

Assignment 1

Aim : To make an EC2 machine on AWS.

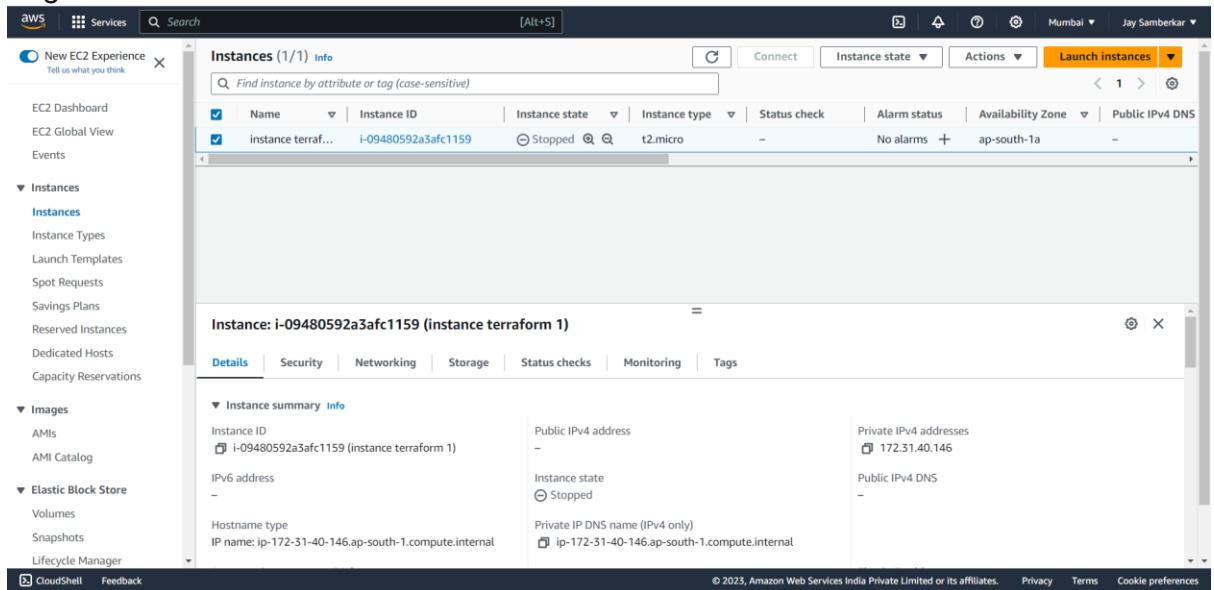
Theory :

Amazon Web Services is a widely used cloud computing platform provided by Amazon. AWS offers a vast array of cloud-based services, including computing power, storage solutions, networking, databases, machine learning, analytics, Internet of Things (IoT) and security.

Amazon Elastic Compute Cloud (Amazon EC2) provides on-demand, scalable computing capacity in the AWS Cloud. Using Amazon EC2 reduces hardware costs so you can develop and deploy applications faster. We can use Amazon EC2 to launch virtual servers according to our needs. We can add capacity (scale up) to handle compute-heavy tasks or reduce capacity based on website traffic demands.

Steps :

1. Login and Search EC2 and scroll down to Launch an Instance.



2. Click on launch instance and select the required machine and instance type (Free Tier Eligible) and then Launch the instance.

The screenshot shows the AWS Lambda console interface. A new function named "HelloWorld" is being created. The "Code" tab is selected, displaying the Lambda@Edge code. The "Handler" dropdown is set to "index.handler". The "Runtime" dropdown is set to "Node.js 14.x". The "Memory" dropdown is set to "128 MB". The "Timeout" dropdown is set to "3 seconds". The "Tracing" dropdown is set to "None". The "Environment variables" section is empty. The "Logs" section shows a log entry from the CloudWatch Logs service. The "Logs" tab is selected.

3. After the instance launches and initializes, select it and click on Connect.

The screenshot shows the AWS EC2 Instances page. A list of instances is displayed, with one instance named "jaysamberkar" selected. The "Details" tab is active, showing the instance's summary information. The instance ID is "i-00726ae995c015c9b", it is running, and its public IP address is "13.232.21.210". The "Actions" dropdown menu is open, showing options like "Stop", "Start", "Reboot", "Termination protection", and "Launch instances". The "CloudShell" and "Feedback" buttons are at the bottom left.

4. The machine will boot up to the console.

```

aws Services Search [Alt+S]
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023
[ec2-user@ip-172-31-43-14 ~]$

i-00726ae995c015c9b (jaysamberkar)
PublicIPs: 13.232.21.210 PrivateIPs: 172.31.43.14

CloudShell Feedback © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

```

5. For shutting down the instance, go back and terminate the instance from the instance state menu.

Name	Instance ID	Instance state	Instance type	Status checks
instance terraf...	i-09480592a3afc1159	Stopped	t2.micro	-
jaysamberkar	i-00726ae995c015c9b	Running	t2.micro	2/2 check

Instance: i-00726ae995c015c9b (jaysamberkar)

Details Security Networking Storage Status checks Monitoring Tags

Instance summary Info

Instance ID: i-00726ae995c015c9b (jaysamberkar)
 Public IPv4 address: 13.232.21.210 [Open address]
 Instance state: Running
 Hostname type: IP name: ip-172-31-43-14.ap-south-1.compute.internal
 Private IP DNS name (IPv4 only): ip-172-31-43-14.ap-south-1.compute.internal

Private IPv4 addresses: 172.31.43.14
 Public IPv4 DNS: ec2-13-232-21-210.ap-south-1.compute.amazonaws.com [Open address]

CloudShell Feedback © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

6. Head to Security Groups and delete the non-default group from the Actions Menu.

Successfully deleted 2 security groups

Security Groups (3/4) Info

Actions Export security groups to CSV Create security group

Name	Security group ID	Security group name	VPC ID	Description	Owner
-	sg-0011626fd7c8592be	launch-wizard-3	vpc-0785ac232d65e01d9	launch-wizard-3 create...	644557152358
-	sg-0b4ab5894d00270bf	launch-wizard-2	vpc-0785ac232d65e01d9	launch-wizard-2 create...	644557152358
-	sg-0bee767c9dec48907	default	vpc-0785ac232d65e01d9	default VPC security gr...	644557152358
-	sg-0e5d5dd9bd51bc894	launch-wizard-1	vpc-0785ac232d65e01d9	launch-wizard-1 create...	644557152358

Lab Outcome:

L01: To understand the fundamentals of cloud computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements.

Conclusion :

Hence learned and implemented the steps to create an EC2 Machine

Assignment 2

Aim : To make and launch AWS Cloud9 IDE.

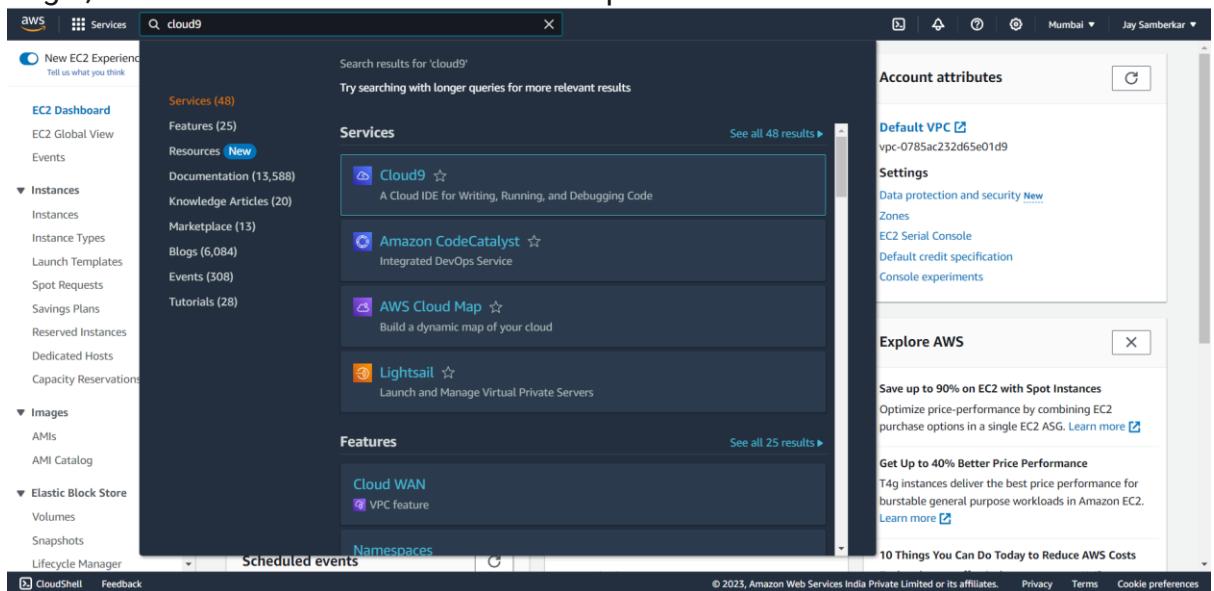
Theory :

AWS Cloud9, originally Cloud9 IDE before its acquisition by Amazon Web Services, revolutionized the development landscape by providing developers with a cloud-based integrated development environment (IDE) that facilitated coding, debugging, and running code with remarkable ease and efficiency.

The inclusion of a terminal within the IDE allowed developers to execute commands and run server-side code directly from the cloud. Beyond its accessibility and flexibility, AWS Cloud9 prioritized data security and scalability, ensuring a reliable and safe environment for businesses and individual developers seeking cutting-edge cloud-based development solutions.

Steps :

1. Login, Search Cloud9 and select the first option.



2. Fill in the name and description. Then scroll down and click on Create an Environment.

AWS Cloud9 > Environments > Create environment

Create environment info

Details

Name Limit of 60 characters, alphanumeric, and unique per user.

Description - optional Limit 200 characters.

Environment type Info
Determines what the Cloud9 IDE will run on.

New EC2 instance
Cloud9 creates an EC2 instance in your account. The configuration of your EC2 instance cannot be changed by Cloud9 after creation.

Existing compute
You have an existing instance or server that you'd like to use.

New EC2 instance

Instance type Info

CloudShell Feedback © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

3. Open the environment and wait for the Cloud9 IDE to launch

AWS Cloud9

Creating JAY. This can take several minutes. While you wait, see Best practices for using AWS Cloud9 info

Environments (1)

Name	Cloud9 IDE	Environment type	Connection	Permission	Owner ARN
JAY	Open	EC2 instance	AWS Systems Manager (SSM)	Owner	arn:aws:iam::644557152358:root

CloudShell Feedback © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

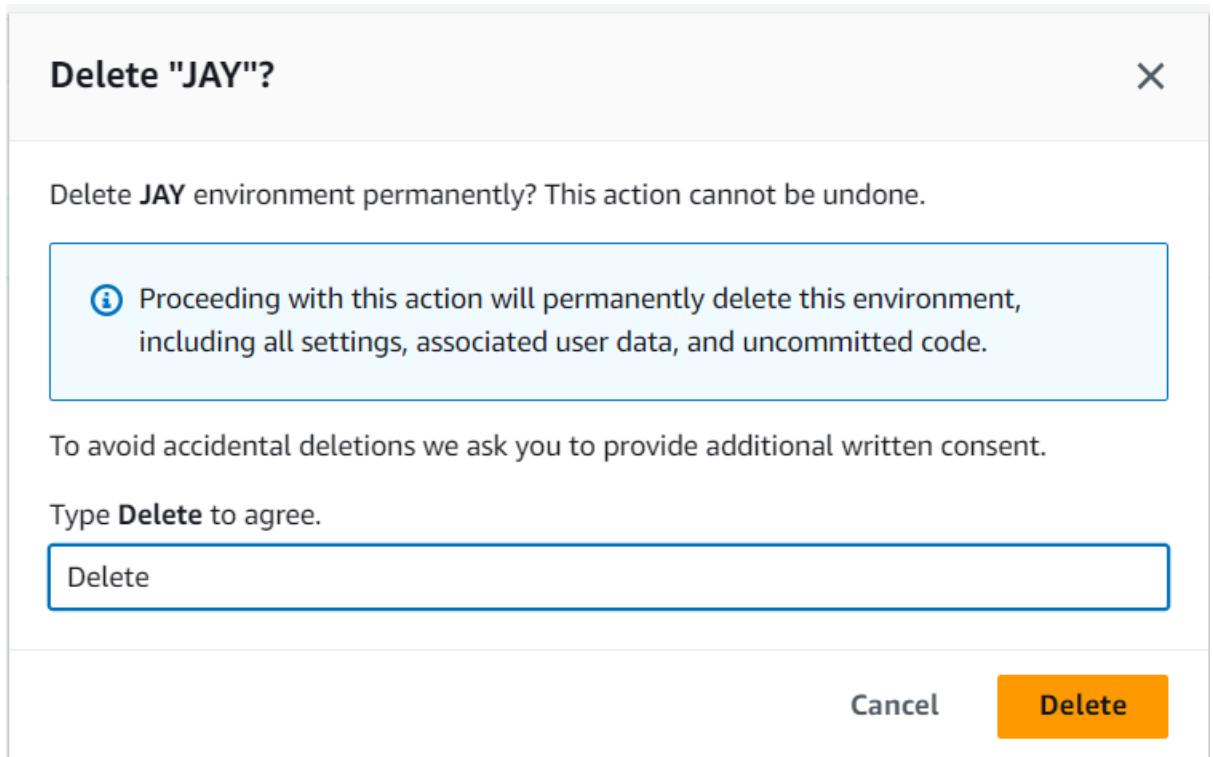
4. Create a new file, write some code in any supported language and execute it.

The screenshot shows the AWS Cloud9 IDE interface. At the top, there are three tabs: 'AWS Cloud9', 'AWS Cloud9', and 'Abhigyan - AWS Cloud9'. Below the tabs, the address bar shows 'eu-north-1.console.aws.amazon.com/cloud9/ide/97b84d16e8b441f181aeecf2a819693c?region=eu-north-1#'. The main workspace contains two tabs: 'Welcome' and 'basicC.cpp'. The 'basicC.cpp' tab displays the following C++ code:

```
1 #include <iostream>
2 using namespace std;
3
4 // main() is where program execution begins.
5 int main() {
6     cout << "Hello World";
7     return 0;
8 }
```

Below the code editor is a terminal window titled 'bash - [ip-172-31-46-6.eu x Immediate x basicC.cpp - Stopped x]'. It shows the output of the 'basicC.cpp' program: 'Hello World'. The status bar at the bottom right indicates '8:2 C and C++ Spaces: 4'.

5. Go back to Dashboard, then click on the delete button, type in Delete and Delete the environment.



6. Lastly, head over to your EC2 instances and terminate the instance related to your cloud9 environment.

The screenshot shows the AWS EC2 Instances page. At the top, there's a header with the AWS logo, a search bar, and navigation links for 'Mumbai' and 'Jay Samberkar'. Below the header, a message says 'New EC2 Experience Tell us what you think' with a close button. On the left, a sidebar has links for 'EC2 Dashboard', 'EC2 Global View', and 'Events'. The main content area is titled 'Instances (1/3) info' with a sub-header 'Find instance by attribute or tag (case-sensitive)'. A table lists one instance: 'aws-cloud9-JA...' (Instance ID: i-097deb93971c8749a), which is 'Terminated' (Status check: -), 't2.micro' (Instance type), and located in 'ap-south-1b' (Availability Zone). There are buttons for 'Connect', 'Actions', and 'Launch instances'.

Lab Outcome:
L01, L02 mapped.

Conclusion :

We understood the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launched AWS Cloud9 IDE and performed Collaboration Demonstration.

ASSIGNMENT No. – 3

Aim: To build your application using AWS CodeBuild and Deploy on S3/SEBS using AWS CodePipeline.

LO Mapped: LO1, LO2

Theory:

Amazon Simple Storage Service (Amazon S3) is an object storage service offering industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can store and protect any amount of data for virtually any use case, such as data lakes, cloud-native applications, and mobile apps. With cost-effective storage classes and easy-to-use management features, you can optimize costs, organize data, and configure fine-tuned access controls to meet specific business, organizational, and compliance requirements.

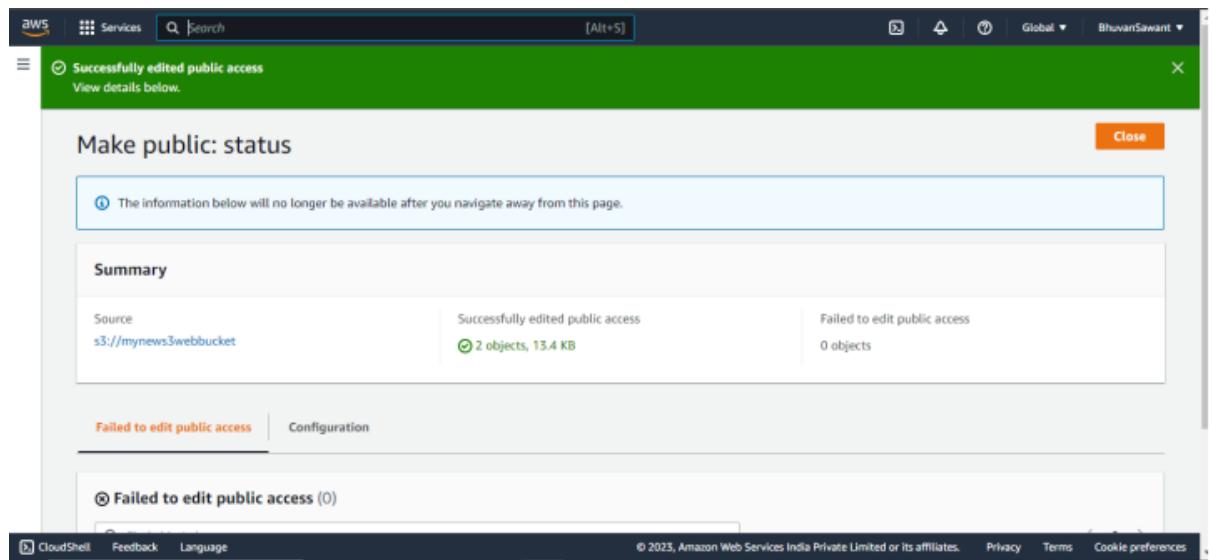
Some of the benefits of AWS S3 are:

- Durability: S3 provides 99.999999999 percent durability.
- Low cost: S3 lets you store data in a range of "storage classes." These classes are based on the frequency and immediacy you require in accessing files.
- Scalability: S3 charges you only for what resources you actually use, and there are no hidden fees or overage charges. You can scale your storage resources to easily meet your organization's ever-changing demands.
- Availability: S3 offers 99.99 percent availability of objects
- Security: S3 offers an impressive range of access management tools and encryption features that provide top-notch security.
- Flexibility: S3 is ideal for a wide range of uses like data storage, data backup, software delivery, data archiving, disaster recovery, website hosting, mobile applications, IoT devices, and much more.
- Simple data transfer: You don't have to be an IT genius to execute data transfers on S3. The service revolves around simplicity and ease of use.

First login to your AWS account.

The screenshot shows the AWS CloudFront console. At the top, a green banner indicates "Upload succeeded" with a link to "View details below". Below this, a summary table shows the destination as "s3://mynewsS3bucket", status as "Succeeded", and metrics for "2 files, 13.4 KB (100.00%)". A "Failed" section shows "0 files, 0 B (0%)". Below the summary is a "Files and folders" section with a table showing two files: "welcome.jpg" (image/jpeg, 13.1 KB, Status: Succeeded) and "index.html" (text/html, 279.0 B, Status: Succeeded). The table has columns for Name, Folder, Type, Size, Status, and Error. At the bottom of the page, there are links for CloudShell, Feedback, Language, and a footer with copyright information and links for Privacy, Terms, and Cookie preferences.

Then create a new S3 bucket.



The screenshot shows a success message: "Successfully edited public access". Below it, a summary table shows the status of public access edits:

Source	Successfully edited public access	Failed to edit public access
s3://mynews3webbucket	2 objects, 13.4 KB	0 objects

Below the table, there are tabs for "Failed to edit public access" (selected) and "Configuration". At the bottom, there are links for CloudShell, Feedback, Language, and a footer with copyright information.

After that host a static web page using the S3 bucket.



Website for AWS

Conclusion: In this assignment we learnt about AWS S3 bucket and hosted a static web page using the S3 bucket.

Assignment Number: 4

Name: Soham Satam Branch: IT/V Roll No.:109

Date:10/09/2023

Aim: To study AWS Code Pipeline and deploy web application using Code Pipeline.

