TYPE        CLUSTER-IP     EXTERNAL-IP    PORT(S)        AGE
kubernetes      ClusterIP   10.96.0.1    <none>        443/TCP       4h43m
nginx-deployment   LoadBalancer 10.101.59.94  <pending>    80:31041/TCP  4m34s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

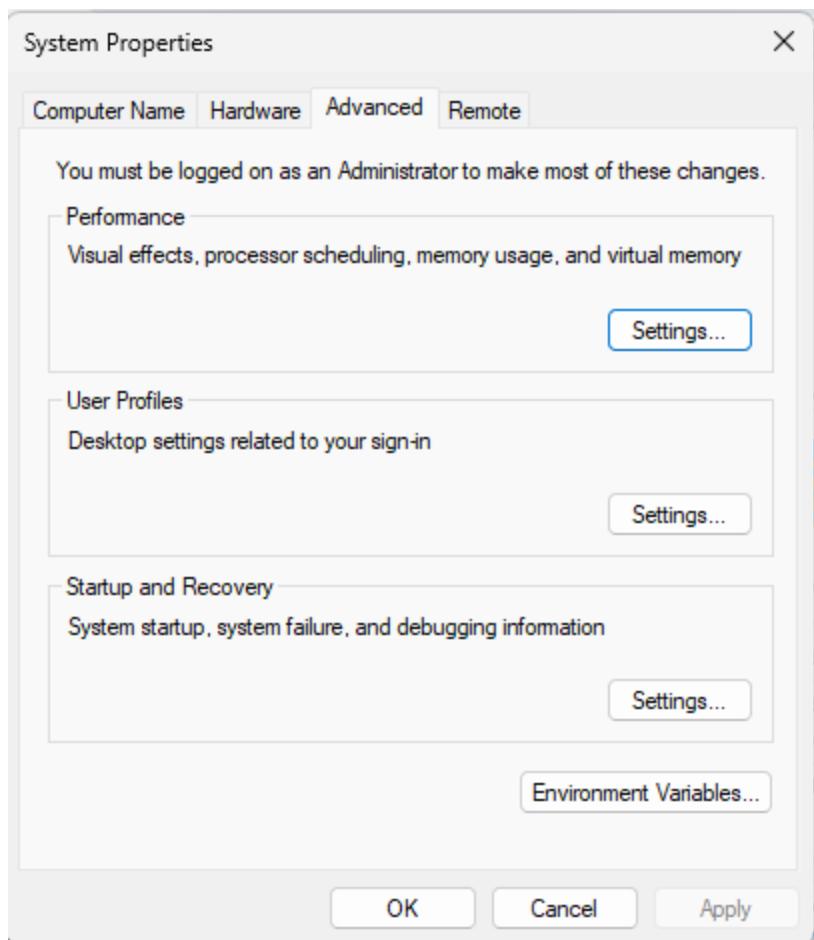
Step 14: Access the application



Experiment No 5 : Terraform

Aim : Installation and Configuration of Terraform in Window





User variables for Student	
System variables	
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
--------	---------	--------

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
```

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

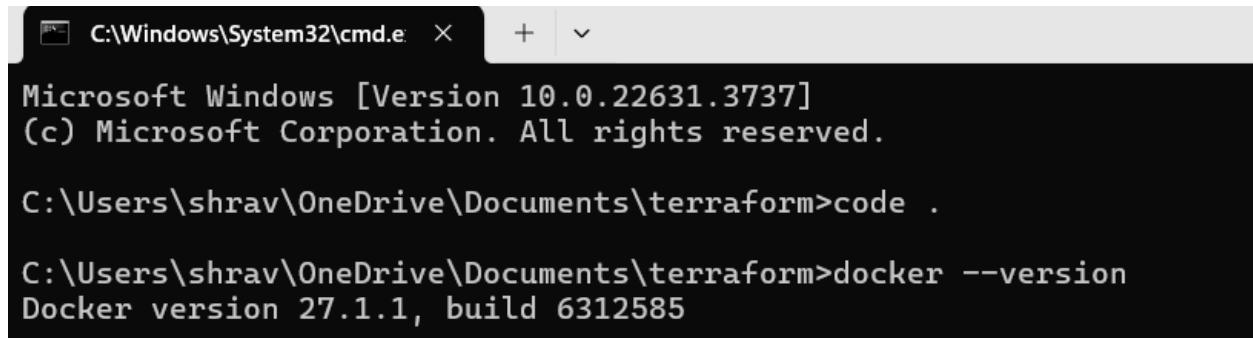
```
C:\Users\shrav\OneDrive\Documents\terraform>docker

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
```



```
C:\Windows\System32\cmd.exe + ^

Microsoft Windows [Version 10.0.22631.3737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\shraw\OneDrive\Documents\terraform>code .

C:\Users\shraw\OneDrive\Documents\terraform>docker --version
Docker version 27.1.1, build 6312585
```

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 }
}
```

The screenshot shows a code editor with several tabs at the top: 'credentials.txt', 'provider.tf', 'docker.tf' (which is the active tab), and 'main.tf'. The code in 'docker.tf' is a Terraform configuration:

```
1 terraform {  
2   required_providers {  
3     docker = {  
4       source = "kreuzwerker/docker"  
5       version = "~> 3.0.1"  
6     }  
7   }  
8 }  
9 provider "docker" {  
10   host    = "npipe:///./pipe/docker_engine"  
11 }  
12 resource "docker_image" "nginx" {  
13   name    = "nginx:latest"  
14   keep_locally = false  
15 }  
16 resource "docker_container" "nginx" {  
17   image = docker_image.nginx.image_id  
18   name = "tutorial"  
19   ports {  
20     internal = 80  
21     external = 8000  
22   }  
23 }  
24 }
```

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-OMELEBG1 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 - c

+ security_opts                                = (known after apply)
+ shm_size                                     = (known after apply)
+ start                                         = true
+ stdio_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=0b844cbfdc3ee0fc75bbf5a6577f7a7d824bd4867787cd570085b011ecdb82e]

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

shrav@LAPTOP-0MELEBG1 MINGW64 ~/OneDrive/Documents/terraform/docker
$ 

```

Step 7 : Docker container after step 6 execution BEFORE -

AFTER -

```

shrav@LAPTOP-0MELEBG1 MINGW64 ~/OneDrive/Documents/terraform/docker
● $ docker container list
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
0b844cbfdc3e 5ef79149e0ec "/docke-entrypoint..." 2 minutes ago Up 2 minutes 0.0.0.0:8000->80/tcp tutorial

shrav@LAPTOP-0MELEBG1 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
1  terraform {
2    required_providers {
3      docker = {
4        source  = "kreuzwerker/docker"
5        version = "~> 3.0.1"
6      }
7    }
8  }
9
10 provider "docker" {
11   host = "npipe:///./pipe/docker_engine"
12 }
13
14 resource "docker_image" "nginx" {
15   name        = "nginx:latest"
16   keep_locally = false
17 }
18
19 resource "docker_container" "nginx" {
20   image = docker_image.nginx.image_id
21   name  = "tutorial"
22   ports {
23     internal = 80
24     external = 8080
25   }
26 }
27
```

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=0b844cbfdc3ee0fc...]
docker_container.nginx: Destruction complete after 3s
docker_container.nginx: Creating...

```

Step 9 : terraform destroy to destroy infrastructure.

```

shraw@LAPTOP-0MELEBG1 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@sha256:447a8665cc1dab95b1ca778e162215839ccb9189104c79d7ec3a81e14577add" -> 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.

shraw@LAPTOP-0MELEBG1 MINGW64 ~/OneDrive/Documents/terraform/docker

```

Step 10 : Docker after destroy command.

```
shrav@LAPTOP-0MELEBGI MINGW64 ~/OneDrive/Documents/terraform/docker
● $ docker container list
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

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

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.

The screenshot shows the Jenkins dashboard with the following details:

- Dashboard Header:** Jenkins, Search (CTRL+K), Notifications, User: Shravani Anil Patil, Log out.
- Left Sidebar:** New Item, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, My Views.
- Build Queue:** No builds in the queue.
- Build Executor Status:** Built-In Node (1 Idle, 2 Idle, 3 Idle).
- Central Content:**
 - A message: "This is first automation testing in Jenkins Hello World by Shravani".
 - A note: "Jenkins is Great".
 - An "Add description" button.
 - A navigation bar with tabs: All (selected), Selenium, project1, +.
 - A table showing build statistics:

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀️	Job 2	1 mo 1 day #2	N/A	1.3 sec
✓	☀️	maventest	1 mo 0 days #3	N/A	23 sec
✓	☀️	maventest2	1 mo 0 days #1	N/A	20 sec
✓	☀️	MAVENTEST3	1 mo 0 days #1	N/A	20 sec
✓	☀️	maventestnew	1 mo 1 day #1	N/A	10 sec
✓	☀️	project1-build	1 mo 0 days #13	1 mo 1 day #2	1.4 sec

2. Run SonarQube in a Docker container using this command

Ensure Docker engine is running

The screenshot shows the Docker Desktop interface with the following details:

- Header:** docker desktop PERSONAL, Search for images, containers, volume... Ctrl+K.
- Left Sidebar:** Containers, Images (selected), Volumes, Builds, Docker Scout, Extensions.
- Images Tab:**
 - Local tab selected, Hub tab available.
 - Storage usage: 2.36 GB / 2.36 GB in use, 4 images.
 - Last refresh: 21 hours ago.
 - Search bar.
 - A table of images:

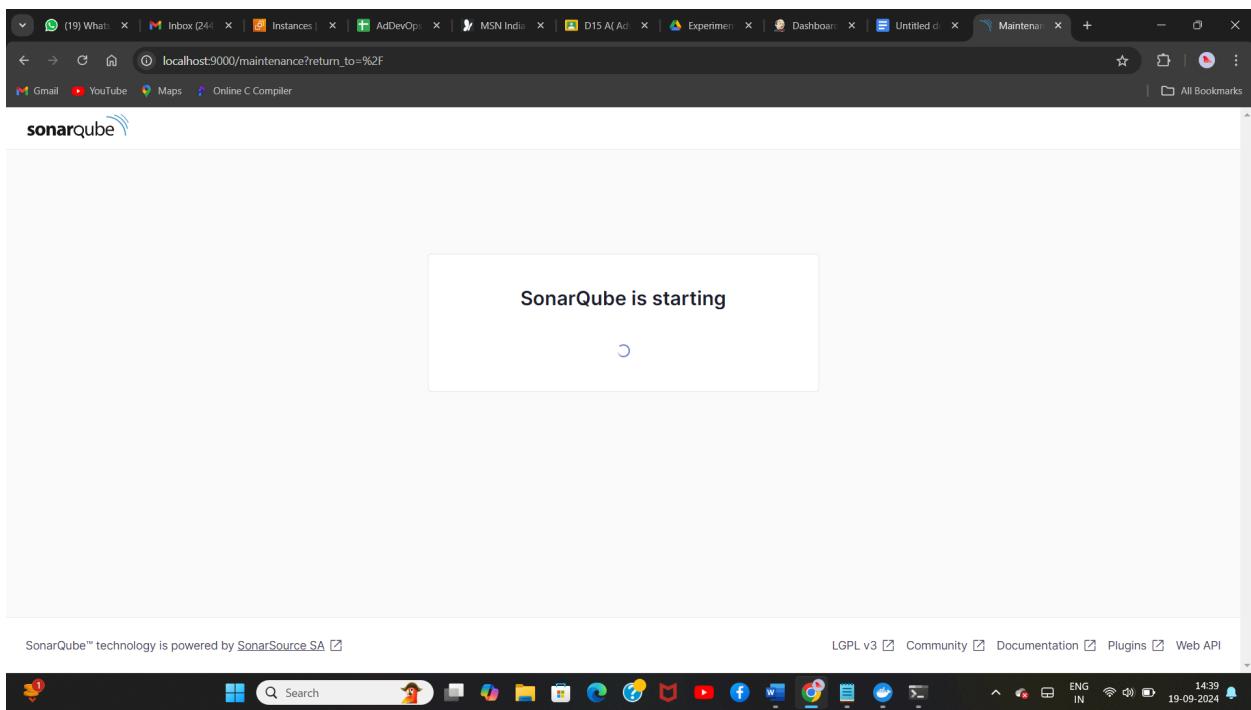
Name	Tag	Status	Created	Size	Actions
node	latest	In use	7 days ago	1.11 GB	⋮ ⚡ 🗑
my-node-app	latest	In use	9 days ago	1.11 GB	⋮ ⚡ 🗑
sonarqube	latest	In use	2 months ago	1.06 GB	⋮ ⚡ 🗑
 - Showing 4 items.
- Walkthroughs:**
 - How do I run a container? (6 mins)
 - Run Docker Hub images (5 mins)
- Bottom Status Bar:** Engine running, RAM 3.65 GB CPU 1.39%, Disk -- GB avail. of -- GB, Terminal v4.34.2, Notifications 3.

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

C:\Users\shraw>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: sha256:72e9fecc71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
e4d231de58da6871a773bb5b77ac510bb10d0ff003c24d82542d1f5cbce4eal

C:\Users\shraw>
```

