

DevSecOps: Blue-Green Deployment of Swiggy-Clone on AWS ECS with AWS Code Pipeline



Introduction:

In the realm of modern software development, DevSecOps practices are gaining prominence for their emphasis on integrating security seamlessly into the software development lifecycle. One critical aspect of this approach is implementing efficient deployment strategies that not only ensure reliability but also maintain security standards. In this blog post, we will delve into the concept of Blue-Green deployment and demonstrate how to apply it to a Swiggy-clone application hosted on AWS ECS (Elastic Container Service) using AWS Code Pipeline.

What is Blue-Green Deployment?

Blue-Green deployment is a technique used to minimize downtime and risk during the release of new versions of an application. In this approach, two identical production environments, termed 'Blue' and 'Green', are maintained. At any given time, only one environment (e.g., Blue) serves live traffic while the other (e.g., Green) remains idle. When a new version is to be deployed, the new version is deployed to the idle environment (Green). Once the deployment is validated, traffic is seamlessly switched to the updated environment (Green), allowing for quick rollback to the previous version if issues arise.

Setting up AWS ECS for Swiggy-Clone:

To demonstrate Blue-Green deployment, we'll use AWS ECS to host our Swiggy-clone application. ECS is a highly scalable container orchestration service provided by AWS.

Implementing Blue-Green Deployment with AWS CodePipeline:

AWS CodePipeline is a fully managed continuous integration and continuous delivery (CI/CD) service that automates the build, test, and deployment phases of your release process. **Let's see how to set up a Blue-Green deployment pipeline using AWS CodePipeline:**

1. Source Stage: Connect your CodePipeline to your source code repository (e.g., GitHub). Trigger the pipeline when changes are detected in the repository.

2. Build Stage: Use AWS CodeBuild to build your Swiggy-clone Docker image from the source code. Run any necessary tests during this stage.

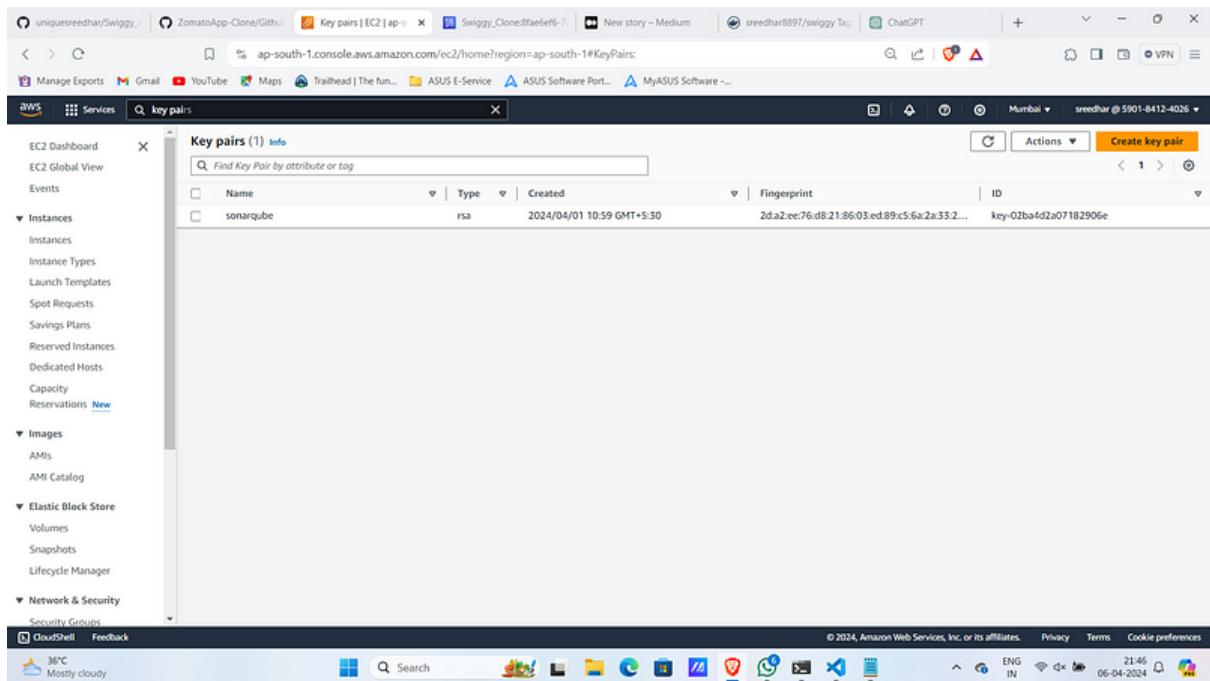
3. Deploy Stage: Configure AWS CodeDeploy for ECS to manage the deployment of your application to ECS clusters. Here's where Blue-Green deployment strategy comes into play:

- A. Define two ECS services: Blue and Green.
- B. Use CodeDeploy to deploy the new version of your Swiggy-clone application to the Green service.
- C. After deployment, automate the ALB routing to gradually shift traffic from the Blue service to the Green service based on predefined health checks.
- D. Monitor the deployment process and rollback automatically if issues occur during the transition.

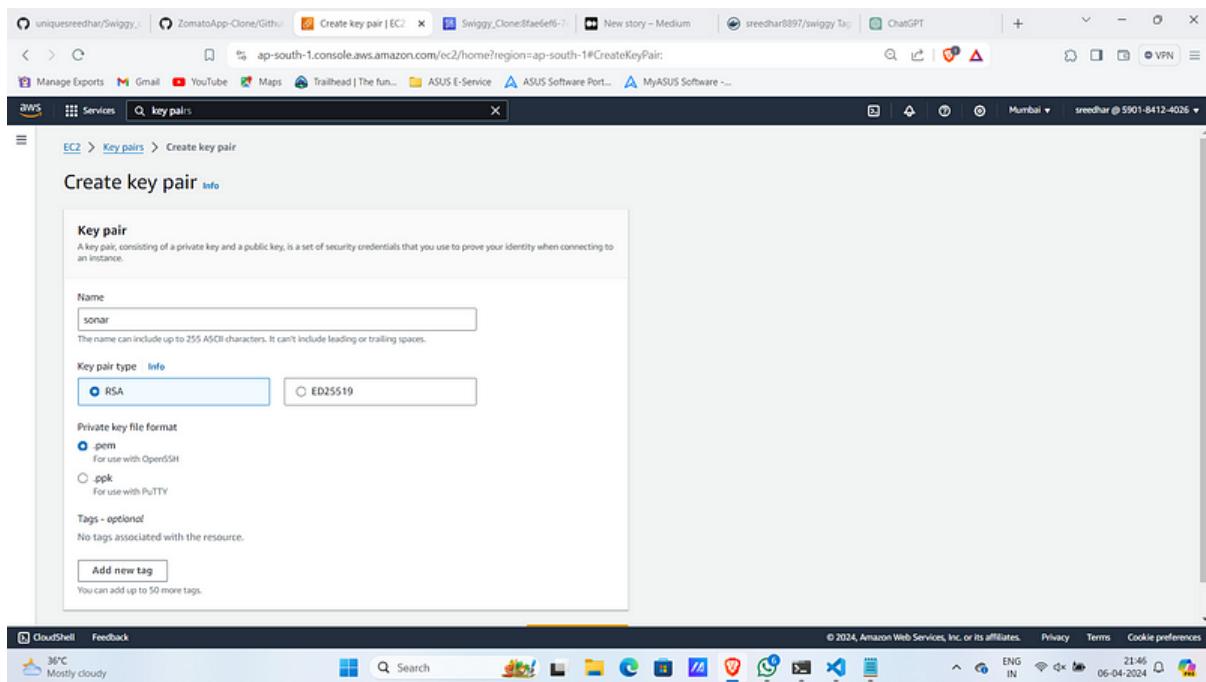
GitHub Source Code Repo: [HERE](#)

Step:-1 : Create a Sonar Server

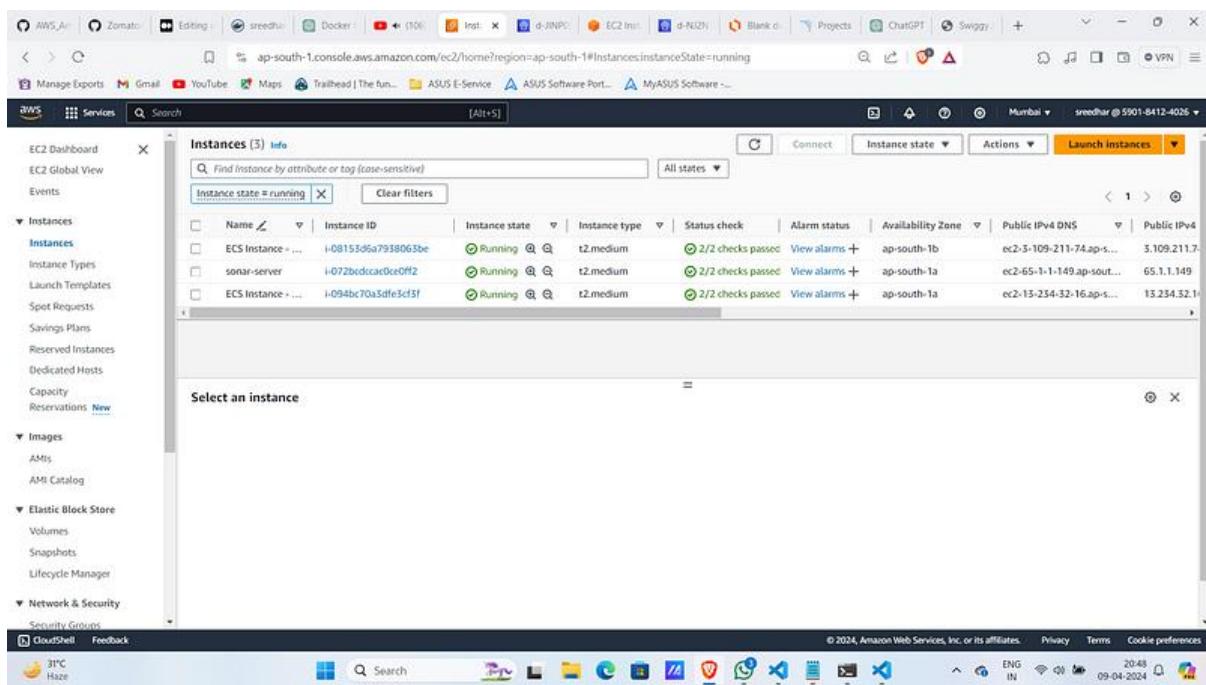
1. To run Static Code Analysis we need a sonar server.
2. Create a key-pair for this purpose.
 - i. Navigate to key-pairs in AWS Console and click on “Create key- pair”.



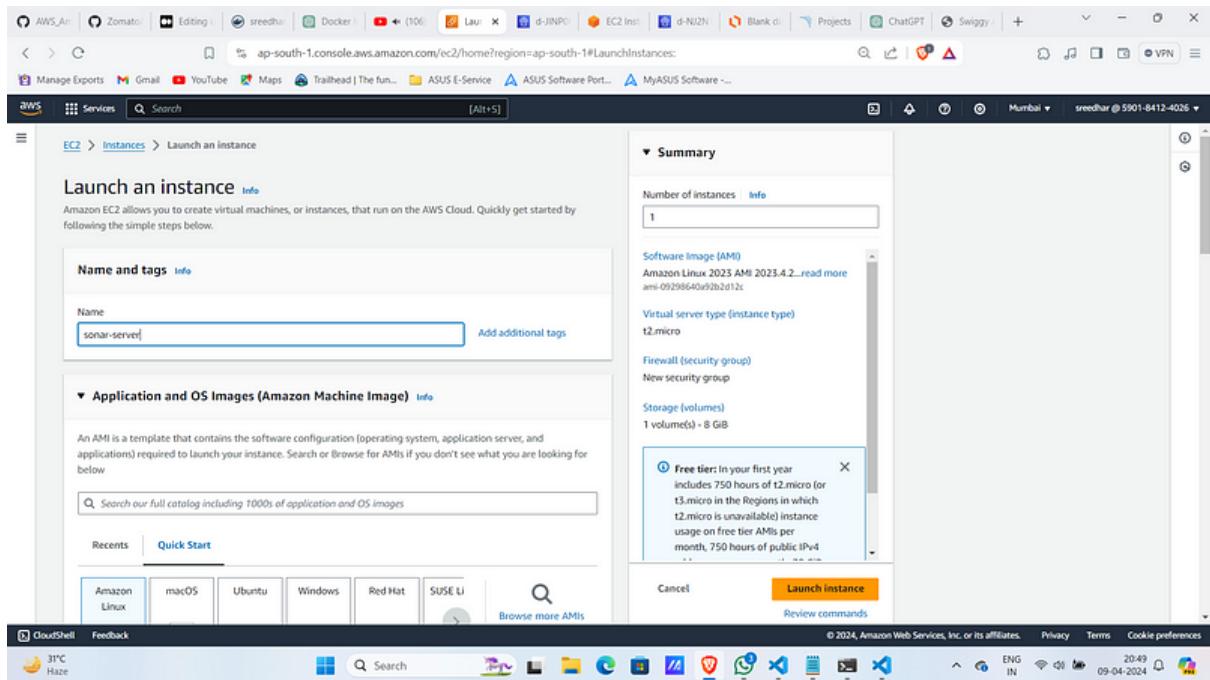
- ii. Provide a name and select key-pair type as .pem then click on create.
The .pem file will be downloaded to your local.



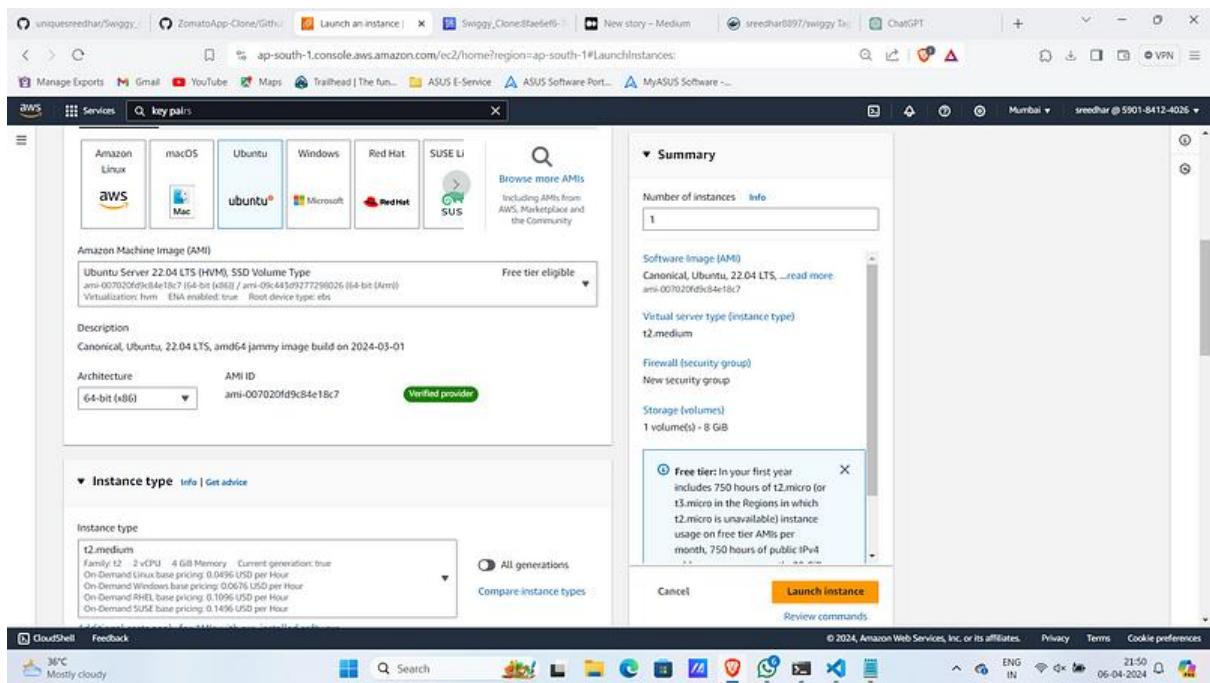
3. Navigate to EC2 console and click on "Launch Instance".



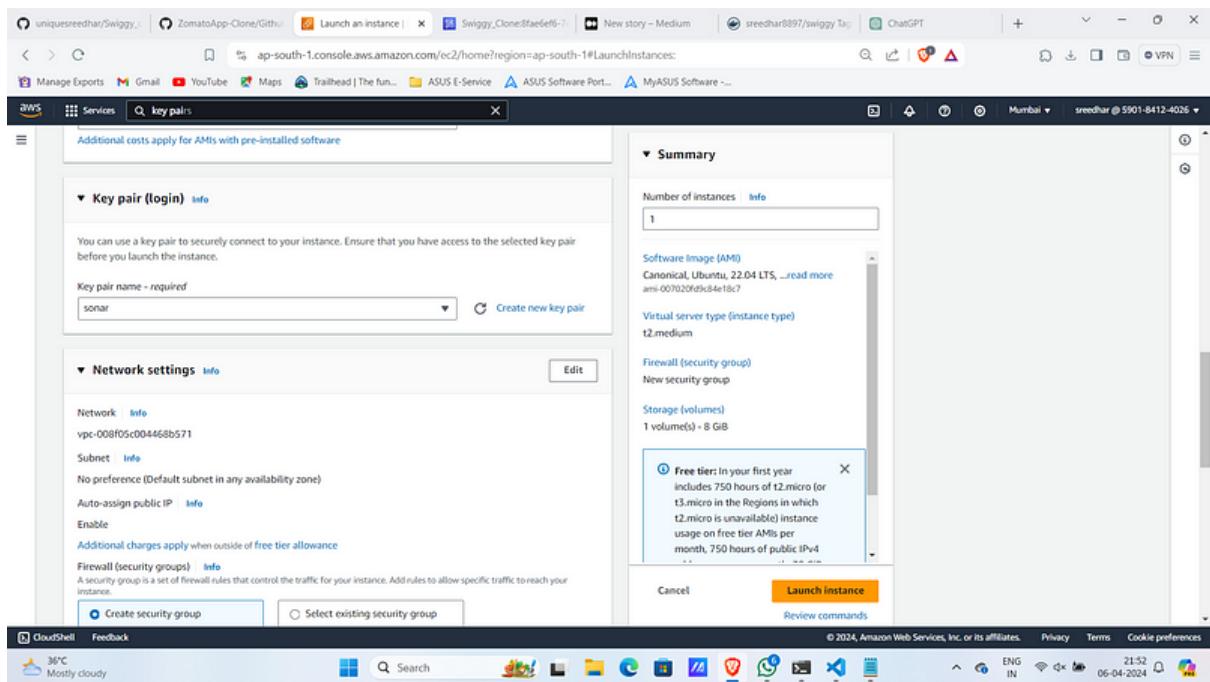
4. Give a name for it.



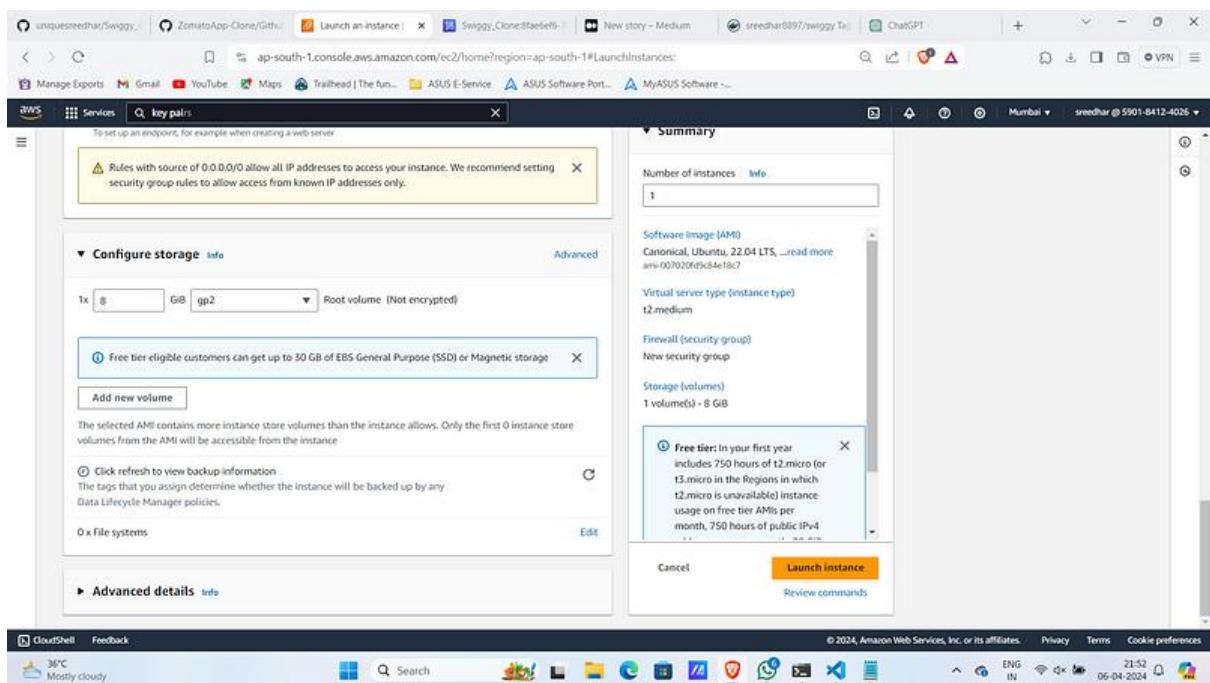
5. Select AMI as “Ubuntu” and instance type as “t2.medium”.



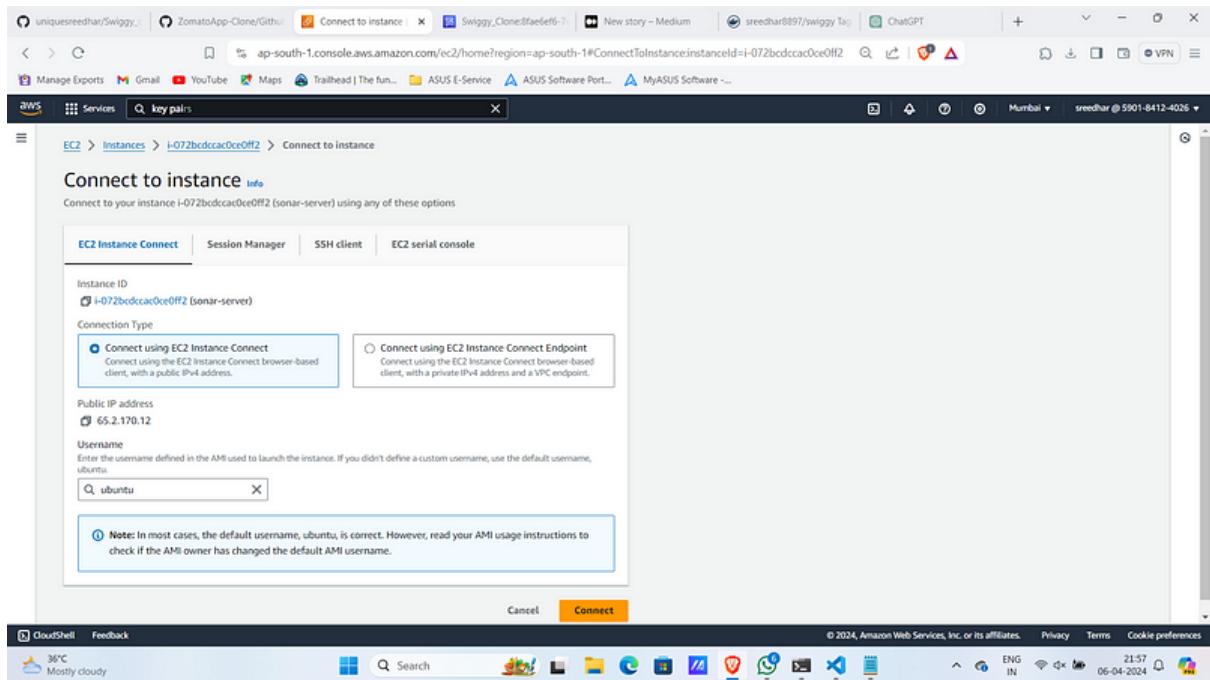
6. Under key-pair select the created one.



7. Click on “Launch Instance” by leaving other things as default.



8. Once the instance is up and running select it and click on “connect”. You can use EC2 instance connect or SSH with the pem file you downloaded.