LO mapped: LO1, LO2

Theory:

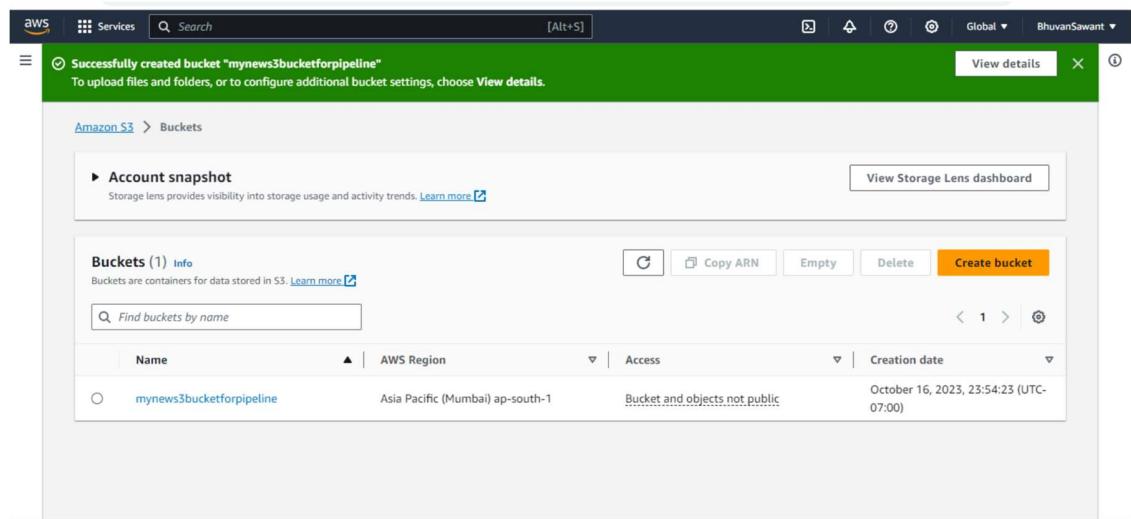
AWS CodePipeline is a continuous integration and continuous delivery (CI/CD) service provided by Amazon Web Services (AWS).

The key aspects of AWS CodePipeline:

Overview:

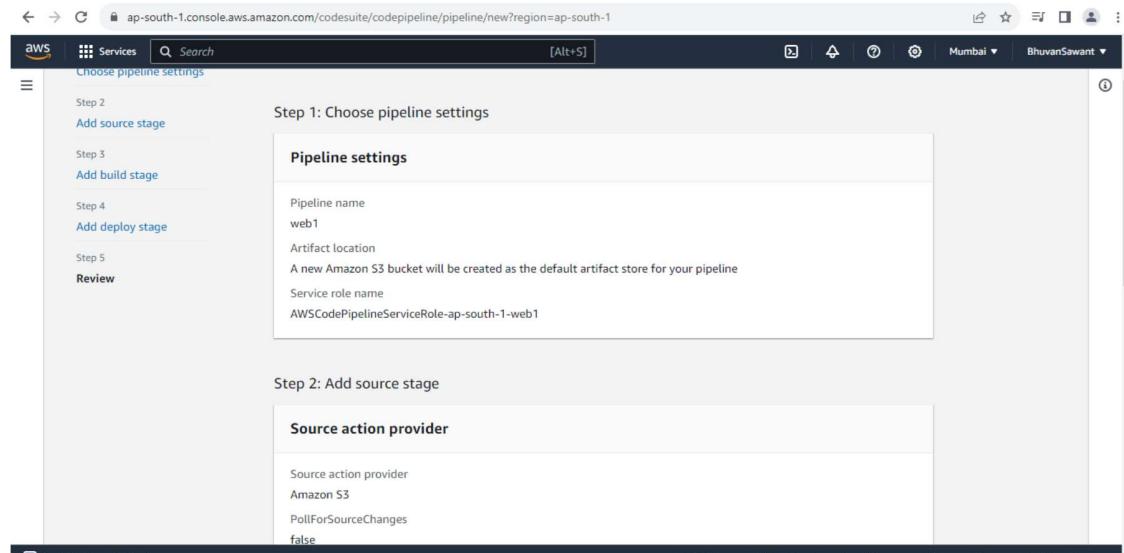
1. CI/CD Workflow:

- AWS CodePipeline facilitates the automation of the build, test, and deployment phases of the release process. It allows you to define a series of stages, each of which can represent a phase in your release pipeline.



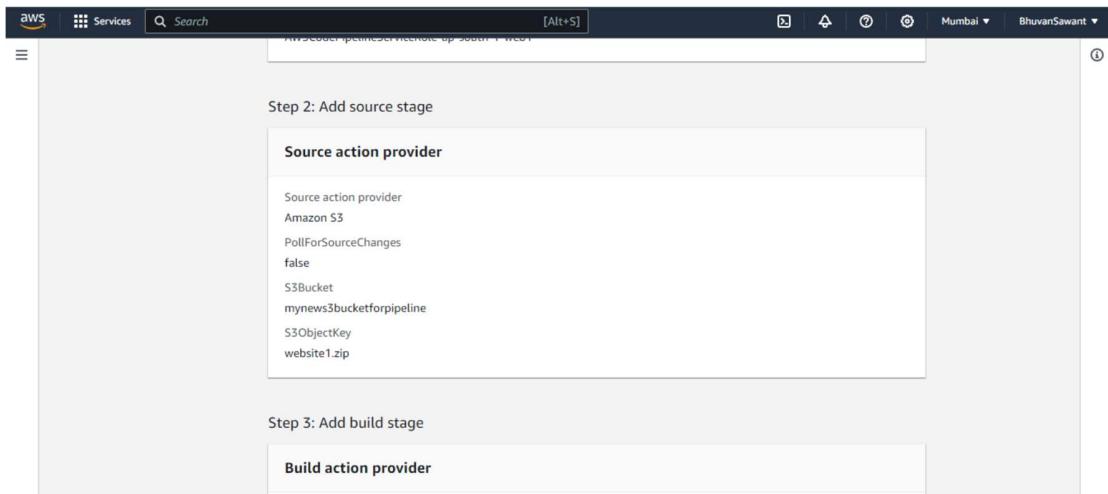
2. Integration with Other AWS Services:

- CodePipeline integrates with various AWS services, such as AWS CodeBuild for building applications, AWS CodeDeploy for automating deployments, and AWS Lambda for running custom actions.



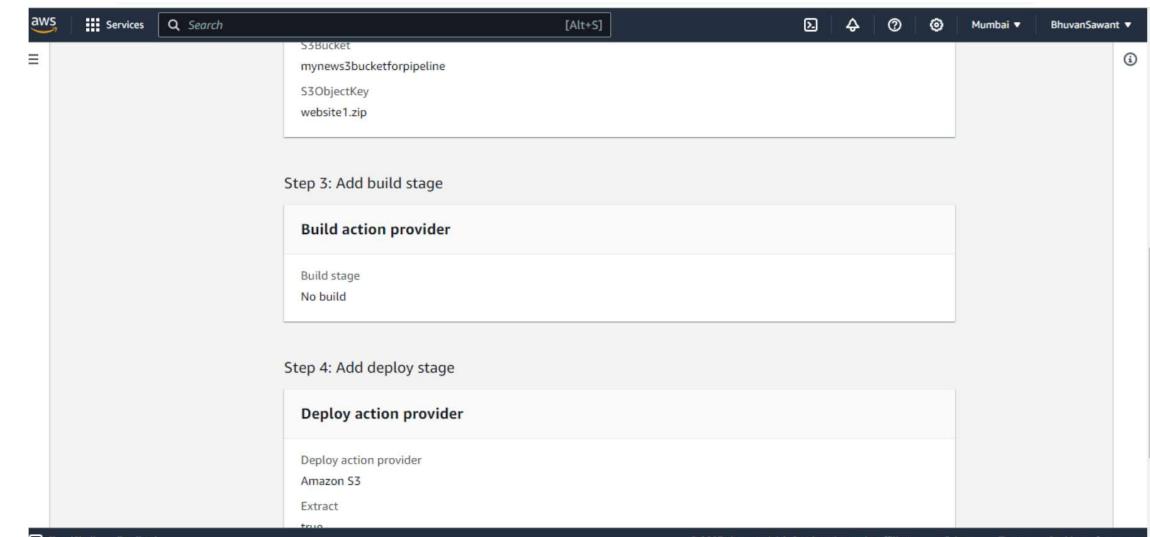
3. Pipeline Execution:

- Pipelines consist of a series of stages, and each stage can have one or more actions. Actions represent a task, such as source code retrieval or deployment to a specific environment.



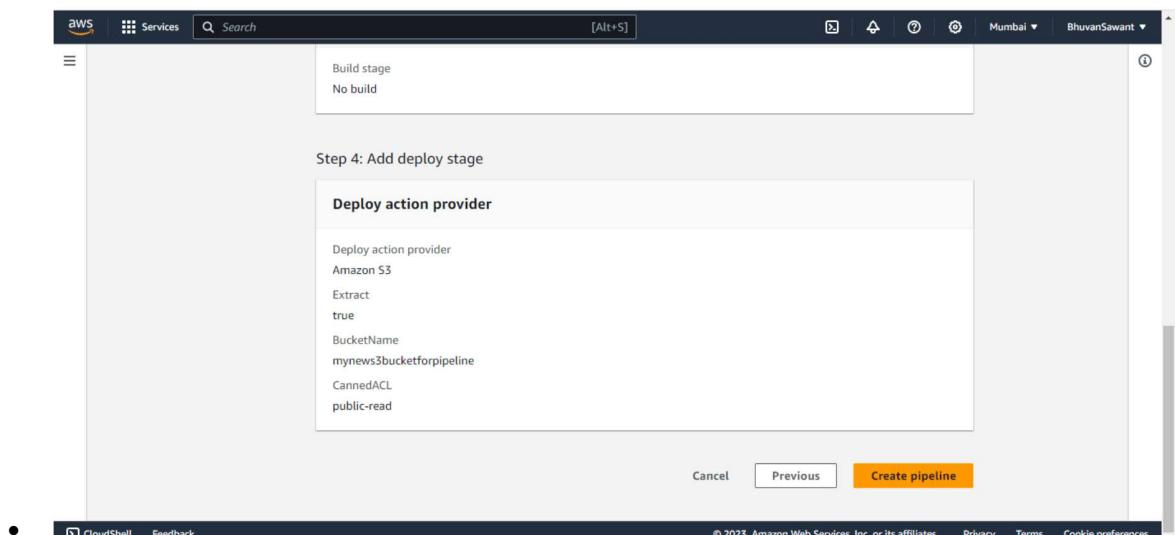
4. Source Providers:

- CodePipeline supports integration with various source code repositories, including AWS CodeCommit, GitHub, and Amazon S3.



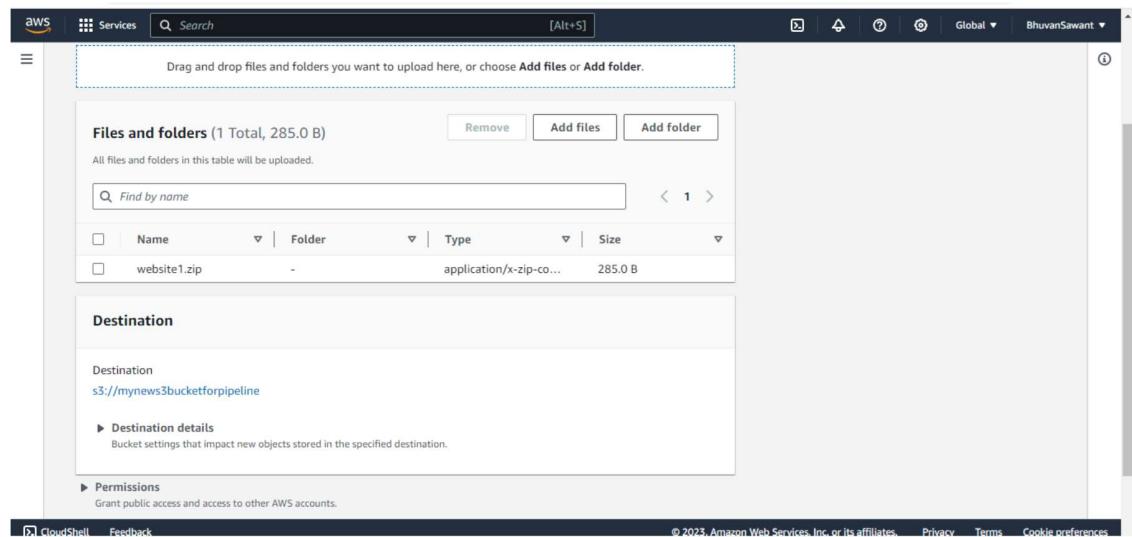
5. Artifact Management:

- CodePipeline uses artifacts to store the files and data needed for each action in a pipeline. Artifacts can be passed between stages to ensure consistency in the deployment process.



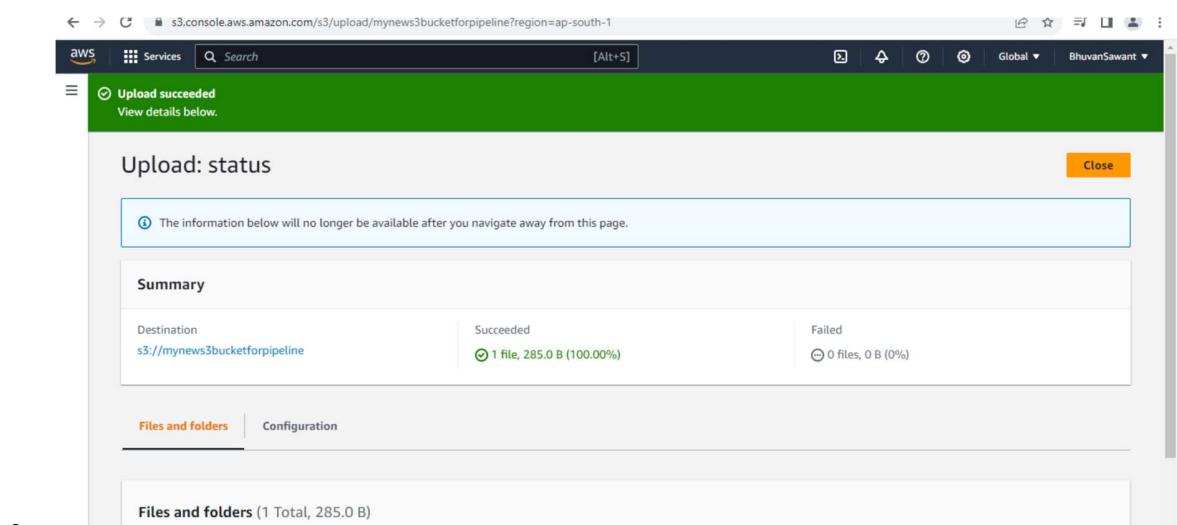
6. Integration with Third-Party Tools:

- Besides AWS services, CodePipeline supports integration with third-party tools. This is achieved through custom actions, which allow you to use external tools and scripts in your pipeline.



7. Pipeline Visualizations:

- CodePipeline provides a visual representation of your release process, making it easy to understand and monitor the status of each stage and action.



Key Concepts:

1. Pipeline:

- A pipeline is a series of stages that represents your release process. Each stage can contain one or more actions.

2. Stage:

- A stage is a logical unit in a pipeline, representing a phase in the release process. Stages are executed sequentially.

3. Action:

- An action represents a task within a stage. Actions can include tasks such as building code, deploying to a test environment, or running tests.

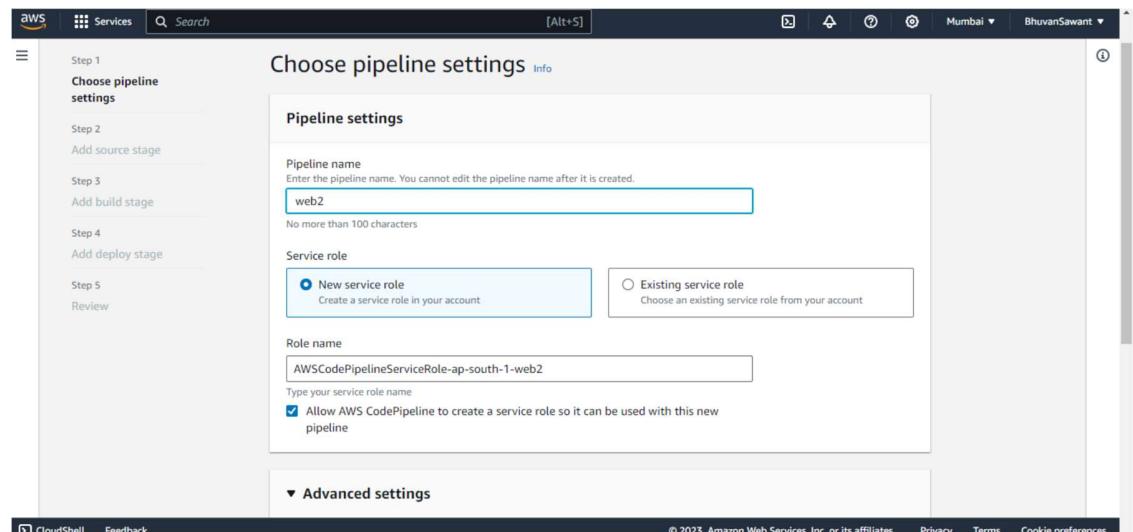
4. Artifact:

- Artifacts are the files and data that are produced as a result of an action. They are used to pass information between stages in a pipeline.

To deploy web application using CodePipeline here are the following steps to be followed:

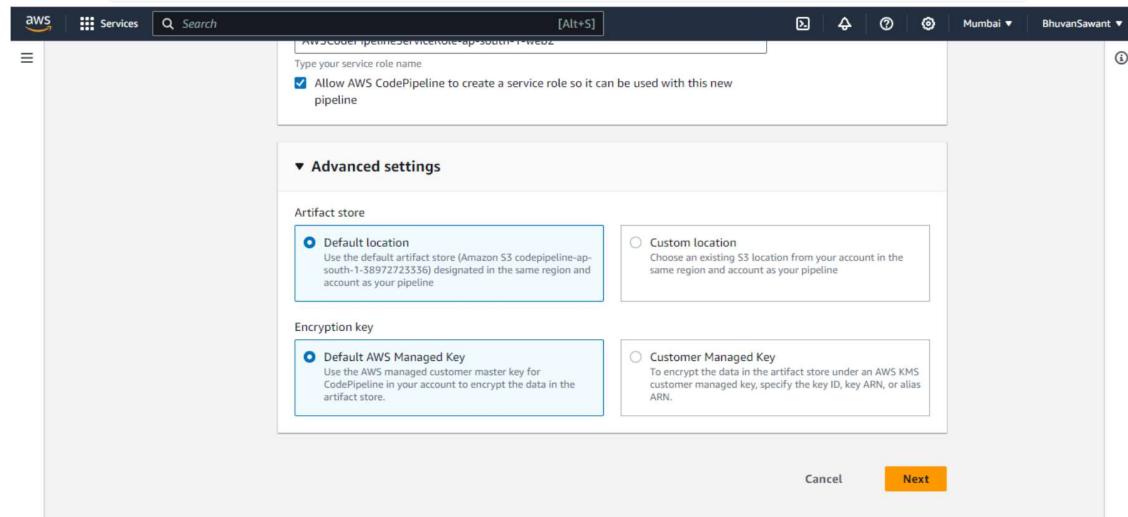
1. Set Up Source Stage:

- Configure a source stage in AWS CodePipeline, linking to your version control system (e.g., CodeCommit, GitHub).



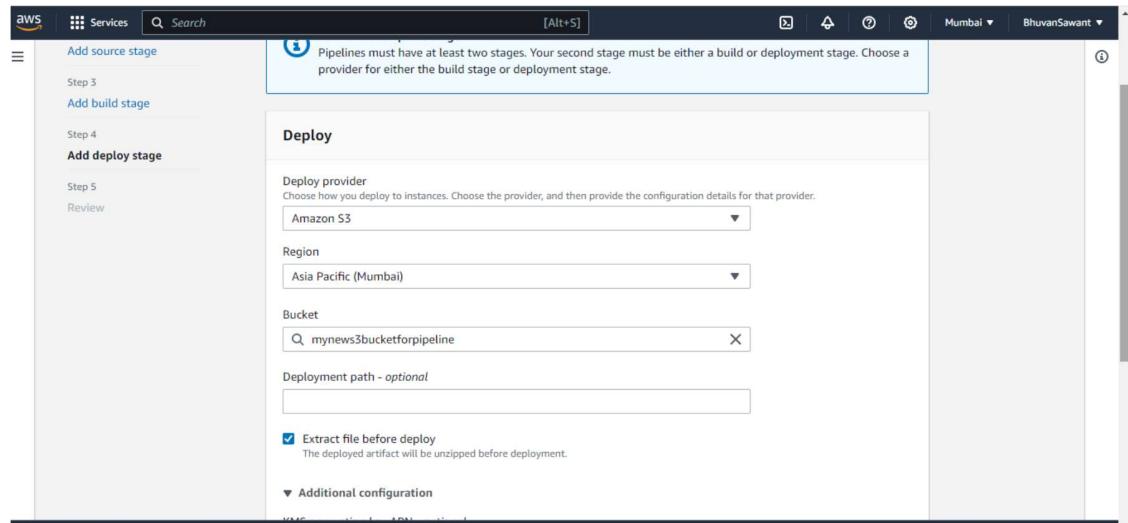
2. Configure Build Stage:

- Set up a build stage using AWS CodeBuild to compile, test, and package your web application.