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

1 of 2

Create a local project

Project display name *

sonarqube-test1



Project key *

sonarqube-test1



Main branch name *

main

The name of your project's default branch [Learn More](#)

Cancel

Next

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

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

The screenshot shows the Jenkins management interface for plugins. The left sidebar has links for 'Dashboard', 'Manage Jenkins', and 'Plugins'. Under 'Plugins', 'Installed plugins' is selected. A search bar at the top right says 'Search installed plugins'. Below the search bar is a table of installed plugins:

Plugin	Description	Status
SCM API Plugin	This plugin provides a new enhanced API for interacting with SCM systems.	Enabled
Script Security	Allows Jenkins administrators to control what in-process scripts can be run by less-privileged users.	Enabled
SnakeYAML API	This plugin provides SnakeYAML for other plugins.	Enabled
SonarQube Scanner for Jenkins	This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.	Enabled
SSH Build Agents plugin	Allows to launch agents over SSH, using a Java implementation of the SSH protocol.	Enabled
SSH Credentials Plugin	Allows storage of SSH credentials in Jenkins	Disabled

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

The screenshot shows the Jenkins 'Configure System' page. In the top navigation bar, the path 'Dashboard > Manage Jenkins > System' is visible. Below this, there is a section titled 'List of SonarQube instances'. A new instance is being added, indicated by a dashed border around the form. The fields are as follows:

- Name:** sonarqube
- Server URL:** http://localhost:9000 (Default is http://localhost:9000)
- Server authentication token:** - none - (dropdown menu)
- Advanced:** (button)

At the bottom of the form are 'Save' and 'Apply' buttons. There is also a 'Add SonarQube' button above the form.

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

The screenshot shows the Jenkins 'Global Tool Configuration' page. In the top navigation bar, the path 'Dashboard > Manage Jenkins > Global Tool Configuration' is visible. Below this, there is a section titled 'SonarQube Scanner'. A new tool is being added, indicated by a dashed border around the form. The fields are as follows:

- Name:** SonarQube
- Install automatically:** (checkbox)
- Install from Maven Central:** (sub-section)
 - Version:** SonarQube Scanner 6.2.0.4584 (dropdown menu)
- Add Installer:** (button)

At the bottom of the form is an 'Add SonarQube Scanner' button.

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

Dashboard > All > New Item

Enter an item name
SonarQube

Select an item type

- Freestyle project**
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different

OK

10. Choose this GitHub repository in Source Code Management.

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

Git ?

Repositories ?

Repository URL ?
https://github.com/shazforiot/MSBuild_firstproject.git

Credentials ?
- none -

+ Add ▾

Advanced ▾

Add Repository

Branches to build ?

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

The screenshot shows the 'Execute SonarQube Scanner' configuration screen. It includes fields for 'JDK', 'Path to project properties' (containing analysis properties), 'Analysis properties' (showing sonar.projectKey=sonarqube, sonar.login=squ_32cbb4ccc16482530991b7a25a9c5e5b980193fa, sonar.sources=HelloWorldCore, sonar.host.url=http://localhost:9000), 'Additional arguments', and 'JVM Options'.

JDK ?
JDK to be used for this SonarQube analysis
(Inherit From Job)

Path to project properties ?
Path to project properties

Analysis properties ?
sonar.projectKey=sonarqube
sonar.login=squ_32cbb4ccc16482530991b7a25a9c5e5b980193fa
sonar.sources=HelloWorldCore
sonar.host.url=http://localhost:9000

Additional arguments ?
Additional arguments

JVM Options ?
JVM Options

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

The screenshot shows the 'Permissions' configuration screen. It lists groups and users with their assigned roles and permissions. The 'Administrator' user has 'Execute Analysis' checked under the 'Administrator' role.

		Administer System	Administer	Execute Analysis	Create
sonar-administrators	System administrators	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Quality Gates <input checked="" type="checkbox"/> Quality Profiles	<input type="checkbox"/>	<input checked="" type="checkbox"/> Projects
sonar-users	Every authenticated user automatically belongs to this group	<input type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Projects
A Administrator	admin	<input checked="" type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input type="checkbox"/> Projects

13. Run The Build.

Status

- </> Changes
- Workspace
- ▷ Build Now
- ⚙ Configure
- Delete Project
- SonarQube
- ✍ Rename

14. Console Output

The screenshot shows the Jenkins interface for a SonarQube project. The top navigation bar includes links for Dashboard, SonarQube, #1, and Console Output. The main content area is titled "Console Output" and shows the following log output:

```
Started by user Shrawani Anil Patil
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\jenkins\workspace\SonarQube
The recommended git tool is: NONE
No credentials specified
> C:\Program Files\git\bin\git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\jenkins\workspace\SonarQube\.git # timeout=10
Fetching changes from the remote Git repository
> C:\Program Files\git\bin\git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/MSBuild_firstproject.git
> C:\Program Files\git\bin\git.exe --version # timeout=10
> git --version # 'git' version 2.46.0.windows.1'
> C:\Program Files\git\bin\git.exe fetch --tags --force --progress -- https://github.com/shazforiot/MSBuild_firstproject.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> C:\Program Files\git\bin\git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf (refs/remotes/origin/master)
> C:\Program Files\git\bin\git.exe config core.sparsecheckout # timeout=10
> C:\Program Files\git\bin\git.exe checkout -f f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
Commit message: "updated"
First time build. Skipping changelog.
[SonarQube] $ C:\ProgramData\Jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\SonarQube\bin\sonar-scanner.bat -
Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=sonarqube -Dsonar.login=squ_32ccb4ccc16482530991b7a25a9c5e5b980193fa -
Dsonar.host.url=http://localhost:9000 -Dsonar.sources=HelloWorldCore -Dsonar.projectBaseDir=C:\ProgramData\Jenkins\jenkins\workspace\SonarQube
```