9. Install docker.

Installing Docker

sudo apt update

sudo apt install docker.io -y

sudo usermod -aG docker ubuntu

sudo systemctl restart docker

sudo chmod 777 /var/run/docker.sock

```
ubuntu@ip-172-31-35-128:~$ sudo apt update
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease [119 kB]
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [109 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64-restricted [217 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64-hybrid [217 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Translation-en [112 kB]
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [6372 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1519 kB]
Get:11 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:12 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [293 kB]
Get:13 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1648 kB]
Get:14 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [275 kB]
Get:15 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1060 kB]
Get:16 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [244 kB]
Get:17 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [22.1 kB]
Get:18 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [49.6 kB]
Get:19 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [12.0 kB]
Get:20 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [472 kB]
Get:21 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [67.1 kB]
Get:22 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [11.0 kB]
Get:23 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Metadata [308 kB]
Get:24 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-f Metadata [116 B]
Get:25 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [28.4 kB]

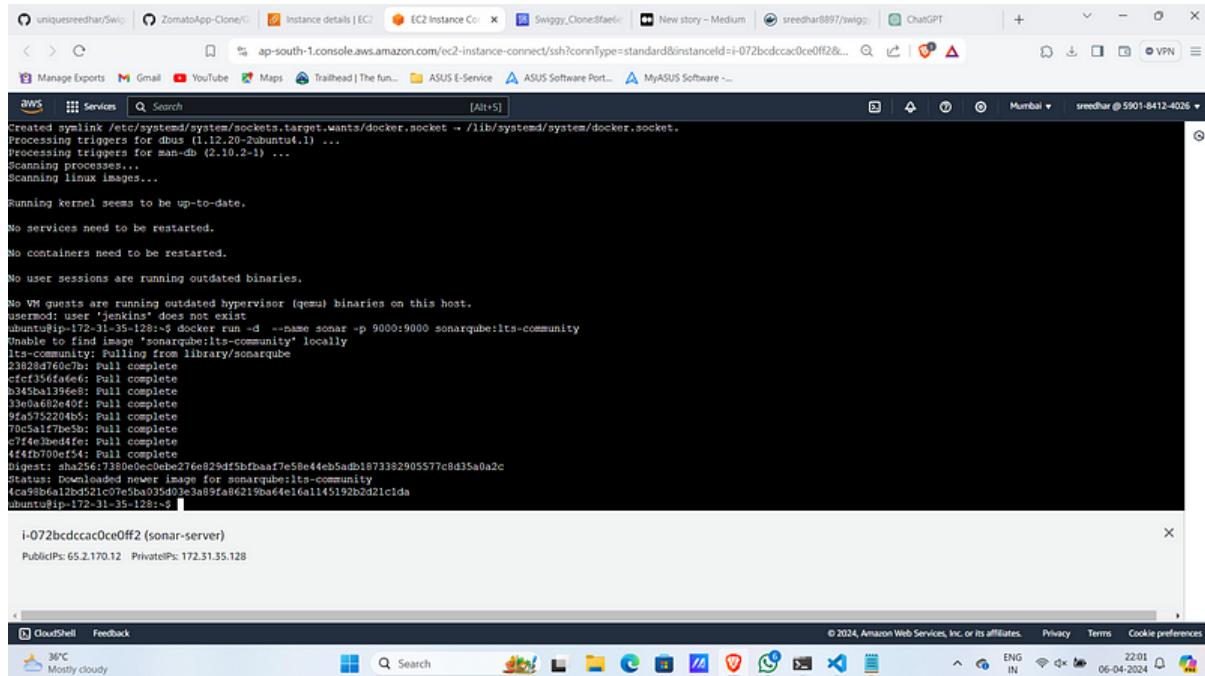
i-072bcdccac0ce0ff2 (sonar-server)
PublicIPs: 65.2.170.12 PrivateIPs: 172.51.55.128
```

10. Run sonarqube as a docker container.

```
# Run Docker Container of Sonarqube
```

```
#!/bin/bash
```

```
docker run -d --name sonar -p 9000:9000 sonarqube:its-community
```

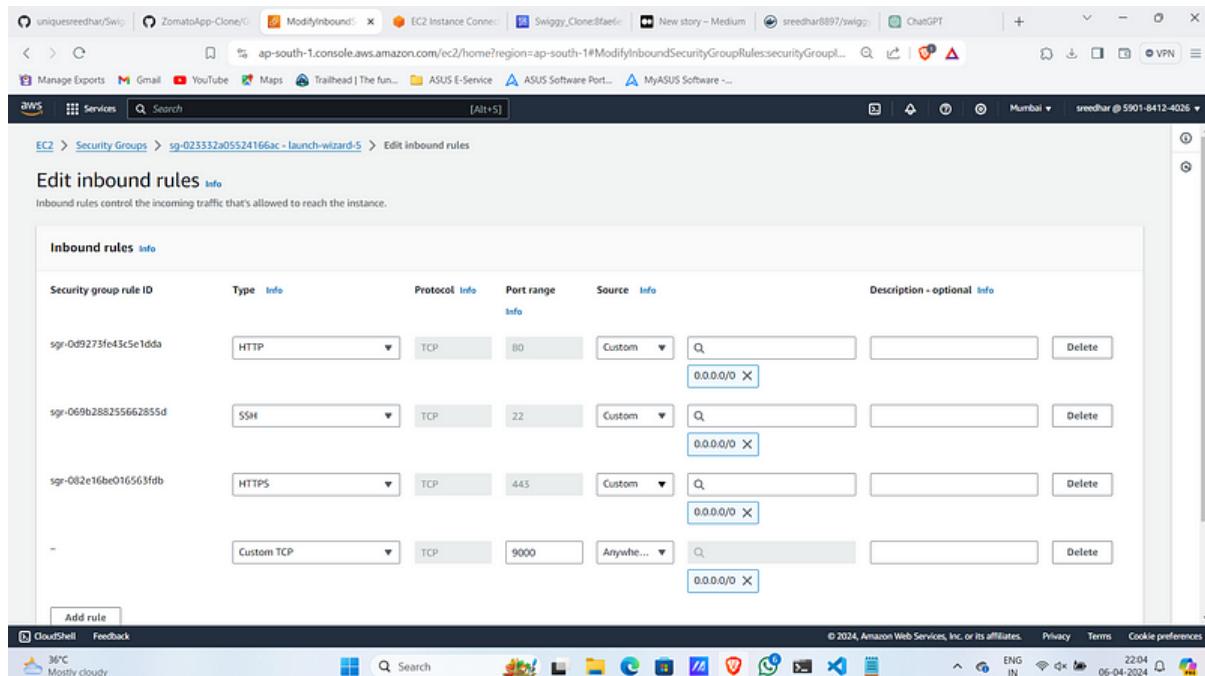


```
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Processing triggers for dbus (1.12.20-2ubuntu04.1) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...
Running kernel seems to be up-to-date.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
usermod: user 'jenkins' does not exist
ubuntu@ip-172-31-35-128:~$ docker run -d --name sonar -p 9000:9000 sonarqube:its-community
Unable to find image 'sonarqube:its-community' locally
its-community: Pulling from library/sonarqube
23828d760c7b: Pull complete
cfef156fa0f0: Pull complete
3d30a66240f0: Pull complete
330a66240f0: Pull complete
5fa5752204b5: Pull complete
70ca5a1f7ba5b1: Pull complete
c774e3bddefe: Pull complete
tf4fb700ef54: Pull complete
Digest: sha256:7380e0ec0eb276e829df5bfbaaf7e58e44eb5ad1b1873382905577c8d35a0a2c
Status: Downloaded newer image for sonarqube:its-community
4ca99b6a12b0d521c07e5ba35d03e3a89fa86219ba4e16a1145192b2d21c1da
ubuntu@ip-172-31-35-128:~$ 
```

i-072bcdccac0ce0ff2 (sonar-server)
PublicIPs: 65.2170.12 PrivateIPs: 172.31.35.128

11. Ensure that port 9000 is opened in security group of that particular instance.



Inbound rules [Info](#)

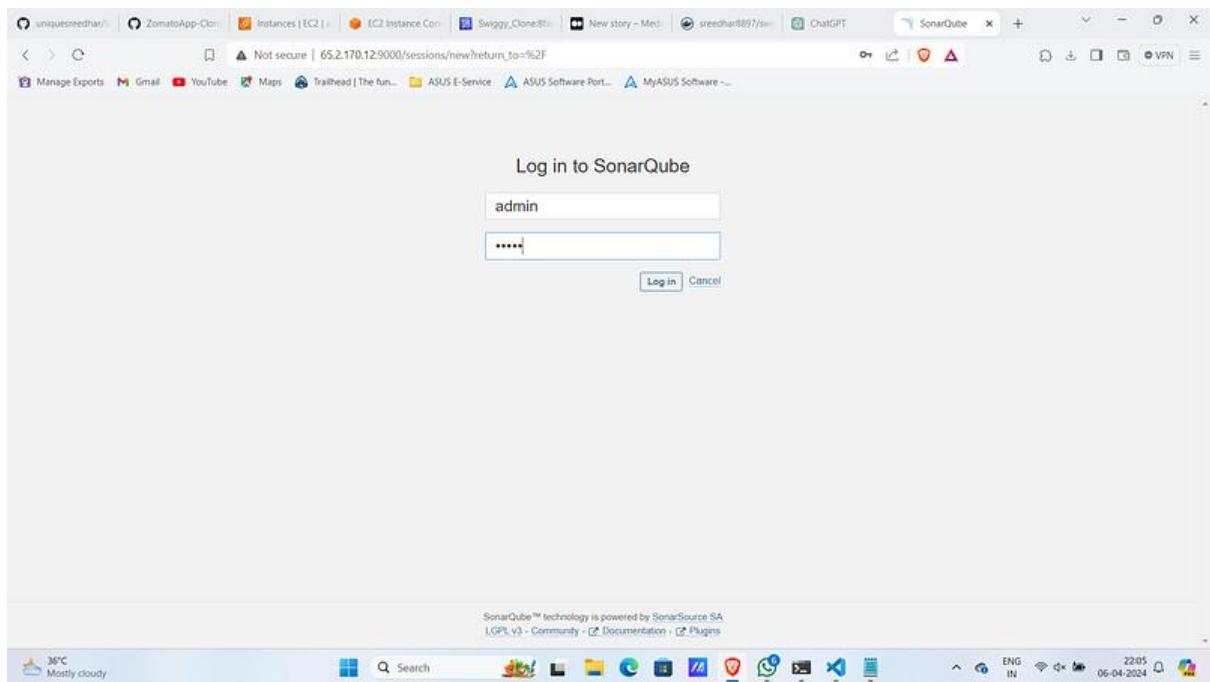
Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0d9273fe43c5e1dda	HTTP	TCP	80	Custom	0.0.0.0/0
sgr-069b288255662855d	SSH	TCP	22	Custom	0.0.0.0/0
sgr-082e16be016563fdb	HTTPS	TCP	443	Custom	0.0.0.0/0
-	Custom TCP	TCP	9000	Anywhere	0.0.0.0/0

[Add rule](#)

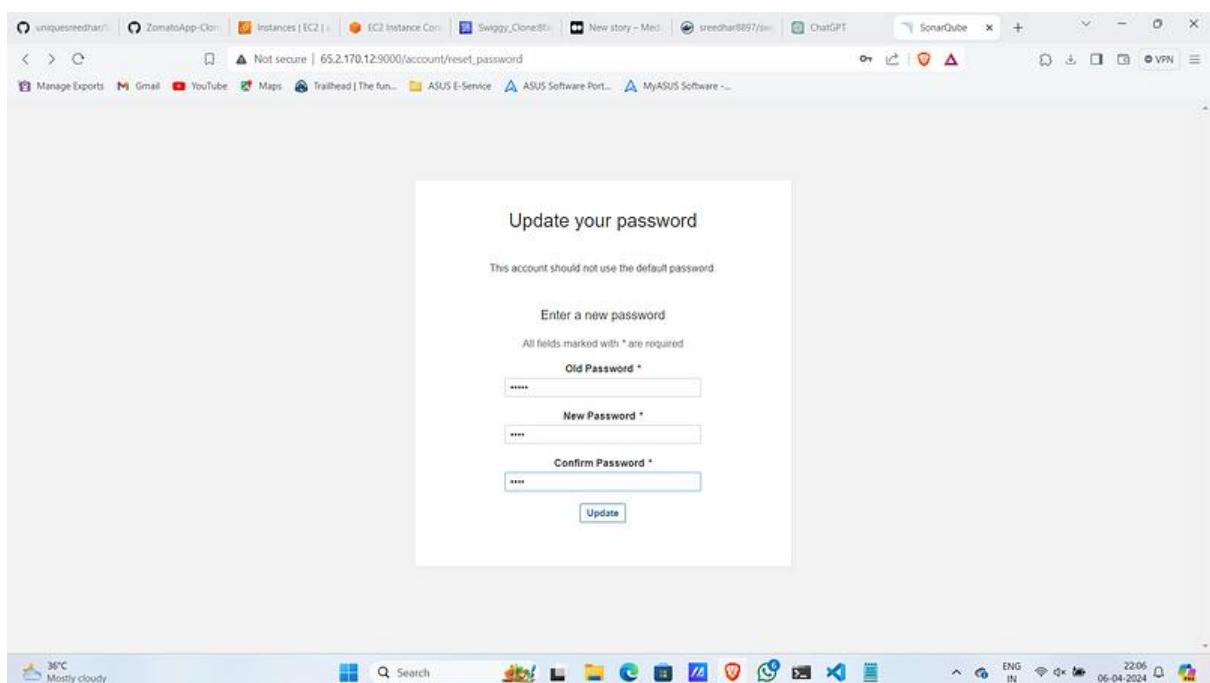
12. Access SonarQube on <public_ip>:9000

Username & Password: admin

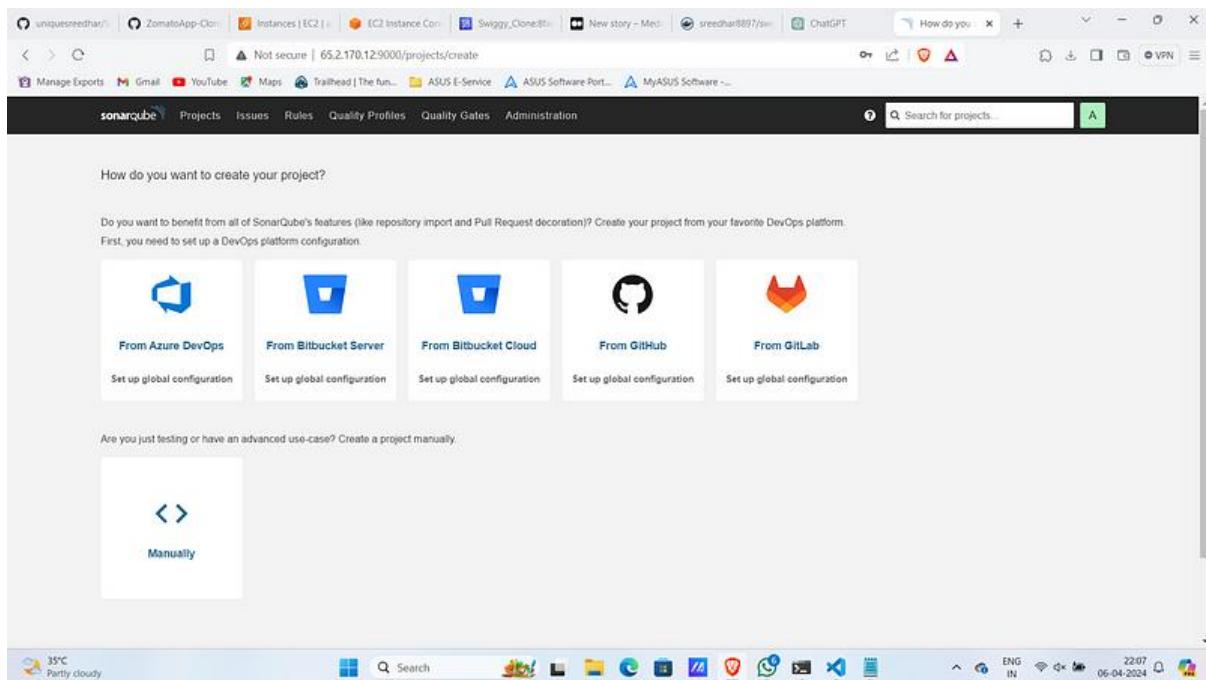


Step:2 :- SonarQube Set-Up.

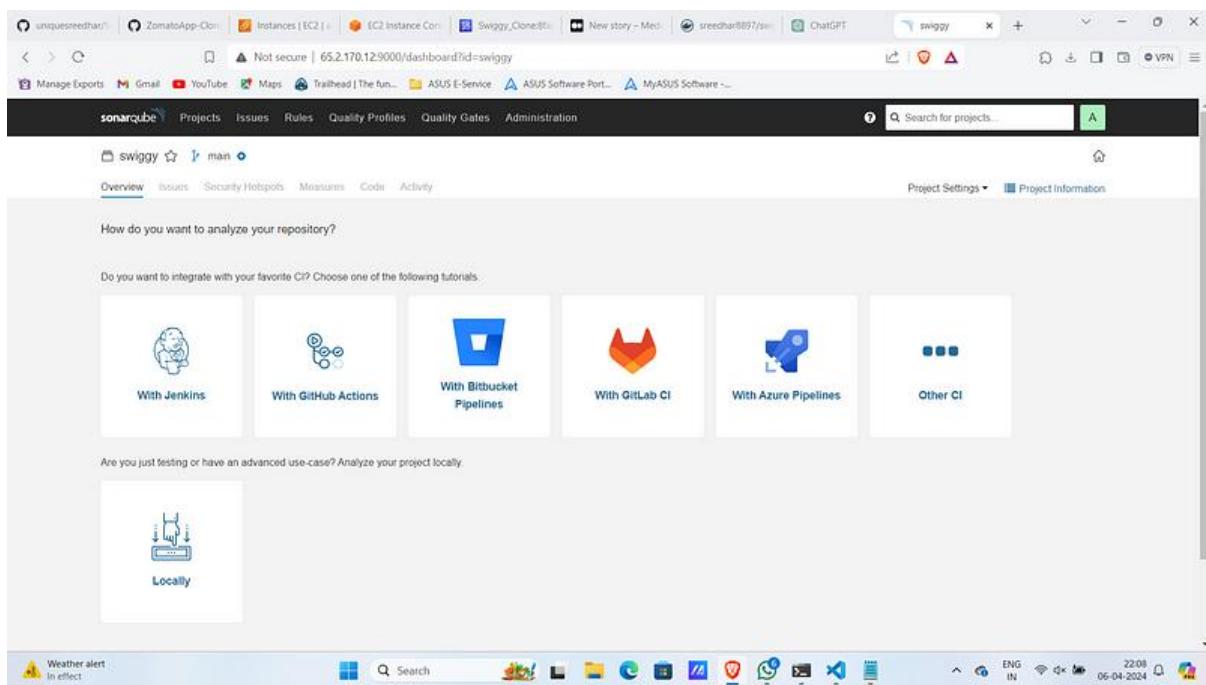
1. After getting log-in create a custom password.



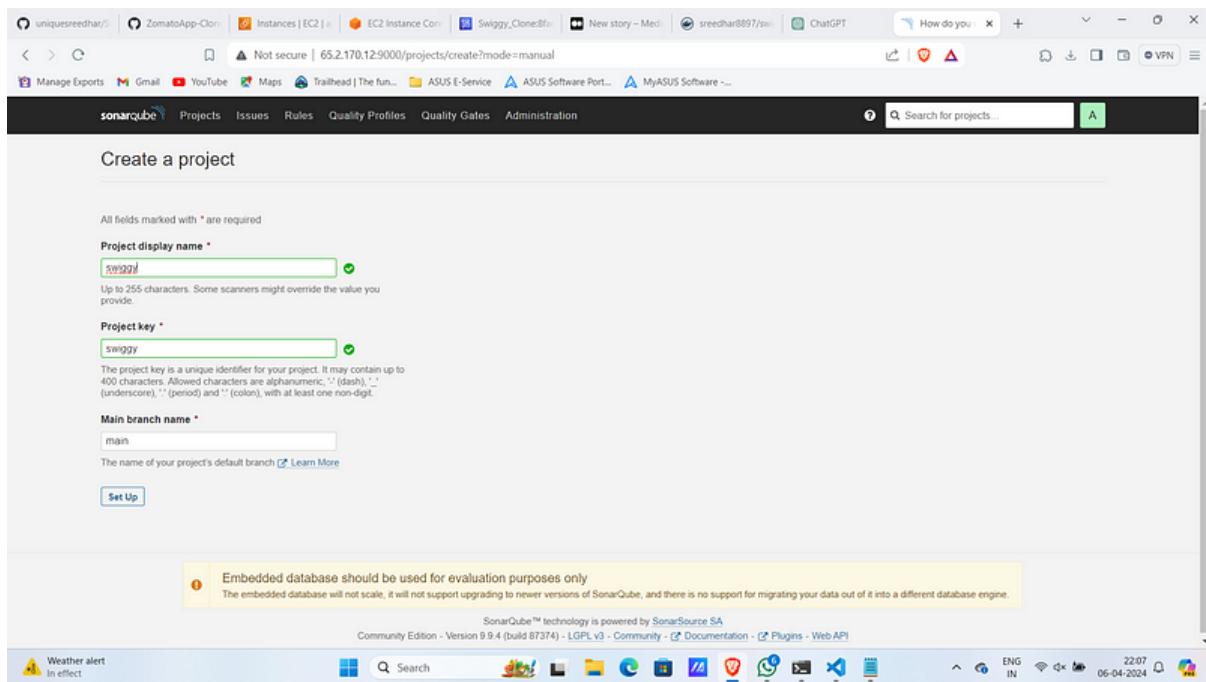
2. Click on “manually” (<>).



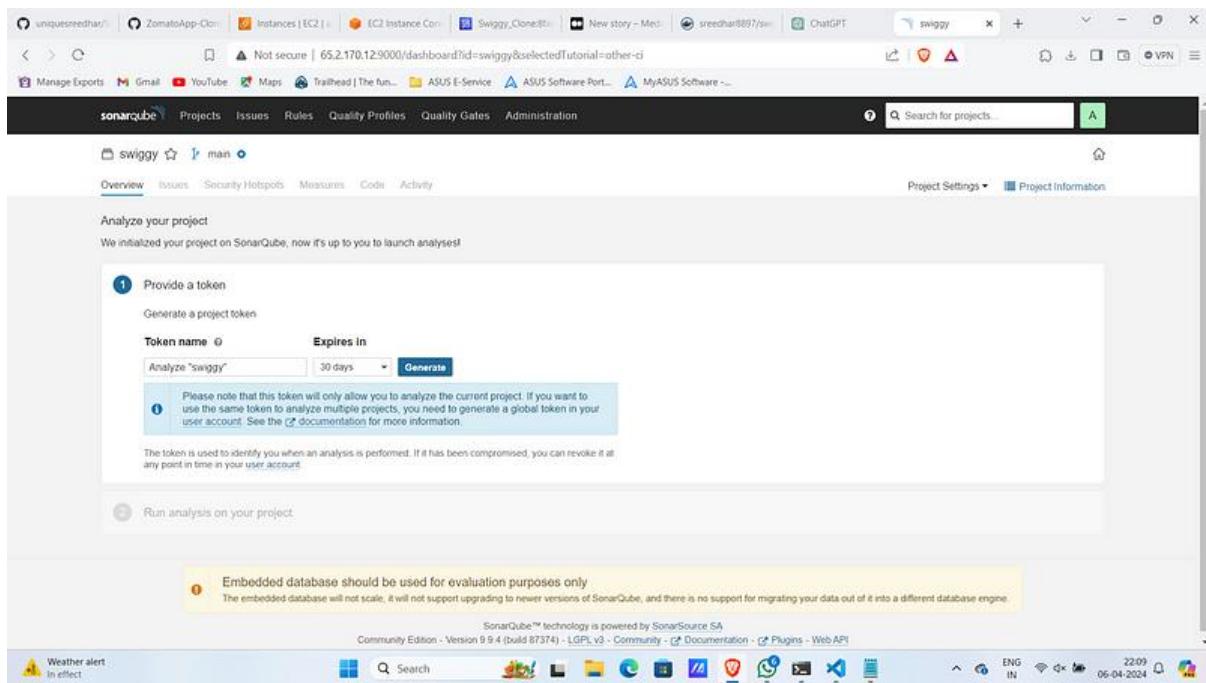
3. Click on “locally”.



4. Provide a name for it.



5. Click on “Set Up” .



6. Click on “Generate” and create one.

Analyze your project
We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token
Analyze "swiggy": sqp_5afbf4a274052e9b935252c07143915ada0c932b

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your user account.

Continue

2 Run analysis on your project

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.

SonarQube™ technology is powered by SonarSource SA
Community Edition - Version 9.9.4 (build 87374) - LGPL v3 - Community - Documentation - Plugins - Web API

7. Then Under code select other and os as linux. Then Copy sonar token.

Run analysis on your project

What option best describes your build?
Maven .NET Other (for JS, TS, Go, Python, PHP, ...)