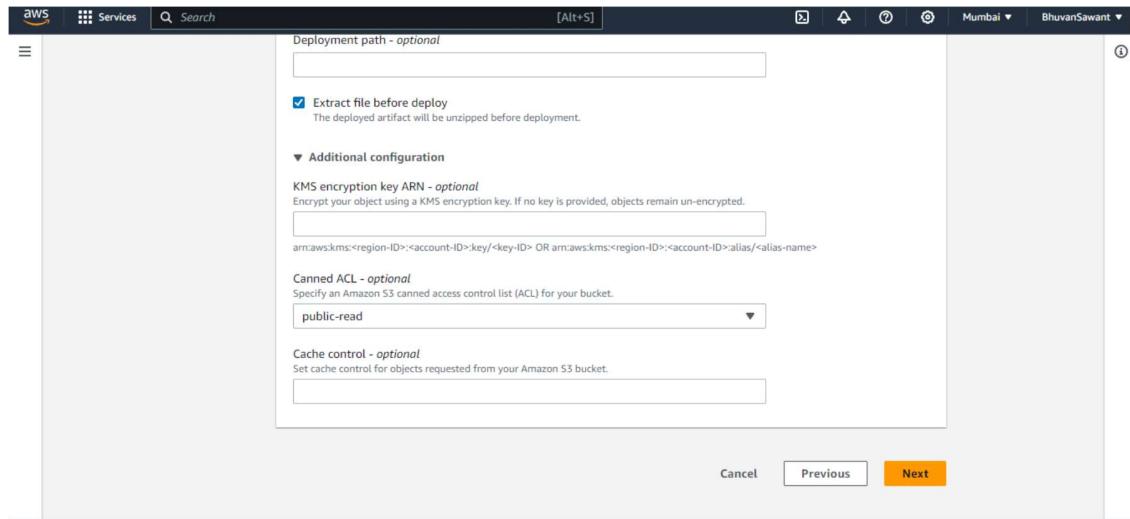
3. Define Deployment Stage:

- Create a deployment stage using AWS CodeDeploy or another deployment provider to deploy your application to target environments.



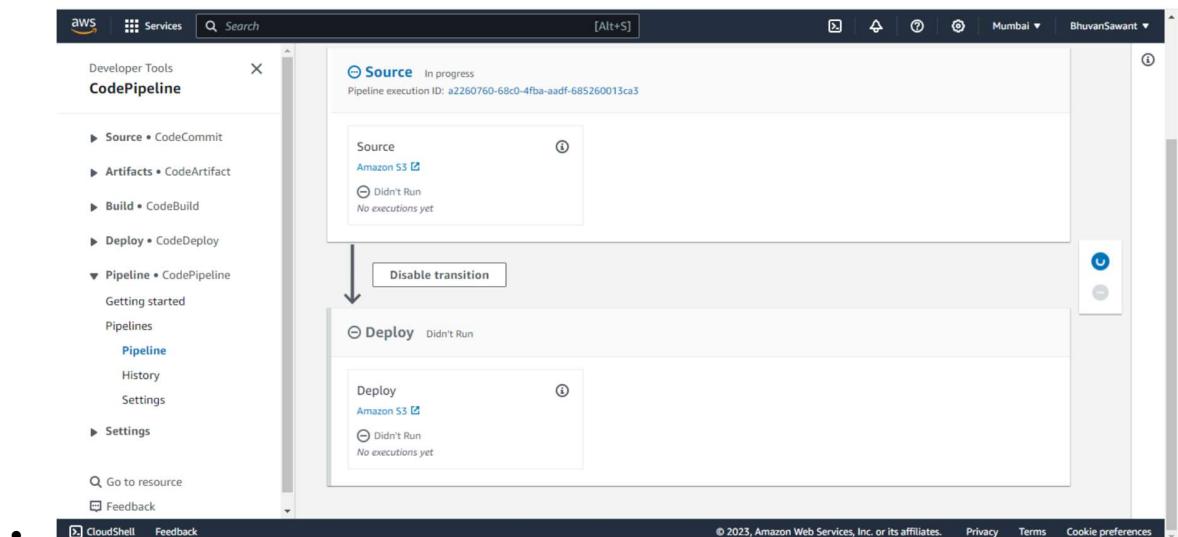
4. Configure Approval (Optional):

- Optionally, add a manual approval stage to review and approve deployments before proceeding to the next stage.



5. Artifact Passing:

- Ensure proper passing of artifacts between stages to maintain consistency in the deployment process.



6. Add Monitoring (Optional):

- Integrate monitoring tools (e.g., AWS CloudWatch) to track the performance and health of your application during and after deployment.

7. Configure Notifications (Optional):

- Set up notifications using AWS SNS or other services to receive alerts about pipeline events and status changes.

8. Test and Validate:

- Test the pipeline by triggering a build, ensuring that each stage executes successfully, and the application deploys as expected.

9. Modify Pipeline as Needed:

- Make adjustments to the pipeline configuration based on the specific requirements of your web application and deployment process.

10. Continuous Improvement:

- Implement continuous improvement practices, such as monitoring feedback, optimizing build and deployment scripts, and iterating on the pipeline structure.

Conclusion: By this assignment we learned how to host static web page through codepipeline.

ASSIGNMENT 5

Aim – To learn basic idea of Kubernetes.

Theory-

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

The name Kubernetes originates from Greek, meaning helmsman or pilot. K8s as an abbreviation results from counting the eight letters between the "K" and the "s". Google open-sourced the Kubernetes project in 2014. Kubernetes combines over 15 years of Google's experience running production workloads at scale with best-of-breed ideas and practices from the community.

Let's take a look at why Kubernetes is so useful by going back in time.

Traditional deployment era: Early on, organizations ran applications on physical servers. There was no way to define resource boundaries for applications in a physical server, and this caused resource allocation issues. For example, if multiple applications run on a physical server, there can be instances where one application would take up most of the resources, and as a result, the other applications would underperform. A solution for this would be to run each application on a different physical server. But this did not scale as resources were underutilized, and it was expensive for organizations to maintain many physical servers.

Virtualized deployment era: As a solution, virtualization was introduced. It allows you to run multiple Virtual Machines (VMs) on a single physical server's CPU. Virtualization allows applications to be isolated between VMs and provides a level of security as the information of one application cannot be freely accessed by another application.

Virtualization allows better utilization of resources in a physical server and allows better scalability because an application can be added or updated easily, reduces hardware costs, and much more. With virtualization you can present a set of physical resources as a cluster of disposable virtual machines. Each VM is a full machine running all the components, including its own operating system, on top of the virtualized hardware.

Container deployment era: Containers are similar to VMs, but they have relaxed isolation properties to share the Operating System (OS) among the applications. Therefore, containers are considered lightweight. Similar to a VM,

a container has its own filesystem, share of CPU, memory, process space, and more. As they are decoupled from the underlying infrastructure, they are portable across clouds and OS distributions.

Why you need Kubernetes and what it can do

Containers are a good way to bundle and run your applications. In a production environment, you need to manage the containers that run the applications and ensure that there is no downtime. For example, if a container goes down, another container needs to start. Wouldn't it be easier if this behavior was handled by a system?

That's how Kubernetes comes to the rescue! Kubernetes provides you with a framework to run distributed systems resiliently. It takes care of scaling and failover for your application, provides deployment patterns, and more. For example, Kubernetes can easily manage a canary deployment for your system. Kubernetes provides you with:

- **Service discovery and load balancing** Kubernetes can expose a container using the DNS name or using their own IP address. If traffic to a container is high, Kubernetes is able to load balance and distribute the network traffic so that the deployment is stable.
- **Storage orchestration** Kubernetes allows you to automatically mount a storage system of your choice, such as local storages, public cloud providers, and more.
- **Automated rollouts and rollbacks** You can describe the desired state for your deployed containers using Kubernetes, and it can change the actual state to the desired state at a controlled rate. For example, you can automate Kubernetes to create new containers for your deployment, remove existing containers and adopt all their resources to the new container.
- **Automatic bin packing** You provide Kubernetes with a cluster of nodes that it can use to run containerized tasks. You tell Kubernetes how much CPU and memory (RAM) each container needs. Kubernetes can fit containers onto your nodes to make the best use of your resources.
- **Self-healing** Kubernetes restarts containers that fail, replaces containers, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.
- **Secret and configuration management** Kubernetes lets you store and manage sensitive information, such as passwords, OAuth tokens, and SSH keys. You can deploy and update secrets and application configuration without rebuilding your container images, and without exposing secrets in your stack configuration.

Understanding Kubernetes objects

Kubernetes objects are persistent entities in the Kubernetes system. Kubernetes uses these entities to represent the state of your cluster. Specifically, they can describe:

What containerized applications are running (and on which nodes)

The resources available to those applications

The policies around how those applications behave, such as restart policies, upgrades, and fault-tolerance

A Kubernetes object is a "record of intent"--once you create the object, the Kubernetes system will constantly work to ensure that object exists. By creating an object, you're effectively telling the Kubernetes system what you want your cluster's workload to look like; this is your cluster's desired state.

To work with Kubernetes objects--whether to create, modify, or delete them--you'll need to use the Kubernetes API. When you use the kubectl command-line interface, for example, the CLI makes the necessary Kubernetes API calls for you. You can also use the Kubernetes API directly in your own programs using one of the Client Libraries.

Conclusion – Hence learned what is Kubernetes.

Assignment 6

Aim : To understand terraform lifecycle, core concepts/ terminologies and install it.

Theory :

Terraform, developed by HashiCorp, is an open-source infrastructure as code (IaC) software tool that allows users to define and provision data center infrastructure using a declarative configuration language. In simpler terms, it lets you codify your infrastructure, enabling consistent and reproducible deployments. It is used for defining, provisioning, and managing cloud infrastructure using a declarative language.

Core Concepts:

Providers: Terraform uses providers to interact with cloud services. Examples include AWS, Azure, Google Cloud, and many others. Each provider offers resource types that can be managed.

Resources: These are the primary components in Terraform. A resource might be a physical component such as an EC2 instance in AWS or a database in Azure.

State: Terraform maintains a state file that maps real-world resources to your configuration. This state is used to determine what Terraform will do on the next apply.

Modules: These are containers for multiple resources that are used together. They provide a way to group resources, and they can be used to create reusable infrastructure components.

Variables and Outputs: Variables allow for parameterization of the Terraform configuration, making it more dynamic and flexible. Outputs are a way to get information about the infrastructure, like IP addresses or DNS names.

Provisioners: While not always recommended due to their imperative nature, provisioners can be used to model specific actions on the local machine or on a remote machine, like executing scripts.

Steps :

1. Login and Search IAM on AWS Console and select the first option.

The screenshot shows the AWS IAM Dashboard. On the left, a sidebar lists various IAM management options like Access management, Access reports, and Identity providers. The main area displays 'Security recommendations' with a red notification badge (1). It shows a warning about adding MFA for the root user and a note that the root user has no active access keys. Below this is a summary of IAM resources: User groups (1), Users (1), Roles (8), Policies (1), and Identity providers (0). A 'What's new' section provides updates for features in IAM. To the right, there's a panel for the 'AWS Account' (Account ID: 644557152358, Account Alias: Create) and a 'Quick Links' section for managing security credentials.

2. Head to users and Click on Create User

The screenshot shows the 'Specify user details' step of the IAM User creation wizard. It asks for a 'User name' (which must be unique) and provides an optional checkbox for 'Provide user access to the AWS Management Console'. A note below explains that you can generate access keys for programmatic access. At the bottom, there are 'Cancel' and 'Next Step' buttons.

3. Enter a new username and click on next

4. Click on create group

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

- Add user to group Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
- Copy permissions Copy all group memberships, attached managed policies, and inline policies from an existing user.
- Attach policies directly Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

User groups (1)

Group name	Users	Attached policies	Created
group1	0	AdministratorAccess	2023-08-22 (1 month ago)

Set permissions boundary - optional

Cancel Previous Next

5. Select the first Policy will full access and enter a group name and create the group.

Permissions policies (1/885)

Policy name	Type	Use...	Description
<input checked="" type="checkbox"/> AdministratorAccess	AWS managed	Permis...	Provides full access to AWS services
<input type="checkbox"/> AdministratorAcc...	AWS managed	None	Grants account administrative perm
<input type="checkbox"/> AdministratorAcc...	AWS managed	None	Grants account administrative perm
<input type="checkbox"/> AlexaForBusinessD...	AWS managed	None	Provide device setup access to Alex
<input type="checkbox"/> AlexaForBusinessF...	AWS managed	None	Grants full access to AlexaForBusin
<input type="checkbox"/> AlexaForBusinessG...	AWS managed	None	Provide gateway execution access t
<input type="checkbox"/> AlexaForBusinessL...	AWS managed	None	Provide access to Lifesize AVS devic
<input type="checkbox"/> AlexaForBusinessP...	AWS managed	None	Provide access to Poly AVS devices
<input type="checkbox"/> AlexaForBusinessR...	AWS managed	None	Provide read only access to AlexaFc

Create user group

6. Then click on next, select the policy and create user.

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Permissions options

Add user to group
Copy permissions
Attach policies directly

User groups (2)			
Group name ▾	Users	Attached policies ▾	Created
group1	0	AdministratorAccess	2023-08-22 (1 month ago)
terraform@1	0	AdministratorAccess	2023-10-16 (Now)

7. Then create an access key with CLI.

Step 1
Access key best practices & alternatives

Step 2 - optional
Set description tag

Step 3
Retrieve access keys

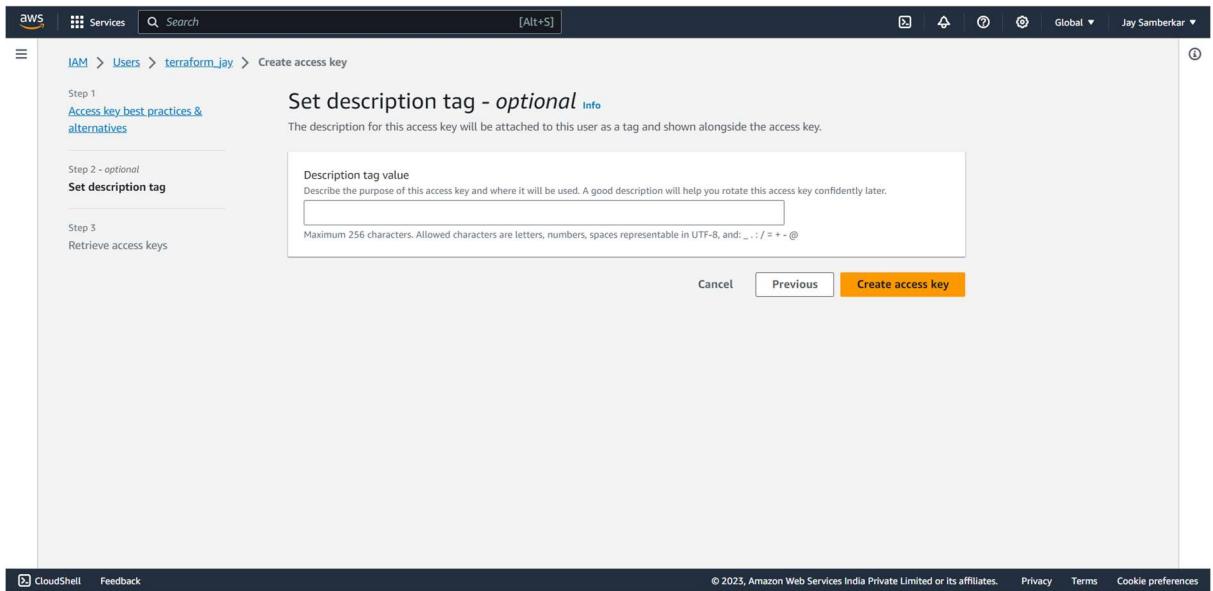
Access key best practices & alternatives Info

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

Command Line Interface (CLI)
Local code
Application running on an AWS compute service
Third-party service
Application running outside AWS
Other

8. Leave the description field empty and create a key. Download the access key in the .csv file.



9. Install terraform.

```
Last login: Sat Aug 19 17:44:56 on ttys000
> brew tap hashicorp/tap
Running `brew update --auto-update`...
Installing from the API is now the default behaviour!
You can save space and time by running:
  brew untap homebrew/core
  brew untap homebrew/cask
==> Auto-updated Homebrew!
Updated 3 taps (mongodb/brew, homebrew/core and homebrew/cask).
==> New Formulae
arm-none-eabi-binutils          medusa
arm-none-eabi-gcc                mjml
arm-none-eabi-gdb                mongodb/brew/mongodb-community@6.0
asnmap                           mongodb/brew/mongodb-enterprise@6.0
cargo-auditable                  mongodb/brew/mongodb-mongocryptd@6.0
cdi                             mysql-client@8.0
cloudlist                        mysql@8.0
coder                           ollama
ctpv                            proxyf
czkawka                         python-certifi
dnsrobocert                     riff
dolphie                         riscv64-elf-binutils
ebook2cw                        riscv64-elf-gcc
go@1.20                          riscv64-elf-gdb
img2pdf                         rpmspectool
```

10. Create the terraform config file main.tf with required configurations.

```

> nano main.tf
> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 4.16"...
- Installing hashicorp/aws v4.67.0...
- Installed hashicorp/aws v4.67.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

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

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
~/terraform_scripts ➜  ✅ 25s

```

11. Run the command `terraform plan` and enter yes

```

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

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

Terraform will perform the following actions:

# aws_instance.Ubuntu will be created
+ resource "aws_instance" "Ubuntu" {
    + ami                                = "ami-0f5ee92e2d63afc18"
    + arn                                = (known after apply)
    + associate_public_ip_address        = (known after apply)
    + availability_zone                  = (known after apply)
    + cpu_core_count                     = (known after apply)
    + cpu_threads_per_core              = (known after apply)
    + disable_api_stop                  = (known after apply)
    + disable_api_termination           = (known after apply)
    + ebs_optimized                     = (known after apply)
    + get_password_data                = false
    + host_id                           = (known after apply)
    + host_resource_group_arn           = (known after apply)
}
```

12. Check EC2 instances on AWS, you'll find an instance running started by terraform.

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with links like EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instances Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, and Images. The main area has a title 'Instances (1) Info' with a 'C' button, 'Connect' button, 'Instance state' dropdown, 'Actions' dropdown, and a 'Launch instances' button. Below that is a search bar with 'Find instance by attribute or tag (case-sensitive)' and a 'Clear filters' button. A table lists one instance: Name (empty), Instance ID (i-018414f145733e4f7), Instance state (Running), Instance type (t2.micro), Status check (Initializing), Alarm status (No alarms), Availability Zone (ap-south-1a), and Public IP (ec2-...). At the bottom, a modal window says 'Select an instance'.

13. Now run terraform destroy to terminate the instance.

```

- enable_resource_name_dns_a_record      = false -> null
- enable_resource_name_dns_aaaa_record = false -> null
- hostname_type                         = "ip-name" -> null
}

- root_block_device {
- delete_on_termination = true -> null
- device_name          = "/dev/sda1" -> null
- encrypted            = false -> null
- iops                 = 100 -> null
- tags                 = {} -> null
- throughput           = 0 -> null
- volume_id             = "vol-07a64df7d00e4e7a1" -> null
- volume_size           = 8 -> null
- volume_type           = "gp2" -> null
}
}

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

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

Enter a value: yes

```

Lab Outcome:

L01, L06 mapped.

Conclusion :

We understood the workings of Terraform to create and manage EC2 instances in AWS which provides a highly automated and reproducible infrastructure-as-code solution. The ability to effortlessly spin up and tear down instances not only enhances operational agility but also ensures optimal resource utilization.