Below the log output, there is another section of Jenkins logs:

```
Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=sonarqube -Dsonar.login=squ_32ccb4ccc16482530991b7a25a9c5e5b980193fa -
Dsonar.host.url=http://localhost:9000 -Dsonar.sources=HelloWorldCore -Dsonar.projectBaseDir=C:\ProgramData\Jenkins\jenkins\workspace\SonarQube
16:22:47.189 WARN Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'
16:22:47.224 INFO Scanner configuration file:
C:\ProgramData\Jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\SonarQube\bin\..\conf\sonar-scanner.properties
16:22:47.230 INFO Project root configuration file: NONE
16:22:47.294 INFO SonarScanner CLI 6.2.0.4584
16:22:47.299 INFO Java 21 Oracle Corporation (64-bit)
16:22:47.311 INFO Windows 11 10.0 amd64
16:22:47.392 INFO User cache: C:\windows\system32\config\systemprofile\.sonar\cache
16:22:49.719 INFO JRE provisioning: os-windows, arch[amd64]
16:22:51.182 INFO Communicating with SonarQube Server 10.6.0.92116
16:22:52.539 INFO Starting SonarScanner Engine...
16:22:52.541 INFO Java 17.0.11 Eclipse Adoptium (64-bit)
16:22:55.497 INFO Load global settings
16:22:56.059 INFO Load global settings (done) | time=560ms
16:22:56.072 INFO Server id: 1478411E-AZIos_TCFdjvGUUATc6Z
16:22:56.105 INFO Loading required plugins
16:22:56.107 INFO Load plugins index
16:22:56.344 INFO Load plugins index (done) | time=237ms
16:22:56.345 INFO Load/download plugins
16:22:56.509 INFO Load/download plugins (done) | time=105ms
16:22:57.306 INFO Process project properties
16:22:57.326 INFO Process project properties (done) | time=20ms
16:22:57.360 INFO Project key: sonarqube
16:22:57.362 INFO Base dir: C:\ProgramData\Jenkins\jenkins\workspace\SonarQube
16:22:57.363 INFO Working dir: C:\ProgramData\Jenkins\jenkins\workspace\SonarQube\scannerwork
16:22:57.384 INFO Load project settings for component key: 'sonarqube'
16:22:57.532 INFO Load quality profiles
```

```
Dashboard > SonarQube > #1 > Console Output

16:23:40.924 INFO Load analysis cache (404) | time=21ms
16:23:41.084 WARN The property 'sonar.login' is deprecated and will be removed in the future. Please use the 'sonar.token' property instead when passing a token.
16:23:41.154 INFO Preprocessing files...
16:23:42.333 INFO 2 languages detected in 23 preprocessed files
16:23:42.334 INFO 0 files ignored because of scm ignore settings
16:23:42.339 INFO Loading plugins for detected languages
16:23:42.341 INFO Load/download plugins
16:23:43.242 INFO Load/download plugins (done) | time=902ms
16:23:43.487 INFO Executing phase 2 project builders
16:23:43.499 INFO Executing phase 2 project builders (done) | time=2ms
16:23:43.514 INFO Load project repositories
16:23:43.546 INFO Load project repositories (done) | time=29ms
16:23:43.611 INFO Indexing files...
16:23:43.613 INFO Project configuration:
16:23:43.700 INFO 23 files indexed
16:23:43.708 INFO Quality profile for cs: Sonar way
16:23:43.705 INFO Quality profile for json: Sonar way
16:23:43.707 INFO -----
16:23:43.707 INFO Run sensors on module sonarqube
16:23:43.874 INFO Load metrics repository
16:23:43.937 INFO Load metrics repository (done) | time=63ms
16:23:46.020 INFO Sensor C# Project Type Information [csharp]
16:23:46.037 INFO Sensor C# Project Type Information [csharp] (done) | time=17ms
16:23:46.042 INFO Sensor C# Analysis Log [csharp]
16:23:46.111 INFO Sensor C# Analysis Log [csharp] (done) | time=76ms
16:23:46.112 INFO Sensor C# Properties [csharp]
16:23:46.113 INFO Sensor C# Properties [csharp] (done) | time=0ms
16:23:46.115 INFO Sensor JaCoCo XML Report Importer [jacoco]
16:23:46.118 INFO 'sonar.coverage.jacoco.xmlReportPaths' is not defined. Using default locations: target/site/jacoco/jacoco.xml,target/site/jacoco-1.4/jacoco.xml,target/site/jacoco/jacoco-1.4/jacoco.xml,target/site/jacoco/jacoco-1.4/jacoco.xml
```

Dashboard > SonarQube > #1 > Console Output

```
16:23:49.539 INFO Sensor C# File Caching Sensor [csharp]
16:23:49.540 WARN Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir' property.
16:23:49.541 INFO Sensor C# File Caching Sensor [csharp] (done) | time=2ms
16:23:49.542 INFO Sensor Zero Coverage Sensor
16:23:49.554 INFO Sensor Zero Coverage Sensor (done) | time=15ms
16:23:49.570 INFO SCM Publisher SCM provider for this project is: git
16:23:49.572 INFO SCM Publisher 2 source files to be analyzed
16:23:51.571 INFO SCM Publisher 2/2 source files have been analyzed (done) | time=1997ms
16:23:51.583 INFO CPD Executor Calculating CPD for 0 files
16:23:51.589 INFO CPD Executor CPD calculation finished (done) | time=0ms
16:23:51.614 INFO SCM revision ID 'f2bc042c046e72427c380caeed6dfee7b49adef'
16:23:51.889 INFO Analysis report generated in 265ms, dir size=19.7 kB
16:23:52.000 INFO Analysis report compressed in 108ms, zip size=20.5 kB
16:23:55.249 INFO Analysis report uploaded in 3245ms
16:23:55.259 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube
16:23:55.260 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
16:23:55.262 INFO More about the report processing at http://localhost:9000/api/ce/task?id=3bd7337b-a267-400d-9686-a4a28c011b88
16:23:55.352 INFO Analysis total time: 58.695 s
16:23:55.356 INFO SonarScanner Engine completed successfully
16:23:55.469 INFO EXECUTION SUCCESS
16:23:55.471 INFO Total time: 1:08.257s
Finished: SUCCESS
```

English (India)
English (India)

To switch input methods, press Windows key + space. 2.1

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

SonarQube Projects Issues Rules Quality Profiles Quality Gates Administration More Q

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 Reliability Maintainability

Open issues	A	Open issues	A	Open issues	A
0 H		0 M		0 L	

Accepted issues Coverage Duplications

This screenshot shows the SonarQube dashboard for the 'main' project. At the top, there's a navigation bar with links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, More, and a search bar. Below that is a secondary navigation bar with Overview, Issues, Security Hotspots, Measures, Code, Activity, Project Settings, and Project Information. The main content area is titled 'main' and shows a large green 'Passed' status with a checkmark icon. It indicates that the last analysis had warnings, with a link to see details. There are two tabs: 'New Code' (selected) and 'Overall Code'. Below these are three sections: Security, Reliability, and Maintainability, each showing 0 open issues and an 'A' grade. At the bottom, there are three more sections: Accepted issues, Coverage, and Duplications.

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.

The screenshot shows the 'Create a local project' wizard in the SonarQube interface. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The main content area is titled 'Create a local project' and shows the first step of the wizard. It has three input fields: 'Project display name *' containing 'sonarqube-pipeline', 'Project key *' containing 'sonarqube-pipeline', and 'Main branch name *' containing 'main'. Below these fields is a note: 'The name of your project's default branch' with a 'Learn More' link. At the bottom are 'Cancel' and 'Next' buttons.

1 of 2

Create a local project

Project display name *

 ✓

Project key *

 ✓

Main branch name *

The name of your project's default branch [Learn More](#)

[Cancel](#) [Next](#)

2 of 2

Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code. This helps you focus attention on the most recent changes to your project, enabling you to follow the Clean as You Code methodology. Learn more: [Defining New Code](#)

Choose the baseline for new code for this project

Use the global setting

Previous version

Any code that has changed since the previous version is considered new code.

Recommended for projects following regular versions or releases.

Define a specific setting for this project

Previous version

Any code that has changed since the previous version is considered new code.

Recommended for projects following regular versions or releases.

Number of days

Any code that has changed in the last x days is considered new code. If no action is taken on a new issue after x days, this issue will become part of the overall code.

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

sonarqube Docs 10.6

Latest | Analyzing source code | Scanners | SonarScanner CLI

SonarScanner CLI

SonarScanner Issue Tracker Show fewer ▾

6.2 2024-09-17

Support PKCS12 truststore generated with OpenSSL

Download scanner for: Linux x64 Linux AArch64 Windows x64 macOS x64 macOS AArch64 Docker Any (Requires a pre-installed JVM)

6.1 2024-06-27

macOS and Linux AArch64 distributions

Download scanner for: Linux x64 Linux AArch64 Windows x64 macOS x64 macOS AArch64 Docker Any (Requires a pre-installed JVM) **Windows x64** (highlighted with a red box)

6.0 2024-06-04

New bootstrapping mechanism and JRE provisioning with SonarQube 10.6+ and SonarCloud

Download scanner for: Linux x64 Windows x64 macOS x64 Docker Any (Requires a pre-installed JVM)

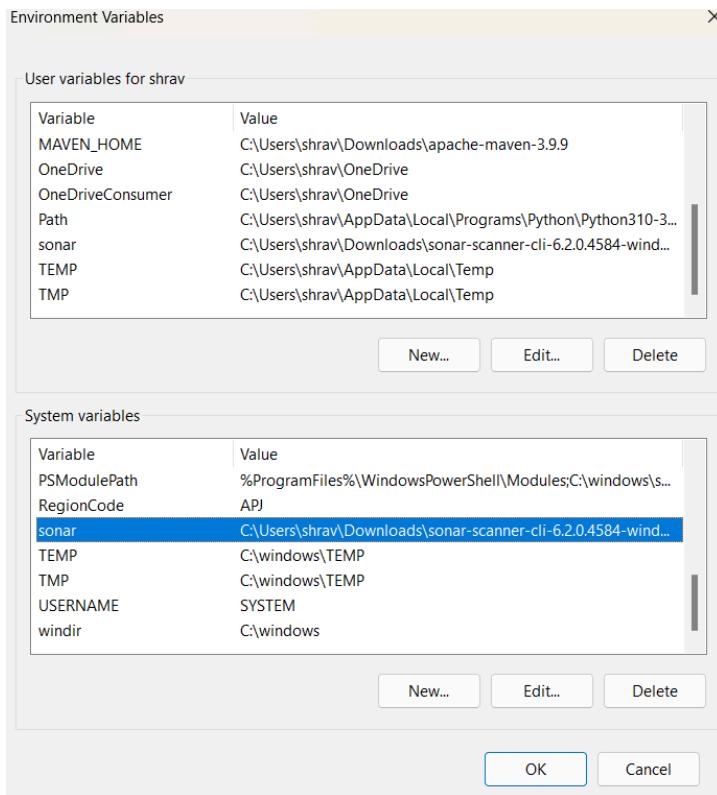
On this page

- Configuring your project
- Running SonarScanner CLI from the zip file
- Running SonarScanner CLI from the Docker image
- Scanning C, C++, or Objective-C projects
- Sample projects
- Alternatives to sonar-project.properties
- Alternate analysis directory
- Advanced configuration
- Troubleshooting

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.