What is your OS?
Linux Windows macOS

Download and unzip the Scanner for Linux

```
export SONAR_SCANNER_VERSION=4.7.0.2747
export SONAR_SCANNER_HOME=$HOME/.sonar/sonar-scanner-$SONAR_SCANNER_VERSION-linux
curl --create-dirs -LsLo $HOME/.sonar/sonar-scanner.zip https://binaries.sonarsource.com/Distribution/sonar-scanner-c11/sonar-scanner-c11-$SONAR_SCANNER_VERSION-linux.zip
unzip -o $HOME/.sonar/sonar-scanner.zip -d $HOME/.sonar/
export PATH=$SONAR_SCANNER_HOME/bin:$PATH
export SONAR_SCANNER_OPTS="-server"
```

Configure a SONAR_TOKEN environment variable in your CI settings

- Add an environment variable called SONAR_TOKEN
- Give it the following value: sqp_5afbf4a274052e9b935252c07143915ada0c932b

Execute the Scanner

Running a SonarQube analysis is straightforward. You just need to execute the following commands in your project's folder:

```
sonar-scanner \
-Dsonar.projectKey=swiggy \
```

8. In the AWS console search for “Systems Manager” and then “Parameter Store”.

The screenshot shows the AWS Systems Manager console with the URL ap-south-1.console.aws.amazon.com/systems-manager/home?region=ap-south-1#. The main heading is "AWS Systems Manager" with the subtext "Gain Operational Insight and Take Action on AWS Resources.". Below this is a "Get Started with Systems Manager" button. The left sidebar contains sections for Operations Management (Explorer, OpsCenter, CloudWatch Dashboard, Incident Manager), Application Management (Application Manager, AppConfig, Parameter Store), and Change Management (Change Manager, Automation, Change Calendar, Maintenance Windows). The central area features three cards: "Group your resources" (Icon: folder with files), "View insights" (Icon: document with charts), and "Take Action" (Icon: laptop with gear). To the right is a "More resources" sidebar with links to Documentation, API reference, and FAQs. The bottom navigation bar includes "CloudShell", "Feedback", and the AWS logo.

9. Click on “create parameter”.

The screenshot shows the AWS Systems Manager Parameter Store with the URL ap-south-1.console.aws.amazon.com/systems-manager/parameters/?region=ap-south-1&tab=Table. A blue banner at the top says "Introducing cross-account parameter sharing" and "You can now use AWS Resource Access Manager to share parameters with other accounts in your organization. Learn more". The main area shows a table titled "My parameters" with columns: Name, Tier, Type, and Last modified. The table lists four parameters: "/oid/docker-credentials/password", "/oid/docker-credentials/username", "/oid/docker-registry/url", and "/oid/sonar/sonar-token". Each row has a "View details", "Edit", "Delete", and "Create parameter" button. The bottom navigation bar includes "CloudShell", "Feedback", and the AWS logo.

9. Give a name for it.

Note: This must be edited in buildspec.yaml file.

The screenshot shows the 'Create parameter' page in the AWS Systems Manager Parameter Store. The 'Name' field is set to '/cicd/sonar/sonar-token'. The 'Type' is selected as 'String'. The 'Value' field contains the copied Sonar token: sqp_Safbf4a274052e9b935252c07143915ada0c932b. The 'Data type' is set to 'text'.

10. Provide the copied token in the value section.

The screenshot shows the 'Create parameter' page in the AWS Systems Manager Parameter Store. The 'Name' field is set to '/cicd/sonar/sonar-token'. The 'Type' is selected as 'String'. The 'Value' field contains the copied Sonar token: sqp_Safbf4a274052e9b935252c07143915ada0c932b. The 'Data type' is set to 'text'.

11. Similarly create parameters for Docker Username, Password and URI.

#In my case

Parameter name: /cicd/sonar/sonar-token Value: <sonar_token>

Parameter name: /cicd/docker-credentials/username Value: <docker_username>

Parameter name: /cicd/docker-credentials/password Value: <docker_password>

Parameter name: /cicd/docker-registry/url Value: docker.io

Note: You need to give your Sonar URL and Project Key in buildspec.yaml

Step:-3 : Create AWS Code Build Project

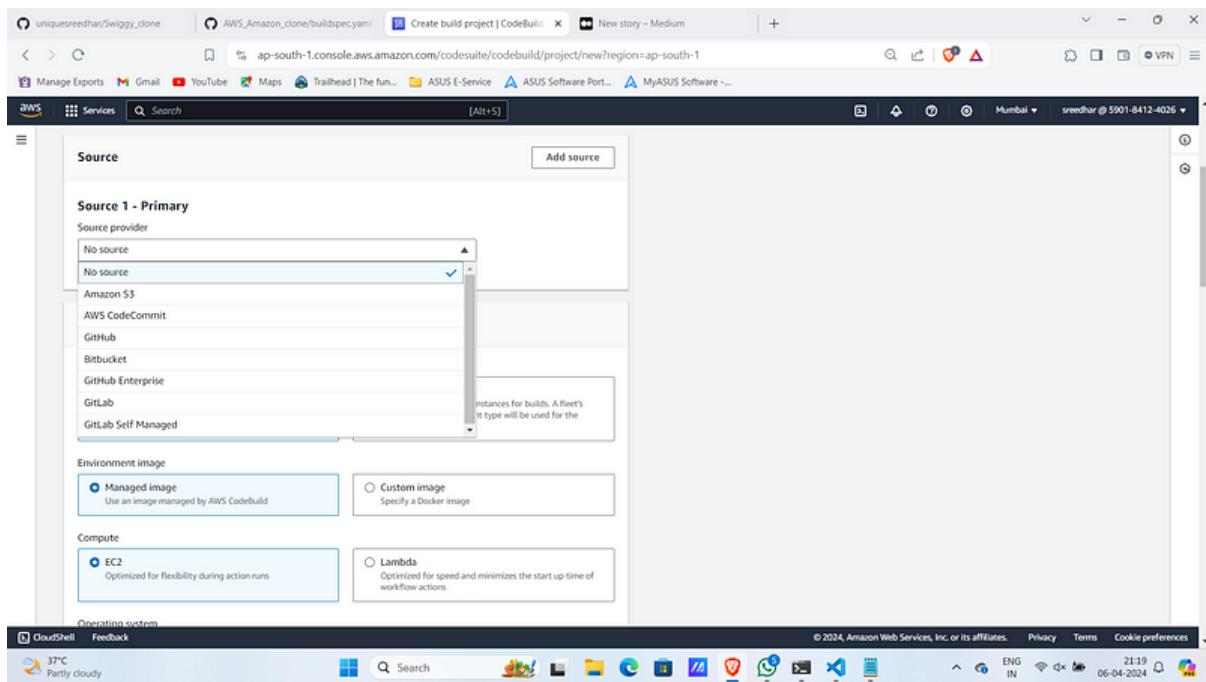
1. Navigate to AWS Codebuild console and click on “create project”.

The screenshot shows the AWS CodeBuild console with the URL <https://ap-south-1.console.aws.amazon.com/codesuite/codebuild/projects?region=ap-south-1&projects-meta=ey/mjp7lnle...>. The left sidebar is titled 'CodeBuild' and includes sections for Source (CodeCommit), Artifacts (CodeArtifact), Build (CodeBuild), Pipeline (CodePipeline), Deploy (CodeDeploy), and Settings. The main content area is titled 'Build projects' and lists two existing projects: 'Swiggy_Clone' and 'amazon_clone'. Both projects are associated with GitHub as the source provider, have 'uniquereedhar/Swiggy_clone' and 'uniquereedhar/AWS_Amazon_clone' as their repositories, and show a 'Succeeded' status with build logs. A prominent orange 'Create project' button is located at the top right of the list table.

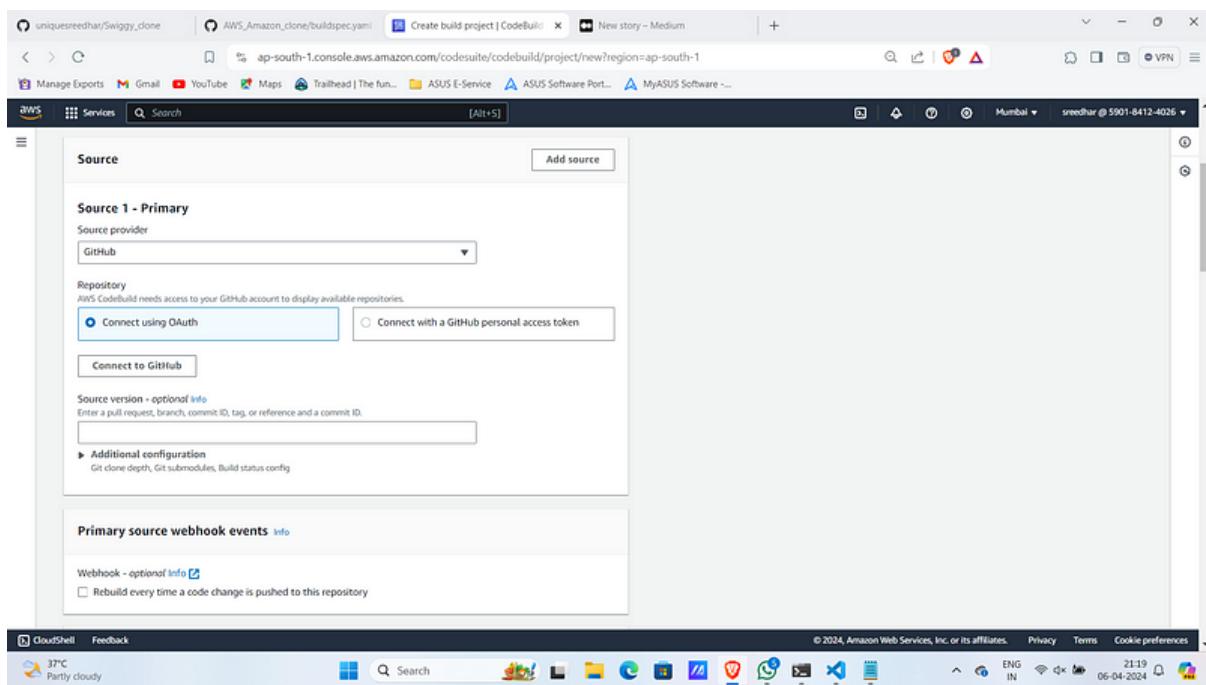
2. Provide a name for it.

The screenshot shows the 'Create build project' configuration page with the URL <https://ap-south-1.console.aws.amazon.com/codesuite/codebuild/project/new?region=ap-south-1>. The left sidebar shows the 'Developer Tools' section. The main form is titled 'Create build project' and contains a 'Project configuration' section where 'Project name' is set to 'Swiggy_Clone'. Below this, there's an 'Additional configuration' section with options for Description, Build badge, Concurrent build limit, and tags. The 'Source' section is expanded, showing 'Source 1 - Primary' with 'Source provider' set to 'No source'. The 'Environment' section is also present. The bottom of the screen shows a Windows taskbar with various icons and system status.

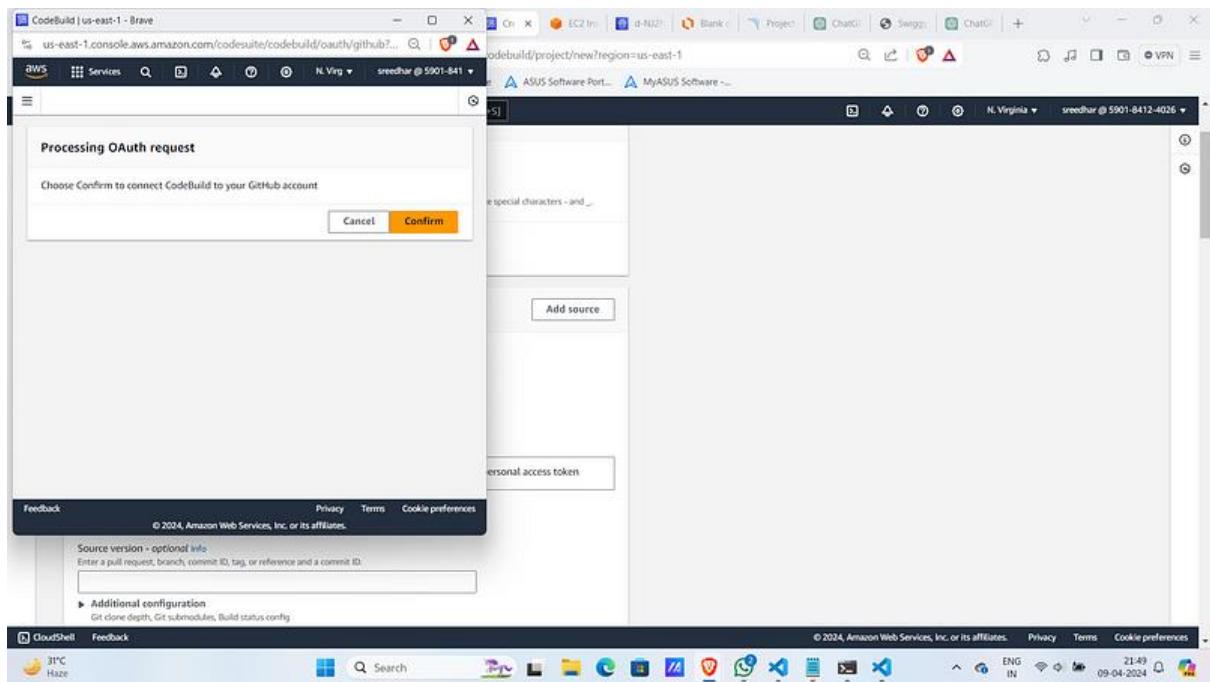
3. Under source select github as a source provider.



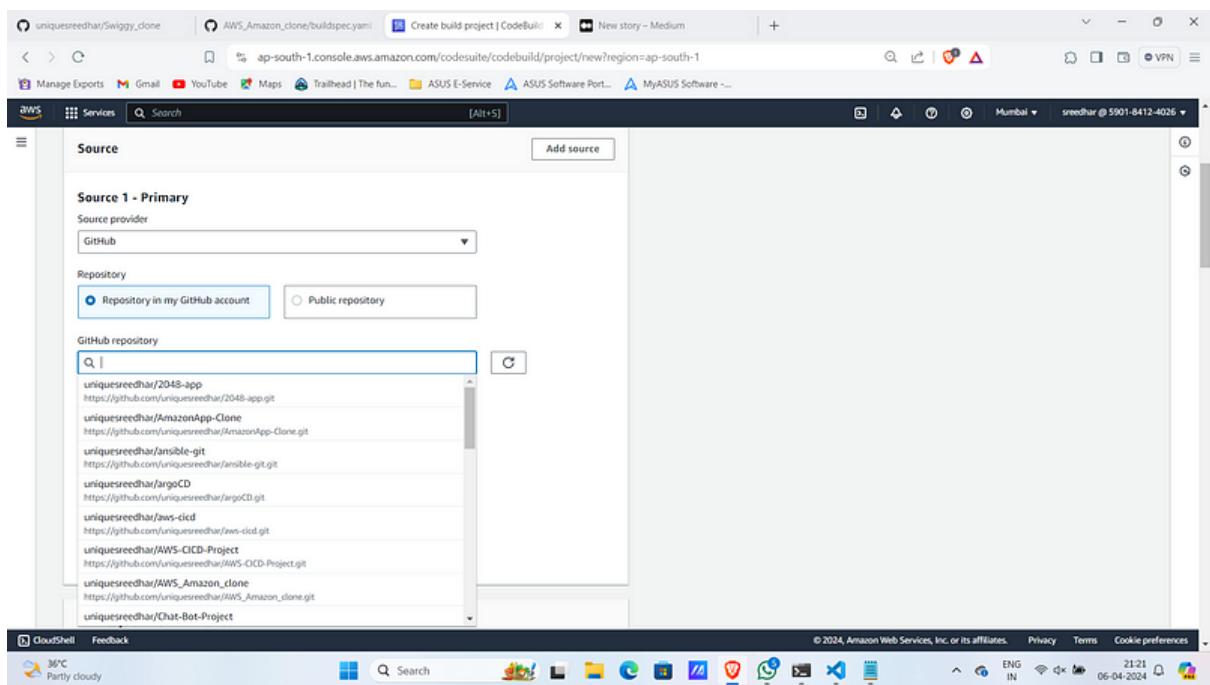
4. Select Connect using OAuth.



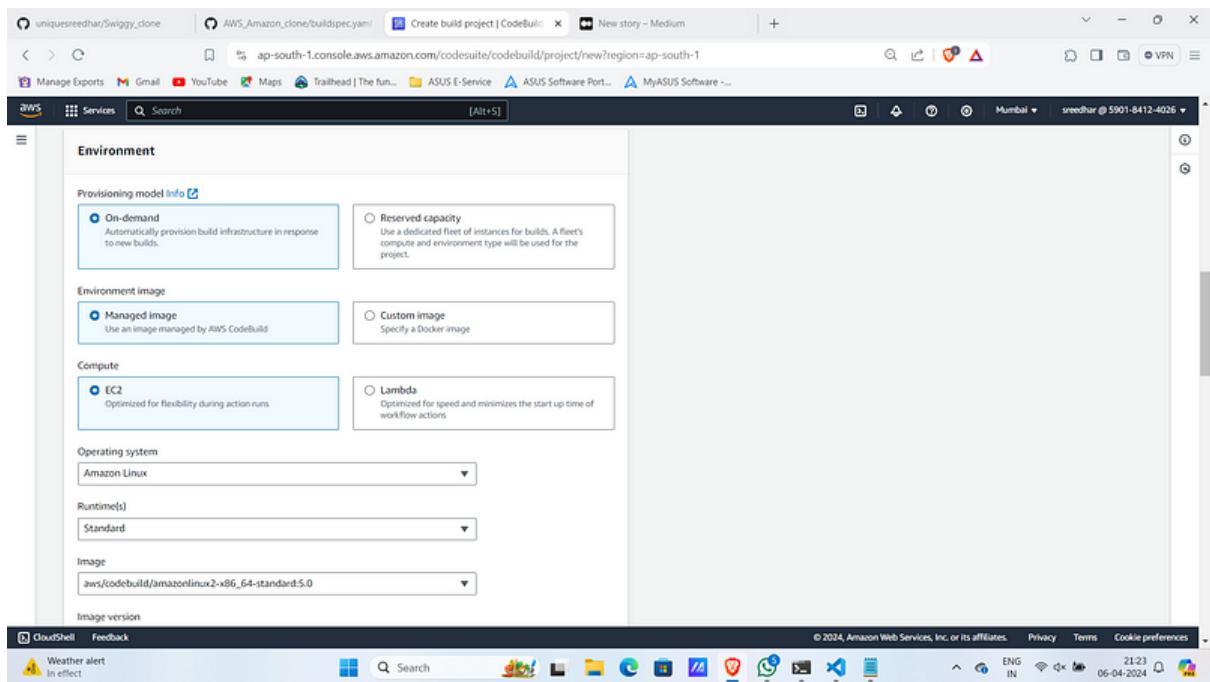
5. After this it will ask for permissions and github login do all the stuff.



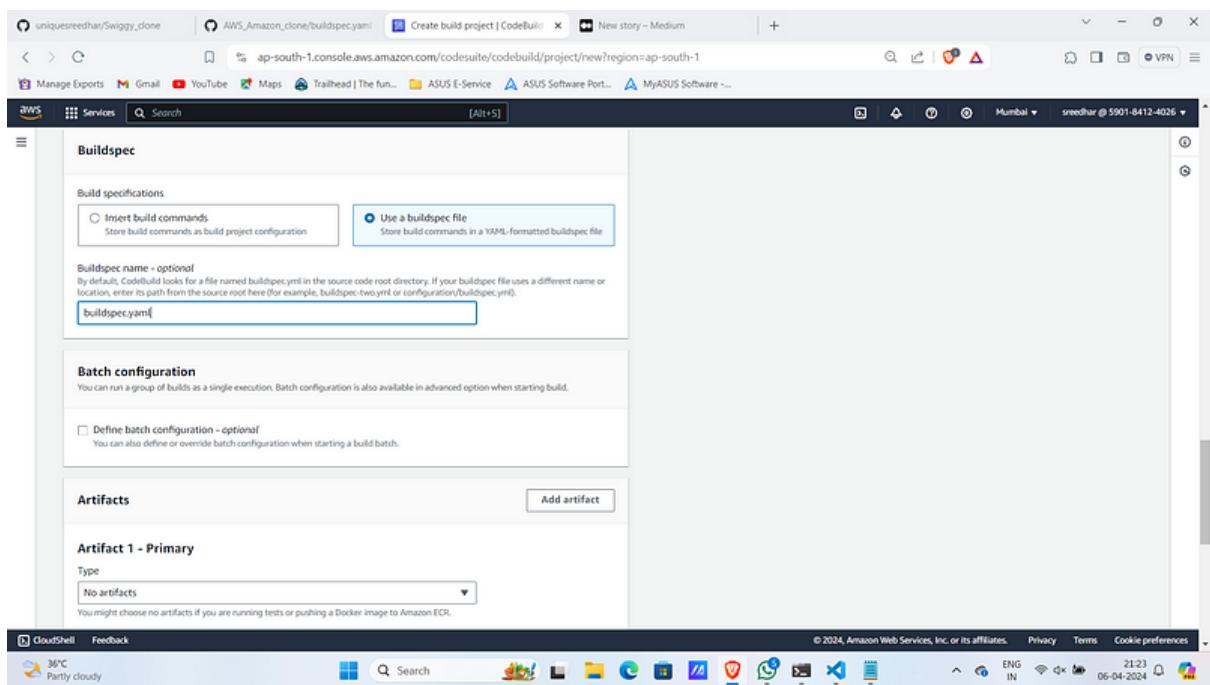
6. Under GitHub repo, select the one your application code relies.



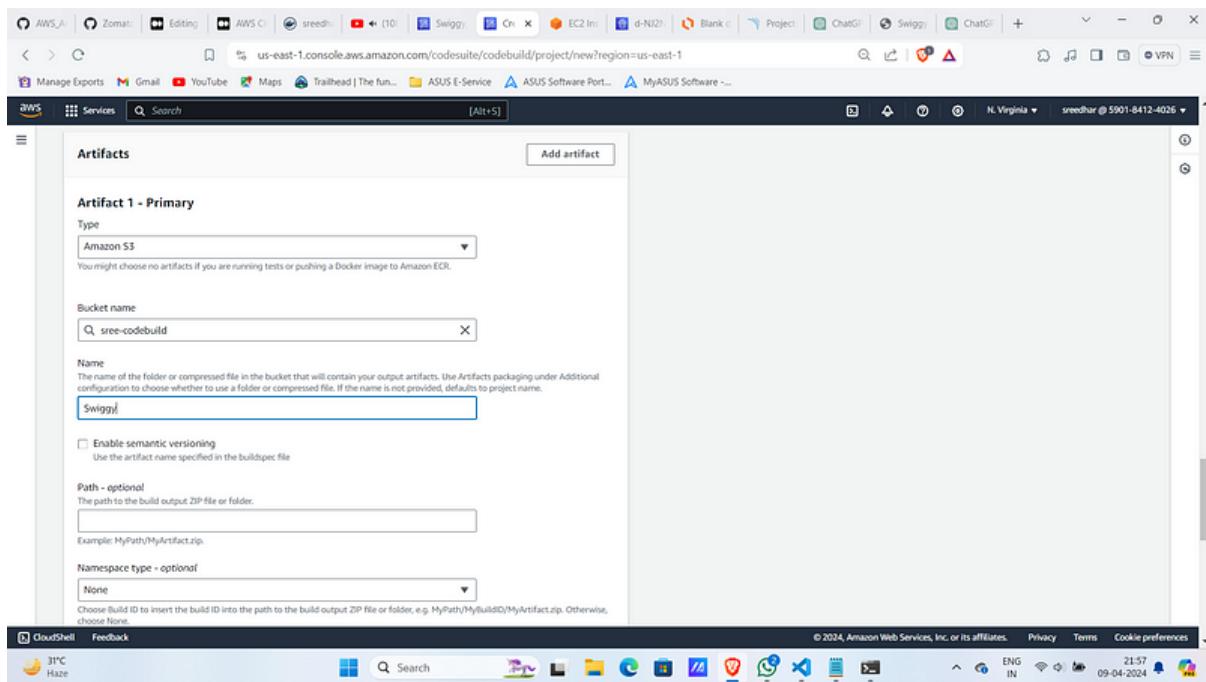
7. Under Environment leave all of them as default.



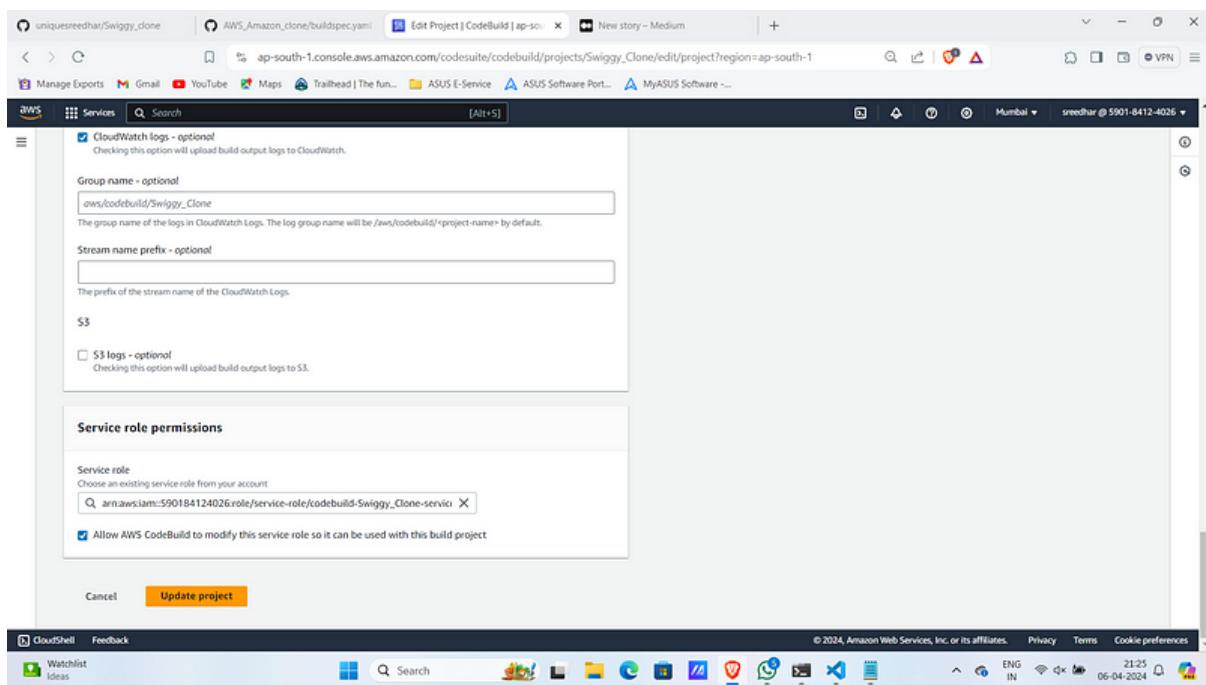
8. Under Buildspec select “Use a buildspec file” and provide the name as “buildspec.yaml”.