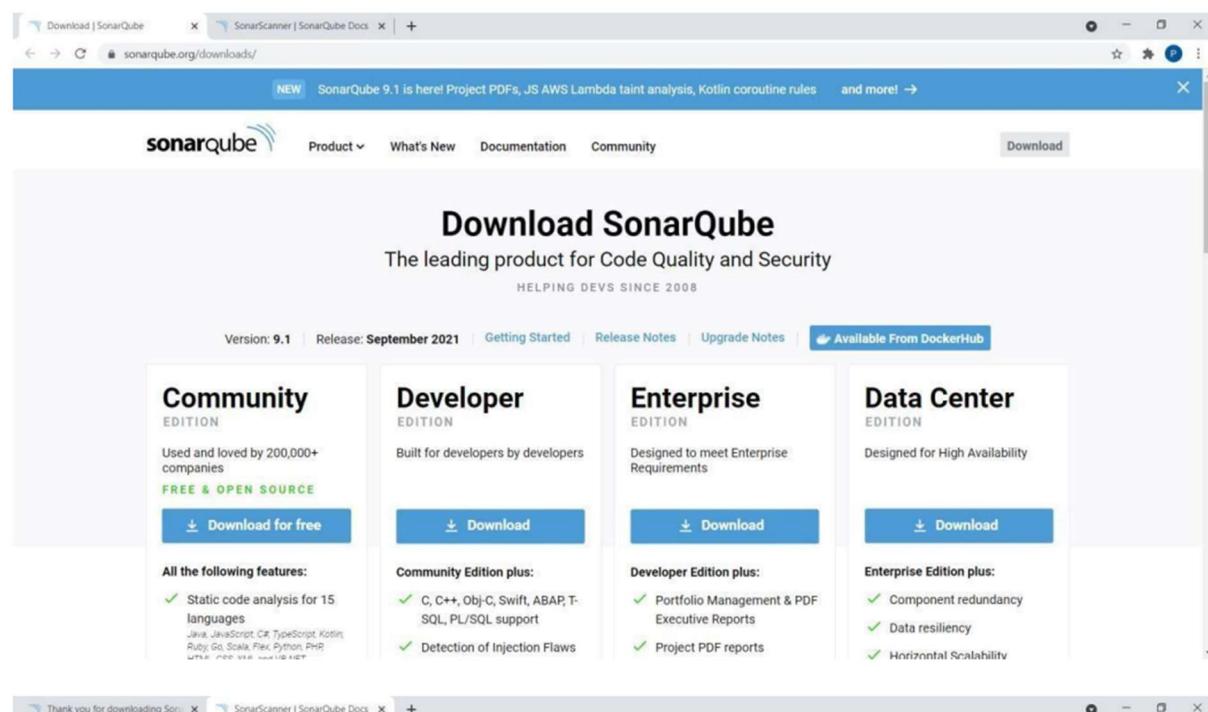
Assignment 7

Aim: TO perform static analysis on python programs using SonarCube SAST processes

LO Mapped: L04

Theory:

Download and Sonar Scanner



After downloading, set Environment Variables. Add "sonarqube-9.1.0.47736\bin" to Path.

Open command prompt. Run commands:

- cd "sonarqube-9.1.0.47736\bin\windows-x86-64"

- StartSonar.bat

```

Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Priyanshi>cd "C:\Users\Priyanshi\Downloads\sonarqube-9.1.0.47736\bin\windows-x86-64">StartSonar.bat

wrapper | --> Wrapper Started as Console
wrapper | Launching a JVM...
jvm 1 | Wrapper (Version 3.2.3) http://wrapper.tanukisoftware.org
jvm 1 | Copyright 1999-2006 Tanuki Software, Inc. All Rights Reserved.
jvm 1 |
jvm 1 | 2021-09-29 13:50:37 INFO app[]|{o.s.a.AppFileSystem} Cleaning or creating temp directory C:\Users\Priyanshi\Downloads\sonarqube-9.1.0.47736\temp
jvm 1 | 2021-09-29 13:50:37 INFO app[]|{o.s.a.es.EsSettings} Elasticsearch listening on [HTTP: 127.0.0.1:9001, TCP: 127.0.0.1:50355]
jvm 1 | [2021-09-29 13:50:37 INFO app[]|{o.s.a.ProcessLauncherImpl} Launch process[[key=es, ipcIndex=1, logfilenamePrefix=es]] from [C:\Users\Priyanshi\Downloads\sonarqube-9.1.0.47736\temp -XX:ErrorFile=-./logs/es_hs_err_pid.log -Des.networkAddress.cache.ttl=60 -Des.networkAddress.cache.size=256 -XX:HeadDumpOnOutOfMemoryError -Delasticsearch.path.home=C:\Users\Priyanshi\Downloads\sonarqube-9.1.0.47736\elasticsearch -Des.path.conf=C:\Users\Priyanshi\Downloads\sonarqube-9.1.0.47736\conf\conf]
jvm 1 | 2021-09-29 13:50:37 INFO app[]|{o.s.a.s.SchedulerImpl} Waiting for Elasticsearch to be up and running
jvm 1 | 2021-09-29 13:50:39 ERROR app[]|{o.s.a.s.EsManagedProcess} Failed to check status
jvm 1 | org.elasticsearch.ElasticsearchException: java.util.concurrent.ExecutionException: java.net.ConnectException: Timeout connecting to [/127.0.0.1:9001]
jvm 1 | at org.elasticsearch.client.RestHighLevelClient.performClientRequest(RestHighLevelClient.java:2078)
jvm 1 | at org.elasticsearch.client.RestHighLevelClient.internalPerformRequest(RestHighLevelClient.java:1732)
jvm 1 | at org.elasticsearch.client.RestHighLevelClient.performRequest(RestHighLevelClient.java:1702)
jvm 1 | at org.elasticsearch.client.RethrowableClient.performRequestAndParseEntity(RestHighLevelClient.java:1672)
jvm 1 | at org.elasticsearch.client.ClusterClient.health(ClusterClient.java:119)
jvm 1 | at org.sonar.application.executor.EsConnectorImpl.getClusterStatus(EsConnectorImpl.java:54)
jvm 1 | at org.sonar.application.executor.EsConnectorImpl.getStatus(esManagedProcess.java:90)
jvm 1 | at org.sonar.application.processor.EsManagedProcess.checkOperational(esManagedProcess.java:75)
jvm 1 | at org.sonar.application.processor.EsManagedProcess.isOperational(esManagedProcess.java:60)
jvm 1 | at org.sonar.application.processor.ManagedProcessHandler.refreshState(ManagedProcessHandler.java:220)
jvm 1 | Caused by: java.util.concurrent.ExecutionException: java.net.ConnectException: Timeout connecting to [/127.0.0.1:9001]
jvm 1 | at org.elasticsearch.common.util.concurrent.BaseFuture$Sync.getValue(BaseFuture.java:62)
jvm 1 | at org.elasticsearch.common.util.concurrent.BaseFuture$Sync.get(BaseFuture.java:249)
jvm 1 | at org.elasticsearch.common.util.concurrent.BaseFuture.get(BaseFuture.java:76)
jvm 1 | at org.elasticsearch.client.RestHighLevelClient.performClientRequest(RestHighLevelClient.java:2075)
jvm 1 | ... 10 common frames omitted
jvm 1 | Caused by: java.net.ConnectException: Timeout connecting to [/127.0.0.1:9001]
jvm 1 | at org.apache.http.nio.pool.RouteSpecificPool.timeout(RouteSpecificPool.java:169)
jvm 1 | at org.apache.http.nio.pool.AbstractNIOConnPool.requestTimeout(AbstractNIOConnPool.java:628)
jvm 1 | at org.apache.http.nio.pool.AbstractNIOConnPool$InternalSessionRequest$Callback.timeout(AbstractNIOConnPool.java:894)
jvm 1 | at org.apache.http.impl.nio.reactor.SessionImpl.timeout(SessionRequestImpl.java:184)
jvm 1 | at org.apache.http.impl.nio.reactor.DefaultConnectingIOReactor.processTimeouts(DefaultConnectingIOReactor.java:214)
jvm 1 | at org.apache.http.impl.nio.reactor.DefaultConnectingIOReactor.execute(DefaultConnectingIOReactor.java:158)
jvm 1 | at org.apache.http.impl.nio.reactor.DefaultConnectingIOReactor.execute(DefaultConnectingIOReactor.java:155)
jvm 1 | at org.apache.http.impl.nio.client.CloseableHttpSyncClientConnectionManager.execute(PoolingHttpSyncClientConnectionManager.java:221)
jvm 1 | at org.apache.http.impl.nio.client.CloseableHttpSyncClientBase$1.run(CloseableHttpSyncClientBase.java:64)
jvm 1 | at java.base/java.lang.Thread.run(Thread.java:834)

```

```

at org.elasticsearch.client.RestHighLevelClient.performRequest(RestHighLevelClient.java:1702)
at org.elasticsearch.client.RestHighLevelClient.performRequestAndParseEntity(RestHighLevelClient.java:1672)
at org.elasticsearch.client.ClusterClient.health(ClusterClient.java:119)
at org.sonar.application.process.EsManagedProcess.checkStatus(EsManagedProcess.java:64)
at org.sonar.application.process.EsManagedProcess.checkOperational(EsManagedProcess.java:75)
at org.sonar.application.process.EsManagedProcess.isOperational(EsManagedProcess.java:66)
at org.sonar.application.process.ManagedProcessHandler.refreshState(ManagedProcessHandler.java:220)
at org.sonar.application.process.ManagedProcessHandler$EventMatcher.run(ManagedProcessHandler.java:285)
Caused by: java.util.concurrent.ExecutionException: java.net.ConnectException: Timeout connecting to [/127.0.0.1:9001]
at org.elasticsearch.common.util.concurrent.BaseFuture$Sync.getValue(BaseFuture.java:262)
at org.elasticsearch.common.util.concurrent.BaseFuture$Sync.get(BaseFuture.java:249)
at org.elasticsearch.common.util.concurrent.BaseFuture.get(BaseFuture.java:76)
at org.elasticsearch.client.RestHighLevelClient.performClientRequest(RestHighLevelClient.java:2075)
... 10 common frames omitted
Caused by: java.net.ConnectException: Timeout connecting to [/127.0.0.1:9001]
at org.apache.http.nio.pool.RouteSpecificPool.timeout(RouteSpecificPool.java:169)
at org.apache.http.nio.AbstractNIOConnPool.requestTimeout(AbstractNIOConnPool.java:628)
at org.apache.http.nio.pool.AbstractNIOConnPool$InternalSessionRequestCallback.timeout(AbstractNIOConnPool.java:894)
at org.apache.http.impl.nio.reactor.SessionRequestImpl.java:184)
at org.apache.http.impl.nio.reactor.DefaultConnectingIOReactor.processTimeouts(DefaultConnectingIOReactor.java:214)
at org.apache.http.impl.nio.reactor.DefaultConnectingIOReactor.processEvents(DefaultConnectingIOReactor.java:158)
at org.apache.http.impl.nio.reactor.AbstractMultiworkerIOReactor.execute(AbstractMultiworkerIOReactor.java:351)
at org.apache.http.impl.nio.reactor.PoolingHttpConnectionManager.execute(PoolingHttpConnectionManager.java:221)
at org.apache.http.impl.nio.client.CloseableHttpClient$Base$1.run(CloseableHttpClient$Base$1.java:67)
at java.base/java.lang.Thread.run(Thread.java:83)
2021.09.29 13:50:50 INFO app[[o.s.a.ProcessLauncherImpl] Process(es) is up
2021.09.29 13:50:50 INFO app[[o.s.a.ProcessLauncherImpl] Launch process[[key='web', ipcIndex=2, logFilenamePrefix=web]] from [C:\Users\Priyansi\Downloads\sonarqube-9.1.0.47736]: C:\Program Files\Java\jdk-11.0.12\bin\java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djava.io.tmpdir=C:\Users\Priyansi\Downloads\sonarqube-9.1.0.47736\temp -XX:+OmitStackTraceInFastThrow --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.rmi=sun.rmi.transport=ALL-UNNAMED --add-exports=java.base/jdk.internal.ref=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/sun.nio.ch=ALL-UNNAMED --add-opens=java.management=sun.management=ALL-UNNAMED --add-opens=jdk.management/com.sun.management.internal=ALL-UNNAMED -Xms512m -Xmx512m -XX:MaxHeapSize=16g -Dhttp.nonProxyHosts=localhost|127.*|[::1] -cp ..\lib\sonar-application-9.1.0.47736.jar;C:\Users\Priyansi\Downloads\sonarqube-9.1.0.47736\lib\jdbc\H2\h2-1.4.199.jar;org.sonar.server.webserver.C:\Users\Priyansi\Downloads\sonarqube-9.1.0.47736\temp\sq-process77945169172441019properties
[2021.09.29 13:51:42 INFO app[[o.s.a.ProcessLauncherImpl] Process[web] is up
[2021.09.29 13:51:42 INFO app[[o.s.a.ProcessLauncherImpl] Launch process[[key='ce', ipcIndex=3, logFilenamePrefix=ce]] from [C:\Users\Priyansi\Downloads\sonarqube-9.1.0.47736]: C:\Program Files\Java\jdk-11.0.12\bin\java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djava.io.tmpdir=C:\Users\Priyansi\Downloads\sonarqube-9.1.0.47736\temp -XX:+OmitStackTraceInFastThrow --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/sun.nio.ch=ALL-UNNAMED --add-opens=java.management=sun.management=ALL-UNNAMED --add-opens=jdk.internal.ref=ALL-UNNAMED -Xms512m -Xmx512m -XX:+HeapDumpOnOutOfMemoryError -Dhttp.nonProxyHosts=localhost|127.*|[::1] -cp ..\lib\sonar-application-9.1.0.47736.jar;C:\Users\Priyansi\Downloads\sonarqube-9.1.0.47736\lib\jdbc\H2\h2-1.4.199.jar;org.sonar.ce.app.CeServer C:\Users\Priyansi\Downloads\sonarqube-9.1.0.47736\temp\sq-process394448714319503.properties
[2021.09.29 13:51:42 WARN app[[o.s.a.SchedulerImpl] Process[ce] is up
[2021.09.29 13:51:42 WARN app[[o.s.a.SchedulerImpl] [start-up] Default Administrator credentials are still being used. Make sure to change the password or deactivate the account.
[2021.09.29 13:51:42 WARN app[[o.s.a.SchedulerImpl] [start-up] -----
[2021.09.29 13:51:46 INFO app[[o.s.a.SchedulerImpl] Process[ce] is up
[2021.09.29 13:51:46 INFO app[[o.s.a.SchedulerImpl] SonarQube is up

```

Open another command prompt. Run command:

e cd "sonar-scanner-4.6.2.2472-windows\bin"

sonar-scanner

```

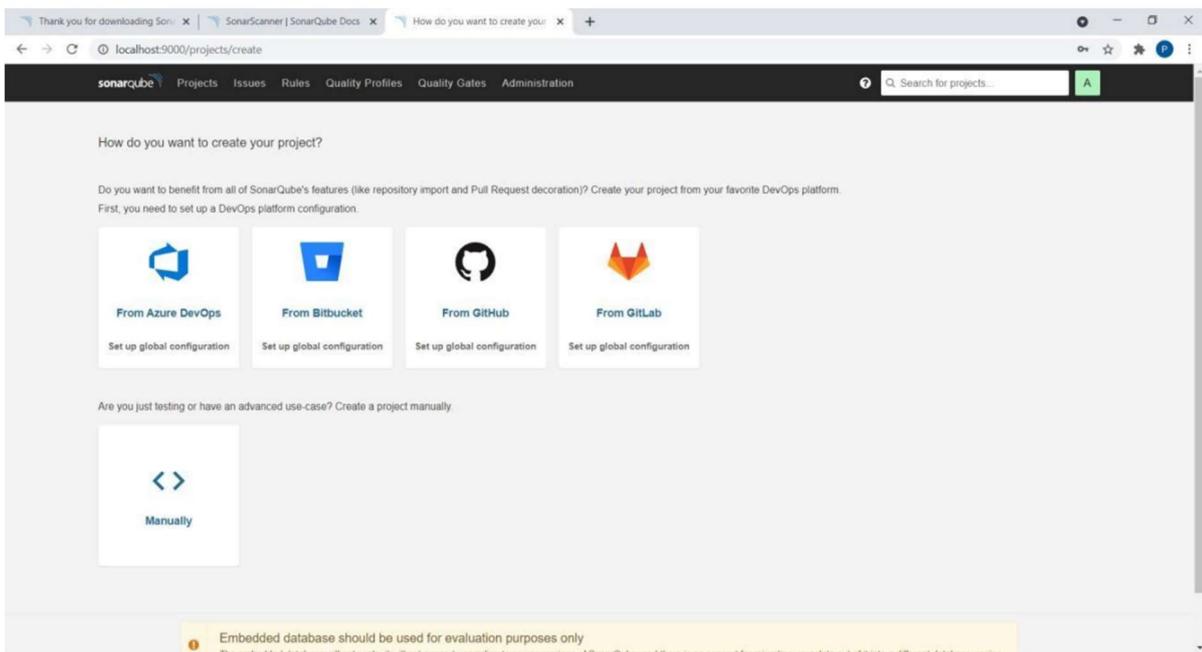
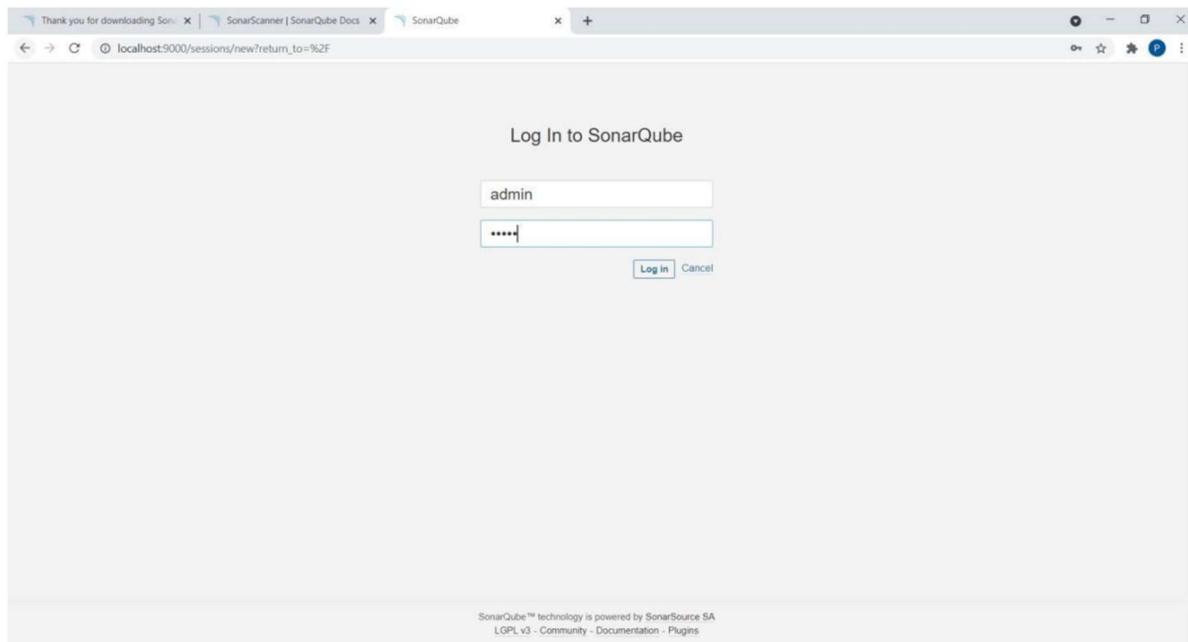
C:\Users\Priyansi\Downloads\sonar-scanner-4.6.2.2472-windows\bin>sonar-scanner
INFO: Scanner configuration file: C:\Users\Priyansi\Downloads\sonar-scanner-4.6.2.2472-windows\bin..\conf\sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: SonarScanner 4.6.2.2472
INFO: Java 11.0.11 AdoptOpenJDK (64-bit)
INFO: Windows 10 10.0 amd64
INFO: User cache: C:\Users\Priyansi\.sonar\cache
INFO: Scanner configuration file: C:\Users\Priyansi\Downloads\sonar-scanner-4.6.2.2472-windows\bin..\conf\sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: Analyzing on SonarQube server 9.1.0
INFO: Default locale: "en_IN", source code encoding: "windows-1252" (analysis is platform dependent)
INFO: Load global settings
INFO: -----
INFO: EXECUTION FAILURE
INFO: -----
INFO: Total time: 3.958s
INFO: Final Memory: 5M/20M
INFO: -----
ERROR: Error during SonarScanner execution
ERROR: Not authorized. Analyzing this project requires authentication. Please provide a user token in sonar.login or other credentials in sonar.login and sonar.password.
ERROR:
ERROR: Re-run SonarScanner using the -X switch to enable full debug logging.

C:\Users\Priyansi\Downloads\sonar-scanner-4.6.2.2472-windows\bin>

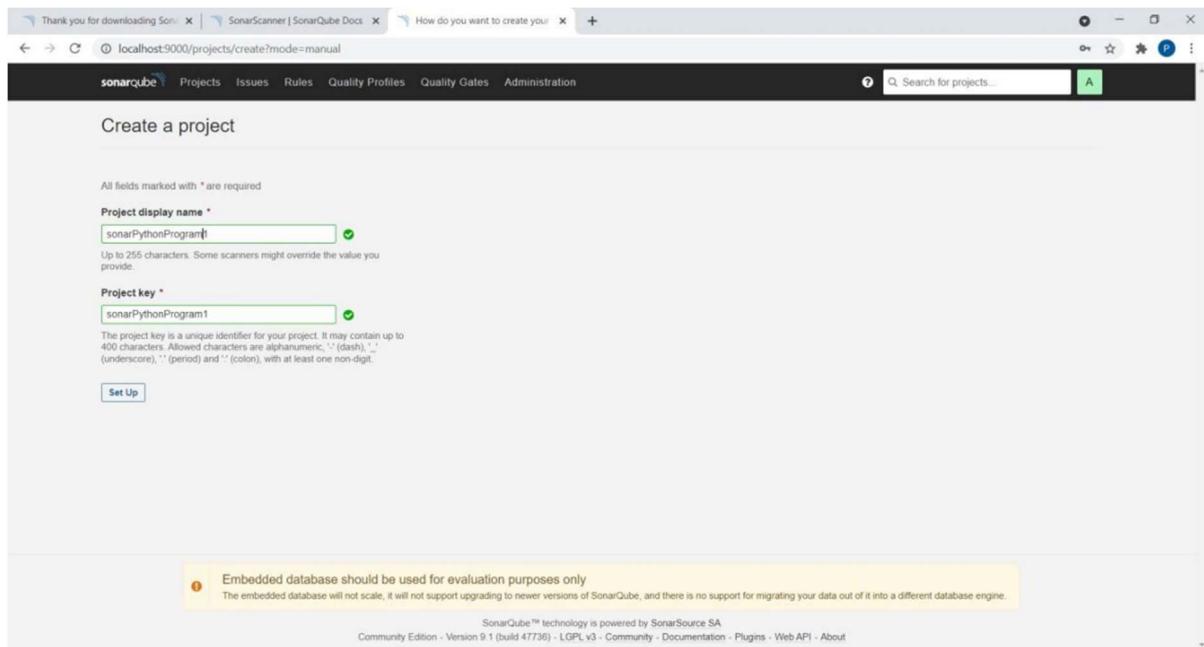
```