New Item

Enter an item name

sonarqube-pipeline

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

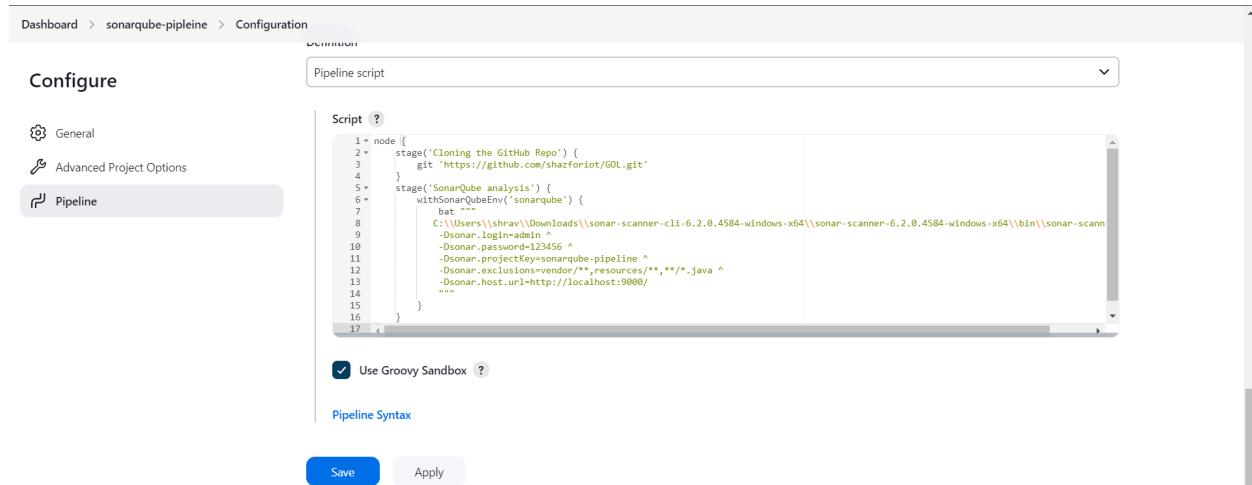


Folder

OK

Add pipeline script :

```
node {  
    stage('Cloning the GitHub Repo') {  
        git 'https://github.com/shazforiot/GOL.git'  
    }  
    stage('SonarQube analysis') {  
        withSonarQubeEnv('sonarqube') {  
            bat """  
  
C:\\\\Users\\\\shraw\\\\Downloads\\\\sonar-scanner-cli-6.2.0.4584-windows-x64\\\\sonar-scanner-6.2.0.4  
584-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/  
    """  
        }  
    }  
}
```



The screenshot shows the Jenkins Pipeline configuration page for a project named "sonarqube-pipeline". The "Pipeline" tab is selected in the left sidebar. The main area contains a code editor titled "Pipeline script" with the following Groovy code:

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

Below the code editor, there is a checkbox labeled "Use Groovy Sandbox" which is checked. At the bottom of the screen are "Save" and "Apply" buttons.

Step 4: Save the pipeline and build it.

The screenshot shows the Jenkins Pipeline interface for the 'sonarqube-pipleine'. On the left, a sidebar contains links for Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, SonarQube, Stages, Rename, and Pipeline Syntax. The main area is titled 'Stage View' and displays two stages: 'Cloning the GitHub Repo' (16s) and 'SonarQube analysis' (16min 25s). A summary bar indicates an average stage time of 16min 25s across 44 stages. Below this is a 'Permalinks' section with a list of recent builds. On the right, a 'Build History' section shows the last four builds, all of which are successful (#1, 15 hr ago).

The screenshot shows the 'Stages' view for the 'sonarqube-pipleine' pipeline. The pipeline is named 'pipeline' and consists of three stages: 'Start', 'Cloning the Git...', and 'SonarQube anal...', followed by an 'End' stage. Stage 'Cloning the Git...' is highlighted with a green checkmark, indicating it has been successfully executed. The 'Configure' and 'Build' buttons are visible at the top right of the stage list.

Console output:

Console Output

[Download](#)[Copy](#)[View as plain text](#)

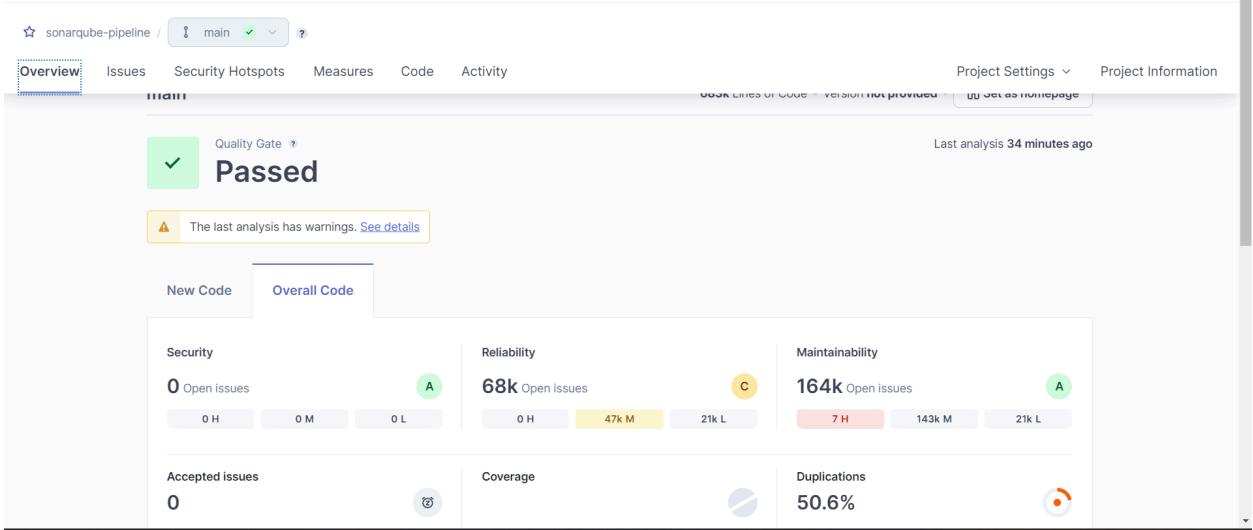
Skipping 4,249 KB.. [Full Log](#)

```
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
```

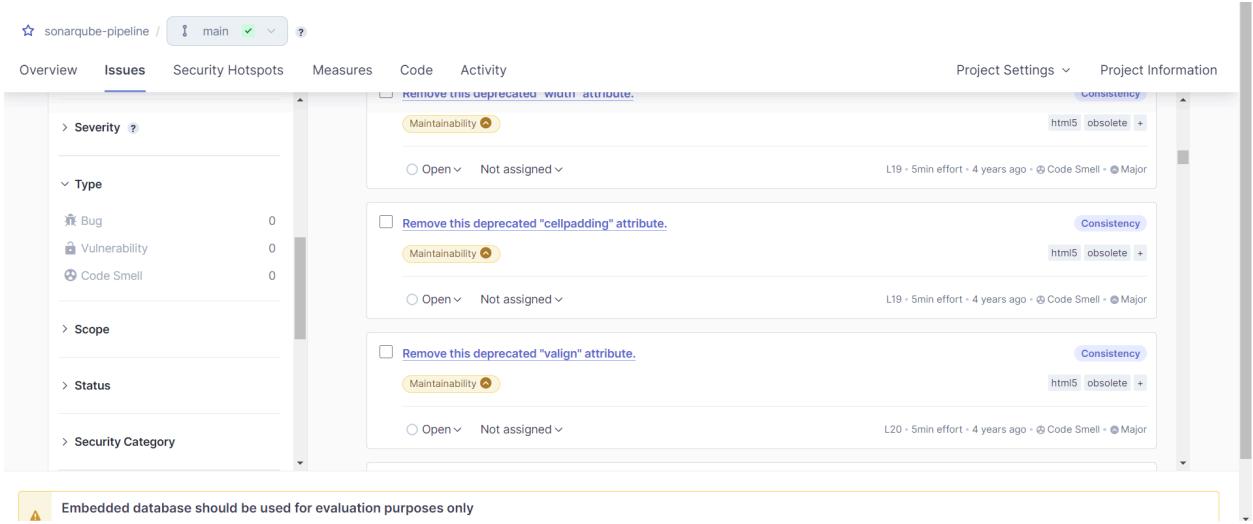
Dashboard > sonarqube-pipeline > #1

```
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 'ba799ba7eb576f04a4612322b0412c5e6e1e5e4'
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=84bf-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



star sonarqube-pipeline / main ?

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

> Severity ?

< Type

- Bug 0
- Vulnerability 0
- Code Smell 0

> Scope

> Status

> Security Category

gameoflife-acceptance-tests/Dockerfile

Use a specific version tag for the image. Intentionality
Maintainability No tags

Surround this variable with double quotes; otherwise, it can lead to unexpected behavior. Intentionality
Maintainability No tags

Surround this variable with double quotes; otherwise, it can lead to unexpected behavior. Intentionality
L12 - 5min effort - 4 years ago - ⚡ Code Smell - ⚡ Major

210,549 issues 3135d effort



Embedded database should be used for evaluation purposes only

The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

star sonarqube-pipeline / main ?

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

> Severity ?

< Type

- Bug 0
- Vulnerability 0
- Code Smell 0

> Scope

> Status

> Security Category

Add "lang" and/or "xml:lang" attributes to this "<html>" element Intentionality
Reliability accessibility wcag2-a

Add "<th>" headers to this "<table>". Intentionality
Reliability accessibility wcag2-a

Remove this deprecated "width" attribute. Consistency
Maintainability html5 obsolete



Embedded database should be used for evaluation purposes only

star sonarqube-pipeline / main ?

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

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%

star 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

localhost:9000/project/issues?id=sonarqube-pipeline&issueStatuses=OPEN%2CCONFIRMED

star 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 See history

HTML 678k

XML 4.7k

JSP 332

CSS 110

Docker 19

sonarqube-pipeline

View as Tree Select files Navigate 6 files

Lines of Code 682,883 See history

- gameoflife-acceptance-tests 164
- gameoflife-build 368
- gameoflife-core 3,675

star 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 ↗

Maintainability ↗

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

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

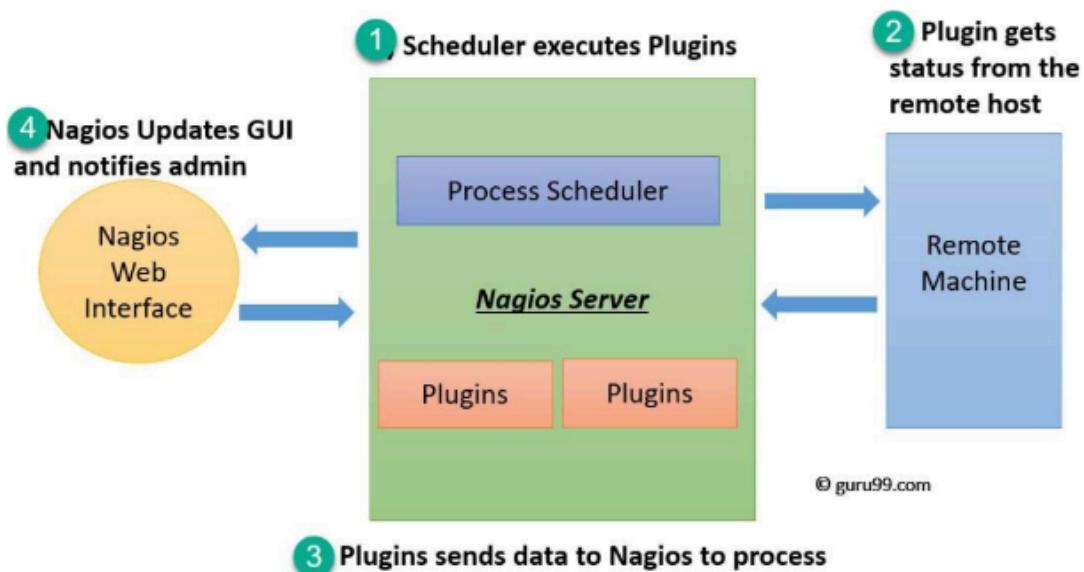
Following are the important features of Nagios monitoring tool:

- 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 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

Instances (1/1) Info								
		Last updated 16 minutes ago		Connect	Instance state	Actions	Launch instances	
<input type="text"/> Find Instance by attribute or tag (case-sensitive)				All states				
<input checked="" type="checkbox"/>	Name Edit	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public
<input checked="" type="checkbox"/>	nagios-host	i-0bb1ccfdbeba9fd0	Running View details Logs	t2.micro	2/2 checks passed View alarms +	View alarms +	us-east-1c	ec2-54

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

Inbound rules (7)						
<input type="text"/> Search						
<input type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol	Port
<input type="checkbox"/>	-	sgr-0a282dd4634b36...	IPv6	All ICMP - IPv6	IPv6 ICMP	All
<input type="checkbox"/>	-	sgr-0a3e4d646f42fe480	IPv4	All traffic	All	All
<input type="checkbox"/>	-	sgr-04a7a2e2d7530a2ff	IPv4	All ICMP - IPv4	ICMP	All
<input type="checkbox"/>	-	sgr-0a55edf7d1641af50	IPv4	HTTP	TCP	80
<input type="checkbox"/>	-	sgr-0bce08a71dc073934	IPv4	Custom TCP	TCP	5666
<input type="checkbox"/>	-	sgr-0b170e9da26a879...	IPv4	HTTPS	TCP	443
<input type="checkbox"/>	-	sgr-03774abfe3a84ac27	IPv4	SSH	TCP	22

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