9. Under Artifacts Use an already created s3 bucket.



10. Click on “Update project”.



11. In IAM click on the role that the codebuild created.

The screenshot shows the AWS IAM Roles page. The left sidebar is collapsed. The main area displays the 'codebuild-Swiggy_Clone-service-role' details. The 'Permissions' tab is selected, showing one managed policy attached: 'CodeBuildBasePolicy-Swiggy_Clone-ap-south-1'. The policy is listed as 'Customer managed'.

12. Give “AmazonSSMFullAccess” to access the parameters in Systems Manager and “AWSS3FullAccess” to upload the artifacts.

The screenshot shows the 'Add permissions' dialog for the 'codebuild-Swiggy_Clone-service-role'. The 'Current permissions policies' section shows one policy attached. The 'Other permissions policies' section lists two AWS managed policies: 'AmazonSSMFullAccess' and 'AWSDataLifecycleManagerSSMFullAccess'. The 'AmazonSSMFullAccess' policy is selected.

13. Click on “Start build”.

Upon successful build it will look like:

Developer Tools > CodeBuild > Build projects > Swiggy_Clone > Swiggy_Clone:1f693ea9-afda-4207-ab5e-7c268c923755

Build status

Status	Initiator	Build ARN	Resolved source version
Succeeded	codepipeline/swiggy_pipeline	arn:aws:codebuild:ap-south-1:590184124026:build/Swiggy_Clone:1f693ea9-afda-4207-ab5e-7c268c923755	a1

Start time: Apr 9, 2024 7:52 PM (UTC+5:30) End time: Apr 9, 2024 7:58 PM (UTC+5:30) Build number: 9

Build logs Phase details Reports Environment variables Build details Resource utilization

Name	Status	Context	Duration	Start time	End time
SUBMITTED	Succeeded	-	<1 sec	Apr 9, 2024 7:52 PM (UTC+5:30)	Apr 9, 2024 7:52 PM (UTC+5:30)
QUEUED	Succeeded	-	<1 sec	Apr 9, 2024 7:52 PM (UTC+5:30)	Apr 9, 2024 7:52 PM (UTC+5:30)
PROVISIONING	Succeeded	-	3 secs	Apr 9, 2024 7:52 PM (UTC+5:30)	Apr 9, 2024 7:52 PM (UTC+5:30)
DOWNLOAD_SOURCE	Succeeded	-	<1 sec	Apr 9, 2024 7:52 PM (UTC+5:30)	Apr 9, 2024 7:52 PM (UTC+5:30)
INSTALL	Succeeded	-	<1 sec	Apr 9, 2024 7:52 PM (UTC+5:30)	Apr 9, 2024 7:52 PM (UTC+5:30)

SonarQube Analysis:

sonarcube Projects Issues Rules Quality Profiles Quality Gates Administration

My Favorites All

Search by project name or key

1 project(s)

swiggy Passed

Last analysis: 2 hours ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
2 C	0 A	0.0% E	1 A	-	0.0%	590 XS HTML, XML

1 of 1 shown

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.

SonarQube™ technology is powered by SonarSource SA
Community Edition - Version 9.9.4 (build 87374) - LGPL v3 - Community - Documentation - Plugins - Web API

Dependency-Check reports:

The screenshot shows the AWS CloudShell interface with the following details:

- S3 Bucket:** seedhar-code
- Objects (9) Info:** The table lists the following files:

Name	Type	Last modified	Size	Storage class
completion-for-dependency-check.sh	sh	April 9, 2024, 22:23:24 (UTC+05:30)	4.1 KB	Standard
dependency-check-junit.xml	xml	April 9, 2024, 22:23:19 (UTC+05:30)	110.0 B	Standard
dependency-check-report.csv	csv	April 9, 2024, 22:23:13 (UTC+05:30)	275.0 B	Standard
dependency-check-report.html	html	April 9, 2024, 22:23:10 (UTC+05:30)	126.6 KB	Standard
dependency-check-report.json	json	April 9, 2024, 22:23:20 (UTC+05:30)	826.0 B	Standard
dependency-check-report.yaml	yaml	April 9, 2024, 22:23:11 (UTC+05:30)	1.5 KB	Standard
dependency-check-report.xml	xml	April 9, 2024, 22:23:26 (UTC+05:30)	789.0 B	Standard
dependency-check.bat	bat	April 9, 2024, 22:23:28 (UTC+05:30)	2.9 KB	Standard
dependency-check.sh	sh	April 9, 2024, 22:23:26 (UTC+05:30)	3.7 KB	Standard
- Buckets:** A sidebar on the left lists buckets: Access Grants, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, Block Public Access settings for this account, Storage Lens (Dashboards, Storage Lens groups, AWS Organizations settings), Feature spotlight, and AWS Marketplace for S3.
- CloudShell:** The bottom left shows CloudShell status (CloudShell) and Feedback link.
- Footer:** The footer includes links for Privacy, Terms, and Cookie preferences, along with system status (30°C Haze), a search bar, and system icons.

Trivy File Scan:

Untitled • Splunk is a powerful platform used • Kubebeam trivyfilescan x +

File Edit View

```
[2024-04-09T16:48:46.887Z] [3d4INFO][0m Need to update DB
2024-04-09T16:48:46.887Z] [3d4INFO][0m Downloading DB...
2024-04-09T16:48:48.357Z] [3d4INFO][0m Number of language-specific files: 1
2024-04-09T16:48:48.357Z] [3d4INFO][0m Detecting gobinary vulnerabilities...

trivy (gobinary)
=====
Total: 31 (UNKNOWN: 13, LOW: 1, MEDIUM: 5, HIGH: 11, CRITICAL: 1)

+-----+-----+-----+-----+-----+
| TITLE | LIBRARY | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION |
+-----+-----+-----+-----+-----+
| github.com/aws/aws-sdk-go-padded | CVE-2020-8911 | MEDIUM | v1.37.0 | aws/aws-sdk-go: CBC |
| golang... | | | | oracle issue in AWS S3 |
| avd.aquasec.com/nvd/cve-2020-8911 | | | | crypto SDK for |
| | | | | --> |
| key | CVE-2020-8912 | LOW | | aws-sdk-go: In-band |
| | | | | negotiation issue in |
| AWS | | | | S3 Crypto SDK for |
| golang... | | | | --> |
| avd.aquasec.com/nvd/cve-2020-8912 | | | | --> |
| | GHSA-7f33-f4f5-xwpx | UNKNOWN | | |
| github.com/advisories/GHSA-7f33-f4f5-xwpx | | | | --> |
| | GHSA-f5pg-7wfw-84q9 | | | |
+-----+-----+-----+-----+-----+
```

Ln 1, Col 1 23,960 characters. 100% Unix (LF) UTF-8

30°C Haze Search

ENG IN 22:26 09-04-2024

Trivy Image Scan:

```

[024-04-09T16:50:16.434Z] [34mINFO[0m] Detected OS: debian
2024-04-09T16:50:16.434Z] [34mINFO[0m] Detecting Debian vulnerabilities...
2024-04-09T16:50:16.650Z] [34mINFO[0m] Number of language-specific files: 67
2024-04-09T16:50:16.650Z] [34mINFO[0m] Detecting jar vulnerabilities...
2024-04-09T16:50:16.683Z] [34mINFO[0m] Detecting gobinary vulnerabilities...

sreedhar8897/swiggy:latest (debian 10.13)
Total: 1926 (UNKNOWN: 60, LOW: 363, MEDIUM: 921, HIGH: 544, CRITICAL: 38)

+-----+-----+-----+-----+-----+
| LIBRARY | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION | TITLE
+-----+-----+-----+-----+-----+
| apt     | CVE-2011-3374   | LOW     | 1.8.2.3          | +             | It was found that apt-key in apt,
|          |                   |          |                  |               | all versions, do not correctly...
|          |                   |          |               +-->avd.aquasec.com/nvd/cve-2011-3374
| bash   | CVE-2019-18276  | HIGH    | 5.0-4            | +             | bash: when effective UID is not
|          |                   |          |                  |               equal to its real UID the...
|          |                   |          |               +-->avd.aquasec.com/nvd/cve-2019-18276
|          | CVE-2022-3715   |          | +             | +             | bash: a heap-buffer-overflow
|          |                   |          |               +-->avd.aquasec.com/nvd/cve-2022-3715
|          |                   |          |               +-->avd.aquasec.com/nvd/cve-2022-3715
| binutils | CVE-2018-12699  | CRITICAL| 2.31.1-16       | +             | binutils: heap-based buffer
|          |                   |          |                  | overflow in finish_stab in stabs.c
|          |                   |          |               +-->avd.aquasec.com/nvd/cve-2018-12699
|          | CVE-2018-1000876| HIGH    | +             | +             | binutils: integer overflow leads to
|          |                   |          |                  | heap-based buffer overflow in objdump
|          |                   |          |               +-->avd.aquasec.com/nvd/cve-2018-1000876
|          | CVE-2018-12697  |          | +             | +             | binutils: NULL pointer dereference
|          |                   |          |               +-->avd.aquasec.com/nvd/cve-2018-12697
|          |                   |          |               +-->avd.aquasec.com/nvd/cve-2018-12697
|          | CVE-2018-12698  |          | +             | +             | binutils: excessive
|          |                   |          |                  | memory consumption in
|          |                   |          |               +-->avd.aquasec.com/nvd/cve-2018-12698
+-----+-----+-----+-----+-----+
Ln 1. Col 1 15.63.616 characters

```

Step:4A :- ECS Cluster Creation

1. Navigate to ECS and click on “Create cluster”.

The screenshot shows the AWS ECS console with the following details:

- Clusters:** 1 Info
- Cluster:** Swiggy
- Services:** 1
- Container instances:** 2 EC2
- CloudWatch monitoring:** Enabled
- Capacity provider strategy:** ASG

A prominent orange "Create cluster" button is visible at the top right of the cluster list.

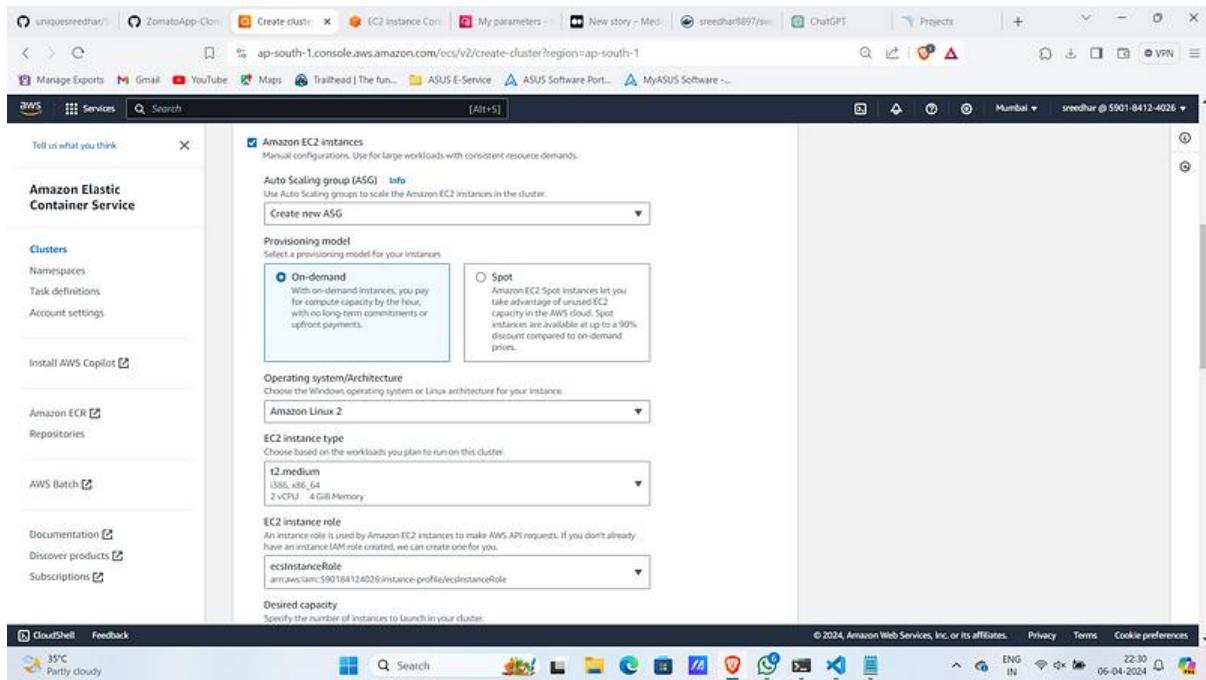
2. Provide a name for it.

The screenshot shows the AWS CloudShell interface with the 'Create cluster' wizard open. The 'Cluster configuration' step is active. In the 'Cluster name' field, 'swiggy_cluster' is entered. Under 'Default namespace - optional', 'swiggy_cluster' is also listed. The 'Infrastructure' section is expanded, showing the 'Serverless' tab selected. Under 'AWS Fargate (serverless)', the checkbox is checked. Under 'Amazon EC2 instances', the checkbox is unchecked. The status bar at the bottom right shows the date as 06-04-2024.

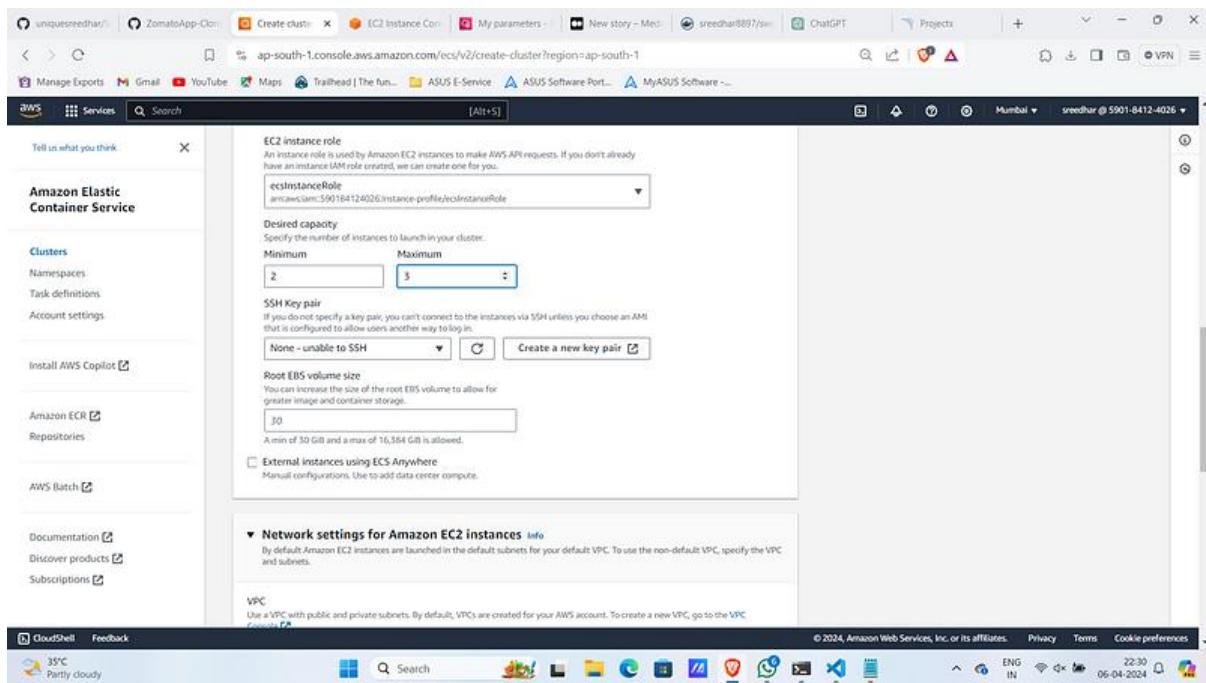
2. Under infrastructure select “Amazon EC2 instances”

The screenshot shows the continuation of the 'Create cluster' wizard. The 'Infrastructure' section is expanded, showing the 'Amazon EC2 instances' option selected. It details the use of Auto Scaling groups (ASG) for scaling EC2 instances. It also shows sections for 'Provisioning model' (On-demand or Spot selected), 'Operating system/Architecture' (Choose the operating system or Linux architecture), 'EC2 instance type' (Choose based on the workloads you plan to run on this cluster), and 'EC2 instance role' (Choose additional instance type). The status bar at the bottom right shows the date as 06-04-2024.

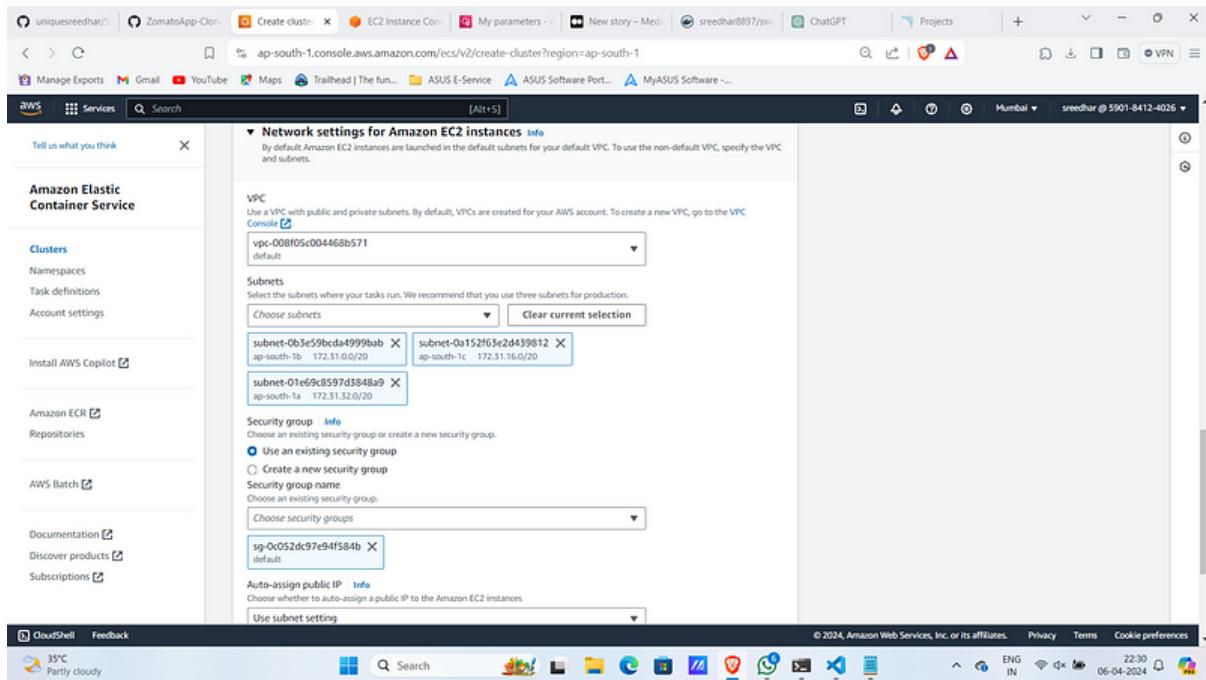
3. Give OS as Amazon Linux 2 and instance type as “t2.medium”.



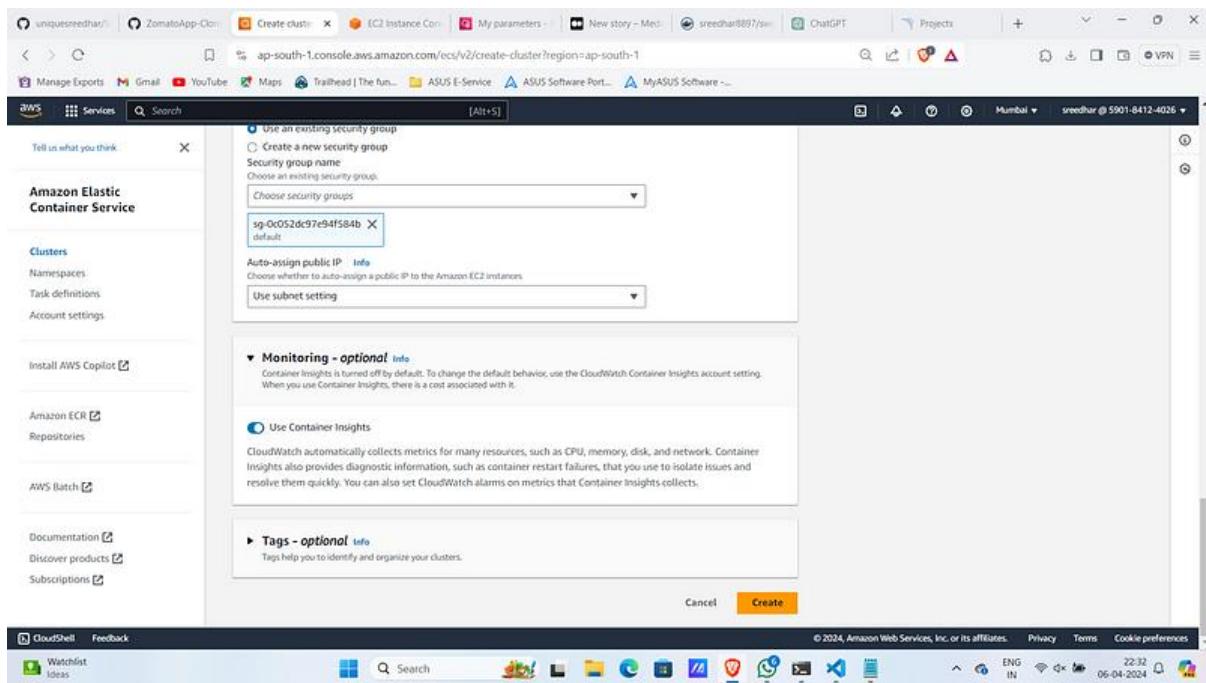
4. Give Desired Capacity min as 2 and max as 3.



5. Under Network settings Select the VPC and the subnets on which instances to be launched.



6. Enable the container insights under monitoring and click “create” .



Step:4B :- ECS Task Definition Creation

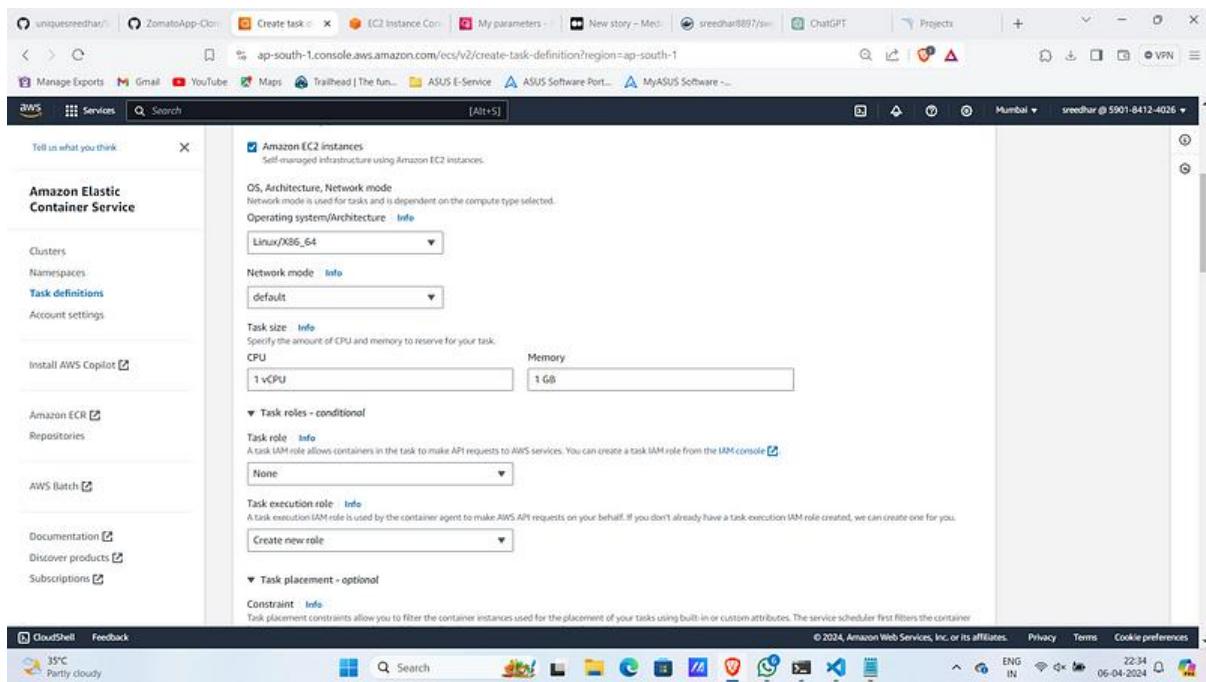
1. In the same ECS console click on “Task Definition” and then “create new task definition”

The screenshot shows the AWS CloudShell interface. On the left, there is a sidebar for the Amazon Elastic Container Service (ECS) with options like Clusters, Namespaces, Task definitions, and Account settings. The main area displays the 'Task definitions (1)' page under 'Amazon Elastic Container Service > Task definitions'. It shows a table with one row for 'amazon', which is marked as 'ACTIVE'. There are buttons for 'Deploy', 'Create new revision', and 'Create new task definition'.