Server up and running on localhost:9000

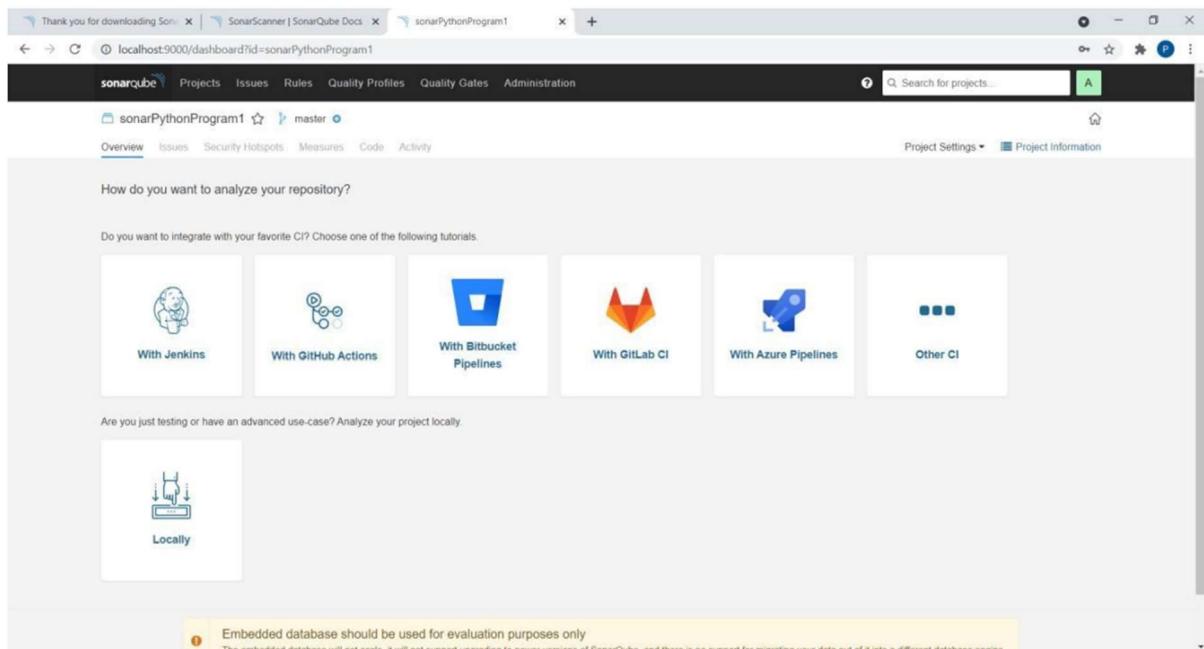
Login using credentials as User: admin and Password: admin and Set a new password



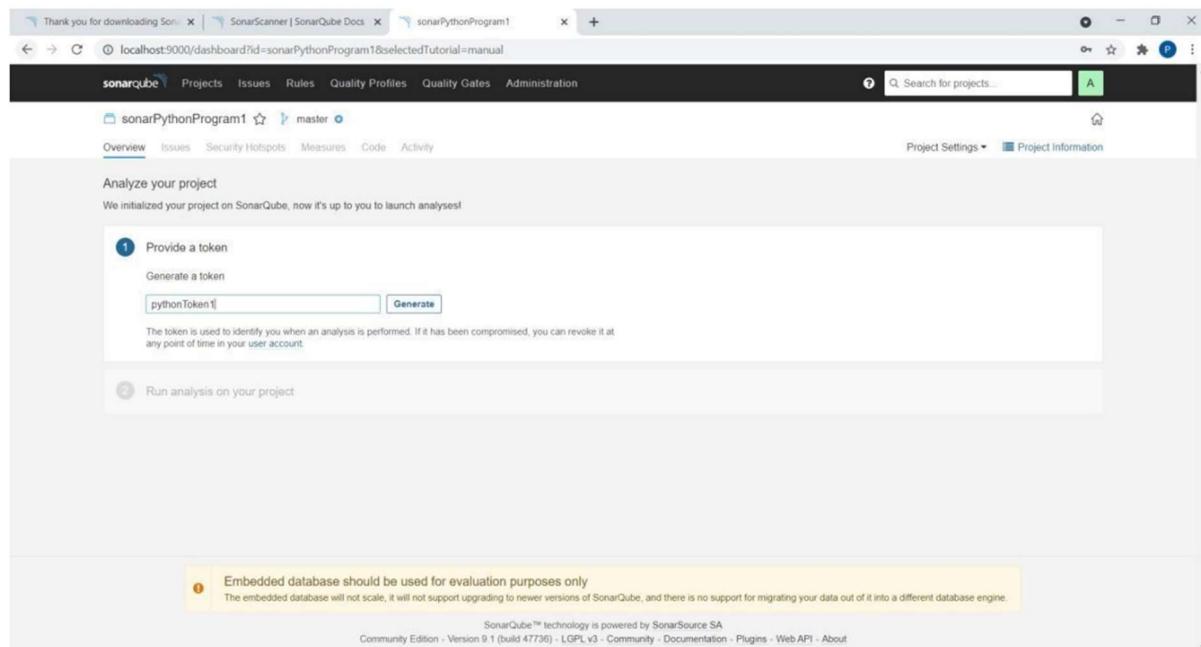
Click on Create a project Manually.



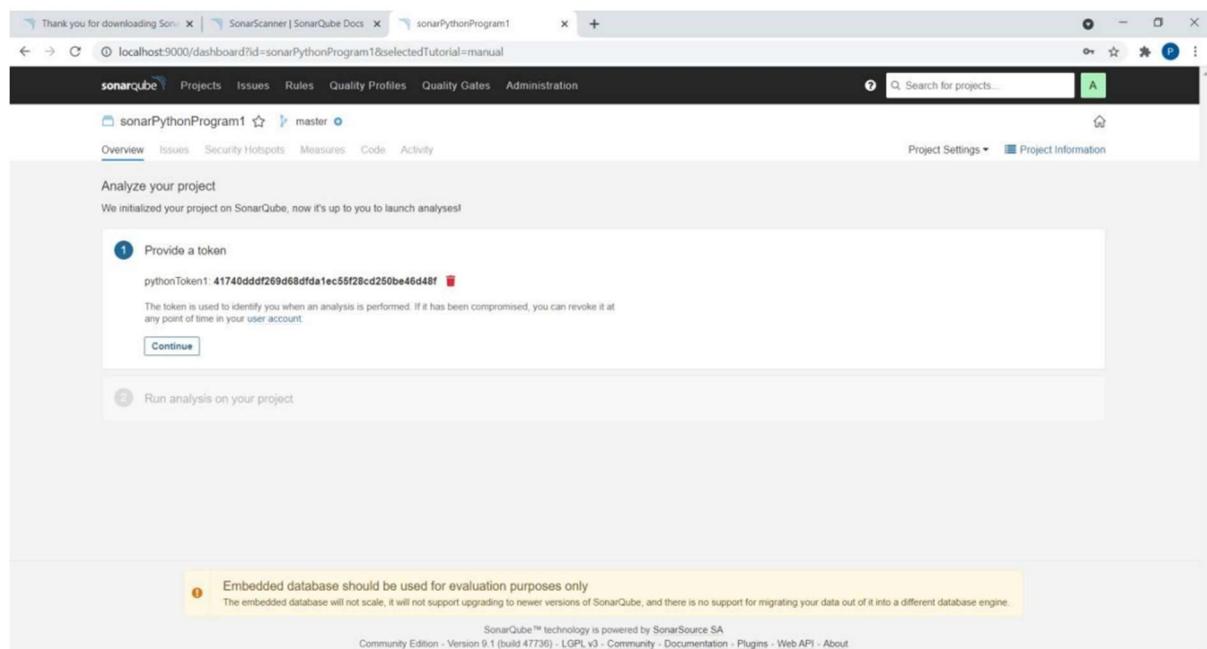
Give any Project display name.



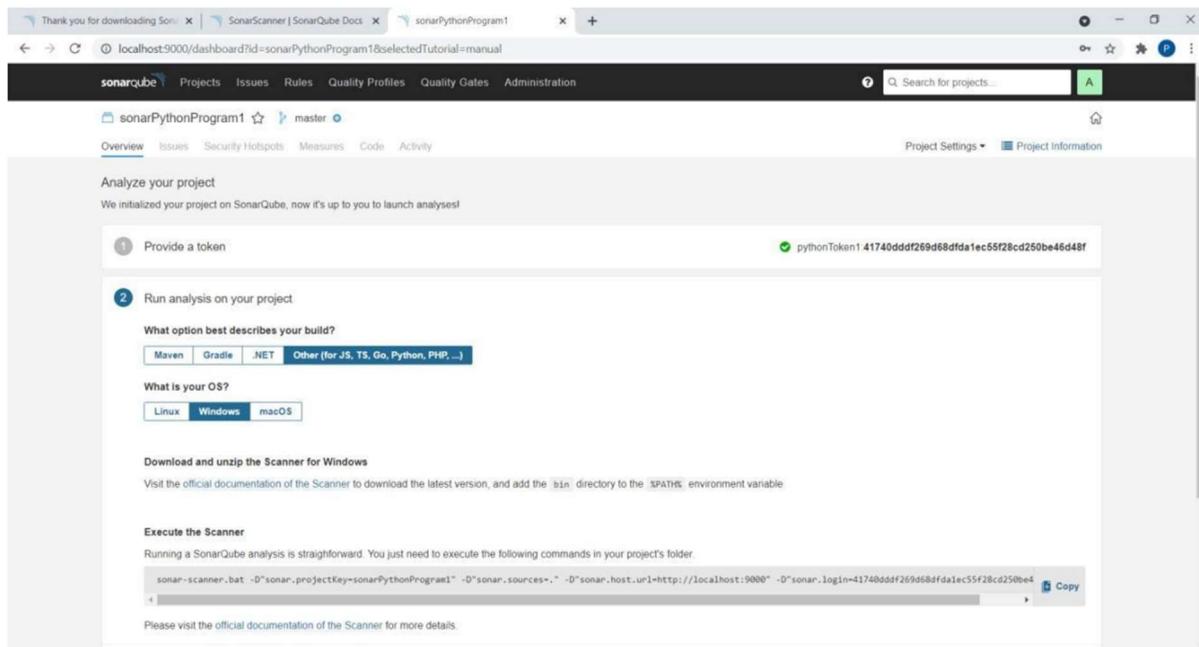
Click on Locally.



Give any name to token and click on Generate.



Click on Continue.



Save a Python program in a folder.

```
class Solution(object):
```

```
    def romanToInt(self, s):
```

```
        roman =
```

```
        num =
```

```
        while i < len(s):
```

```
            if i+1<len(s) and s[i:i+2] in roman:
```

```
                num+=roman[s[i:i+2]]
```

```
            else:
```

```
#print(i)
```

```
            num+=roman[s[i]]
```

```
        return num
```

```
obl = Solution()
```

```
print(obl I")
```

```
print(obl . l")
```

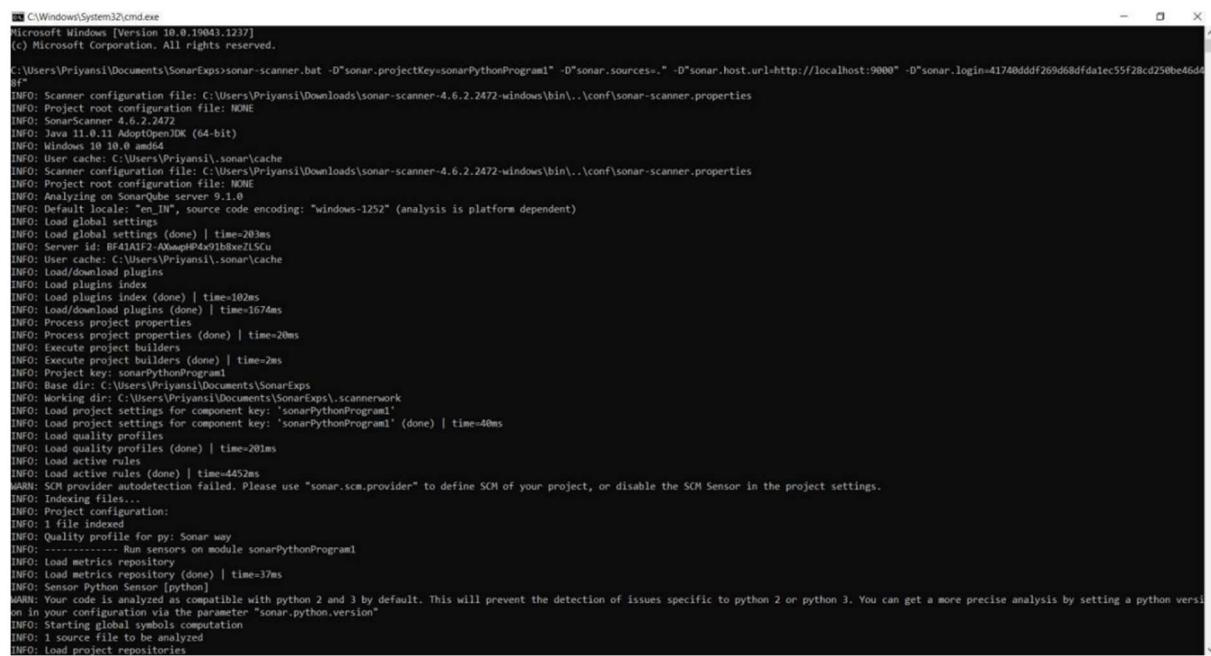
Open command prompt in this folder and Run program using copied command.

```
sonar-scanner.bat -D"sonar.projectKey=<YourDisplayName>"
```

```
-
```

```
D"sonar.sources=." -D"sonar.host.url=http://localhost:9000" -
```

```
D"sonar.login=<YourTokenGeneratedID>'
```



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Priyansi\Documents\SonarExps>sonar-scanner.bat -D"sonar.projectKey=sonarPythonProgram1" -D"sonar.sources=." -D"sonar.host.url=http://localhost:9000" -D"sonar.login=41740ddd269d68dfdalec55f28cd250be46d4bf"

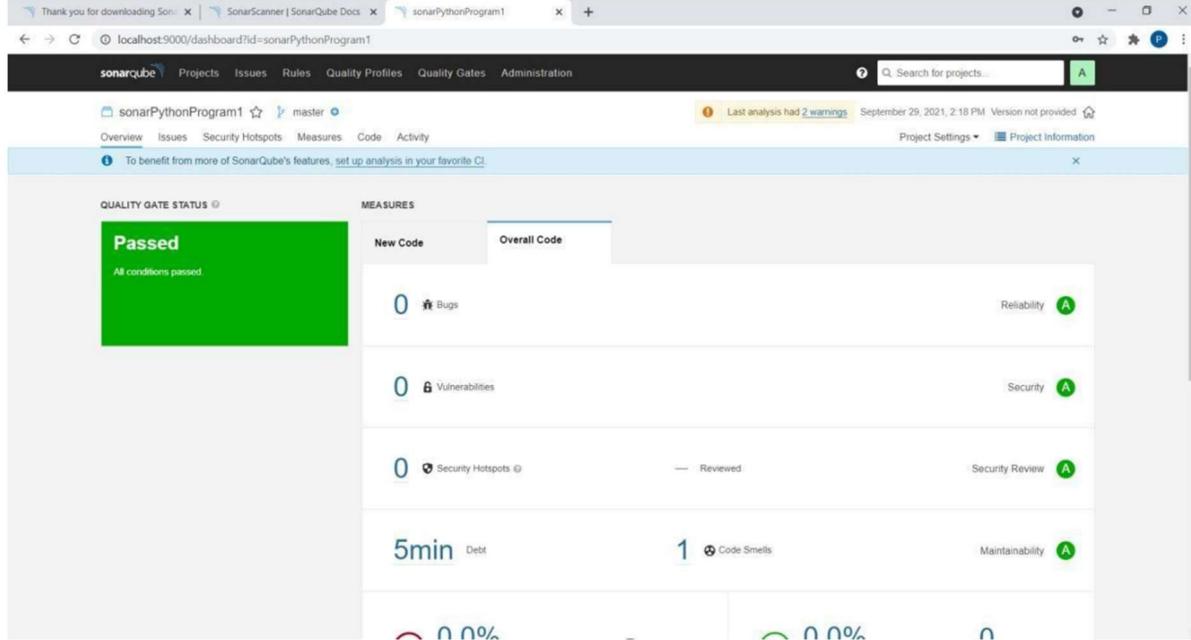
INFO: Scanner configuration file: C:\Users\Priyansi\Downloads\sonar-scanner-4.6.2.2472-windows\bin..\conf\sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: SonarScanner 4.6.2.2472
INFO: Java 11.0.11 AdoptOpenJDK (64-bit)
INFO: Windows 10 10.0 amd64
INFO: User cache: C:\Users\Priyansi\sonar\cache
INFO: Scanner configuration file: C:\Users\Priyansi\Downloads\sonar-scanner-4.6.2.2472-windows\bin..\conf\sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: Analyzing on SonarQube server 9.1.0
INFO: Default locale: "en_IN", source code encoding: "windows-1252" (analysis is platform dependent)
INFO: Load global settings (done) | time=20ms
INFO: Load global settings (done) | time=20ms
INFO: Server id: BF1A1F2-XXaa9P491bbxezLSCu
INFO: User cache: C:\Users\Priyansi\sonar\cache
INFO: load/download plugins
INFO: load plugins index
INFO: load plugins index (done) | time=102ms
INFO: load/download plugins (done) | time=1674ms
INFO: Process project properties
INFO: Process project properties (done) | time=20ms
INFO: Load project builder index
INFO: Execute project builders (done) | time=2ms
INFO: Project key: sonarPythonProgram1
INFO: Base dir: C:\Users\Priyansi\Documents\SonarExps
INFO: Working dir: C:\Users\Priyansi\Documents\SonarExps\scannerwork
INFO: Load project settings for component key: 'sonarPythonProgram1'
INFO: Load project settings for component key: 'sonarPythonProgram1' (done) | time=40ms
INFO: Load quality profiles
INFO: Load quality profiles (done) | time=20ms
INFO: Load active rules
INFO: Load active rules (done) | time=45ms
WARN: SCM provider auto-detection failed. Please use "sonar.scm.provider" to define SCM of your project, or disable the SCM Sensor in the project settings.
INFO: Indexing files...
INFO: Project configuration:
INFO: 1 file indexed
INFO: Quality profile for py: Sonar way
INFO: ----- Run sensors on module sonarPythonProgram1
INFO: Load metrics repository
INFO: Load metrics repository (done) | time=37ms
INFO: Sensor Python (done) [py]
WARN: Python is analyzed as compatible with python 2 and 3 by default. This will prevent the detection of issues specific to python 2 or python 3. You can get a more precise analysis by setting a python version in your configuration via the parameter "sonar.python.version"
INFO: Starting global symbols computation
INFO: 1 source file to be analyzed
INFO: Load project repositories
```

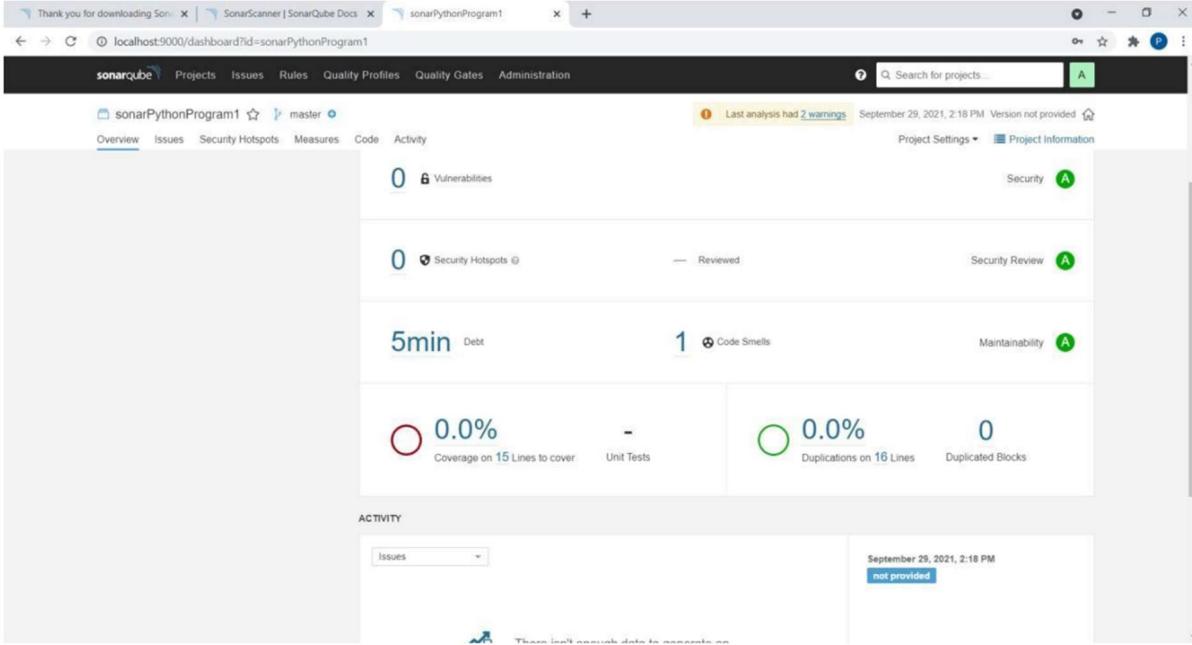
```