```
,      #
~\_\_ #####          Amazon Linux 2023
~~ \_\_#####\
~~   \###|
~~     \#/  https://aws.amazon.com/linux/amazon-linux-2023
~~     V~'__->
~~~      /
~~ ._. /_
~/m/' [ec2-user@ip-172-31-34-87 ~]$
```

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
libwebp-1.2.4-1.amzn2023.0.6.x86_64
libxcb-1.13.1-7.amzn2023.0.2.x86_64
libxml2-devel-2.10.4-1.amzn2023.0.6.x86_64
pcre2-utf16-10.40-1.amzn2023.0.3.x86_64
pixman-0.40.0-3.amzn2023.0.3.x86_64
xml-common-0.6.3-56.amzn2023.0.2.noarch
xz-devel-5.2.5-9.amzn2023.0.2.x86_64

libtiff-devel-4.4.0-4.amzn2023.0.18.x86_64
libwebp-devel-1.2.4-1.amzn2023.0.6.x86_64
libxcb-devel-1.13.1-7.amzn2023.0.2.x86_64
pcre2-devel-10.40-1.amzn2023.0.3.x86_64
pcre2-utf32-10.40-1.amzn2023.0.3.x86_64
sysprof-capture-devel-3.40.1-2.amzn2023.0.2.x86_64
xorg-x11-proto-devel-2021.4-1.amzn2023.0.2.noarch
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

```
GNU nano 5.8                               /usr/local/nagios/etc/objects/contacts.cfg                         Modified
#
# CONTACTS
#
#####
#
# Just one contact defined by default - the Nagios admin (that's you)
# This contact definition inherits a lot of default values from the
# 'generic-contact' template which is defined elsewhere.
#
define contact {
    contact_name      nagiosadmin          ; Short name of user
    use               generic-contact       ; Inherit default values from generic-contact template (defined above)
    alias             Nagios Admin        ; Full name of user
    email             shravani@1427@gmail.com ;<***** CHANGE THIS TO YOUR EMAIL ADDRESS *****
}

#####
#
# CONTACT GROUPS
#
^G Help          ^Q Write Out     ^W Where Is      ^K Cut           ^T Execute      ^C Location     M-U Undo      M-A Set Mark   M-[ To Bracket M-Q Previous
^X Exit          ^R Read File     ^E Replace      ^U Paste         ^J Justify      ^Y Go To Line   M-E Redo      M-G Copy      M-C Where Was  M-W Next
i-0bb1ccfdbaba9fd0 (nagios-host)
PublicIP: 54.90.165.191 PrivateIP: 172.31.34.87
```

15. Configure the web interface.

```
sudo make install-webconf
```

```
[ec2-user@ip-172-31-34-87 nagios-4.4.6]$ sudo nano /usr/local/nagios/etc/objects/contacts
[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/requirements  
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/requirements  
[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.gh
              ├─67758 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.gh
              ├─67759 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.gh
              ├─67760 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.gh
              └─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.gh' 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

```

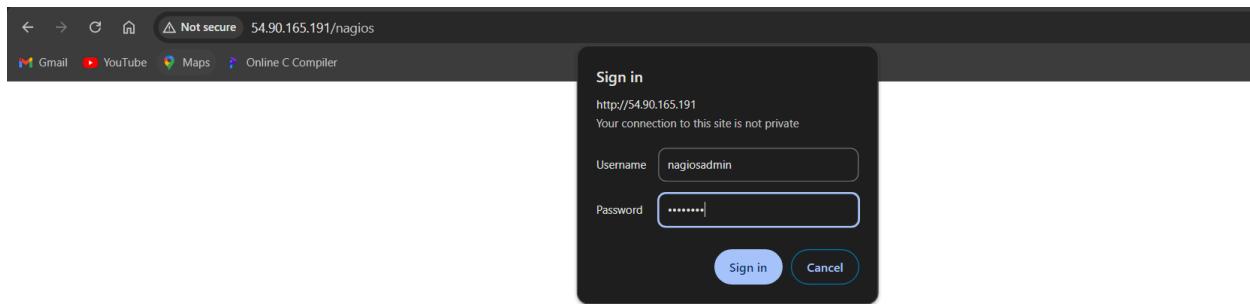
22. Go back to EC2 Console and copy the Public IP address of this instance

The screenshot shows the AWS EC2 Instance Details page for an instance named "i-0bb1ccfdbbaeba9fd0 (nagios-host)". The instance is currently running. Key details shown include:

- Public IPv4 address:** 54.90.165.191 | [open address](#)
- Instance state:** Running
- Private IPv4 addresses:** 172.31.34.87
- Public IPv4 DNS:** ec2-54-90-165-191.compute-1.amazonaws.com | [open address](#)

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.

The screenshot shows the Nagios Core 4.4.6 dashboard. At the top right, the Nagios logo is displayed with the text "Nagios® Core™" and a green checkmark indicating "Daemon running with PID 67756". Below the logo, the version information "Nagios® Core™ Version 4.4.6 April 28, 2020" is shown, along with a "Check for updates" link. A blue banner at the top center states "A new version of Nagios Core is available! Visit nagios.org to download Nagios 4.5.5". On the left side, there is a vertical navigation menu with sections: General (Home, Documentation), Current Status (Tactical Overview, Map (Legacy), Hosts, Services, Host Groups, Service Groups, Problems), Reports (Availability, Trends (Legacy), Alerts, History, Summary, Histogram (Legacy), Notifications, Event Log), and System (Comments, Downtime, Process Info, Performance Info, Scheduling Queue). The main content area is divided into several panels: "Get Started" (with a list of bullet points: Start monitoring your infrastructure, Change the look and feel of Nagios, Extend Nagios with hundreds of addons, Get support, Get training, Get certified), "Quick Links" (with links to Nagios Library, Nagios Labs, Nagios Exchange, Nagios Support, Nagios.com, and Nagios.org), "Latest News" (empty), and "Don't Miss..." (empty). A "Page Tour" button is located on the far right.

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

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

[EC2](#) > ... > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

Add additional tags

▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

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 --v
nagios    69366  69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios --v
nagios    69367  69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios --v
nagios    69368  69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios --v
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
                                                N
```

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