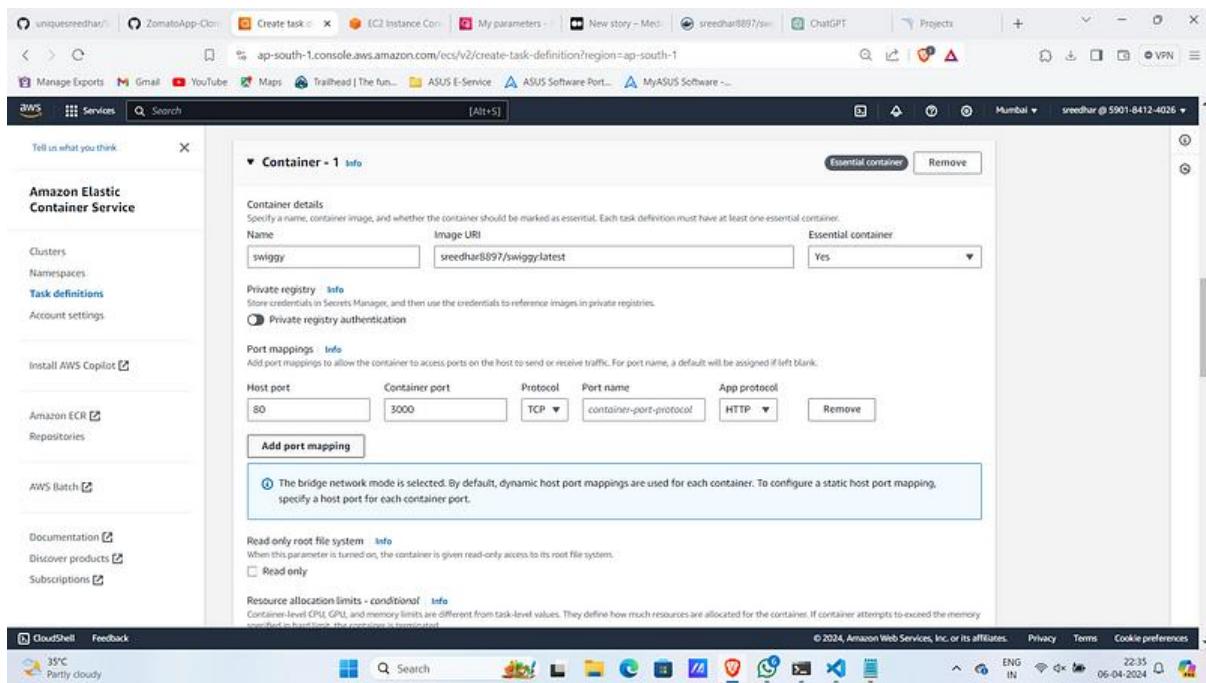
2. Give a name for it and under “infra requirements” select “Amazon EC2 instances”.

The screenshot shows the 'Create new task definition' page under 'Amazon Elastic Container Service > Create new task definition'. In the 'Task definition configuration' section, the 'Task definition family' is set to 'Swiggy'. Under the 'Infrastructure requirements' section, the 'Launch type' is selected as 'Amazon EC2 instances'. Other options like 'AWS Fargate' and 'OS, Architecture, Network mode' are also visible.

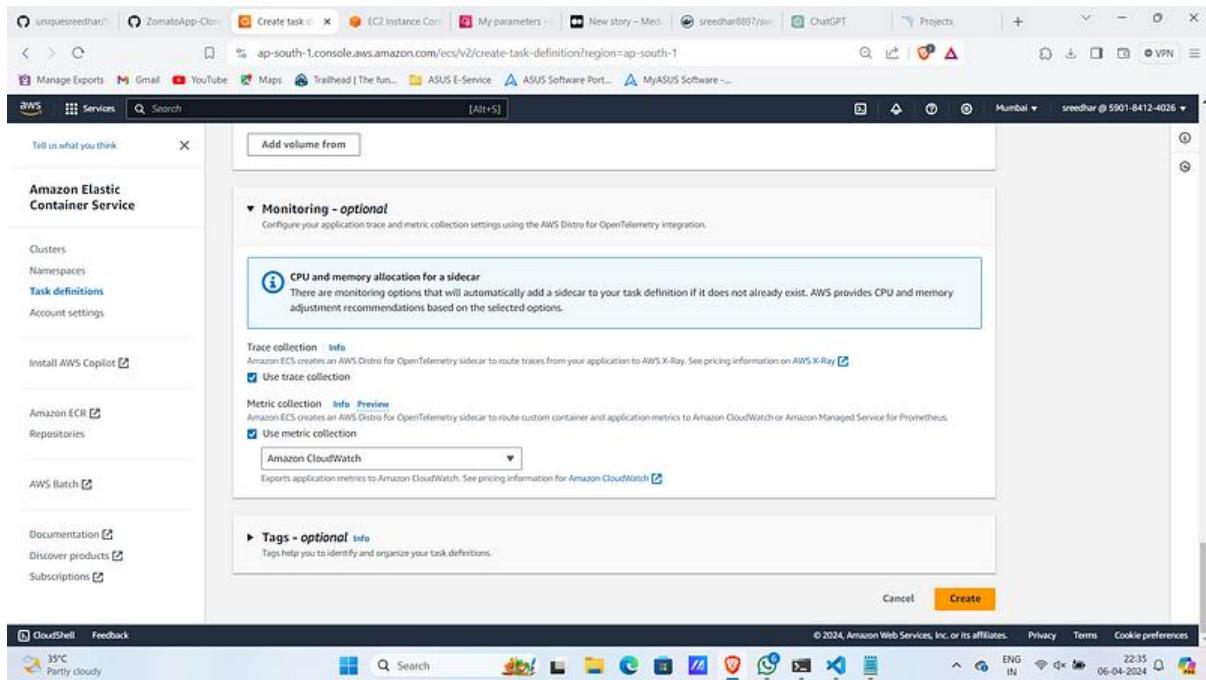
3. Give the configuration as below.



4. Under container provide Name, Image and container port .

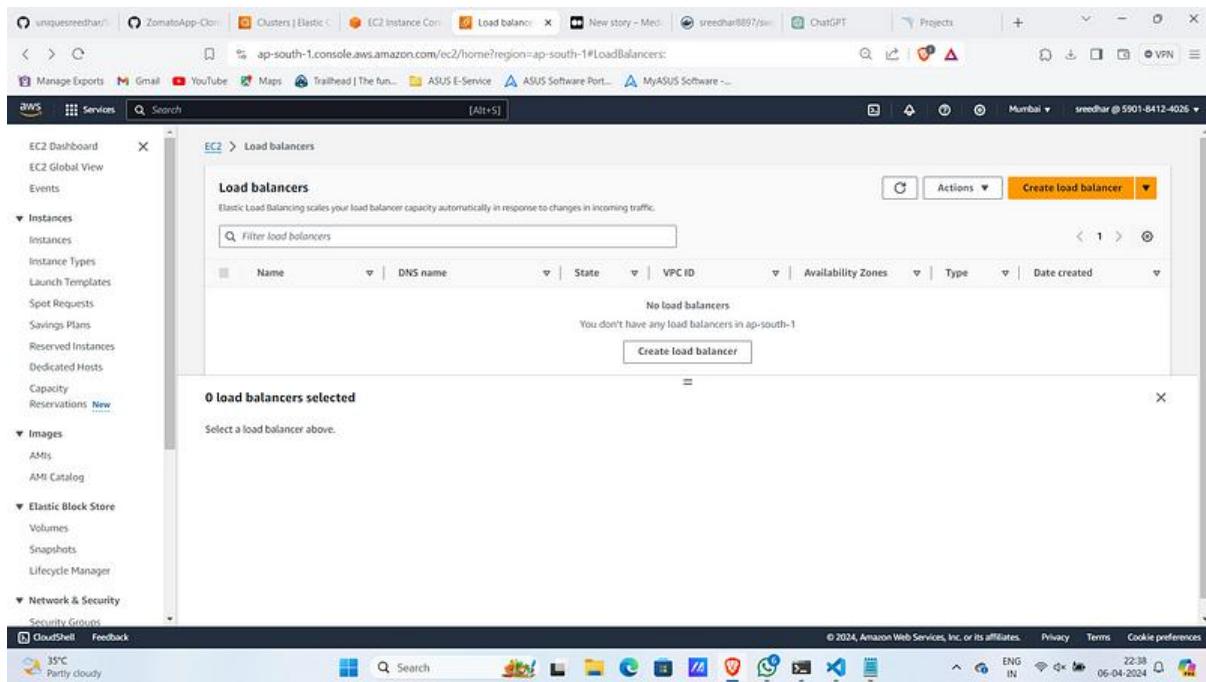


5. Under monitoring select as below and click “create”.

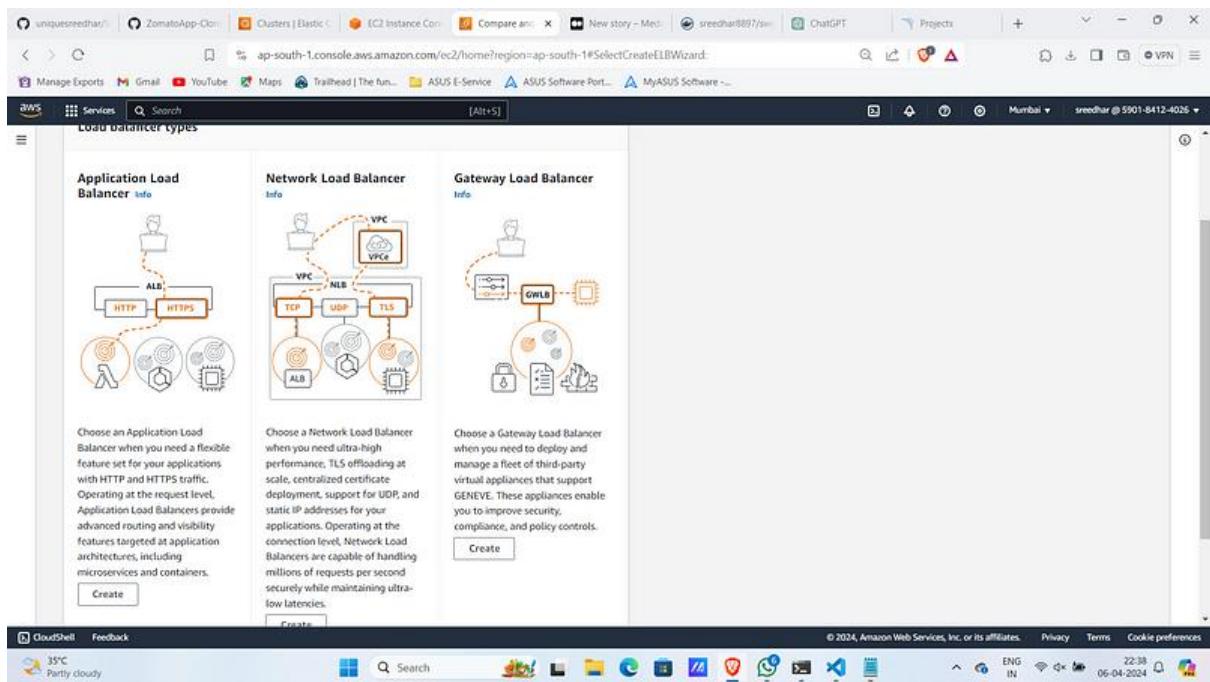


Step:4C :- Load Balancer Creation

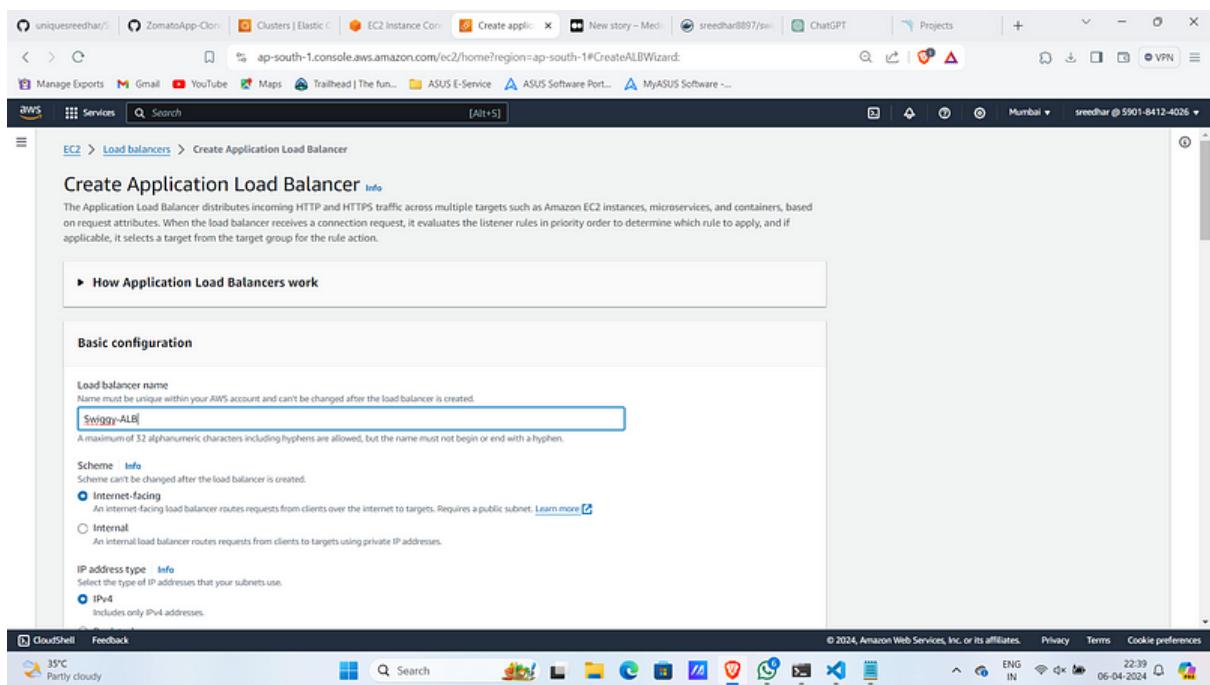
1. Navigate to EC2 and under Load Balancer click on “Create load balancer”.



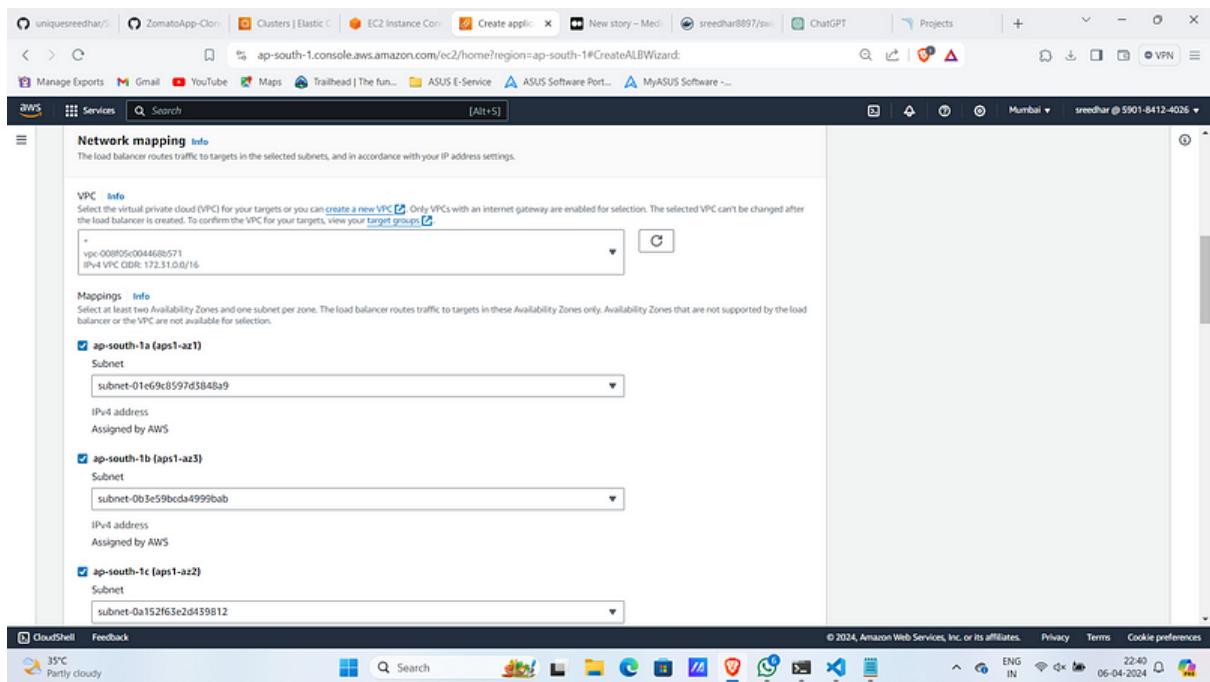
2. Select “Application Load Balancer”.



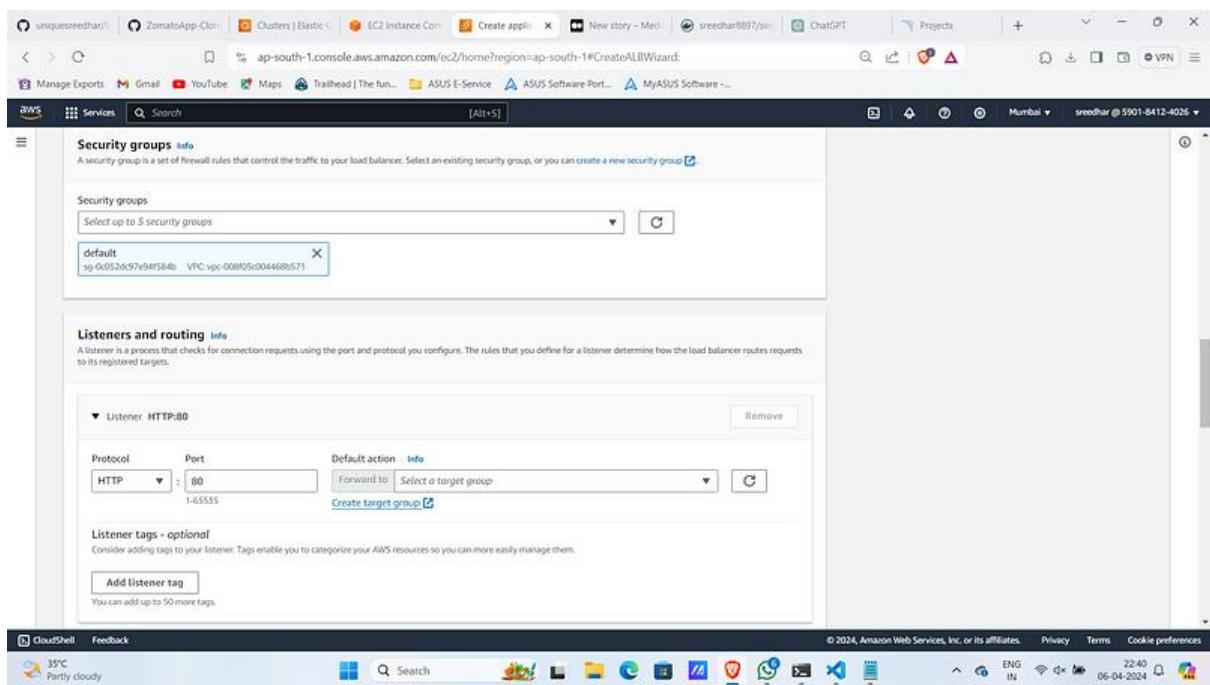
2. Provide a name for it.



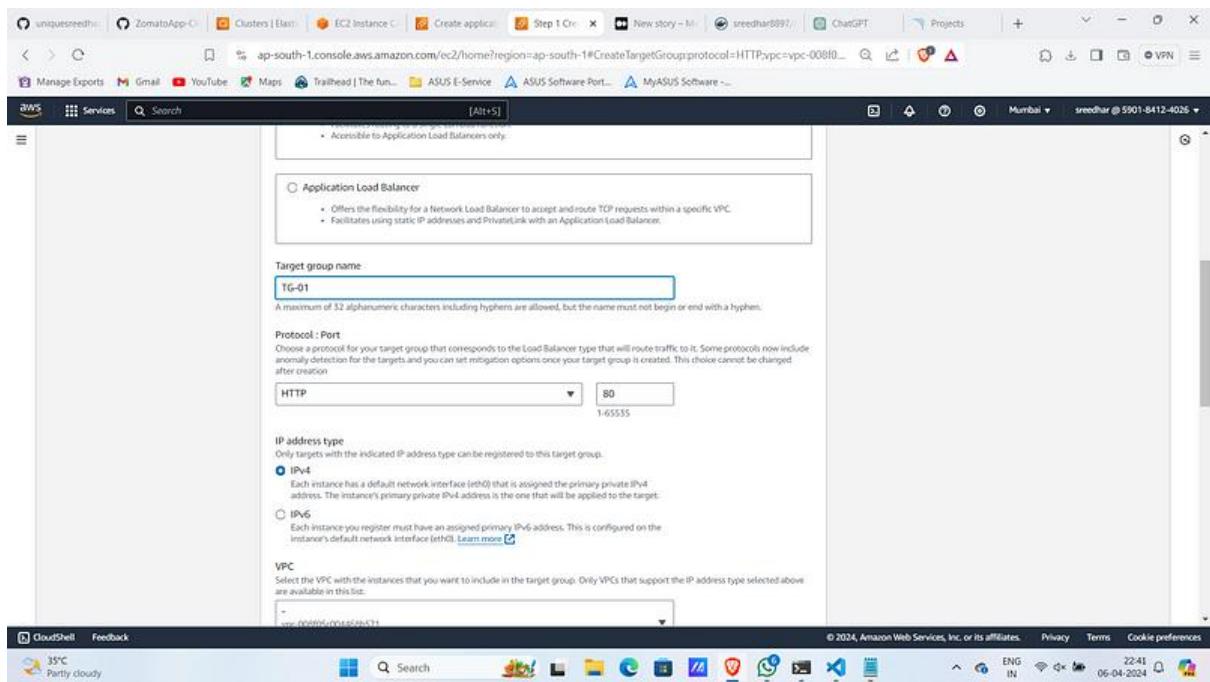
3. Under Network mapping select the vpc sand subnets.



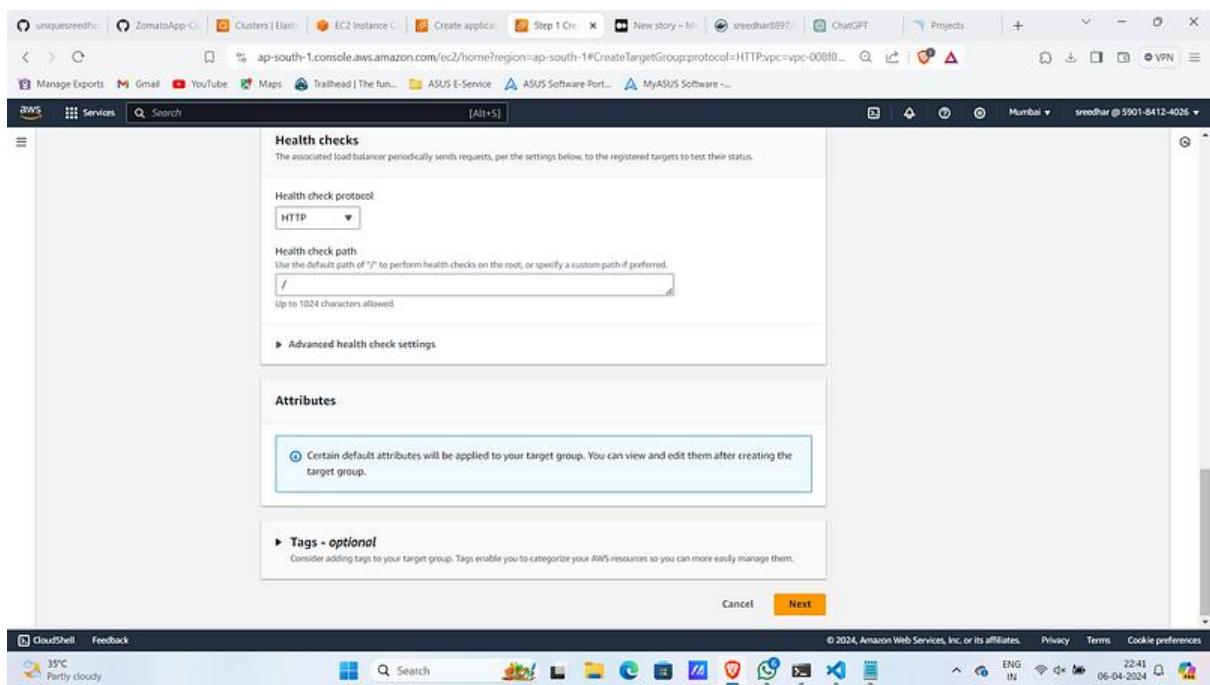
4. Under Listeners and routing click on “Create target group”.