C:\Windows\System32\cmd.exe
[INFO]: Sensor HTML [web] {done} | time=2ms
[INFO]: Sensor VB.NET Project Type Information [vbnet]
[INFO]: Sensor VB.NET Project Type Information [vbnet] {done} | time=1ms
[INFO]: Sensor VB.NET Analysis Log [vbnet]
[INFO]: Sensor VB.NET Analysis Log [vbnet] {done} | time=12ms
[INFO]: Sensor VB.NET Properties [vbnet]
[INFO]: Sensor VB.NET Properties [vbnet] {done} | time=0ms
[INFO]: -----
[INFO]: Run Sensors on project
[INFO]: Sensor Zero Coverage Sensor
[INFO]: Sensor Zero Coverage Sensor {done} | time=12ms
[INFO]: SCM Publisher: No SCM system was detected. You can use the 'sonar.scm.provider' property to explicitly specify it.
[INFO]: CPD Executor: Calculating CPD for 1 file
[INFO]: CPD Executor: CPD calculation finished {done} | time=10ms
[INFO]: Analysis report generated in 10ms, dir size=103.9 kB
[INFO]: Analysis report compressed in 19ms, zip size=14.7 kB
[INFO]: Analysis report uploaded in 76ms
[INFO]: ANALYSIS SUCCESSFUL, you can browse: http://localhost:9000/dashboard?id=sonarPythonProgram
[INFO]: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
[INFO]: More about the report processing at http://localhost:9000/api/ce/task?id=AxawvYlhx91bbxeZLXH
[INFO]: Analysis total time: 7.502 s
[INFO]: -----
[INFO]: EXECUTION SUCCESS
[INFO]: -----
[INFO]: Total time: 10.887s
[INFO]: Final Memory: 7M/30M
[INFO]: -----
C:\Users\Priyanshi\Documents\SonarExps>

```

Given below is the inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities.

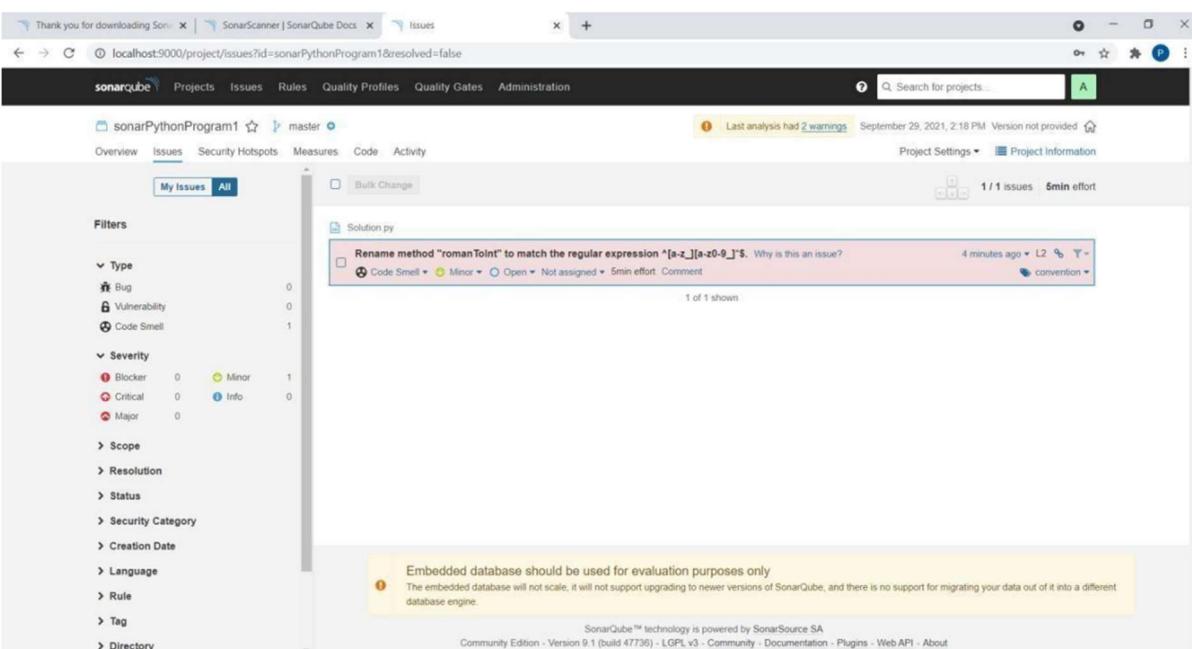




The screenshot shows the SonarQube dashboard for the project 'sonarPythonProgram1'. The dashboard provides an overview of the code quality with various metrics:

- Vulnerabilities:** 0
- Security Hotspots:** 0
- Debt:** 5min
- Code Smells:** 1
- Coverage:** 0.0% (Coverage on 15 Lines to cover)
- Duplications:** 0.0% (Duplications on 16 Lines)
- Maintainability:** 0

The 'ACTIVITY' section shows the last analysis was on September 29, 2021, at 2:18 PM, with 'not provided' details.



The screenshot shows the 'Issues' page for the same project. It displays a single issue found in 'Solution.py':

- Title:** Rename method "romanToInt" to match the regular expression "[a-z][a-z0-9]". Why is this an issue?
- Type:** Code Smell
- Severity:** Minor
- Status:** Open
- Assignee:** Not assigned
- Effort:** 5min
- Comment:** convention

A note at the bottom of the page states: "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."

Press "Ctrl + C" to stop the server.

```

[jvm 1] at org.sonar.application.process.ESManagedProcess.isOperational(ESManagedProcess.java:60)
[jvm 1] at org.sonar.application.process.ManagedProcessHandler.refreshState(ManagedProcessHandler.java:220)
[jvm 1] at org.sonar.application.process.ManagedProcessHandler.event(ManagedProcessHandler.java:285)
[jvm 1] Caused by: java.util.concurrent.ExecutionException: java.net.SocketTimeoutException: Timeout connecting to [/127.0.0.1:9001]
[jvm 1] at org.elasticsearch.common.util.concurrent.BaseFuture$Sync.getValue(BaseFuture.java:102)
[jvm 1] at org.elasticsearch.common.util.concurrent.BaseFuture$Sync.get(BaseFuture.java:249)
[jvm 1] at org.elasticsearch.common.util.concurrent.BaseFuture.get(BaseFuture.java:249)
[jvm 1] at org.elasticsearch.client.RestHighLevelClient.performClientRequest(RestHighLevelClient.java:2075)
[jvm 1] ... 10 common frames omitted
[jvm 1] Caused by: java.net.ConnectException: Timeout connecting to [/127.0.0.1:9001]
[jvm 1] at org.apache.http.nio.pool.RouteSpecificPool.timeout(RouteSpecificPool.java:169)
[jvm 1] at org.apache.http.nio.pool.AbstractNIOConnPool$SessionRequestImpl.timeout(AbstractNIOConnPool.java:626)
[jvm 1] at org.apache.http.nio.pool.AbstractNIOConnPool$InternalSessionRequestImpl.timeout(AbstractNIOConnPool.java:894)
[jvm 1] at org.apache.http.impl.nio.reactor.SessionRequestImpl.timeout(SessionRequestImpl.java:104)
[jvm 1] at org.apache.http.impl.nio.reactor.DefaultConnectingIOReactor.processTimeouts(DefaultConnectingIOReactor.java:214)
[jvm 1] at org.apache.http.impl.nio.reactor.DefaultConnectingIOReactor.processEvents(DefaultConnectingIOReactor.java:158)
[jvm 1] at org.apache.http.impl.nio.reactor.AbstractMultiworkerIOReactor.execute(AbstractMultiworkerIOReactor.java:351)
[jvm 1] at org.apache.http.impl.nio.client.PoolingHttpClientConnectionManager.execute(PoolingHttpClientConnectionManager.java:221)
[jvm 1] at org.apache.http.impl.nio.client.CloseableHttpAsyncClientBase$1.run(CloseableHttpAsyncClientBase.java:64)
[jvm 1] at java.base/java.lang.Thread.run(Thread.java:834)
[jvm 1] | 2021.09.29 13:50:50 INFO app[]|o.s.a.SchedulerImpl] Process[es] is up
[jvm 1] | 2021.09.29 13:50:50 INFO app[]|o.s.a.ProcessLauncherImpl] Launch process[[key='web', ipcIndex<2, logfilenamePrefix=web]] from [C:\Users\Priyanshu\Downloads\sonarqube-9.1.0.47736]; C:\Program Files\Java\jdk-11.0.12\bin\java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djava.io.tmpdir=C:\Users\Priyanshu\Downloads\sonarqube-9.1.0.47736\temp -XX:-OmitStackTraceInFastThrow --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-exports=java.base/sun.nio.ch=ALL-UNNAMED --add-exports=jdk.management/com.sun.management=ALL-UNNAMED --add-opens=jdk.management/com.sun.management.internal=ALL-UNNAMED -Xms512m -Xmx512m -XX:+HeapDumpOnOutOfMemoryError -Dhttp.nonProxyHosts=localhost[127.*::1] -cp ./lib/sonar-application-9.1.0.47736.jar;C:\Users\Priyanshu\Downloads\sonarqube-9.1.0.47736\lib\jdbch2\h2-2.1.4.199.jar org.sonar.server.app.WebServer C:\Users\Priyanshu\Downloads\sonarqube-9.1.0.47736\temp\sq-process177945169172410119properties
[jvm 1] | 2021.09.29 13:51:42 INFO app[]|o.s.a.SchedulerImpl] Process[web] is up
[jvm 1] | 2021.09.29 13:51:42 INFO app[]|o.s.a.ProcessLauncherImpl] Launch process[[key='ce', ipcIndex<3, logfilenamePrefix=ce]] from [C:\Users\Priyanshu\Downloads\sonarqube-9.1.0.47736]; C:\Program Files\Java\jdk-11.0.12\bin\java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djava.io.tmpdir=C:\Users\Priyanshu\Downloads\sonarqube-9.1.0.47736\temp -XX:-OmitStackTraceInFastThrow --add-opens=java.base/java.util=ALL-UNNAMED --add-exports=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/sun.nio.ch=ALL-UNNAMED --add-exports=jdk.management/com.sun.management=ALL-UNNAMED --add-opens=jdk.management/com.sun.management.internal=ALL-UNNAMED -Xms512m -Xmx512m -XX:+HeapDumpOnOutOfMemoryError -Dhttp.nonProxyHosts=localhost[127.*::1] -cp ./lib/sonar-application-9.1.0.47736.jar;C:\Users\Priyanshu\Downloads\sonarqube-9.1.0.47736\lib\jdbch2\h2-2.1.4.199.jar org.sonar.ce.app.CeServer C:\Users\Priyanshu\Downloads\sonarqube-9.1.0.47736\temp\sq-process39444687418319503\sqProperties
[jvm 1] | 2021.09.29 13:51:42 WARN app[]|[startUp] #####
[jvm 1] | 2021.09.29 13:51:42 WARN app[]|[startUp] Default Administrator credentials are still being used. Make sure to change the password or deactivate the account.
[jvm 1] | 2021.09.29 13:51:42 WARN app[]|[startUp] #####
[jvm 1] | 2021.09.29 13:51:46 INFO app[]|o.s.a.SchedulerImpl] Process[ce] is up
[jvm 1] | 2021.09.29 13:51:46 INFO app[]|o.s.a.SchedulerImpl] SonarQube is up
[wrapper | C:\Windows\system32\cmd.exe: Stopping down...
[wrapper | 2021.09.29 14:38:57 INFO app[]|o.s.a.SchedulerImpl] Stopping SonarQube
[jvm 1] | 2021.09.29 14:38:58 INFO app[]|o.s.a.SchedulerImpl] Process[ce] is stopped
[jvm 1] | 2021.09.29 14:38:58 INFO app[]|o.s.a.SchedulerImpl] Process[web] is stopped
[jvm 1] | 2021.09.29 14:38:58 INFO app[]|o.s.a.SchedulerImpl] Process[es] is stopped
[jvm 1] | 2021.09.29 14:38:58 INFO app[]|o.s.a.SchedulerImpl] SonarQube is stopped
[wrapper | <- Wrapper Stopped
Terminate batch job (Y/N)? y
C:\Users\Priyanshu\Downloads\sonarqube-9.1.0.47736\bin\windows-x86-64>

```

Conclusion:

In this assignment we implemented static analysis on python programs using SonarCube

SAST processes.

Assignment 8

Name Soham Satam

Roll No.: 109 Branch: IT/T2

Date:17/10/2023

Aim: To understand Continuous Monitoring using Nagios.

LO Mapped: LO5

Theory:

Login to Aws.

Create Ec2 instances on Aws account of any linux os

Then Run the following command in SS

```
ubuntu@ip-172-31-13-219:~$ sudo apt update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 Packages [8628 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [909 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/universe Translation-en [5124 kB]
```

cd /usr/src/

sudo wget https://github.com/NagiosEnterprises/nagioscore/archive/nagios-4.4.2.tar.gz

```
ubuntu@ip-172-31-13-219:/usr/src$ sudo wget https://github.com/NagiosEnterprises/nagioscore/archive/nagios-4.4.2.tar.gz
--2021-10-05 10:24:05-- https://github.com/NagiosEnterprises/nagioscore/archive/nagios-4.4.2.tar.gz
Resolving github.com (github.com)... 13.234.176.102
Connecting to github.com (github.com)|13.234.176.102|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/NagiosEnterprises/nagioscore/tar.gz/nagios-4.4.2 [following]
--2021-10-05 10:24:05-- https://codeload.github.com/NagiosEnterprises/nagioscore/tar.gz/nagios-4.4.2
Resolving codeload.github.com (codeload.github.com)... 13.233.43.20
Connecting to codeload.github.com (codeload.github.com)|13.233.43.20|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-gzip]
Saving to: 'nagios-4.4.2.tar.gz'

nagios-4.4.2.tar.gz                                              [ <=>                               ] 10.78M 16.9MB/s   in 0.6s

2021-10-05 10:24:06 (16.9 MB/s) - 'nagios-4.4.2.tar.gz' saved [11301457]
```

sudo tar zxf nagios-*tar.gz

cd nagioscore-nagios-*/

```
ubuntu@ip-172-31-13-219:/usr/src$ cd nagioscore-nagios-*/
ubuntu@ip-172-31-13-219:/usr/src/nagioscore-nagios-4.4.2$ █
```

Now finally run the following command

```
sudo ./configure --with-httpd-conf=/etc/apache2/sites-enabled
```

if error comes install c compiler on the linux

by following this link

<https://linuxize.com/post/how-to-install-gcc-compiler-on-ubuntu-18-04/>

Finally-

```
4. Rerun the configure script.

NOTE: If you can't get the configure script to recognize the GD libs
on your system, get over it and move on to other things. The
CGIs that use the GD libs are just a small part of the entire
Nagios package. Get everything else working first and then
revisit the problem. Make sure to check the nagios-users
mailing list archives for possible solutions to GD library
problems when you resume your troubleshooting.

*****
checking ltdl.h usability... no
checking ltdl.h presence... no
checking for ltdl.h... no
checking dlfcn.h usability... yes
checking dlfcn.h presence... yes
checking for dlfcn.h... yes
checking for dlopen in -ldl... yes
checking for extra flags needed to export symbols... -Wl,-export-dynamic
checking for linker flags for loadable modules... -shared
checking for traceroute... no
checking for type va_list... yes
checking for perl... /usr/bin/perl
checking for unzip... no
```

Now let us install plugins by

```
sudo wget -O nagios-plugins.tar.gz https://github.com/nagios-plugins/nagios-plugins/archive/release-2.2.1.tar.g
```

then

```
sudo tar zxf nagios-plugins.tar.gz
```

then

```
cd nagios-plugins-release-2.2.1
```

finally start and then check status of nagi os