```
ubuntu@ip-172-31-33-27:~$ sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InR
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports I
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe am
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Pack
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Tr
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe am
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe am
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse
```

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.

#
# 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

The screenshot shows the Nagios web interface. On the left, there is a navigation sidebar with links for General (Home, Documentation), Current Status (Tactical Overview, Map (Legacy), Hosts, Services, Host Groups Summary, Grid, Service Groups Summary, Grid), Problems (Services (Unhandled), Hosts (Unhandled), Network Outages), and Reports (Availability). The main content area displays the following information:

- Current Network Status:** Last Updated: Mon Oct 7 18:26:34 UTC 2024. Updated every 90 seconds. Nagios® Core™ 4.4.6 - www.nagios.org. Logged in as nagiosadmin.
- Host Status Totals:** Up: 2, Down: 0, Unreachable: 0, Pending: 0. All Problems: 0, All Types: 2.
- Service Status Totals:** Ok: 12, Warning: 2, Unknown: 0, Critical: 2, Pending: 0. All Problems: 4, All Types: 16.
- Host Status Details For All Host Groups:** Limit Results: 100. Host: linuxserver, Status: UP, Last Check: 10-07-2024 18:22:38, Duration: 0d 0h 23m 18s, Status Information: PING OK - Packet loss = 0%, RTA = 0.03 ms. Host: localhost, Status: UP, Last Check: 10-07-2024 18:23:07, Duration: 0d 1h 57m 49s, Status Information: PING OK - Packet loss = 0%, RTA = 0.03 ms.

At the bottom right of the dashboard, there is a red button labeled "Tour".

Nagios®

General

- [Home](#)
- [Documentation](#)

Current Status

- [Tactical Overview](#)
- [Map \(Legacy\)](#)
- [Hosts](#)
- [Services](#)
- [Host Groups](#)
- [Summary](#)
- [Grid](#)
- [Service Groups](#)
- [Summary](#)
- [Grid](#)
- [Problems](#)
- [Services \(Unhandled\)](#)
- [Hosts \(Unhandled\)](#)
- [Network Outages](#)
- [Quick Search:](#)

Reports

- [Availability](#)
- [Trends \(Legacy\)](#)
- [Alerts](#)

Host Information

Last Updated: Mon Oct 7 18:28:15 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.4.6 - www.nagios.org
Logged in as nagiosadmin

[View Status Detail For This Host](#)
[View Alert History For This Host](#)
[View Trends For This Host](#)
[View Alert Histogram For This Host](#)
[View Availability Report For This Host](#)
[View Notifications For This Host](#)

Host
linuxserver (linuxserver)

Member of
No hostgroups

127.0.0.1

Host State Information

Host Status:	UP (for 0d 0h 24m 59s)
Status Information:	PING OK - Packet loss = 0%, RTA = 0.03 ms
Performance Data:	rtt=0.034000ms;3000.000000;5000.000000;0.000000 pl=0%;80;100;0
Current Attempt:	1/10 (HARD state)
Last Check Time:	10-07-2024 18:27:38
Check Type:	ACTIVE
Check Latency / Duration:	0.000 / 4.160 seconds
Next Scheduled Active:	10-07-2024 18:32:38
Check:	
Last State Change:	10-07-2024 18:03:16
Last Notification:	N/A (notification 0)
Is This Host Flapping?	NO (0.00% state change)
In Scheduled Downtime?	NO
Last Update:	10-07-2024 18:28:05 (0d 0h 0m 10s ago)
Active Checks:	ENABLED
Passive Checks:	ENABLED
Obsessing:	ENABLED

Nagios®

General

- [Home](#)
- [Documentation](#)

Current Status

- [Tactical Overview](#)
- [Map \(Legacy\)](#)
- [Hosts](#)
- [Services](#)
- [Host Groups](#)
- [Summary](#)
- [Grid](#)
- [Service Groups](#)
- [Summary](#)
- [Grid](#)
- [Problems](#)
- [Services \(Unhandled\)](#)
- [Hosts \(Unhandled\)](#)
- [Network Outages](#)
- [Quick Search:](#)

Reports

- [Availability](#)
- [Trends \(Legacy\)](#)
- [Alerts](#)

Current Network Status

Last Updated: Mon Oct 7 18:33:39 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.4.6 - www.nagios.org
Logged in as nagiosadmin

[View History For All hosts](#)
[View Notifications For All Hosts](#)
[View Host Status Detail For All Hosts](#)

Host Status Totals

Up	2	0	0	0	Pending	0
All Problems	0	2	0	0	All Types	0

Service Status Totals

Ok	12	2	0	2	Critical	0	Pending	0
All Problems	4	2	0	16	All Types	0	Pending	0

Service Status Details For All Hosts

Host **	Service **	Status **	Last Check **	Duration **	Attempt **	Status Information	
						OK	WARNING
linuxserver	Current Load	OK	10-07-2024 18:28:53	0d 0h 20m 49s	1/4	OK - load average: 0.00, 0.00, 0.00	
	Current Users	OK	10-07-2024 18:29:31	0d 0h 20m 8s	1/4	USERS OK - 2 users currently logged in	
	HTTP	WARNING	10-07-2024 18:33:08	0d 0h 25m 31s	4/4	HTTP WARNING: HTTP/1.1 403 Forbidden - 319 bytes in 0.001 second response time	
	PING	OK	10-07-2024 18:30:46	0d 0h 27m 53s	1/4	PING OK - Packet loss = 0%, RTA = 0.03 ms	
	Root Partition	OK	10-07-2024 18:31:23	0d 0h 27m 16s	1/4	DISK OK - free space / 6000 MB (74.91% inode=9%)	
	SSH	WARNING	10-07-2024 18:32:01	0d 0h 26m 38s	1/4	SSH OK - OpenSSH_8.7 (protocol 2.0)	
	Swap Usage	Critical	10-07-2024 18:30:38	0d 0h 23m 1s	4/4	SWAP CRITICAL - 0% free (0 MB out of 0 MB) - Swap is either disabled, not present, or of zero size.	
	Total Processes	OK	10-07-2024 18:33:16	0d 0h 25m 23s	1/4	PROCS OK, 37 processes with STATE = RNSDT	
localhost	Current Load	OK	10-07-2024 18:29:22	0d 2h 4m 17s	1/8	OK - load average: 0.00, 0.00, 0.00	
	Current Users	OK	10-07-2024 18:30:00	0d 0h 3m 39s	1/4	USERS OK - 2 users currently logged in	
	HTTP	WARNING	10-07-2024 18:28:37	0d 2h 9m 2s	4/4	HTTP WARNING: HTTP/1.1 403 Forbidden - 319 bytes in 0.001 second response time	
	PING	OK	10-07-2024 18:31:15	0d 0h 2m 24s	1/4	PING OK - Packet loss = 0%, RTA = 0.03 ms	
	Root	OK	10-07-2024 18:32:01	0d 0h 2m 47s	1/4	DISK OK - free space / 6000 MB (74.91% inode=9%)	

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.

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

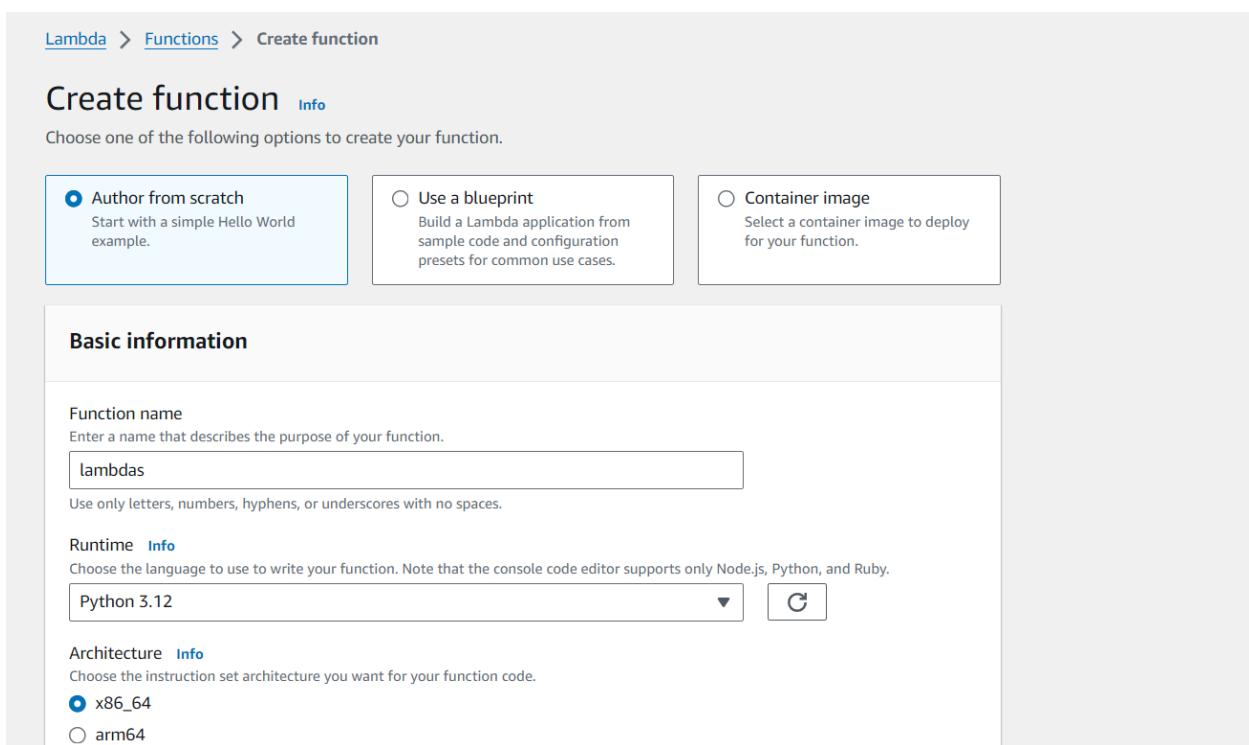
1. Open up the Lambda Console and click on the Create button.



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.



The screenshot shows the 'Create function' wizard. Step 1: 'Choose one of the following options to create your function.' It has three options: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. Step 2: 'Basic information' section. It asks for a 'Function name' and provides a placeholder 'lambdas'. Step 3: 'Runtime' section. It asks for the language ('Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.') and shows 'Python 3.12' selected. Step 4: 'Architecture' section. It asks for the instruction set architecture ('Choose the instruction set architecture you want for your function code.') and shows 'x86_64' selected.

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

▼ Change default execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Create a new role with basic Lambda permissions

Use an existing role

Create a new role from AWS policy templates

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

LabRole

[View the LabRole role](#) on the IAM console.

► Advanced settings

Cancel **Create function**

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.

Successfully created the function **lambdas**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > lambdas

lambdas

Throttle Copy ARN Actions ▾

▼ Function overview [Info](#)

Diagram [Template](#)

lambdas

Layers (0)

+ Add destination

Description

-

Last modified **in 0 seconds**

Function ARN

arn:aws:lambda:us-east-1:533162464157:function:lambdas

Function URL [Info](#)

-

The screenshot shows the AWS Lambda function editor interface. At the top, a green banner displays the message: "Successfully created the function lambdas. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." Below the banner is a toolbar with File, Edit, Find, View, Go, Tools, Window, Test, Deploy, and a Help button. The main area has tabs for lambda_function and Environment Var. The code editor shows the following Python code:

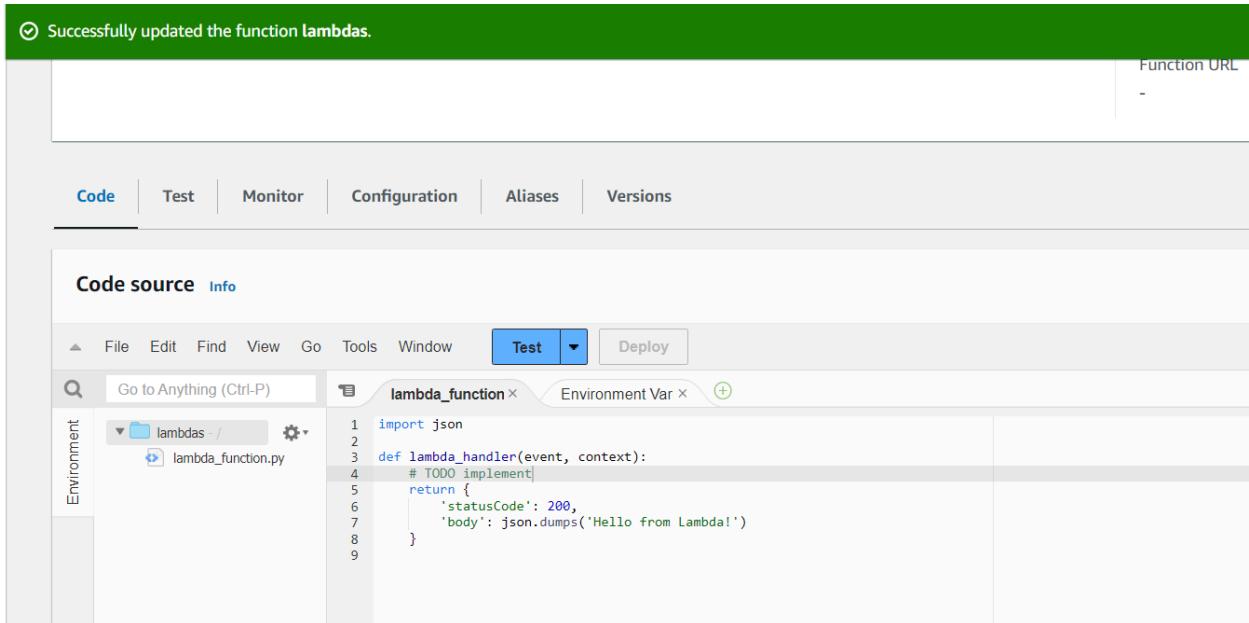
```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')
8     }
```

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.

The screenshot shows the AWS Lambda function configuration page. At the top, a note says: "function code is resilient to snapshot operations, review the [SnapStart compatibility considerations](#)". Below that, a dropdown menu is set to "None". A note below says: "Supported runtimes: Java 11, Java 17, Java 21.". Under "Timeout", there is a field set to "0 min 1 sec". Under "Execution role", it says: "Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#)". Two radio buttons are shown: "Use an existing role" (selected) and "Create a new role from AWS policy templates". Under "Existing role", it says: "Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs." A dropdown menu shows "LabRole" and a "View" link. A "Create" button is at the bottom right.

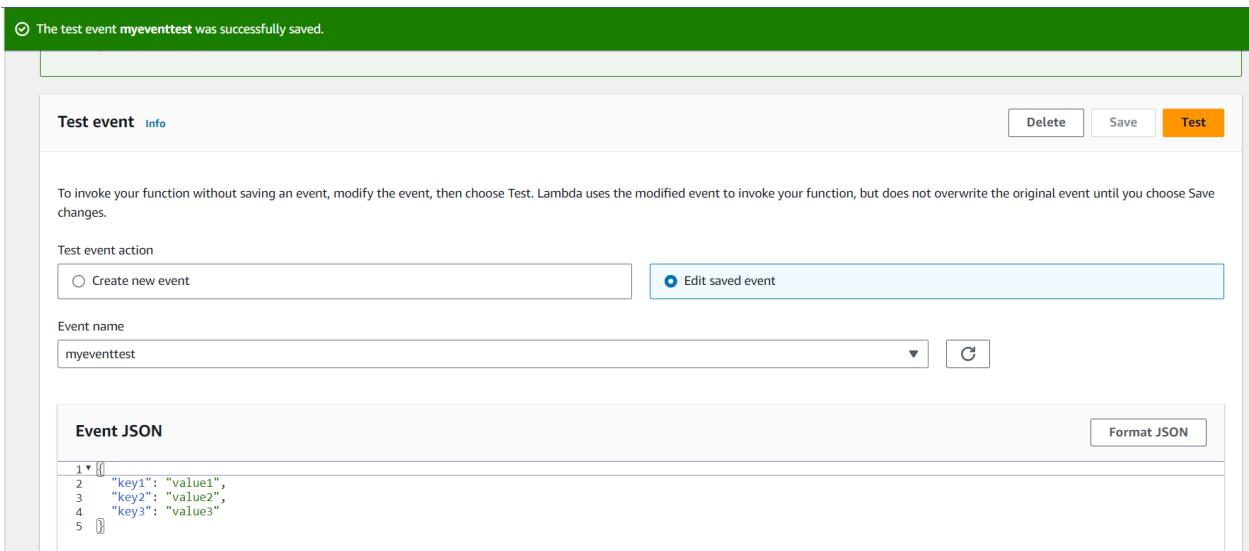
5. You can make changes to your function inside the code editor. You can also upload a zip file of your function or upload one from an S3 bucket if needed.
Press Ctrl + S to save the file and click Deploy to deploy the changes.



The screenshot shows the AWS Lambda console's code editor interface. At the top, a green success bar displays the message "Successfully updated the function lambdas.". Below the bar, the navigation tabs are "Code", "Test", "Monitor", "Configuration", "Aliases", and "Versions". The "Code source" tab is selected. The main area contains a code editor with the following Python code:import json
def lambda_handler(event, context):
 # TODO implement
 return {
 'statusCode': 200,
 'body': json.dumps('Hello from Lambda!')
 }