5. Give a target group name.



6. Click on “Next”.



7. Select the ECS registered instances and give port for them.

The screenshot shows the 'Register targets' step of the 'Create target group' wizard. It displays a list of available instances from the previous step. Two instances are selected: 'i-0399e513719bdd6f5' and 'i-0d3eb61551e4c0ea7', both of which are running in the 'ECS Instance - swiggy_cluster'. The 'Ports for the selected instances' field contains '3000'. A button 'Include as pending below' is visible.

8. Click on “Create target group”.

The screenshot shows the 'Review targets' step of the 'Create target group' wizard. It displays the selected target 'i-072bcdccac0ce0ff2' with its details: Name 'sonar-server', State 'Running', Security groups 'launch-wizard-5', and Zone 'ap-south-1a'. Below this, the 'Targets (0)' section shows a note 'No instances added yet' and a message 'Specify instances above, or leave the group empty if you prefer to add targets later.' At the bottom right, there is a 'Create target group' button.

9. In the Load balancer select the created one.

The screenshot shows the AWS Lambda console with the 'Create ALB Wizard' step 1: Set up security groups. In the 'Security groups' section, a dropdown menu is open, showing 'Select up to 5 security groups'. Under 'default', it lists 'sg-0052dc97e94f584b' and 'VPC: vpc-008f05e004468b571'. In the 'Listeners and routing' section, there is a table for 'Listener HTTP:80' with three target groups: TG-01, TG-1, and TG-3, all set to forward to 'HTTP' on port 80. Below this, there is a section for 'Listener tags - optional'.

10. Click on “Create load balancer”.

The screenshot shows the 'Create ALB Wizard' step 2: Set up basic configuration. It displays the 'Basic configuration' section with 'Swiggy-ALB' (Internet-facing, IPv4), 'Security groups' (VPC: vpc-008f05e004468b571, target groups: ap-south-1a, ap-south-1b, ap-south-1c), 'Network mapping' (HTTP:80 to TG-01), 'Listeners and routing' (HTTP:80 to TG-01), 'Service integrations' (AWS WAF: None, AWS Global Accelerator: None), 'Tags' (None), and 'Attributes' (Note: Certain default attributes will be applied). The 'Creation workflow and status' section shows 'Server-side tasks and status' (After completing and submitting the above steps, all server-side tasks and their statuses become available for monitoring). At the bottom, there is a 'Create load balancer' button.

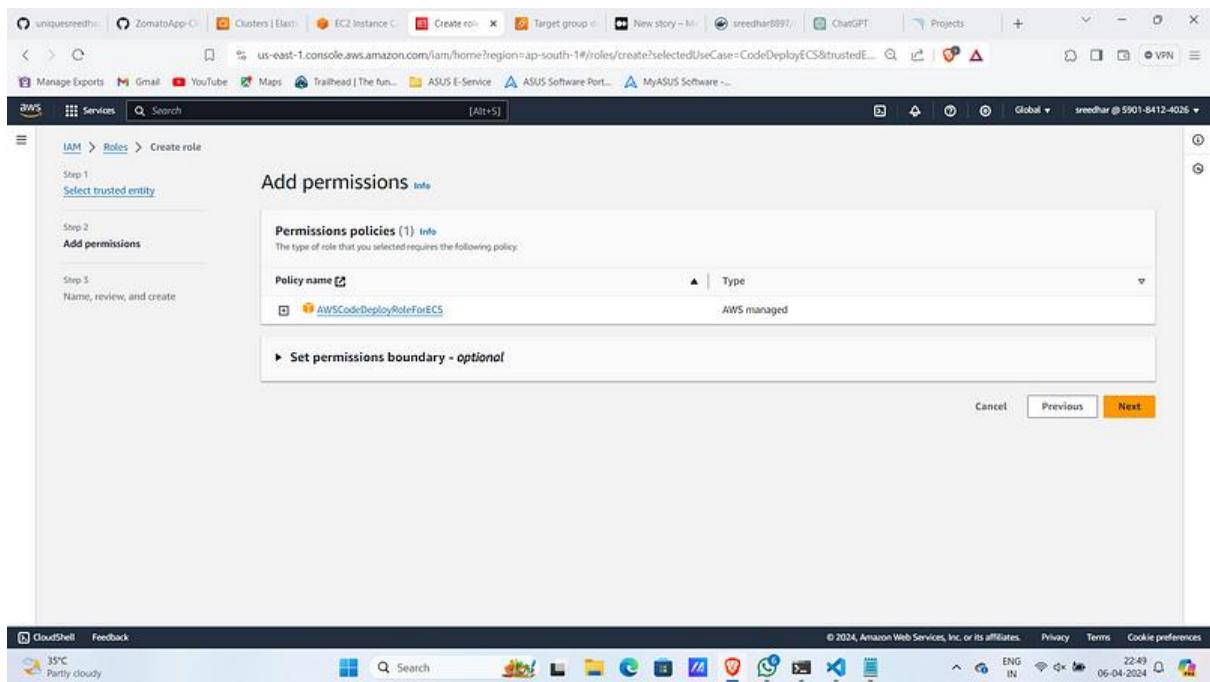
11. To access ECS Code Deploy needs a role so let's create a one. Navigate to roles in IAM and click on “Create role”.

The screenshot shows the AWS IAM Roles page. On the left, there's a sidebar with navigation links like Dashboard, Access management, and Access reports. The main area displays a table of roles, each with a name, trusted entities, and last activity. Some roles listed include `Amazon_EventBridge_Scheduler_LAMBDA_4876d4b7f4`, `api-lambda-function-role-av7d7kdb`, and `AWSCodePipelineServiceRole-ap-south-1-amazon_cld`.

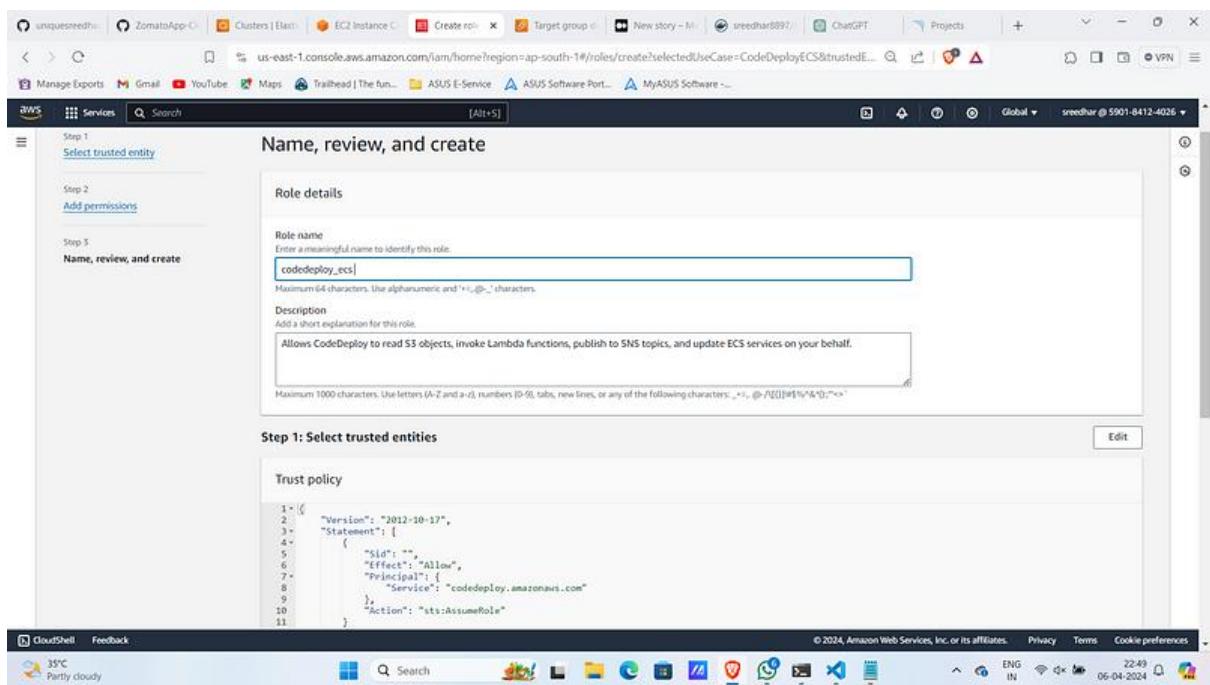
12. Select Use case as “CodeDeploy-ECS”

This screenshot shows the 'Create role' wizard at Step 3. It asks for a name, reviews the details, and creates the role. The 'Use case' section is expanded, showing five options: AWS service (selected), AWS account, SAML 2.0 federation, and Custom trust policy. Under 'Service or use case', 'CodeDeploy' is chosen. In the 'Use case' dropdown, 'CodeDeploy - ECS' is selected. At the bottom right, there are 'Cancel' and 'Next' buttons.

13. Click on “Next”.



14. Give a name for it.



15. Click on Create.

Step 2: Add permissions

```

    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": "*",
          "Service": "codedeploy.amazonaws.com"
        },
        {
          "Action": "sts:AssumeRole"
        }
      ]
    }
  
```

Permissions policy summary

Policy name	Type	Attached as
AWSCodeDeployRoleForECS	AWS managed	Permissions policy

Step 3: Add tags

Add tags - optional Info
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add new tag
You can add up to 50 more tags.

Cancel Previous Create role

Step:4D :- ECS Service Creation

- Under Created ECS cluster and Service section click on “create”.

Cluster overview

ARN: arn:aws:ecs:ap-south-1:590184124026:cluster/swiggy_cluster Status: Active CloudWatch monitoring: Container Insights Registered container instances: 2

Services

Draining	Status	Tasks
*	Active	Pending

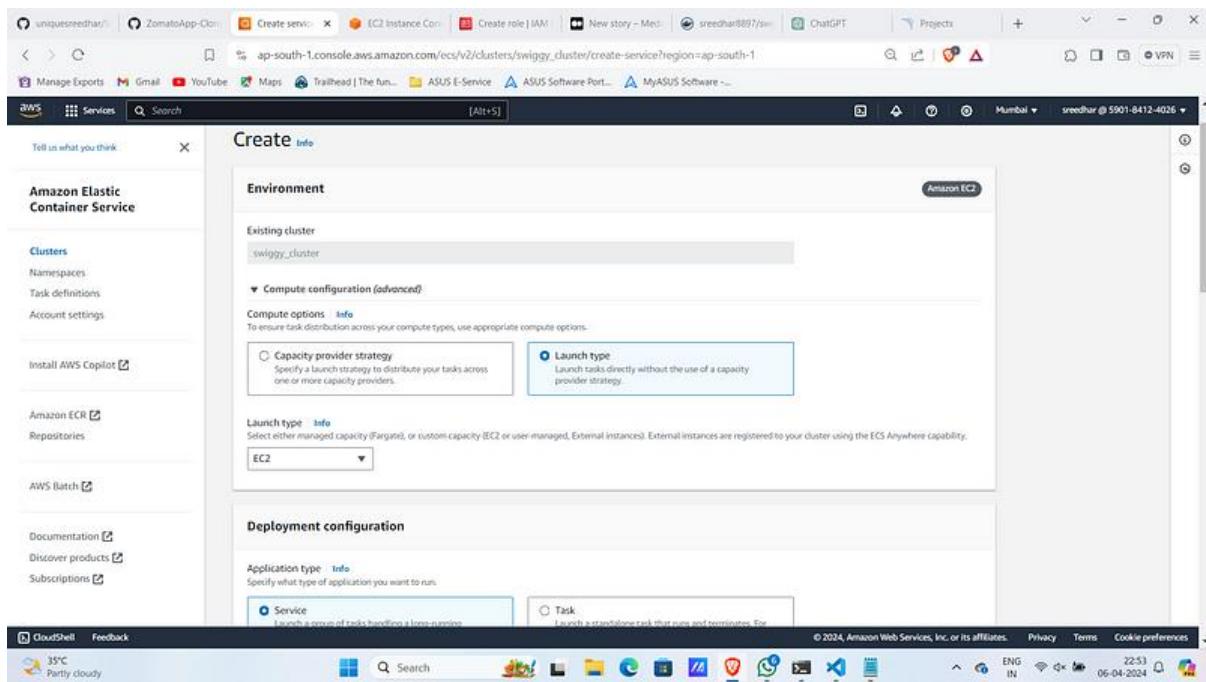
Services (0) Info

Filter launch type: Any launch type Filter service type: Any service type

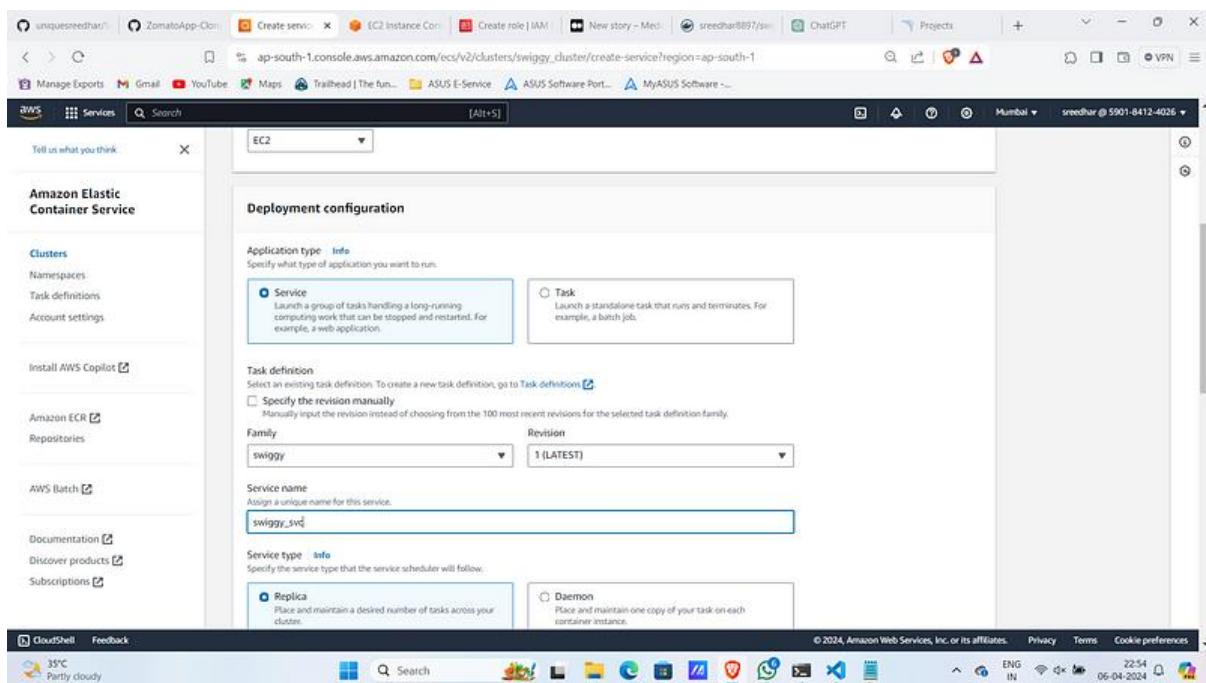
Service name	ARN	Status	Service type	Deployments and tasks	Last deploy...	Task def...
No services No services to display.						

Create

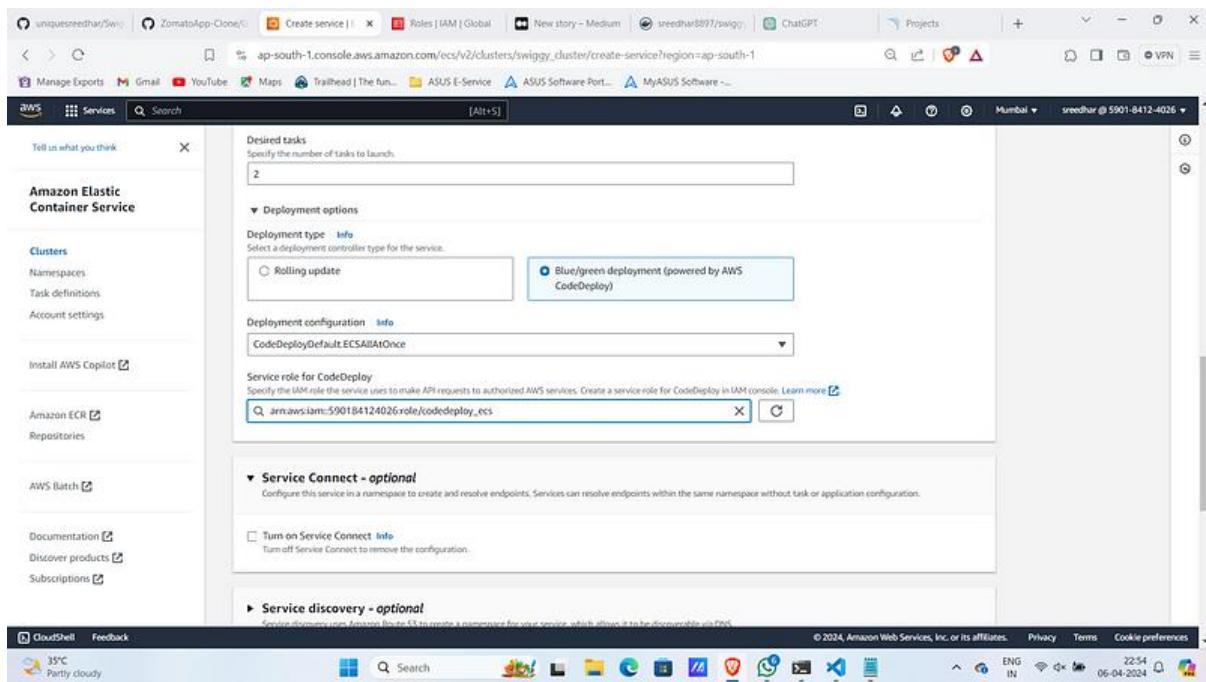
- Opt Compute as “Launch type”.



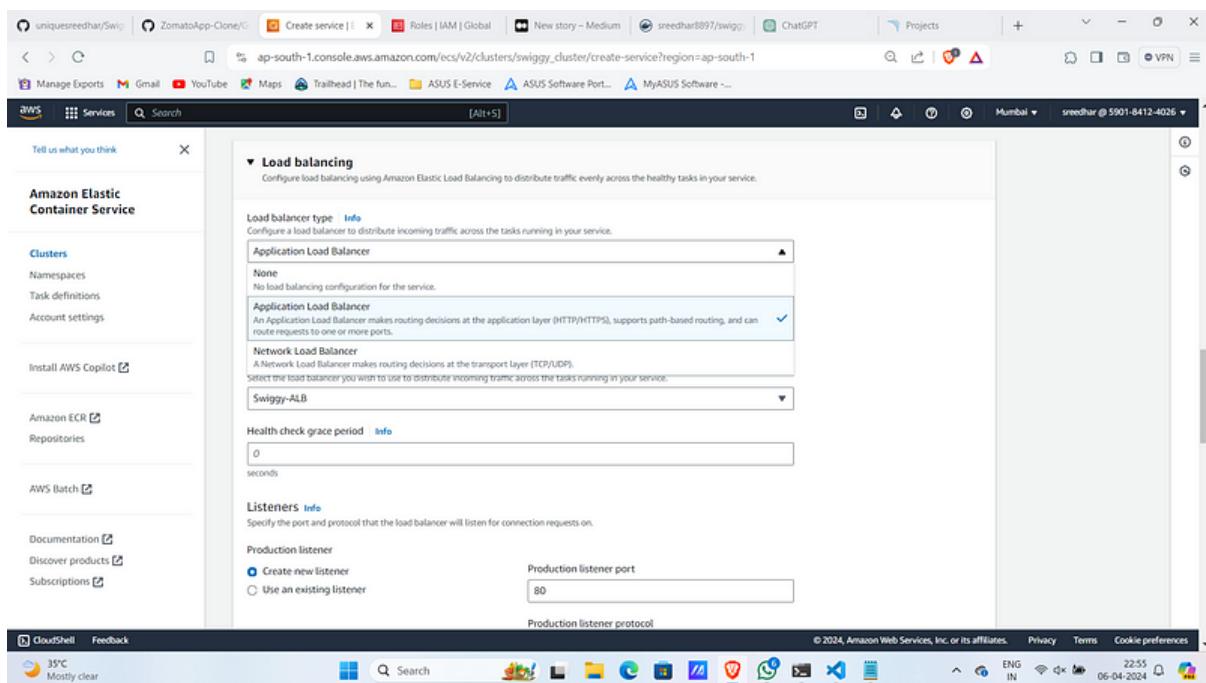
3. Under Deployment config select Family as the created task definition and give it a name.



4. Give desired tasks as 1 and Under Deployment options select Deployment type as “Blue/green deployment” then provide service role you created earlier.



5. In the Load balancing section select type as ALB and opt the one you have created.

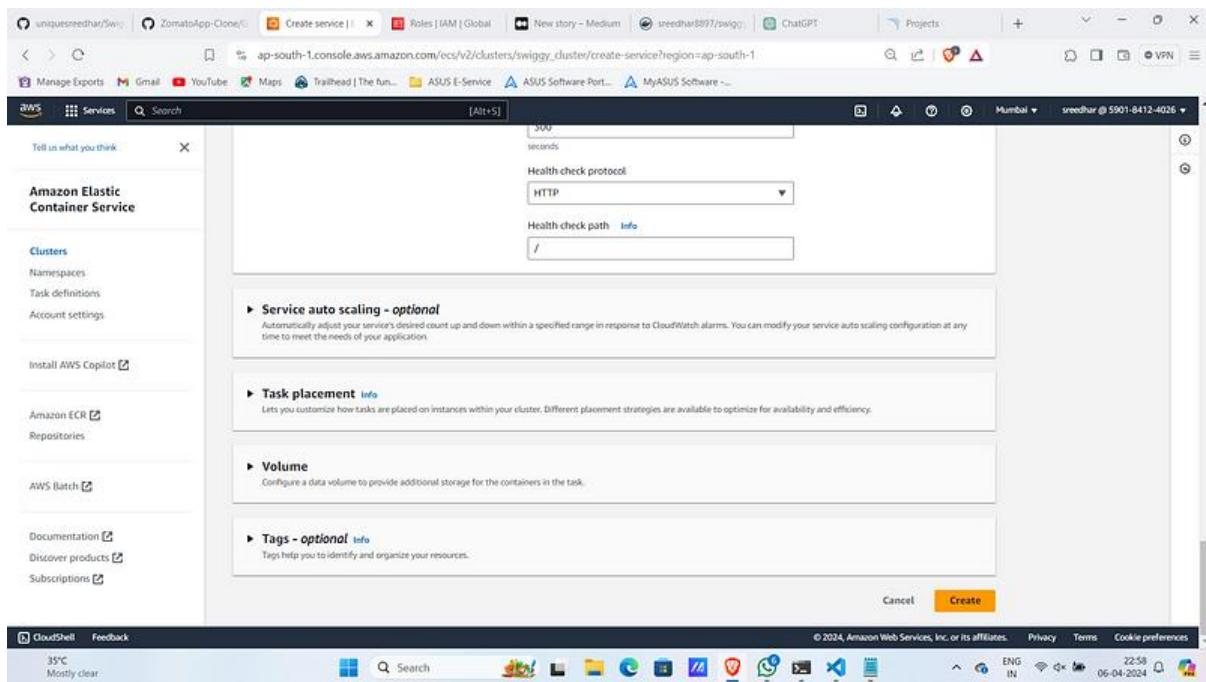


6. Use existing Listener and Target group and TG-2 create a one.

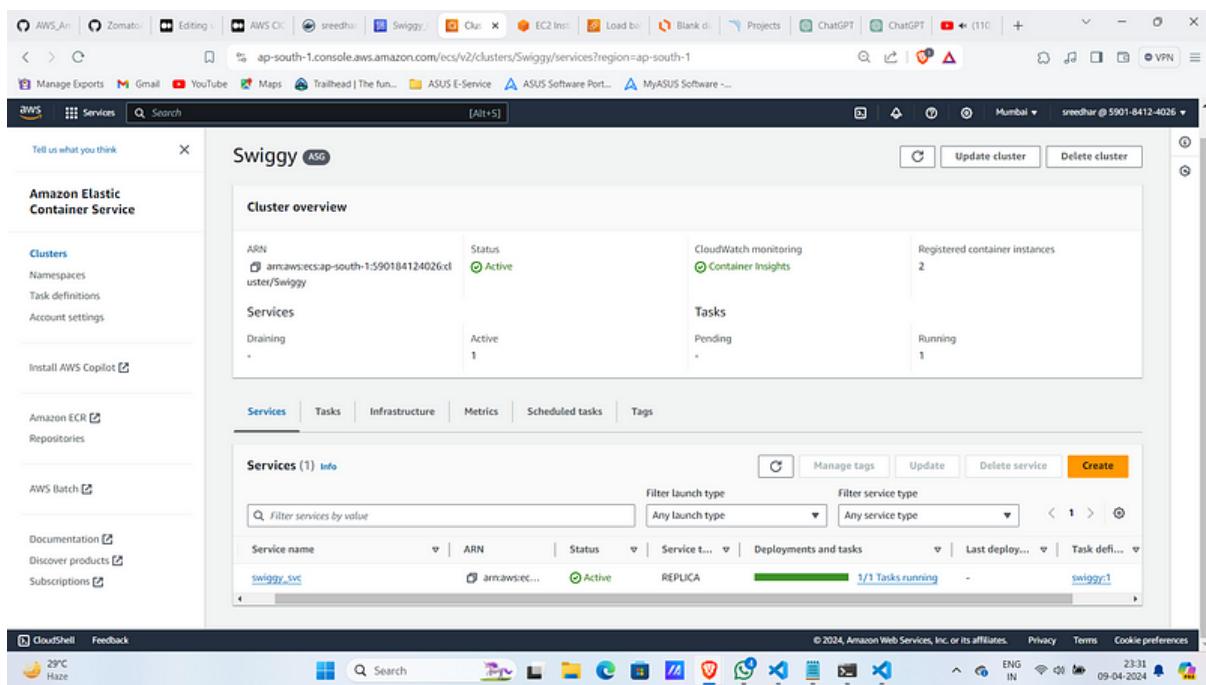
The screenshot shows the AWS Cloud Console interface for creating a new service in the Amazon Elastic Container Service (ECS). The 'Listeners' tab is active, displaying a configuration for a production listener on port 80 using the HTTP protocol. A single listener rule is defined, pointing to a target group named 'tg-swiggy-swiggy_svc'. The 'Target groups' section shows this target group has been created.