```
sudo systemctl start nagios
```

```
sudo systemctl status nagios
```

```
* nagios.service - Nagios Core 4.4.2
   Loaded: loaded (/lib/systemd/system/nagios.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2018-11-16 14:54:21 PST; 1s ago
     Docs: https://www.nagios.org/documentation
   Process: 18294 ExecStopPost=/bin/rm -f /usr/local/nagios/var/rw/nagios.cmd (code=exited, status=0/SUCCESS)
   Process: 18293 ExecStop=/bin/kill -s TERM ${MAINPID} (code=exited, status=0/SUCCESS)
   Process: 18315 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Process: 18313 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
 Main PID: 18325 (nagios)
    Tasks: 6 (limit: 2319)
   CGroup: /system.slice/nagios.service
```

Steps-

Go to google.com, Search NagiOs Demo

Click on the first link shown below

Google search results for "nagios demo":

- <https://exchange.nagios.org> › directory › Demos › details
- Nagios XI Online Demo**
An online **demo** of Nagios XI. The **demo** allows you to test configuration wizards, dashlets, dashboards, views, and more. Reviews (0).
- <https://exchange.nagios.org> › directory › Demos
- Demos - Nagios Exchange**
An online **demo** of Nagios Log Server. The **demo** allows you to view system logs and event logs, giving some examples on how you can visualize data sent into Nagios ...
- <https://exchange.nagios.org> › directory › Demos › details

Now click on the website-

Nagios®

Network: | Enterprise | Support | Library | Project | Exchan

Home Directory About

Home | Directory | Demos | Nagios XI Online Demo

Directory Tree

Nagios XI Online Demo

Submit review | Recommend | Print | Visit | Claim |

Rating  Favoured: 0
0 votes

Owner [egalstad](#)

Website nagiosxi.demos.nagios.com

Hits 141800

Search Exchange

search... Advanced Search

Search All Sites

Nagios Live Webinars
Let our experts show you how Nagios can help your organization.

Now click on login as administrator

Demo Account Options

- **Administrator Access** - An account with administrative privileges.
Log in as Administrator Username: nagiosadmin
 Password: nagiosadmin
- **Read-Only User Access** - A user account that can view all hosts and services, but not make any changes or issue commands.
Log in as Read-Only User Username: readonly
 Password: readonly
- **Advanced User Access** - An advanced user account that can see and control (schedule downtime, edit, etc) all hosts and services.
Log in as Advanced User Username: advanced
 Password: advanced
- **Normal User Access** - A sample "normal" user account that has rights to view only a defined subset of all hosts and services.
Log in as Normal User Username: jdoe
 Password: jdoe
- **Administrator Access** - showing the dark theme.
Log in as Administrator Username: darktheme
 Password: darktheme

Demo Notes

It will have interface like this

Home Dashboard

Getting Started Guide

Host Status Summary

Up	Down	Unreachable	Pending
56	1	0	6

Last Updated: 2021-10-05 05:06:48

Service Status Summary

OK	Warning	Unknown	Critical	Pending
764	100	5	131	7

Last Updated: 2021-10-05 05:06:48

Administrative Tasks

Start Monitoring

We're Here To Help!

Our knowledgeable techs are happy to help you with any questions or problems you may have getting Nagios up and running.

- Support Forum / Customer Support Forum
- Help Resources
- Customer Ticket
- Support Center
- Customer Phone Support: +1 651-204-9102 Ext. 4

Now click on Host status-

The screenshot shows the Nagios XI 5.7.2 interface. On the left is a sidebar with navigation links like Home Dashboard, Host Status, Host Group Summary, and Maps. The main area has two summary tables: 'Host Status Summary' and 'Service Status Summary'. Below them is a detailed table of host status.

Host	Status	Duration	Attempt	Last Check	Status Information
europa.nagios.local	Down	42d 19h 2m 42s	5/5	2021-10-05 05:04:53	CRITICAL - 192.168.4.54: Host unreachable @ 192.168.5.66. rta nan, lost 100%
www.acme.com	Down	1190d 17h 28m 49s	5/5	2021-10-05 05:05:20	CRITICAL - 216.27.178.28: rta nan, lost 100%
www.chaoticmoon.com	Up	851d 16h 42m 45s	5/5	2021-10-05 05:05:50	check_http: invalid hostname/address - www.chaoticmoon.com
Firewall	Up	1190d 17h 28m 11s	1/10	2021-10-05 05:02:49	OK - 127.0.0.1 rta 0.020ms lost 0%
Log-Server.nagios.local	Up	2275d 8h 1m 2s	1/5	2021-10-05 05:05:22	OK - localhost rta 0.022ms lost 0%
Log-Server2.nagios.local	Up	1180d 14h 8m 21s	1/5	2021-10-05 05:05:53	OK - localhost rta 0.026ms lost 0%
NOAA	Up	3763d 12h 56m 36s	1/3	2012-01-02 09:43:01	HTTP OK HTTP/1.1 200 OK - 99753 bytes in 0.478 seconds
Netw	Pending	N/A	1/5	N/A	No check results for host yet...
Network-Analyzer.nagios	Pending	N/A	1/5	N/A	No check results for host yet...
Network-Analyzer.nagios.local	Up	2275d 7h 58m 0s	1/5	2021-10-05 05:07:12	OK - localhost rta 0.021ms lost 0%
Network-Analyzer2	Pending	N/A	1/5	N/A	No check results for host yet...
Network-Analyzer2.nagios.local	Up	2275d 7h 57m 50s	1/5	2021-10-05 05:06:42	OK - localhost rta 0.021ms lost 0%
Router	Up	1190d 17h 31m 9s	1/10	2021-10-05 05:03:25	OK - 127.0.0.1 rta 0.020ms lost 0%

In the above image one can see Host Status Summary and Service Status Summary also how many host are up, down and also errors in detail

Now click on Host Group Status.

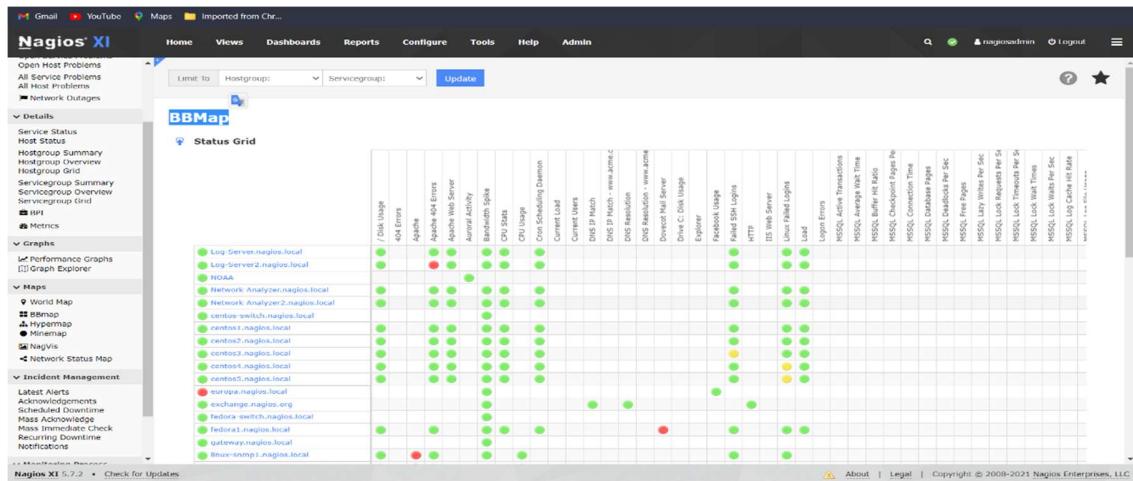
The screenshot shows the Nagios XI 5.7.2 interface. The main area displays 'Host Group Status' with a summary view and a detailed 'Status Summary For All Host Groups' table.

Host Group	Hosts	Services
Monitoring Servers (Monitoring Servers)	5 Up	93 OK 14 Warning 2 Critical
Hostgroup Two (hg2)	1 Up	11 OK 4 Warning 2 Unknown 1 Critical
Some Other Hostgroup (hg3)	2 Up	11 OK 1 Warning 1 Critical
Linux Servers (linux-servers)	11 Up	218 OK 27 Warning 2 Unknown 12 Critical
Network Devices (network-devices)	7 Up	215 OK 35 Warning 12 Critical

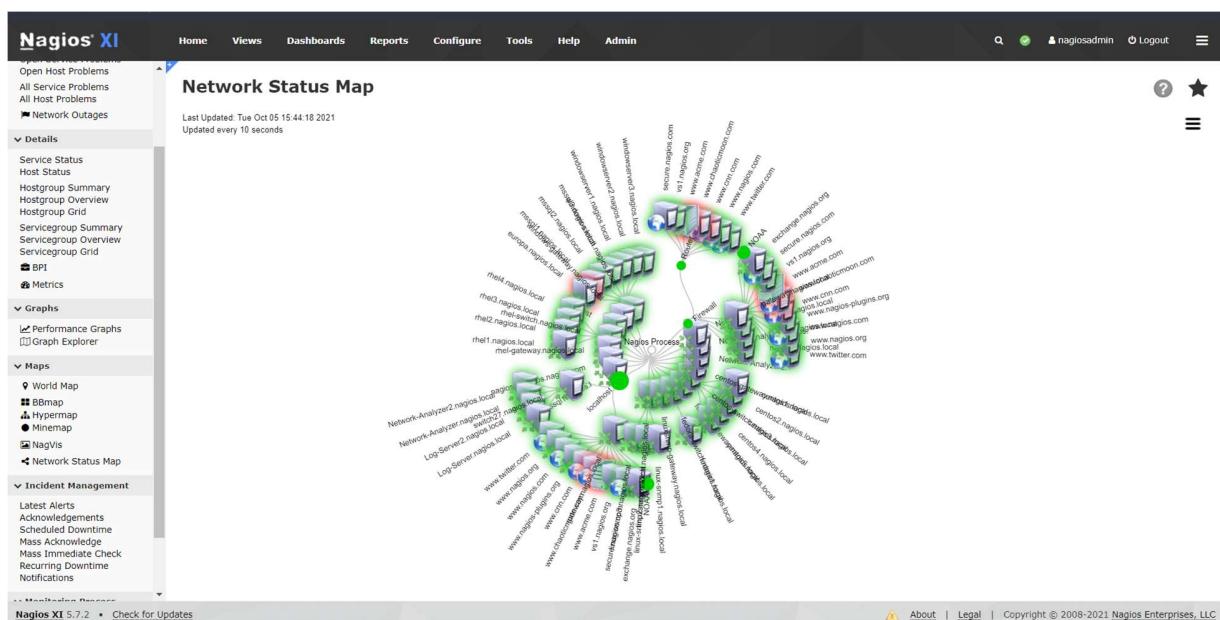
Here we can see Status Summary For All Host Groups

Now we click on BBMap

In this we can see status of following stuff in each host-



Now we have Network status map which is graphical representation of the network status



Conclusion: In this assignment we learned about how to continuously monitor Nagios commands

ASSIGNMENT 9

AIM: To understand Lambda Function and create a Lambda function which will log "An Image has been added" once you add an object to a specific bucket in S3.

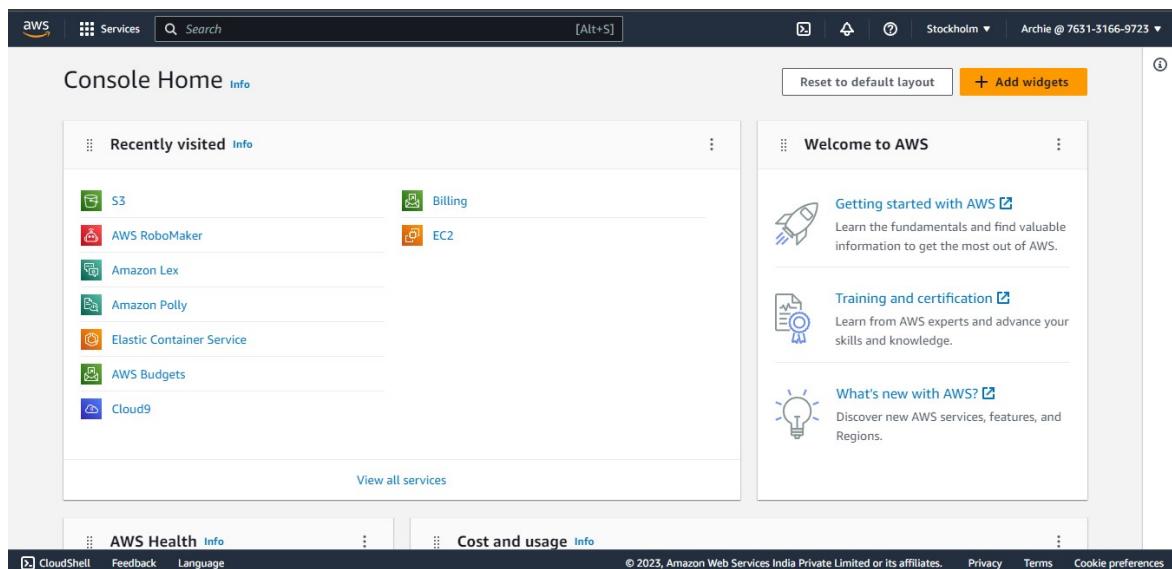
LO MAPPED: LO1, LO5

THEORY:

You can use Lambda to process event notifications from Amazon Simple Storage Service. Amazon S3 can send an event to a Lambda function when an object is created or deleted. You configure notification settings on a bucket, and grant Amazon S3 permission to invoke a function on the function's resource-based permissions policy.

STEPS TO FOLLOW:

1. Log in as IAM User



2. Create a S3 bucket and Enable the "Block all public access"

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.
- Block public access to buckets and objects granted through new public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- Block public and cross-account access to buckets and objects through any public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Successfully created bucket "labdabucket117"

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

[View details](#) (1)

Amazon S3 > Buckets

Account snapshot

Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

Buckets (1) Info

Buckets are containers for data stored in S3. [Learn more](#)

Name	AWS Region	Access	Creation date
labdabucket117	Europe (Stockholm) eu-north-1	Objects can be public	August 30, 2023, 22:56:19 (UTC-07:00)

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

3. Search IAM on the console and go to "Roles". Click on Create Roles

Identity and Access Management (IAM)

[Search IAM](#)

IAM > Roles

Roles (4) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
AWSCloud9SSMAccessRole	AWS Service: cloud9, and 1 more. Edit	29 days ago
AWSServiceRoleForAWSCloud9	AWS Service: cloud9 (Service-Linked Role) Edit	29 days ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role) Edit	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role) Edit	-

Roles Anywhere [Info](#)

Authenticate your non AWS workloads and securely provide access to AWS services.

[Manage](#)

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

4. Select the options of "AWS service" and "lambda"

Screenshot of the AWS IAM "Create New Role" wizard Step 2: Add permissions. The "Trusted entity type" section shows "AWS service" selected. The "Use case" section shows "Lambda" selected. The "Permissions policies" section shows "CloudWatchFullAccess" selected.

5. Enable the "CloudWatchFullAccess" and "AmazonS3FullAccess"

Screenshot of the AWS IAM "Create New Role" wizard Step 2: Add permissions. The "Permissions policies" section shows "CloudWatchFullAccess" selected.

Screenshot of the AWS IAM "Create New Role" wizard Step 2: Add permissions. The "Permissions policies" section shows "AmazonS3FullAccess" selected.

<https://us-east-1.console.aws.amazon.com/iamv2/home?region=eu-north-1#/policies/details/arn%3Awss%3Iam%3A%3Aaw%3Apolicy%2FAmazonS3ReadOnlyAccess>

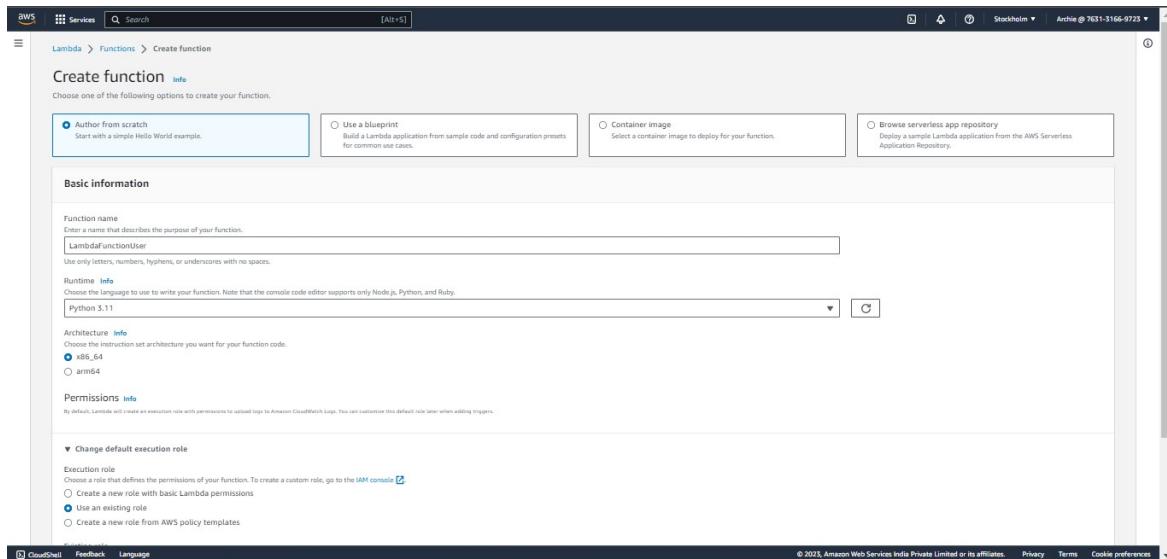
6. Click on Create Role and you will be redirected to this dashboard. Give the Role a Name and Click on Done.

The screenshot shows the 'Name, review, and create' step of the IAM role creation wizard. It includes three tabs: Step 1 (Select trusted entity), Step 2 (Add permissions), and Step 3 (Name, review, and create). The 'Role name' field contains 'LambdaUser'. The 'Description' field contains 'Allows Lambda functions to call AWS services on your behalf.' The 'Edit' button is visible at the bottom right of the main form area.

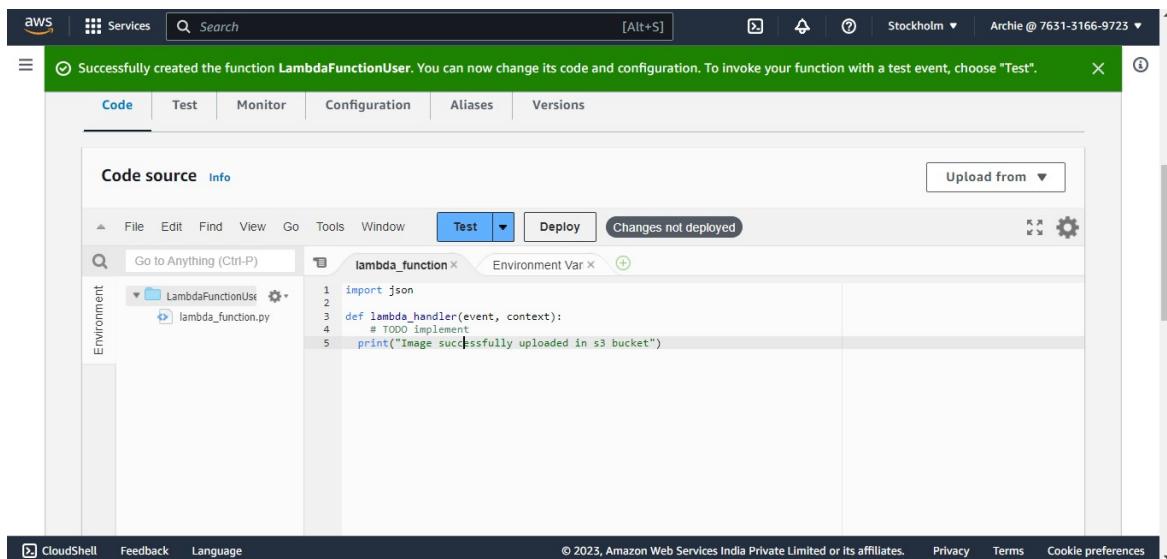
7. Search for Lambda in the console and click on Create Function.

The screenshot shows the AWS Lambda home page. It features a dark header with the AWS logo and a search bar. Below the header, there's a large 'Compute' section with the heading 'AWS Lambda' and the subtext 'lets you run code without thinking about servers.' A 'Get started' box contains the text 'Author a Lambda function from scratch, or choose from one of many preconfigured examples.' with a 'Create a function' button. At the bottom, there's a 'How it works' section with tabs for '.NET', 'Go', 'Java', 'Node.js', 'Python', 'Ruby', and 'Custom runtime'. The 'Node.js' tab is selected. The URL in the browser address bar is <https://eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/create/function?firstrun=true>.

8. Change the settings of the Lambda Function.



9. Add the python code and click on deploy to save the changes.



10. Scroll above and click on ADD TRIGGER. Select the following options and click on Done.

Trigger configuration [Info](#)

S3 aws asynchronous storage

Bucket
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

[Select](#)

Bucket region: eu-north-1

Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

All object create events [X](#)

Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.

Recursive invocation
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)

[cloudShell](#) [Feedback](#) [Language](#)

© 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

11. Go to S3 bucket and click on Add files. Select a image and click on Upload.

Amazon S3 [X](#)

labdabucket117 [Info](#)

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

Objects (0)

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](#)

[Actions](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#)

[Create folder](#) [Upload](#)

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

[CloudShell](#) [Feedback](#) [Language](#)

© 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose [Add files](#) or [Add folder](#).

Files and folders (1 Total, 266.2 KB)

All files and folders in this table will be uploaded.

	Name	Folder	Type	Size
<input type="checkbox"/>	Screenshot (3).png	-	image/png	266.2 KB

Destination

Destination
[s3://labdabucket117](#)

Destination details
Bucket settings that impact new objects stored in the specified destination.

[CloudShell](#) [Feedback](#) [Language](#)

© 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

The screenshot shows the AWS S3 console with a green header bar indicating "Upload succeeded". Below it, a summary table shows one file uploaded to "s3://labdabucket117" with a success rate of 100.00%. At the bottom, there are tabs for "Files and folders" and "Configuration", with "Files and folders" selected. A message at the top says "The information below will no longer be available after you navigate away from this page."

12. Search CloudWatch and go to Log groups. Select the existing Log Group.

The screenshot shows the AWS CloudWatch Logs console. The left sidebar is expanded to show "Logs" and "Log groups". The main area displays a table for "Log groups (1)". It shows one log group named "/aws/lambda/LambdaFunctionUser" with a retention policy of "Never expire". There are buttons for "Actions", "View in Logs Insights", "Start tailing", and "Create log group". A search bar and filter options are also present.

13. Click on the link provided and then you will see the message displayed.

The screenshot shows the AWS CloudWatch Logs console. The left sidebar is expanded to show "Logs" and "Log groups". The main area displays a table for "Log streams (1)". It shows one log stream with the name "2023/08/31/[LATEST]97bf6ca2f8dd4cf2b383b55caf72...". The table includes columns for "Log stream", "Last event time", and "Subscription filters". There are buttons for "Actions", "Delete", "Create log stream", and "Search all log streams". A search bar and filter options are also present.

The screenshot shows the AWS CloudWatch Log Events interface. The left sidebar has a 'Logs' section with 'Log groups' selected, showing 'Live Tail' and 'Logs Insights'. The main area is titled 'Log events' and displays log entries for a Lambda function. The log entries are as follows:

Timestamp	Message
2023-08-30T23:08:48.141-07:00	INIT_START Runtime Version: python:3.11.v10 Runtime Version ARN: arn:aws:lambda:...
2023-08-30T23:08:48.249-07:00	START RequestId: d87be3bc-640d-4d9a-864d-2d10c71bc4a9 Version: \$LATEST
2023-08-30T23:08:48.249-07:00	Image successfully uploaded in s3 bucket
2023-08-30T23:08:48.250-07:00	END RequestId: d87be3bc-640d-4d9a-864d-2d10c71bc4a9
2023-08-30T23:08:48.250-07:00	REPORT RequestId: d87be3bc-640d-4d9a-864d-2d10c71bc4a9 Duration: 1.51 ms Billed D...

No newer events at this moment. [Auto retry paused](#). [Resume](#)

CONCLUSION: In this assignment, we learnt how to create a Lambda function which will log "An Image has been added" once you add an object to a specific bucket in S3.

Assignment Number 10

Name: Soham Satam ; Branch: I.T (T.E.); Roll Number: 109

17/10/2023

Aim: To create a Lambda function using Python for adding data to Dynamo DB database. LO

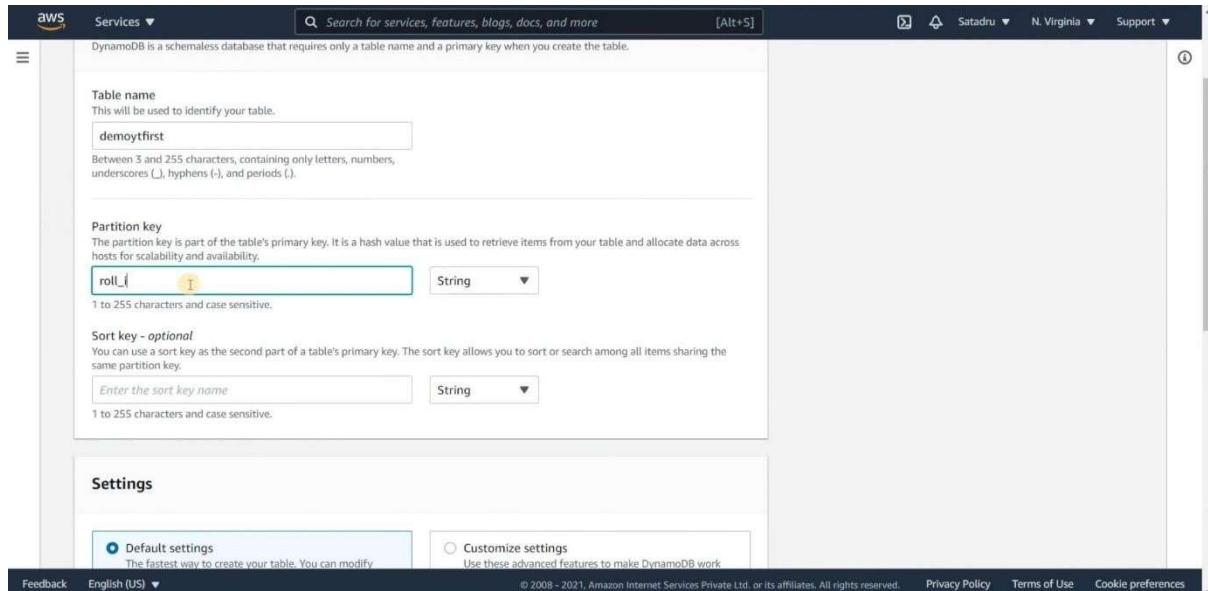
mapped: LO6

Theory:

Step 1: Set Up AWS Environment

1. **Create an AWS Account:** If you don't have an AWS account, sign up for one at [AWS Console](#).

2. **Access AWS Lambda Console:**



- Go to the [AWS Lambda Console](#).
- Click on "Create function."

3. **Create a New Lambda Function**

The screenshot shows two overlapping AWS service consoles. The top window is the 'DynamoDB' service, specifically the 'Tables' section. A green success message at the top states: 'The demoytfirst table was created successfully.' Below this, the table list shows one entry: 'demoytfirst' (Status: Active, Partition key: roll_no (Number), Sort key: -, Read capacity mode: Provisioned with auto scaling (5), Write capacity mode: Provisioned with auto scaling (5)). The bottom window is the 'Create role' step in the 'AWS Lambda' service. It's the second step in a four-step process, indicated by a blue circular progress indicator. The step title is 'Attach permissions policies'. A search bar shows 'Dynam'. A table lists available policies, with 'AmazonDynamoDBFullAccess' selected (indicated by a checked checkbox). Other policies listed include 'AmazonDynamoDBReadOnlyAccess', 'AWSApplicationAutoscalingDynamoDBTablePolicy', 'AWSLambdaDynamoDBExecutionRole', 'AWSLambdaInvocation-DynamoDB', 'DynamoDBCloudWatchContributorInsightsServiceRolePolicy', 'DynamoDBKinesisReplicationServiceRolePolicy', and 'DynamoDBReplicationServiceRolePolicy'. The table has columns for 'Policy name' and 'Used as'.

- Choose "Author from scratch."
- Enter a name for your function (e.g., **AddToDynamoDBFunction**).
- Choose "Python" as the runtime.

4. Configure Execution Role

The screenshot shows the AWS Lambda 'Create role' interface. The user is on Step 2, 'Attach permissions policies'. A search bar at the top right is set to 'Cloudwatch'. Below it, a table lists policies under 'Showing 27 results'. The 'CloudWatchFullAccess' policy is selected (indicated by a checked checkbox). Other policies listed include CloudWatchEventsInvocationAccess, CloudWatchEventsReadOnlyAccess, CloudWatchEventsServiceRolePolicy, CloudWatchLambdaInsightsExecutionRolePolicy, CloudWatchLogsFullAccess, CloudWatchLogsReadOnlyAccess, and CloudWatchReadOnlyAccess.

Policy name	Used as
CloudWatchEventsInvocationAccess	None
CloudWatchEventsReadOnlyAccess	None
CloudWatchEventsServiceRolePolicy	None
<input checked="" type="checkbox"/> CloudWatchFullAccess	None
CloudWatchLambdaInsightsExecutionRolePolicy	None
CloudWatchLogsFullAccess	None
CloudWatchLogsReadOnlyAccess	None
CloudWatchReadOnlyAccess	None

- Create a new role with basic Lambda permissions and additional permissions to interact with DynamoDB.
- Attach the role to your Lambda function.

Step 2: Set Up DynamoDB

1. Access DynamoDB Console:

- Go to the [AWS DynamoDB Console](#).

Function name: ytdynamodemo

Runtime: Python 3.9

Architecture: x86_64

Permissions: lambdaroledynamo

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

2. Create a DynamoDB Table:

- Click on "Create table."
- Enter a table name and a primary key (e.g., **ID**).
- Configure other settings as needed and create the table.