```
On the left sidebar, there is a file tree showing a folder named "lambdas" containing "lambda_function.py". The "Environment" section is collapsed. At the bottom right of the code editor, there are "Test" and "Deploy" buttons, with "Test" being highlighted.
```

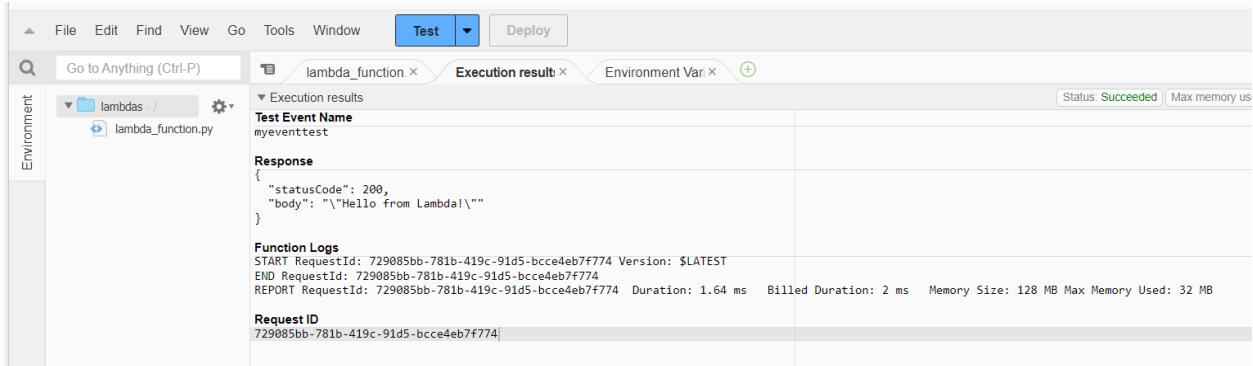
6. Click on Test and you can change the configuration, like so. If you do not have anything in the request body, it is important to specify two curly braces as valid JSON, so make sure they are there.



The screenshot shows the "Test event" configuration screen. At the top, a green success bar displays the message "The test event myeventtest was successfully saved.". The main area has a "Test event" header with "Delete", "Save", and "Test" buttons. A note below the header says: "To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save changes." Under "Test event action", there are two options: "Create new event" (radio button) and "Edit saved event" (radio button, which is selected). The "Event name" field contains "myeventtest". In the "Event JSON" section, there is a code editor with the following JSON:1 []
2 { "key1": "value1",
3 "key2": "value2",
4 "key3": "value3"
5 }

```
At the top right of the "Event JSON" editor, there is a "Format JSON" button.
```

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



The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with File, Edit, Find, View, Go, Tools, Window, a Test button (which is highlighted in blue), and Deploy. Below the navigation bar, there's a search bar labeled "Go to Anything (Ctrl-P)" and a sidebar titled "Environment" which lists "lambda_function.py". The main area is titled "Execution results" and shows the following details:

- Test Event Name:** myeventtest
- Response:**

```
{  
    "statusCode": 200,  
    "body": "{\"Hello from Lambda!\""  
}
```
- Function Logs:**

```
START RequestId: 729085bb-781b-419c-91d5-bcce4eb7f774 Version: $LATEST  
END RequestId: 729085bb-781b-419c-91d5-bcce4eb7f774  
REPORT RequestId: 729085bb-781b-419c-91d5-bcce4eb7f774 Duration: 1.64 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 32 MB
```
- Request ID:** 729085bb-781b-419c-91d5-bcce4eb7f774

In the top right corner of the main area, it says "Status: Succeeded" and "Max memory used".

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.

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.

General configuration

AWS Region
US East (N. Virginia) us-east-1

Bucket type [Info](#)

General purpose
Recommended for most use cases and access patterns.
General purpose buckets are the original S3 bucket type.
They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name [Info](#)

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Format: s3://bucket/prefix

Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership

Successfully created bucket "slambdabucket"
To upload files and folders, or to configure additional bucket settings, choose [View details](#).

[View details](#) [X](#)

Amazon S3 > Buckets

Account snapshot - updated every 24 hours [All AWS Regions](#)
Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

[View Storage Lens dashboard](#)

[General purpose buckets](#) [Directory buckets](#)

General purpose buckets (2) [Info](#) [All AWS Regions](#)

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
slambdabucket	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 3, 2024, 15:13:18 (UTC+05:30)
test-2468-shravani	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 22, 2024, 20:13:56 (UTC+05:30)

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.

The screenshot shows the 'Create function' wizard. At the top, there are three options: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. Below this is a 'Basic information' section where the function name is set to 'shraimageloader'. Under 'Runtime', 'Python 3.12' is selected. Under 'Architecture', 'x86_64' is selected. A success message at the bottom states: 'Successfully created the function shraimageloader. You can now change its code and configuration. To invoke your function with a test event, choose "Test".'

The screenshot shows the 'Function overview' page for the 'shraimageloader' function. It includes tabs for 'Diagram' (selected) and 'Template'. The function name is 'shraimageloader'. There are buttons for '+ Add trigger' and '+ Add destination'. On the right, there are sections for 'Description' (empty), 'Last modified' (1 minute ago), 'Function ARN' (arn:aws:lambda:us-east-1:533162464157:function:shraimageloader), and 'Function URL' (Info). Top navigation shows 'Lambda > Functions > shraimageloader' and buttons for 'Throttle', 'Copy ARN', and 'Actions'.

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

The screenshot shows the AWS Lambda function configuration interface. At the top, a green banner indicates "Successfully updated the function shraimageloader." Below the banner, there's a button "+ Add trigger" and a "Function URL" field with a "Info" link. The main area is titled "Code source" with an "Info" link. A toolbar at the top of the code editor includes "File", "Edit", "Find", "View", "Go", "Tools", "Window", "Test" (which is selected), and "Deploy". The code editor displays the following Python code:

```

1 import json
2
3 def lambda_handler(event, context):
4     # Extract bucket name and object key from the event
5     bucket_name = event['Records'][0]['s3']['bucket']['name']
6     object_key = event['Records'][0]['s3']['object']['key']
7
8     # Log a message
9     print(f"An image has been added to the bucket {bucket_name}: {object_key}")
10
11    return {
12        'statusCode': 200,
13        'body': json.dumps('Log entry created successfully')
14    }
15

```

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).

The screenshot shows the "Add triggers" configuration page for a Lambda function. The top navigation bar shows "Lambda > Add triggers". The main section is titled "Add trigger" and contains a "Trigger configuration" section with an "Info" link. The "Bucket" dropdown is set to "S3" (aws asynchronous storage). The "Bucket" input field shows "s3/slambdabucket" with a clear button "X" and a copy button "C". The "Bucket region" is listed as "us-east-1". The "Event types" section allows selecting events, with "All object create events" selected. The "Prefix - optional" section includes a note about matching keys and a text input field.

Lambda > Functions > shraimagerloader

shraimagerloader

The trigger slambdabucket was successfully added to function shraimagerloader. The function is now receiving events from the trigger.

Function overview [Info](#)

[Diagram](#) [Template](#)

shraimagerloader

S3

[+ Add destination](#)

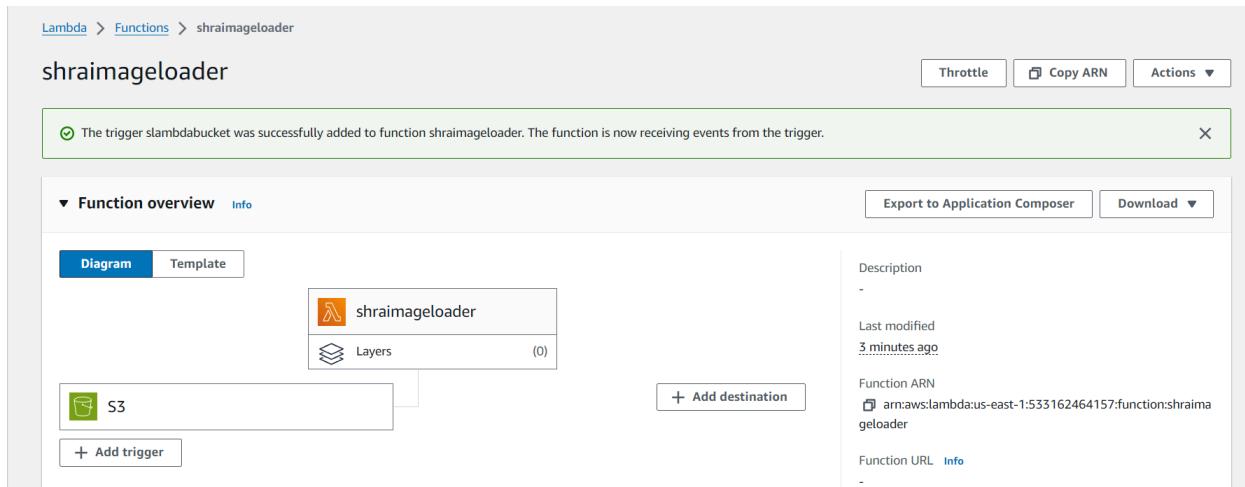
[+ Add trigger](#)

Description

Last modified 3 minutes ago

Function ARN arn:aws:lambda:us-east-1:533162464157:function:shraimagerloader

Function URL [Info](#)



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.

Amazon S3 > Buckets > slambdabucket

slambdabucket [Info](#)

[Objects](#) [Properties](#) [Permissions](#) [Metrics](#) [Management](#) [Access Points](#)

Objects (0) [Info](#)

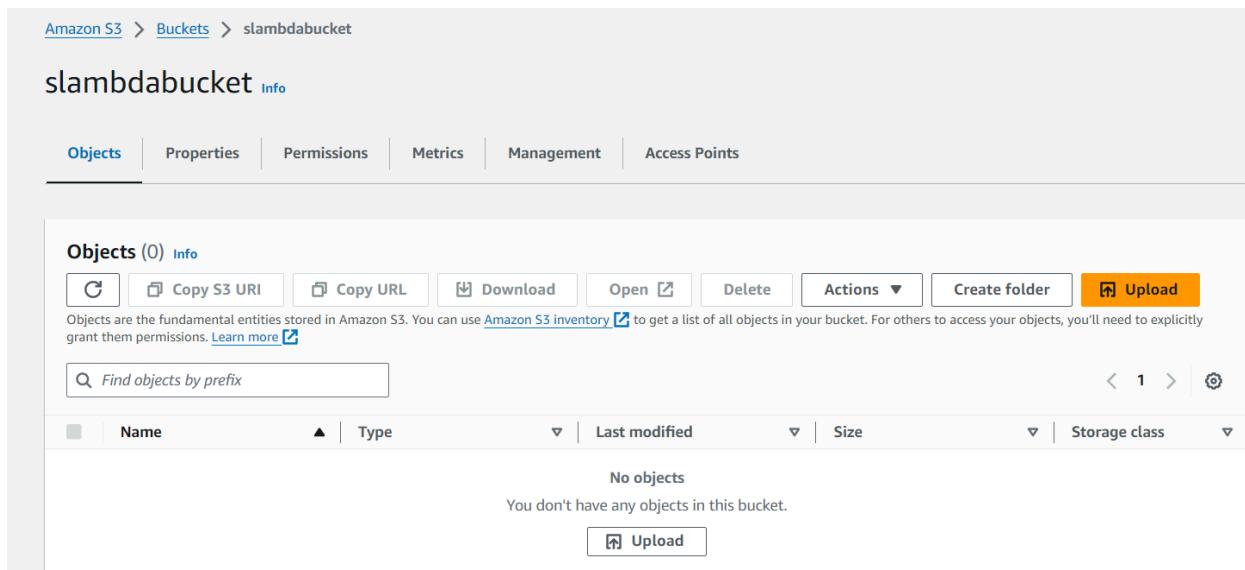
[C](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions ▾](#) [Create folder](#) [Upload](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

Name	Type	Last modified	Size	Storage class
No objects				
You don't have any objects in this bucket.				

[Upload](#)



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)

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.

<input type="checkbox"/> 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

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](#)

<input type="checkbox"/> 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: python3.12.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.

07/09/24

Advance Works - 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 servers of Amazon CloudFront, so you can watch any time.)

Amazon CloudFront is a service that speeds up distribution of static & dynamic web content to users. CloudFront delivers contents through worldwide network of data centers called edge locations.

CloudFront uses the cache for the lowest latency. If content is already in edge locations, 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 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 system, networks & infrastructure. It helps organization identify and resolve IT infrastructure problems before they impact actual business processes.

Nagios uses in e-services:

→ They are publicly available services such as HTTP, STP, SMTP etc. These services are network accessible. Services like web services, email services while private services need intermediate agents for monitoring.

Nagios uses plugins to monitor e-services many of which come pre-installed & additional plugins can be found, found online or developed by users. To monitor a service, a host must first be defined in Nagios configuration. Nagios provides alert if services failed to respond.

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

→ Overview:

BMW is a leading automotive manufacturer known for its luxury vehicles, while Netflix 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 derive the simulation to improve customer experiences by optimising their operations.

BMW's use of AWS
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 of 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 Maintenance: 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 on down load on demand, ensuring they only pay for what they use.
- Global reach - AWS's global infrastructure allows BMW to deploy applications closer to their customer's, reducing latency & improving service delivery.

Storage use of AWS

1. Content Delivery & streaming services

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

deliver high quality streaming experiences to
million of users

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

Benefits:

- Scalability: During events like IPL, when there is a massive spike, AWS allows Plotstar to scale resources dynamically to handle those spikes without compromising performance.
 - Global Content Reach: AWS enables Plotstar to distribute content across multiple regions, ensuring that users worldwide can access their services seamlessly.
- Q. Data Analytics for User Engagement
- Plotstar leverages AWS data analytics tools to gather insights about user behaviour, content preferences & viewing patterns.
- AWS Redshift & Athena - These services help Plotstar analyse large datasets to improve content recommendations & 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 to serve more targeted ads, increasing ads revenue & improving user satisfaction.

Challenging & solutions:

While both BMW & Volvo 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 certificates.

2. Cost Management - As the usage scales, managing cost can become challenging.
Solutions: Implementing AWS cost explorer & using AWS Budgets helps both companies

monitor & optimise their cloud expenditure.

Conclusion ..

Therefore, by leveraging AWS's scalability, data analysis & web 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 the containerized applications. Originally developed by Google, it has become a standard for managing cloud-native applications e.g. microservices architecture.

Advantages:-

- i. Automated Deployment & scaling: facilitates easy deployment and scaling of apps based on demand.
- ii. Self healing: automatically restarts or replaces failed containers.

iii. load balancing - distributes traffic among 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.

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 services.
- 1. Services framework: A powerful tool that simplifies deployment of serverless applications across various cloud providers such as AWS, Azure, and Google Cloud.
- 2. Serverless architecture: This design model allows developers to build applications without worrying about underlying infrastructure, enabling focus on code e.g. business logic.
- + 3. Rest API: Representational State Transfer is a architectural style for designing network applications.

steps for creating REST API for serverless framework:

1. Install serverless framework:
You start by installing serverless framework on your system 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 functions 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 & sets 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 IP'.
 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 end points API key, log streams.
- Q.2. Case study on sonarqube:
- Create your own profile in sonarqube for testing project quality
 - Use sonarcloud to analyse your github code.
 - Install sonarlint in your Java sublime IDE or eclipse IDE on eclipse IDE and analyse your java code.
 - Analyse python project with sonarqube.
 - Analyse node.js project with sonarqube.

→ Solution

- create the sonarqube profile for testing project quality.
- open Intellij setting, find tools > sonarlink setup & select to open connection wizard.
- enter 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 in 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 this connection in global setting by clicking OK.

Bind python project to sonarqube

- select Sonarlink > Bind project to sonar cloud.
- 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-service" 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 teams to efficiently deploy services in compliance with your organisations practices. Terraform cloud can also integrate with tooling system like sonar now to automatically generate new infrastructure requests.

→ self-service infrastructure model with Terraform modules:

At a large organisation, implementing a self-service 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 the

organisational standards by best practices.

key aspects of this self-service model includes:

a. Standardisation through Terraform modules
by creating and utilising Terraform modules
big -scale organisations can codify their
infrastructure deployment by management
standards. These modules serve as reusable
packages of terraform configurations that
encapsulate common patterns by best practices.

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

c. Compliance

By using pre-defined modules, teams ensure
their deployments comply with the organisations
established practices by security guidelines.

d. Automation:

The use of Terraform modules promotes automation
reducing manual intervention by potential
human errors in infrastructure management.

e. Version control: with modules stored in version control system like git, teams can track changes, collaborate on improvements & 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. Centralised Management: By centralised infrastructure management through Terraform cloud organisation & 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 of practices, large organisations can effectively leverage their own to build a robust, scalable & compliant infrastructure management system that supports both centralised control & decentralised autonomy.