The screenshot continues the service creation process, switching to the 'Target groups' tab. It shows the configuration for two target groups: 'tg-swiggy-swiggy_svc' and 'tg-swiggy-swiggy_svc-2'. Both target groups are set to use the HTTP protocol. The 'tg-swiggy-swiggy_svc-2' target group has a 'Deregistration delay' of 300 seconds.

7. Click on “Create”



8. Upon Successful service creation it looks like as:



9. Navigate to load balancer and copy the DNS name. Observe that the traffic to routing to target group TG-1.

The screenshot shows the AWS CloudWatch Metrics interface. A log stream named 'Swiggy' is selected, displaying a single log entry:

```
2024-04-09T19:56:23.500Z INFO Deployment successful.
```

10. Paste it on your favorite browser.

Observe that the tab name is “Swiggy Application”.

The screenshot shows the Swiggy Application homepage. Key elements include:

- Best offers for you:**
 - It's Snack Time!**: Get 50% OFF & Free Delivery on your first order. ORDER NOW
 - Get 40% OFF**: On heavenly delights from P60 Pizza. ORDER NOW
 - Flat ₹125 OFF***: Celebrate International Coffee Day with Starbucks. ORDER NOW
- What's on your mind Today?**: A grid of food icons including Pothichoru, Biryani, Burger, Pizza, South Indian, Chinese, and North Indian.

11. This will also create an application and deployment group under “code deploy” section.

The screenshot shows the AWS CodeDeploy console with the URL ap-south-1.console.aws.amazon.com/codesuite/codedeploy/applications?region=ap-south-1. The left sidebar navigation includes 'Developer Tools', 'CodeDeploy', 'Source', 'Artifacts', 'Build', 'Deploy', 'Pipeline', and 'Settings'. Under 'Deploy', 'Applications' is selected. The main content area displays the 'Applications' section with a table:

Application name	Compute platform	Created
AppECS-swiggy_cluster-swiggy_svc	Amazon ECS	29 minutes ago

The screenshot shows the 'Application details' page for the 'AppECS-swiggy_cluster-swiggy_svc' application. The URL is ap-south-1.console.aws.amazon.com/codesuite/codedeploy/applications/AppECS-swiggy_cluster-swiggy_svc?region=ap-south-1. The left sidebar is identical to the previous screenshot. The main content area shows the 'Application details' section with the application name 'AppECS-swiggy_cluster-swiggy_svc' and compute platform 'Amazon ECS'. Below this, the 'Deployment groups' tab is selected, showing a table:

Name	Status	Last attempted deployment	Last successful deployment	Trigger count
DgpECS-swiggy_cluster-swiggy... (ellipsis)	-	-	-	0

12. Create a file named “appspec.yaml” and past the below snippet replacing the task definition arn with yours.

version: 0.0

Resources:

- TargetService:

Type: AWS::ECS::Service

Properties:

TaskDefinition: "arn:aws:ecs:ap-south-1:<account_id>:task-definition/swiggy:1"

LoadBalancerInfo:

ContainerName: "swiggy"

ContainerPort: 3000

The screenshot shows the AWS ECS Task Definitions page. On the left, there's a sidebar with navigation links like Clusters, Namespaces, Task definitions, and Account settings. The main content area is titled 'swiggy:1'. It has tabs for Overview, Containers, JSON, Task placement, Volumes (0), Requires attributes, and Tags. Under Overview, it shows details such as ARN (arn:aws:ecs:ap-south-1:1590184124026:task-definition/swiggy:1), Status (ACTIVE), Time created (2024-04-06T17:06:01.343Z), and App environment (EC2). Under Containers, it lists a single container with Task role (ecsTaskExecutionRole), Task execution role (ecsTaskExecutionRole), Operating system/Architecture (Linux/X86_64), and Network mode (default). Below this, there's a 'Task size' section with Task CPU (1024 units (1 vCPU)) and Task memory (1024 MiB (1 GiB)). There are also sliders for Task CPU maximum allocation for containers and Task memory maximum allocation for container memory reservation.

Step:5 :- AWS Code Pipeline Creation

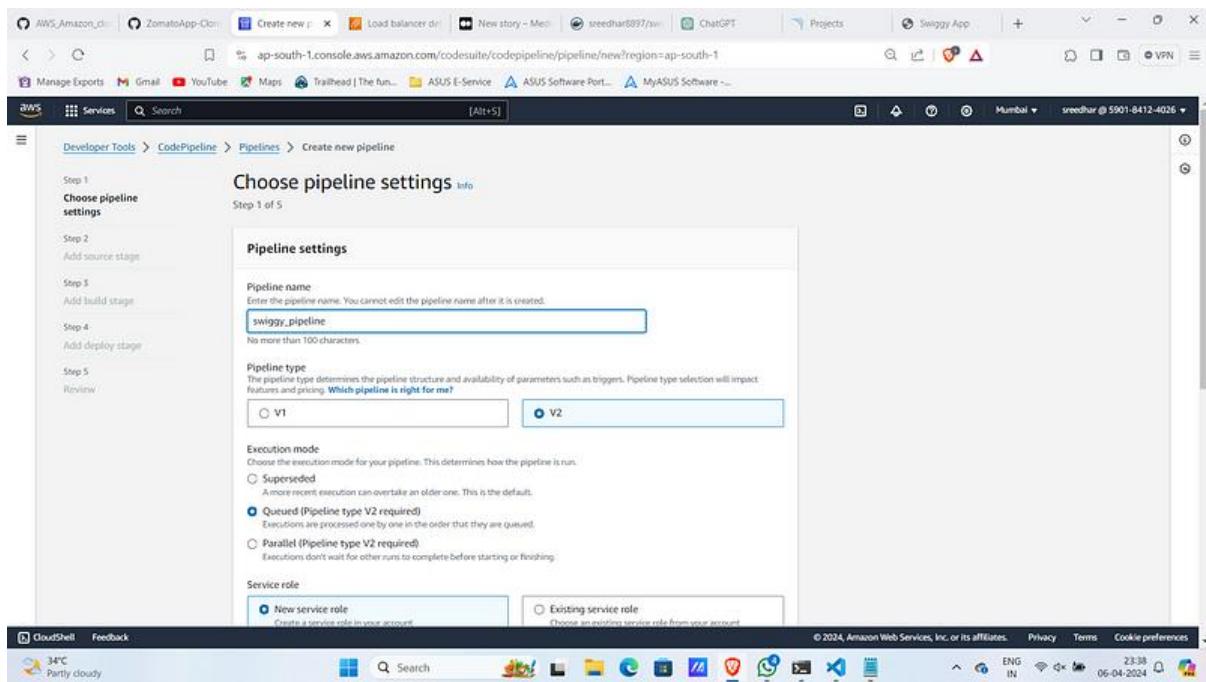
1. Navigate to Code Pipeline in AWS console and click on “create pipeline”.

The screenshot shows the AWS CodePipeline Pipelines page. On the left, there's a sidebar with sections like Source, Artifacts, Build, Deploy, Pipeline, and Settings. The main content area is titled 'Pipelines'. It shows a table with columns: Name, Latest execution status, Latest source revisions, Latest execution started, and Most recent executions. Three pipelines are listed:

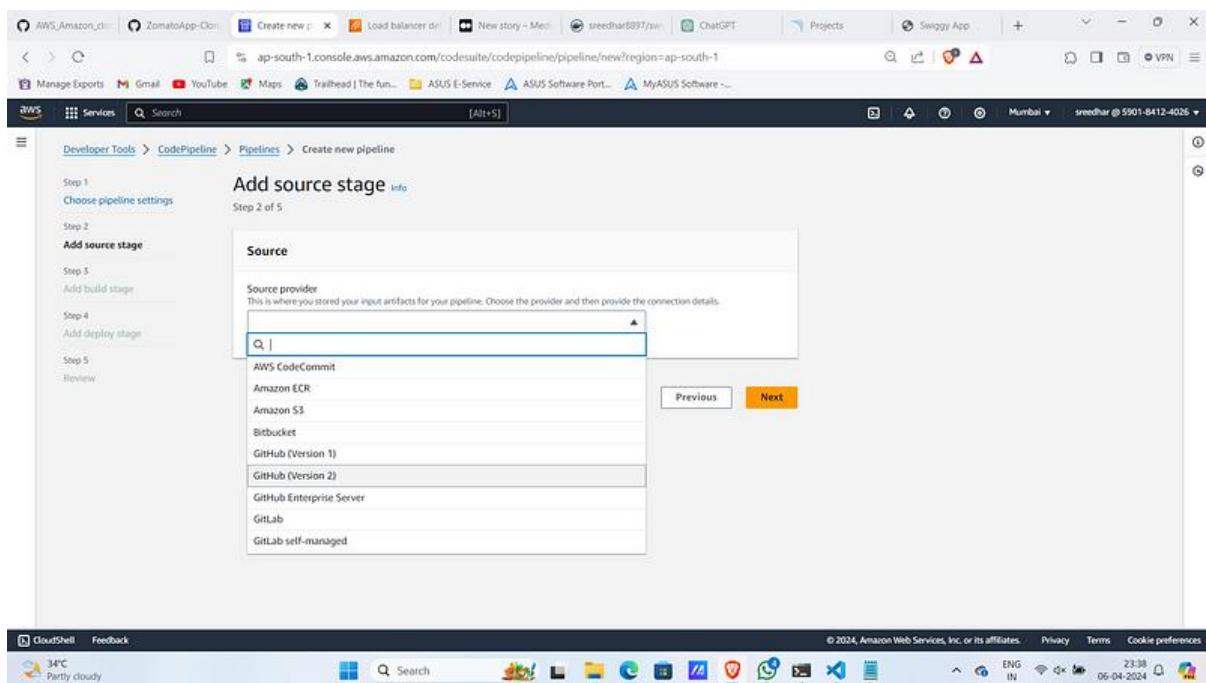
- swiggy_pipeline (Type: V2 | Execution mode: QUEUED) - In progress
- amazon_pipeline (Type: V2 | Execution mode: QUEUED) - Succeeded
- shekhar-reddy (Type: V2 | Execution mode: QUEUED) - Failed

Each row includes a 'View details' link.

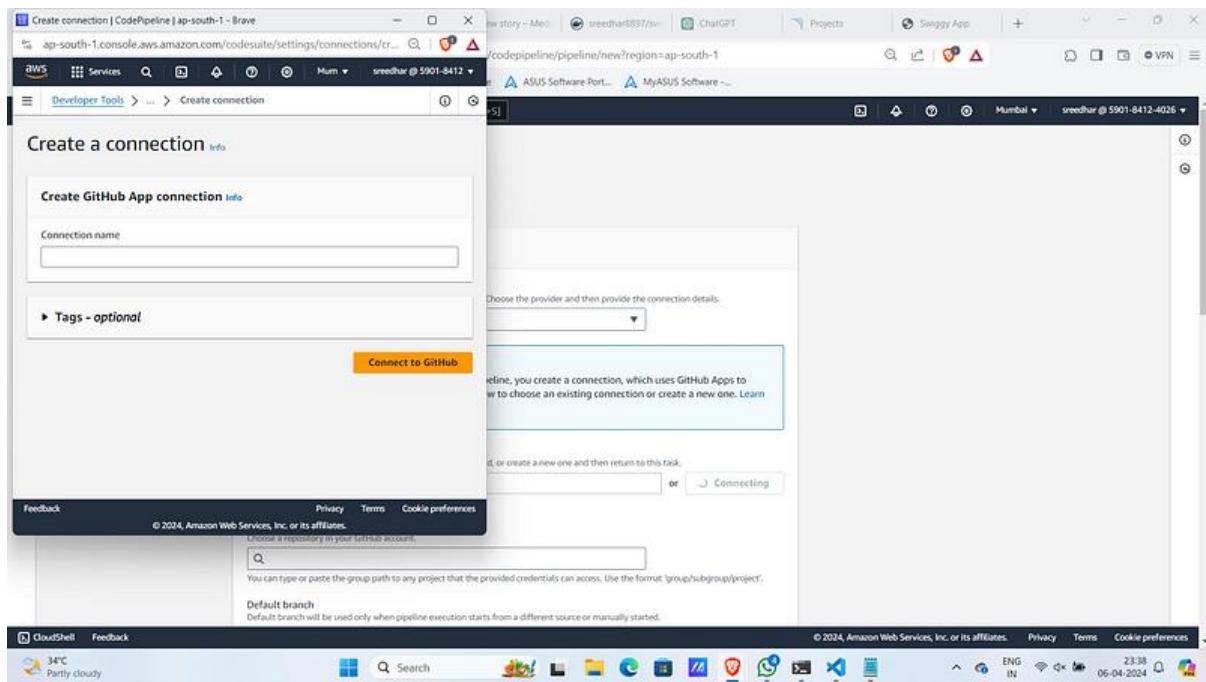
2. Provide a name for it and click on “next”.



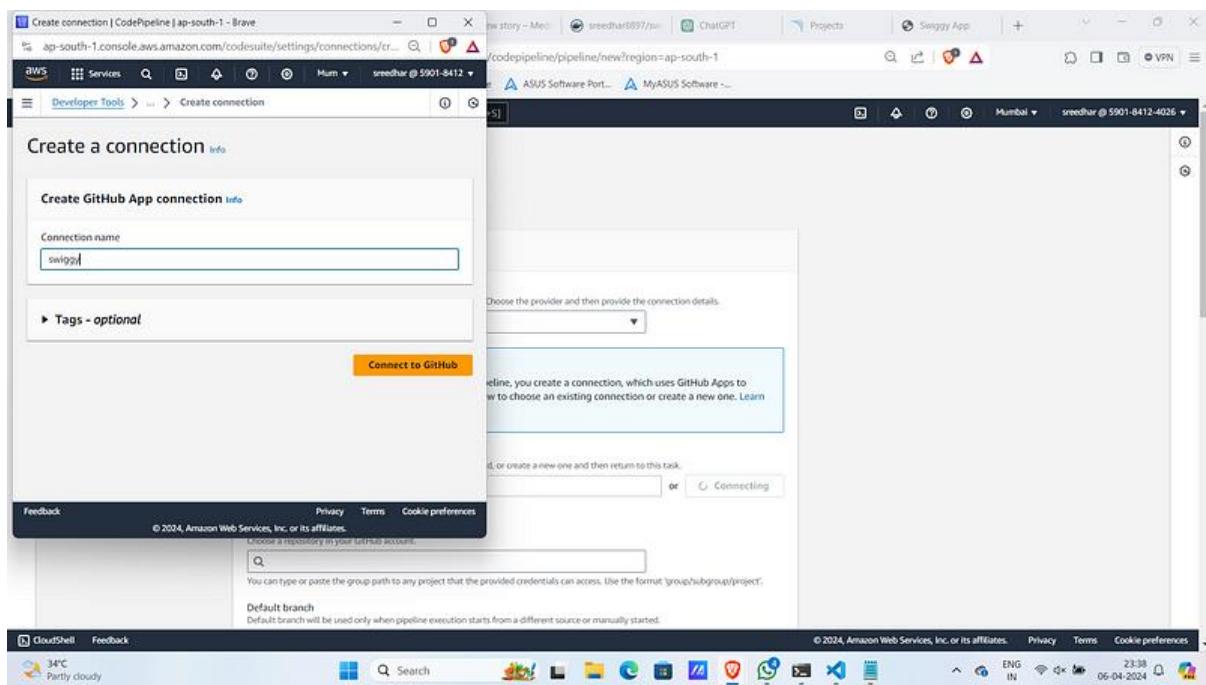
3. In source stage give GitHub(Version 2) as Source.



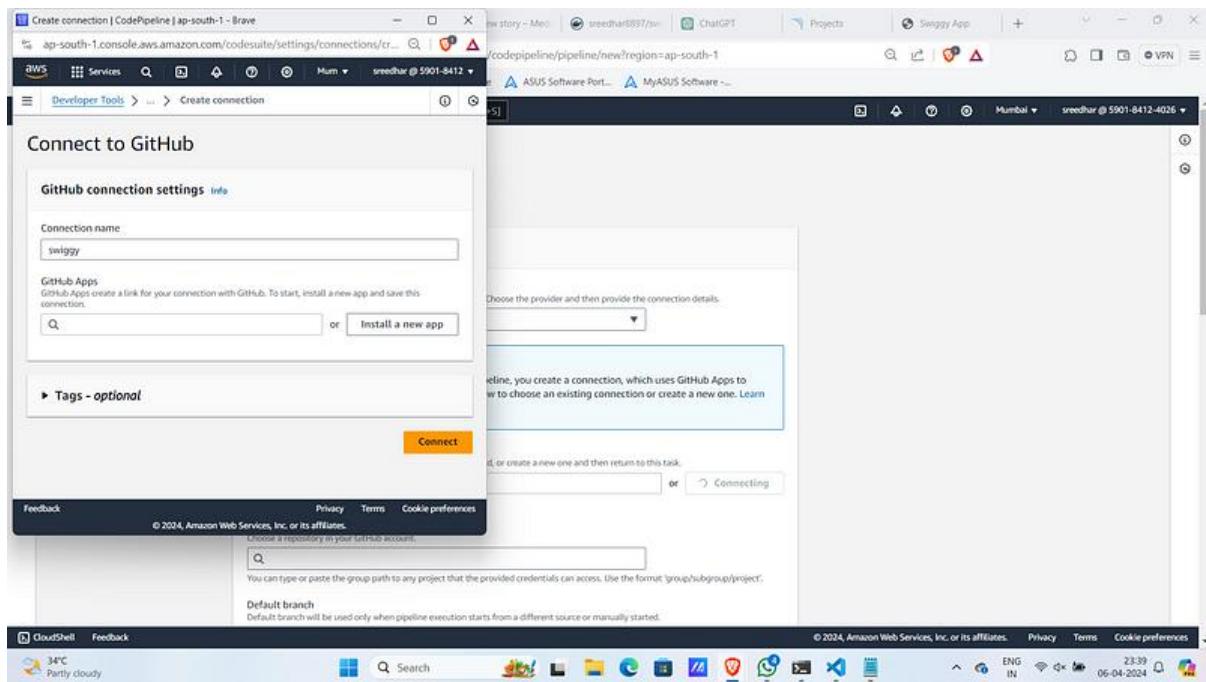
4. For source code you need to provide access for github so click on "Connect to GitHub".



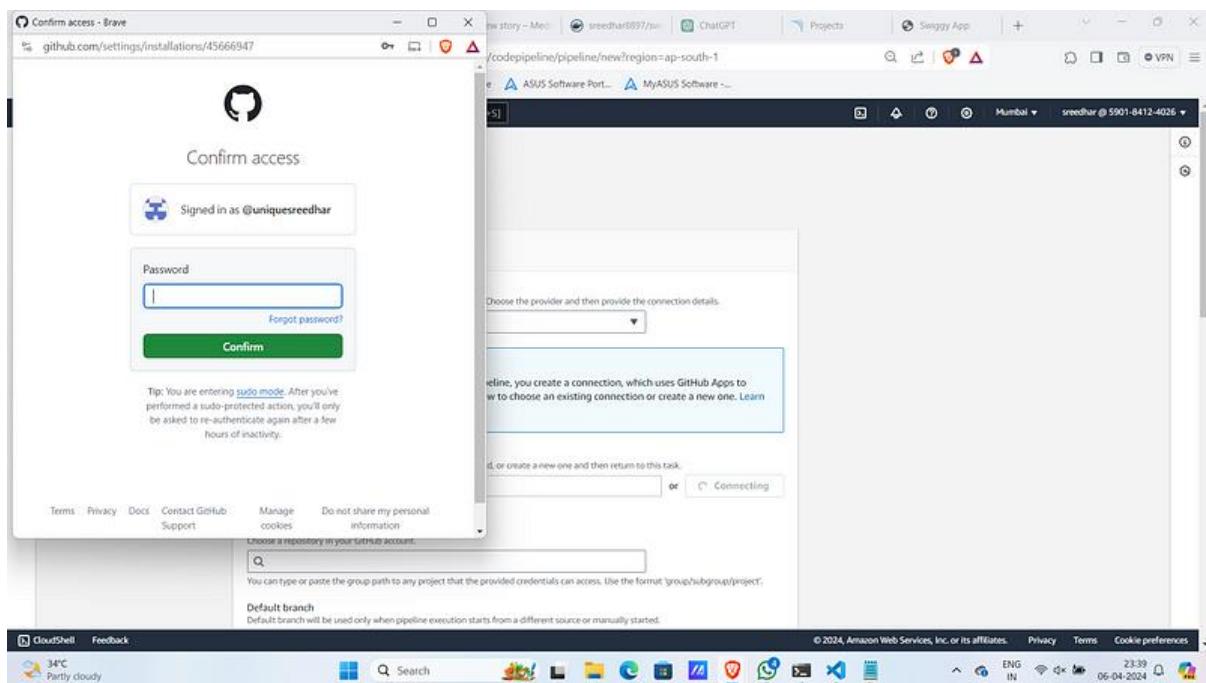
5. Provide a connection name.



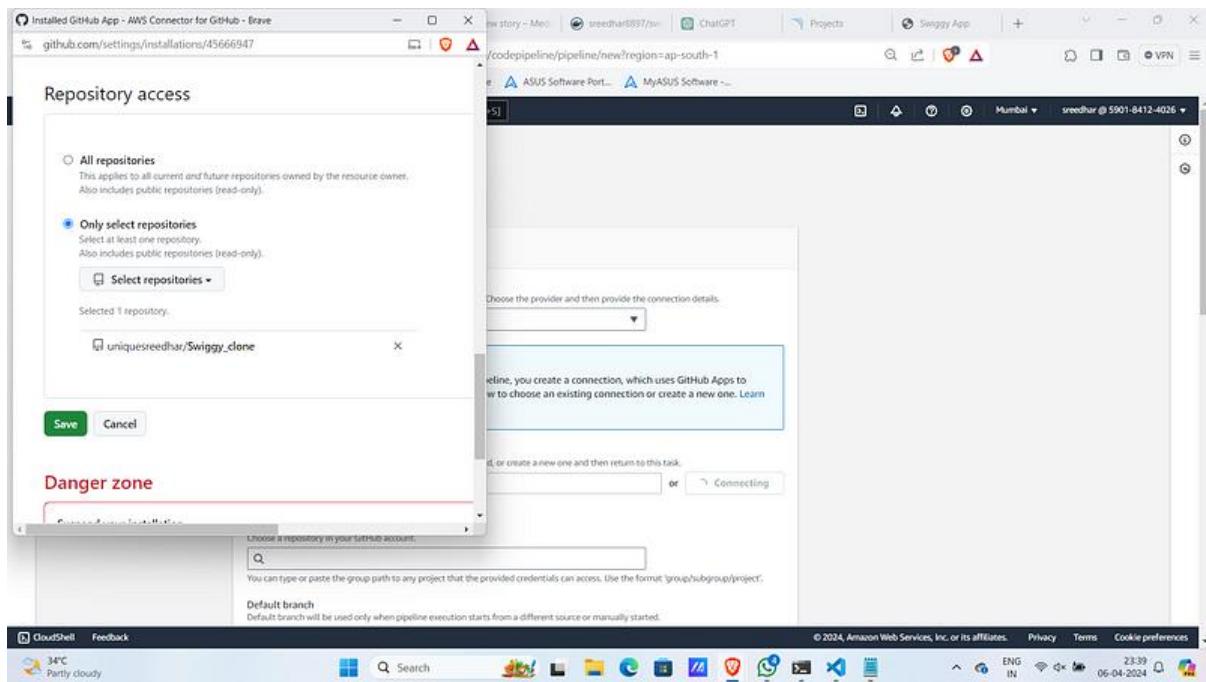
6. Click on "Install a new app".