```

1 import json
2 import boto3
3
4 def lambda_handler(event, context):
5     # TODO implement
6     client_dynamo=boto3.resource('dynamodb')
7     table=client_dynamo.Table('testing')
8     try:
9         response=table.put_item(Item=event)
10    return "Done"
11 except:
12    raise
13

```

Step 3: Write Lambda Function Code

REPORT RequestId: 0479da14-d297-4a84-982b-3d597a6b67e2 Duration: 1623.97 ms Billed Duration: 1624 ms Memory Size: 128 MB Max Memory Used: 65 MB Init Duration: 402.18 ms

Test event

Invoke your function with a test event. Choose a template that matches the service that triggers your function, or enter your event document in JSON.

New event
 Saved event

Template: hello-world

Name: MyEventName

```

1+ {
2   "roll_no": 20,
3   "Name": "Rajdeep Sil",
4   "University": "VIT",
5   "Address": "Delhi"
6 }

```

1. Ensure to replace '**YourDynamoDBTableName**' with the actual name of your DynamoDB table.

2. Configure Lambda Handler:

- In the Lambda function configuration, set the handler to **filename.lambda_handler** (e.g., **yourfilename.lambda_handler**).

Step 4: Configure Environment Variables

1. Add DynamoDB Table Name:

- In the Lambda function configuration, add an environment variable (e.g., **DYNAMODB_TABLE**) with the value set to your DynamoDB table name.

Step 5: Test Locally

1. Test Lambda Function Locally:

- Create a test event with sample data.
- Test the Lambda function using the "Test" button in the Lambda console.

Step 6: Deploy Lambda Function

1. Deploy the Lambda Function:

- Click on the "Deploy" button in the Lambda console.

Step 7: Test in AWS Lambda Console

1. Test in AWS Lambda Console:

- Use the Lambda console to test the function by creating a test event with sample data.
- Verify that the function executes successfully.

Step 8: Monitor and Troubleshoot

1. Check CloudWatch Logs:

- Use CloudWatch Logs to check logs generated by your Lambda function for troubleshooting.

Step 9: Documentation

1. Create Documentation:

- Document the purpose, input parameters, and expected output of your Lambda function.
- Explain any challenges faced during development and how they were resolved.

By following these steps, you can create a Lambda function using Python to add data to a DynamoDB database in the AWS environment.

The screenshot shows the AWS DynamoDB Items page. On the left, there's a sidebar with 'DynamoDB' selected. The main area has a search bar at the top. Below it, a table titled 'demoyfirst' is displayed with two rows of data. The columns are labeled 'roll_no', 'Address', 'Name', and 'University'. The first row has values 10, Soham, KIT, and KIT. The second row has values 20, Delhi, Rajdeep Sill, and VIT. There are buttons for 'Run' and 'Reset' filters, and a 'Completed' status message indicating 0.5 capacity units consumed.

roll_no	Address	Name	University
10	Soham	KIT	KIT
20	Delhi	Rajdeep Sill	VIT

Conclusion: By this assignment we learn how to create a Lambda function using Python for adding data to Dynamo DB database.

Adv. DevOps Written Assignment : 01

1. what security measures can be taken while using Kubernetes?

1. Role-Based Access Control (RBAC): RBAC restricts who can perform actions within a Kubernetes cluster. It defines roles and role bindings to specify what resources and operations users or service accounts can access. This prevents unauthorized access and actions within the cluster.
2. Regular Updates: Keeping Kubernetes and its components up to date is crucial. New releases often include security patches. Regular updates help mitigate known vulnerabilities and ensure your cluster remains secure.
3. Network Policies: Network policies allow you to define rules for communication between pods. By specifying which pods can communicate with each other, you can limit the attack surface and prevent unauthorized access.
4. Container Security Tools: Employ container security tools like vulnerability scanners to assess the security of container images. These tools can identify and remediate vulnerabilities in the containerized applications before they are deployed.
5. Monitoring and Audit: Implement monitoring and auditing solutions to track cluster activity. This helps detect and respond to suspicious or unauthorized behavior. Tools like Prometheus and Grafana can be used for monitoring, while audit logs provide insights into cluster activity.
6. Secrets Management: Sensitive data like API keys, passwords, and certificates should be stored securely using Kubernetes secrets or external vaults. This prevents sensitive information from being exposed within containers or configuration files.
7. PodSecurityPolicies (PSP): PSP is a Kubernetes feature that enforces security policies at the

pod level. It allows you to define restrictions on privilege escalation, host access, and other security-sensitive configurations for pods.

8. Namespaces: Use Kubernetes namespaces to logically isolate workloads. This provides a level of separation between different applications or teams, reducing the risk of unauthorized access or interference between them.

9. Admission Controllers: Admission controllers are webhook plugins that intercept and validate requests to the Kubernetes API server. You can use them to enforce custom policies and ensure that only compliant resources are admitted to the cluster.

10. Container Runtime Security: Implement container runtime security solutions like Docker Security Scanning or container runtime protection tools. These tools monitor containers at runtime for abnormal behavior, helping to detect and respond to potential threats.

Combining these measures into a comprehensive security strategy is essential for safeguarding your Kubernetes cluster and the applications running within it. It's important to stay informed about best practices and evolving security threats in the Kubernetes ecosystem.

2. What are the three security techniques that can be used to protect data?

Three security techniques commonly used to protect data are:

1. Encryption: Encryption is the process of converting data into a secure format that can only be read by someone with the decryption key. It ensures that even if unauthorized parties access the data, they cannot understand it without the correct key. Two common types of encryption are:

- Data-at-rest Encryption: Protects data when it's stored on disk or in a database.
- Data-in-transit Encryption: Secures data as it's transmitted between systems over networks.

2. Access Control: Access control mechanisms regulate who can access data and what actions they can perform on it. This involves setting permissions, roles, and policies to ensure that only authorized users or applications can access and manipulate data. Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC) are commonly used access control models.

3. Data Masking/Redaction: Data masking or redaction involves obscuring or replacing sensitive data with fictitious or scrambled values. This is often used in non-production environments or when sharing data with third parties. It ensures that even if someone gains access to the data, they cannot see the actual sensitive information.

These techniques are often used in combination to create a layered approach to data security, providing multiple levels of protection to safeguard sensitive information from unauthorized access and disclosure.

3. How do you expose a service using ingress in Kubernetes?

To expose a service using Ingress in Kubernetes, you need to follow these steps:

1. Set up Kubernetes: Ensure you have a Kubernetes cluster up and running, and you have the `kubectl` command-line tool configured to communicate with the cluster.

2. Deploy Your Application: Deploy your application as a Kubernetes Deployment or a Pod, and create a Kubernetes Service to expose it internally within the cluster. This Service will be the target for the Ingress.

3. Install an Ingress Controller: You need to have an Ingress controller installed in your cluster. Some popular options include Nginx Ingress Controller, Traefik, or HAProxy Ingress. The controller will manage the Ingress resources and configure the load balancer.

For example, to install the Nginx Ingress Controller, you can use:

```
```bash
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.0.0/deploy/static/provider/cloud/deploy.yaml
```

```
```
```

4. Create an Ingress Resource: Define an Ingress resource that specifies the rules for routing traffic to your service. Here's an example Ingress resource manifest:

```
```yaml
```

```
apiVersion: networking.k8s.io/v1
```

```
kind: Ingress
```

```
metadata:
```

```
name: my-ingress
```

```
spec:
```

```
rules:
```

```
- host: example.com
```

```
 http:
```

```
 paths:
```

```
 - path: /path
```

```
 pathType: Prefix
```

```
 backend:
```

```
 service:
```

```
 name: your-service
```

```
 port:
```

```
 number: 80
```

```
```
```

In this example, traffic for `example.com/path` will be routed to `your-service`.

5. Apply the Ingress Resource: Use `kubectl apply` to create the Ingress resource in your cluster:

```
```bash
```

```
kubectl apply -f your-ingress.yaml
```

```
```
```

6. Configure DNS: Ensure that the DNS records for the specified hostname (e.g., `example.com`) point to the external IP address of your Ingress controller.

7. Access Your Service: After DNS propagation, you should be able to access your service externally via the hostname and path you defined in the Ingress resource.

4. Which service protocols does Kubernetes ingress expose?

Kubernetes Ingress is primarily designed to expose HTTP and HTTPS services, making it suitable for routing and load balancing web traffic. However, with the evolution of Kubernetes and Ingress controllers, it has expanded to support additional protocols and features:

1. HTTP: Ingress is commonly used to expose HTTP services. You can define routing rules based on URL paths, hostnames, and other HTTP attributes.

2. HTTPS: Secure HTTP services can be exposed through Ingress by configuring TLS certificates. This allows you to terminate SSL/TLS encryption at the Ingress controller and route decrypted traffic to your services.

3. TCP: Some Ingress controllers, like Nginx Ingress, support TCP services. This enables you to expose non-HTTP services such as databases or custom protocols. TCP-based routing typically relies on port numbers.

4. UDP: While less common, some Ingress controllers support UDP services. UDP is a connectionless protocol used for various purposes, including DNS and VoIP. Exposing UDP services may require specific controller support.

5. gRPC: If your services use the gRPC protocol, you can configure Ingress resources to handle gRPC traffic. gRPC is a high-performance RPC (Remote Procedure Call) framework often used for communication between microservices.

6. WebSocket: Ingress controllers can be configured to support WebSocket connections. WebSocket is a protocol that enables full-duplex communication over a single TCP connection and is used for real-time applications.

7. Custom Protocols: In some cases, you may need to expose services using custom or proprietary protocols. Depending on your Ingress controller and its capabilities, you might be able to configure it to handle these custom protocols.

Additionally, Ingress controllers often evolve, so it's essential to refer to the documentation and features of the specific controller you plan to use to ensure compatibility with your service protocols.

WRITTEN ASSIGNMENT 2

Q1. How to deploy lambda function on AWS?

Deploying a Lambda function on AWS involves several steps. Lambda is a serverless compute service that lets you run code without provisioning or managing servers. Here's a step-by-step guide on how to deploy a Lambda function:

Step 1: Create a Lambda Function

1. Log in to the AWS Management Console.
2. Navigate to the Lambda service by searching for it in the AWS Services section.
3. Click the "Create function" button.

Step 2: Configure Your Function

4. Choose "Author from scratch."
5. Fill in the basic information for your function, including the name, runtime, and execution role.
6. For "Execution role," you can create a new role from a template or use an existing role if you have one with the necessary permissions.
7. Click the "Create function" button.

Step 3: Upload Your Code

8. In the "Function code" section, you can upload your code either directly (ZIP file or folder) or by specifying a repository from AWS CodeCommit, Amazon S3, or other options.
9. Configure the handler if needed. The handler is the method that AWS Lambda invokes.

10. Set environment variables if your code requires them.
11. Adjust the runtime settings if necessary.
12. Click the "Deploy" button to upload your code.

Step 4: Define the Trigger

13. In the "Add triggers" section, you can specify event sources that will trigger your Lambda function. This can be an API Gateway, an S3 bucket, an SNS topic, etc.
14. Configure the trigger according to your needs and grant permissions as required.

Step 5: Test Your Function

15. You can test your Lambda function by creating a test event or using a sample test event provided by AWS.
16. Click the "Test" button to run your function and see the results.

Step 6: Monitor and Troubleshoot (Optional)

17. AWS Lambda provides various monitoring and logging options. You can configure CloudWatch alarms, inspect logs, and set up notifications to keep an eye on your function's performance.

Step 7: Save and Deploy the Function

18. After you've configured everything to your satisfaction, click the "Save" button to save your changes.
19. Click the "Deploy" button to make your function live and ready to respond to triggers.

Step 8: Invocation and Scaling

Your Lambda function is now deployed and ready to be invoked by the trigger you configured. AWS Lambda handles the scaling of resources based on the incoming workload automatically.

Q2. What are the deployment options for AWS Lambda?

AWS Lambda offers several deployment options:

1. Code Upload: You can directly upload your function code as a ZIP file or a deployment package when creating or updating your Lambda function.

2. AWS Lambda Layers: You can use Lambda Layers to separate your function code from its dependencies. This allows you to manage and version common libraries independently and share them across multiple functions.

3. AWS SAM (Serverless Application Model): AWS SAM is a framework for building serverless applications. You can define your Lambda functions and their associated resources in a SAM template file, then deploy the entire application using AWS SAM CLI.

4. AWS CloudFormation: You can use AWS CloudFormation templates to define and deploy Lambda functions along with other AWS resources. This enables infrastructure as code (IaC) for your serverless applications.

5. AWS Serverless Application Repository: You can publish and share serverless applications and Lambda functions using the AWS Serverless Application Repository. Users can deploy your applications directly from the repository.

6. AWS CodePipeline: You can integrate AWS Lambda deployment into a continuous integration/continuous deployment (CI/CD) pipeline using AWS CodePipeline. This automates the building, testing, and deployment of your Lambda functions.

7. Container Image Support: AWS Lambda now supports deploying functions as container images, allowing you to package your function and dependencies in a Docker container and deploy them as a Lambda function.

These deployment options provide flexibility and enable you to choose the one that best fits your application architecture and development workflow.

Q3 What are the 3 full deployment modes that can be used for AWS?

In the context of AWS Lambda, there are three primary deployment modes:

1. Automatic Deployment: AWS Lambda provides automatic deployment when you directly upload your function code as a ZIP file or specify a deployment package. This is the most common and straightforward deployment method, and it's suitable for most use cases.

2. Infrastructure as Code (IaC) with CloudFormation: AWS CloudFormation is a service that allows you to define and provision AWS infrastructure and resources in a template. You can use CloudFormation to define your Lambda functions, their associated resources, and any other AWS resources your application needs. When you create or update the CloudFormation stack, it deploys the Lambda functions and other resources defined in the template. This approach is more suitable for complex serverless applications and infrastructure orchestration.

3. Serverless Application Model (AWS SAM): AWS SAM is an open-source framework for building serverless applications. It extends AWS CloudFormation to provide a simplified way to define serverless resources, including Lambda functions. You can use AWS SAM to define your functions and related resources in a SAM template, and then deploy the entire application using AWS SAM CLI. This approach is a balance between the simplicity of direct deployment and the power of CloudFormation for more complex applications.

These deployment modes offer different levels of control and abstraction, allowing you to choose the one that best matches your deployment needs and development workflow.

Q4. What are the 3 components of AWS Lambda?

AWS Lambda consists of three primary components:

1. Function Code: This is the core component of AWS Lambda. It's the code that you want to run in response to events. You can write your code in various supported programming languages (e.g., Node.js, Python, Java, C#, etc.). You can package your code along with its dependencies into a deployment package, which you upload to Lambda.

2. Event Sources: Event sources are triggers that invoke your Lambda function. AWS Lambda supports various event sources, such as Amazon S3, Amazon DynamoDB, Amazon SNS, AWS API Gateway, and more. When an event occurs in one of these services, it triggers the execution of your Lambda function.

3. Execution Environment: AWS Lambda manages the execution environment for your function. It automatically scales and provisions the necessary infrastructure to run your code. You don't need to worry about server provisioning or resource management. AWS Lambda also provides runtime support for various programming languages, so your code

runs in an isolated environment with the necessary resources to execute.