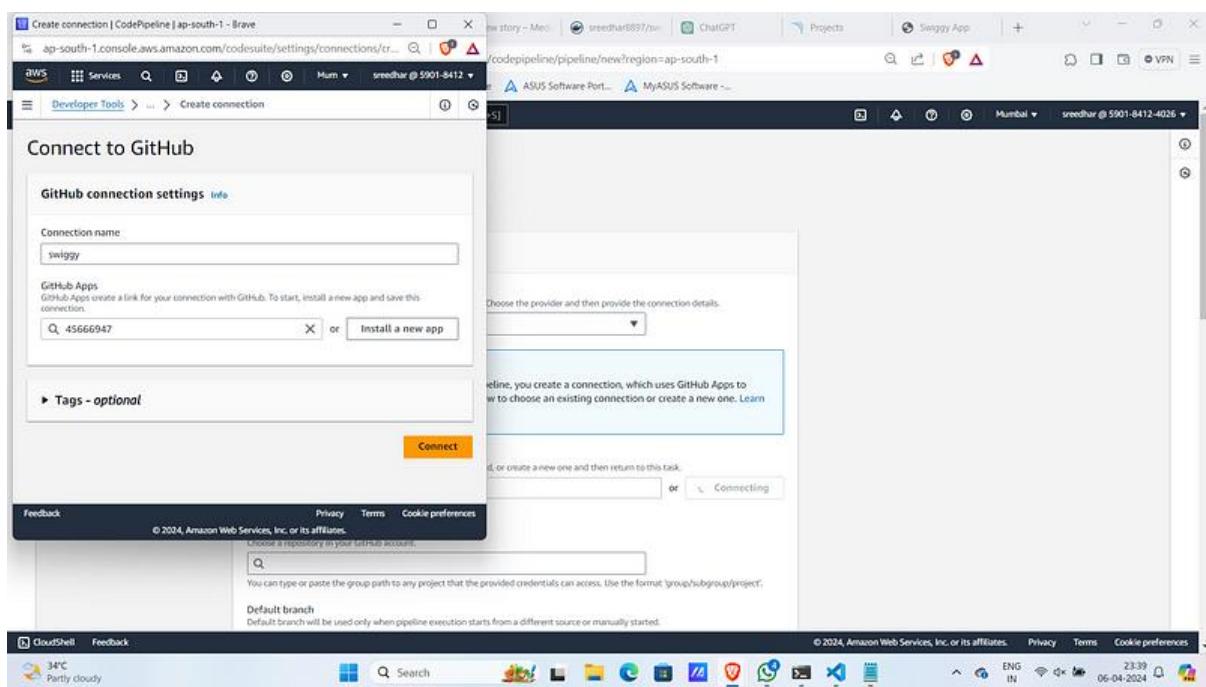
7. Login to GitHub using your credentials.



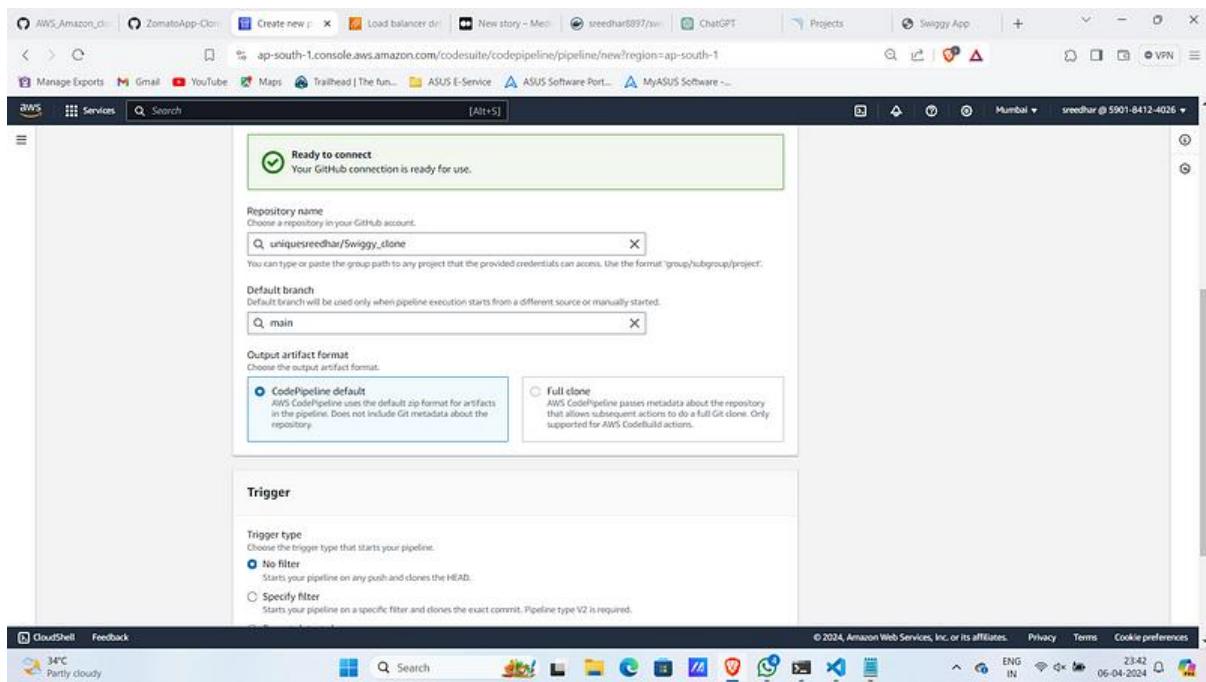
8. Grant access for that particular repo and save it.



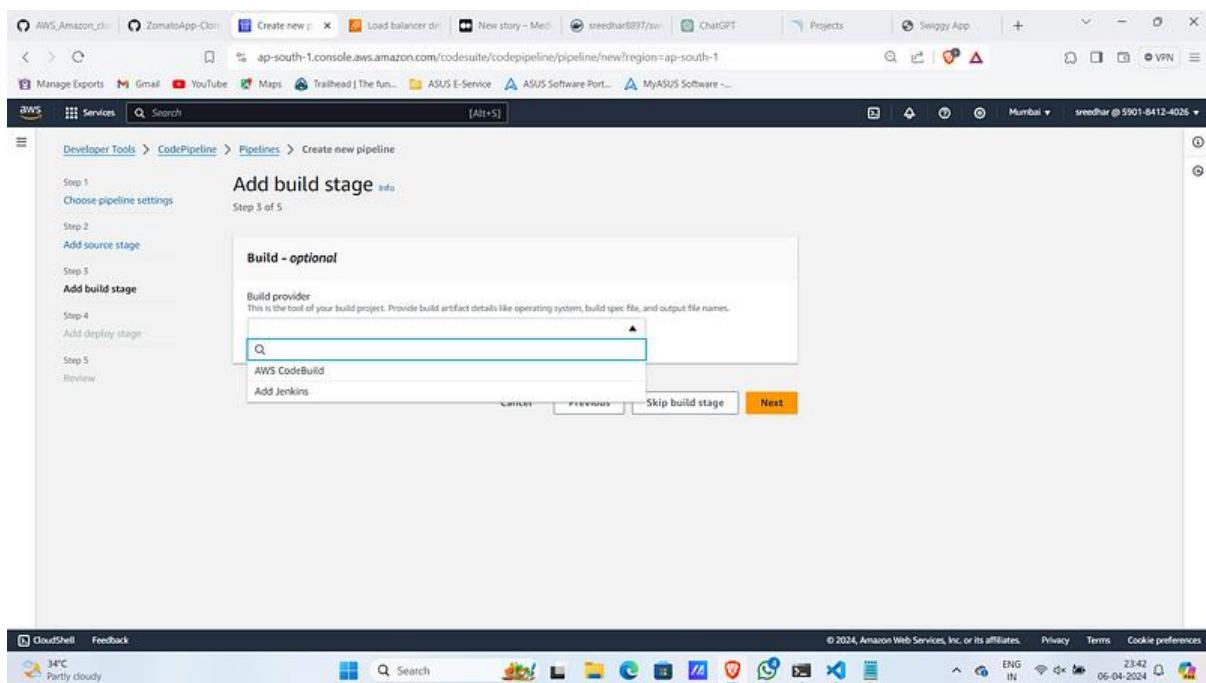
9. Click on connect.



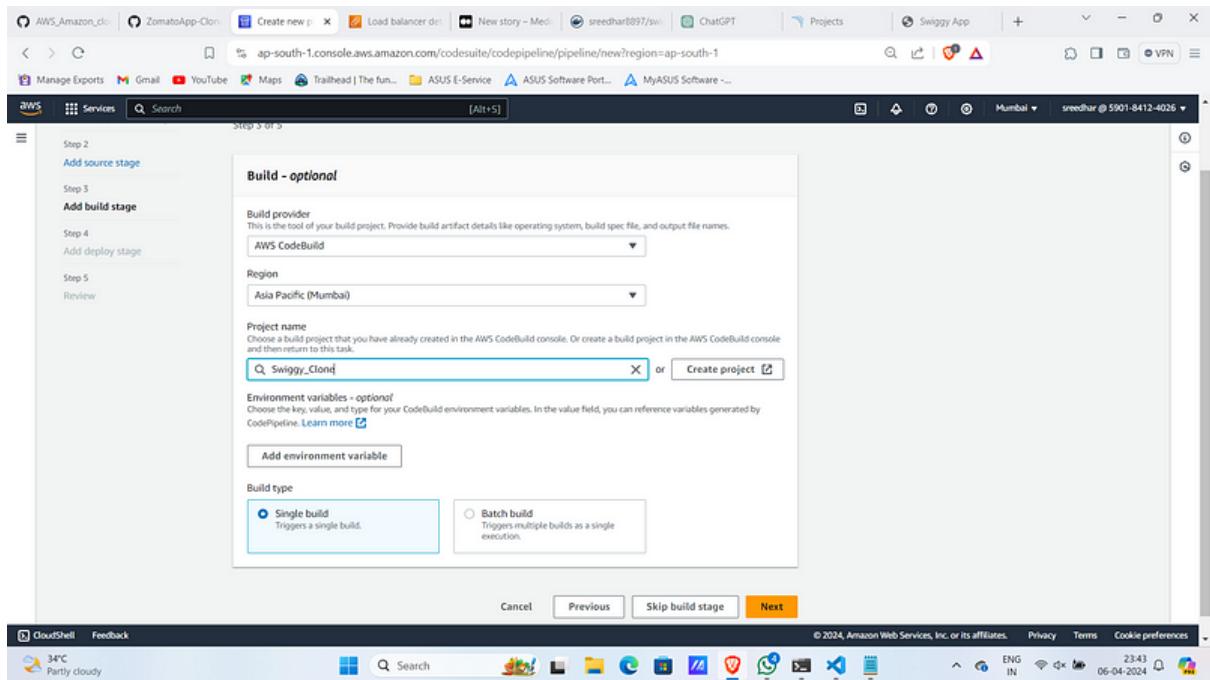
10. Select the repo and the branch. Under trigger type select "No filter". Click "Next".



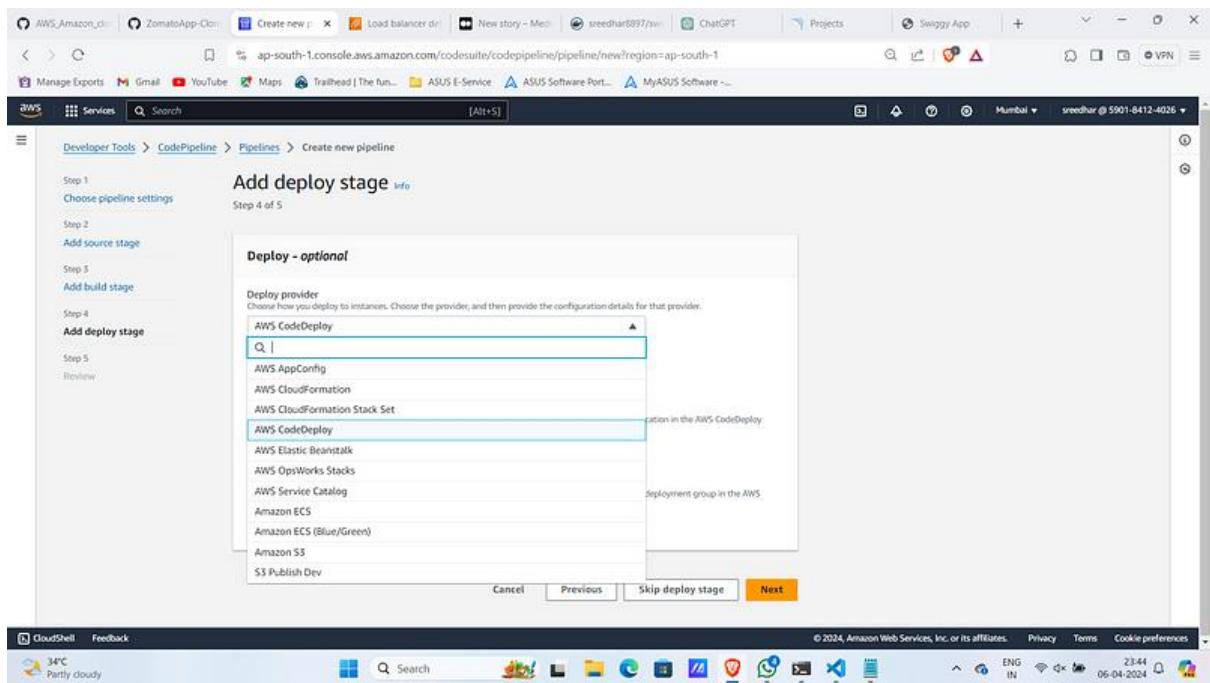
11. Under Build stage add provider as "AWS Code Build".



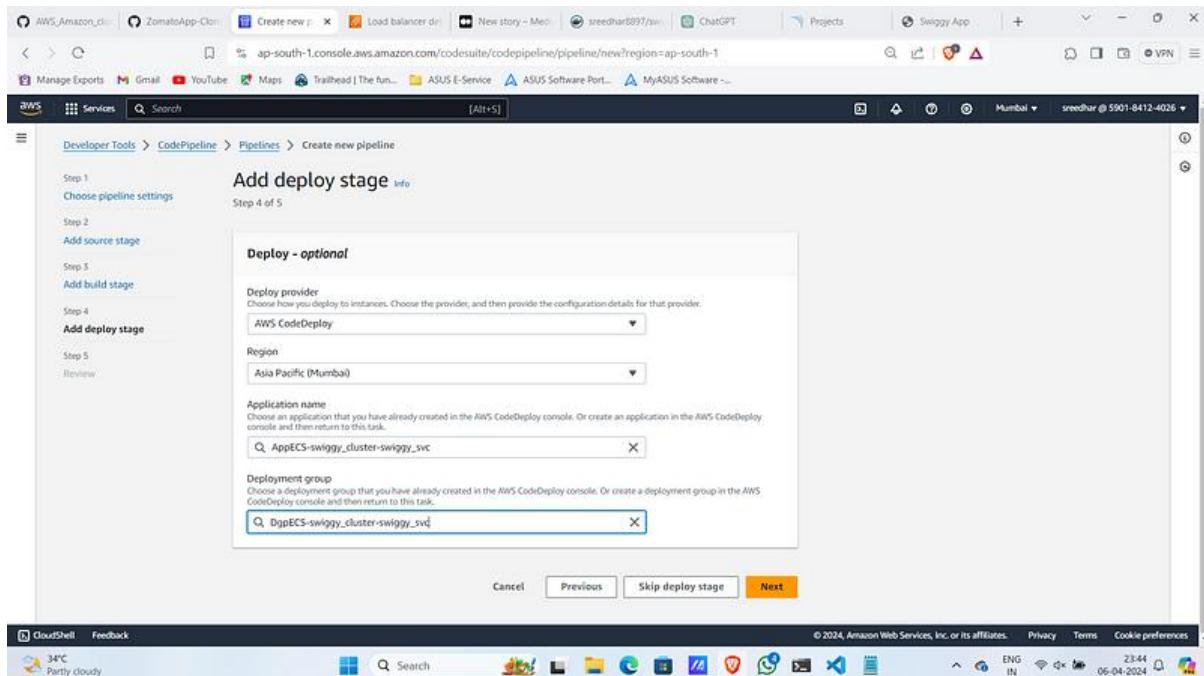
12. Select Project as one created. Click on “next”.



13. Under the deploy stage add AWS CodeDeploy as deploy provider.



13. Select the application name and deployment group created by the ECS service.



14. Review and Click on “create”.

Step:6 :- ECS Deployment.

1. Now make some changes to the application code.
 2. I am doing a change in public/index.html by changing the title.

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure: `SWIGGY_CLONE [WSL: Ubuntu-22.04]`, `Photos`, `public`, `index.html`, `src`, `appspec.yaml`, `buildspec.yaml`, `Dockerfile`, `package-lock.json`, and `package.json`.
- Editor (Center):** The `index.html` file is open, displaying its content. The code includes meta tags for charset, icon, viewport, and fonts.googleapis.com, as well as a title tag for "swiggy Application".
- Bottom Status Bar:** Shows the environment as `WSL:Ubuntu-22.04`, the file path as `main`, and the status bar with icons for battery, signal, and network.

3. Change it as “Swiggy app” and commit to the GitHub.
When the push happens code pipeline triggers automatically with webhook.

```

public > index.html > html > head > title
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="utf-8" />
5    <link rel="icon" href="%PUBLIC_URL%/.favIcon.icon" />
6    <meta name="viewport" content="width=device-width, initial-scale=1" />
7    <link rel="preconnect" href="https://fonts.googleapis.com">
8    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
9    <link href="https://fonts.googleapis.com/css2?family=Yantra+Manava:wght@100;300;400;500;700;900&display=swap" rel="stylesheet">
10   <link href="https://fonts.googleapis.com/css2?family=Font-Awesome:6.4.2/css/all.min.css" integrity="sha512-z3gipd2yNf1Y0bC0pk4qpor8akUjan+cShubuuan19qptobGoreCFKk1he1659CQKFBbbkuq1g80A==" crossorigin="anonymous" referrerpolicy="no-referrer" />
11   <title>Swiggy App</title>
12 </head>
13 <body>
14   <div id="root"></div>
15 </body>
16 </html>
17
18
19
20
21

```

4. Source and Build took usual time but code deploy took much.

5. Navigate to that deployment.

Step	Description	Status	Progress (%)
Step 1:	Deploying replacement task set	Completed	100%
Step 2:	Rerouting production traffic to replacement task set	Completed	100%
Step 3:	Wait 1 hour 0 minutes	Waiting	12%
Step 4:	Terminate original task set	In progress	0%

Traffic shifting progress

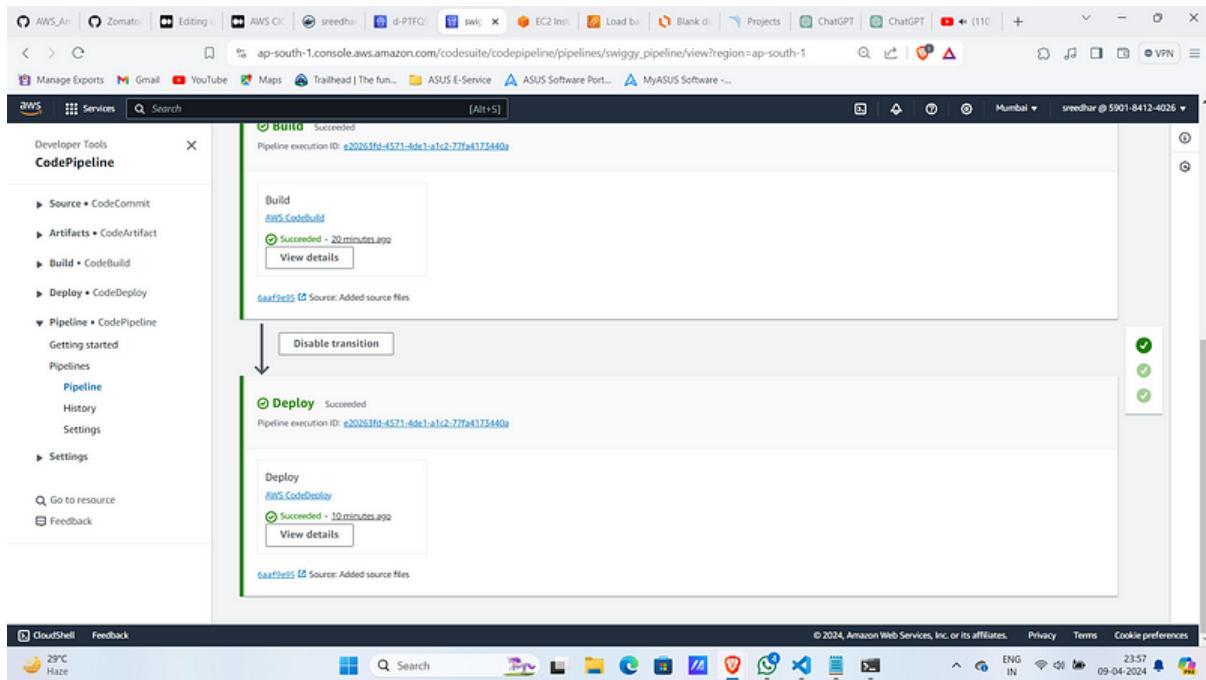
- Original: 0%
- Replacement: 100%

6. If you want both blue and green versions to be running leave it else if you don't want click on "Terminate original task set". It takes less time.

The screenshot shows the AWS CodeDeploy console for a deployment named d-PTFQ9PXB4. The deployment status indicates four steps have completed successfully: Deploying replacement task set, Rerouting production traffic to replacement task set, Wait, and Terminating original task set. The traffic shifting progress shows 0% for the original task set and 100% for the replacement task set.

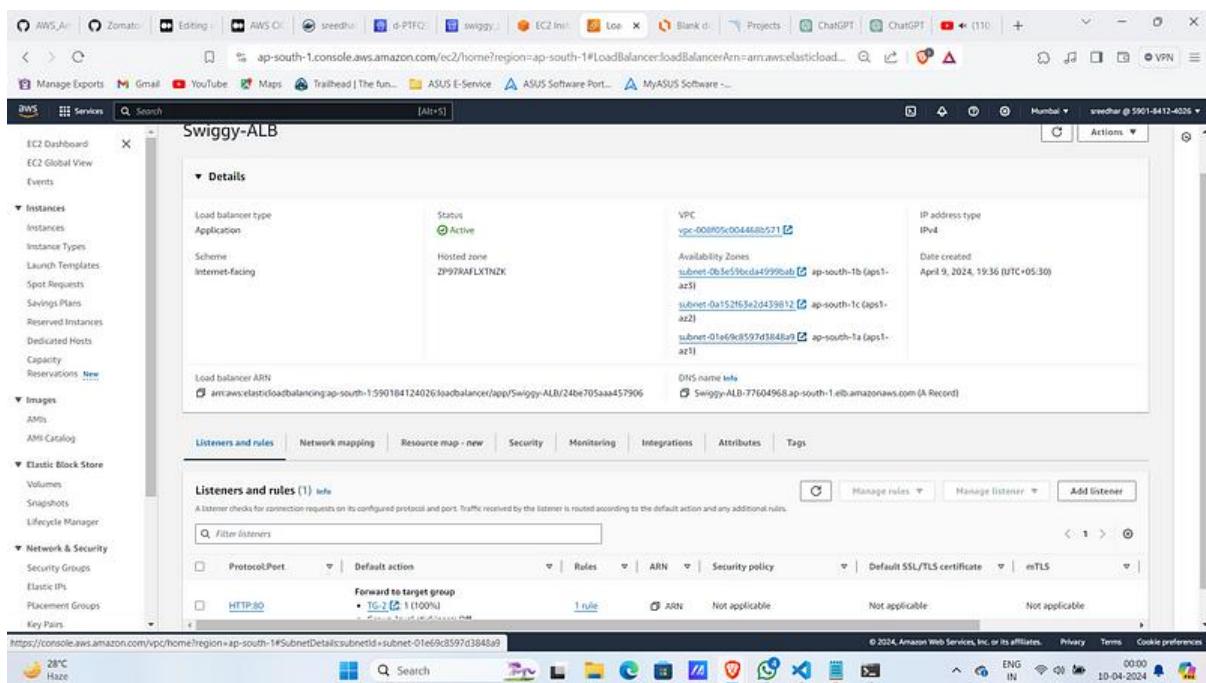
7. Upon success code pipeline looks as :

The screenshot shows the AWS CodePipeline console for the swiggy_pipeline. The pipeline type is V2 and the execution mode is QUEUED. The Source stage is listed as Succeeded with a GitHub (Version 2) source. The Build stage is listed as Succeeded with an AWS CodeBuild build.



8. Now navigate to load balancer section and click on the created one.

9. Copy the DNS of that load balancer.



Observe that the traffic is routing to TG-2 (Green) instead of TG-1(Blue).

10. Paste that in your favorite browser.

Observe that the title changed as expected.

Step:-7 : Clean Up

1. Deleted created Code Pipeline.
2. Delete ECS Cluster.
3. Delete Created Code Build.
4. Delete Sonar-Server EC2 Instance.