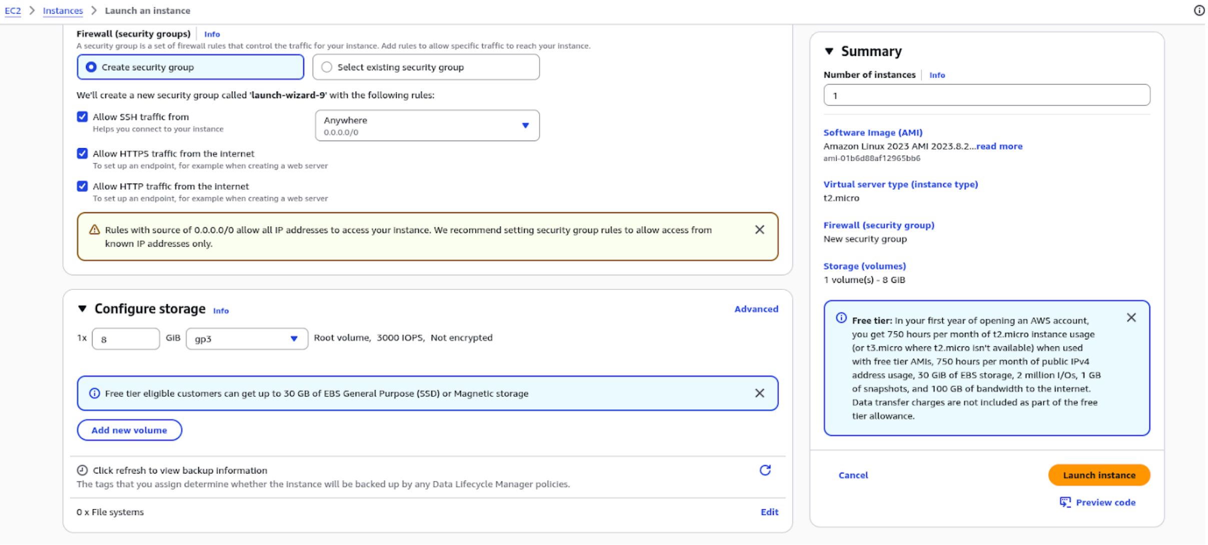
TASK 1- Automate Code Deployment Using CI/CD Pipeline(GitHub Actions)

Step 1- Launched a AWS Instance(t2-free tier) and setup the security configurations, also taken the private key for taking SSH-access



EC2 > Instances > Launch an instance

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

● Create security group ○ Select existing security group

We'll create a new security group called '**launch-wizard-9**' with the following rules:

☑ Allow SSH traffic from — Anywhere 0.0.0.0/0
Helps you connect to your instance

☑ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

☑ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Configure storage Info

1x 8 GiB gp3 Root volume, 3000 IOPS, Not encrypted

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

Click refresh to view backup information
The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

0 x File systems

Summary

Number of instances: 1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.8.2...read more
ami-01b6d88af12965bb6

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet. Data transfer charges are not included as part of the free tier allowance.

Step2- installed Docker, and started the service

1️⃣ Update your system

```
sudo apt update
sudo apt upgrade -y
```

2️⃣ Install required packages

```
sudo apt install apt-transport-https ca-certificates curl
software-properties-common -y
```

These are needed to allow `apt` to use HTTPS repositories.

3️⃣ Add Docker’s official GPG key

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
--dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

4️⃣ Add the Docker repository

```
echo "deb [arch=$(dpkg --print-architecture)
signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
| sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

5️⃣ Install Docker

```
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io -y
```

Check version:

```
docker --version
```

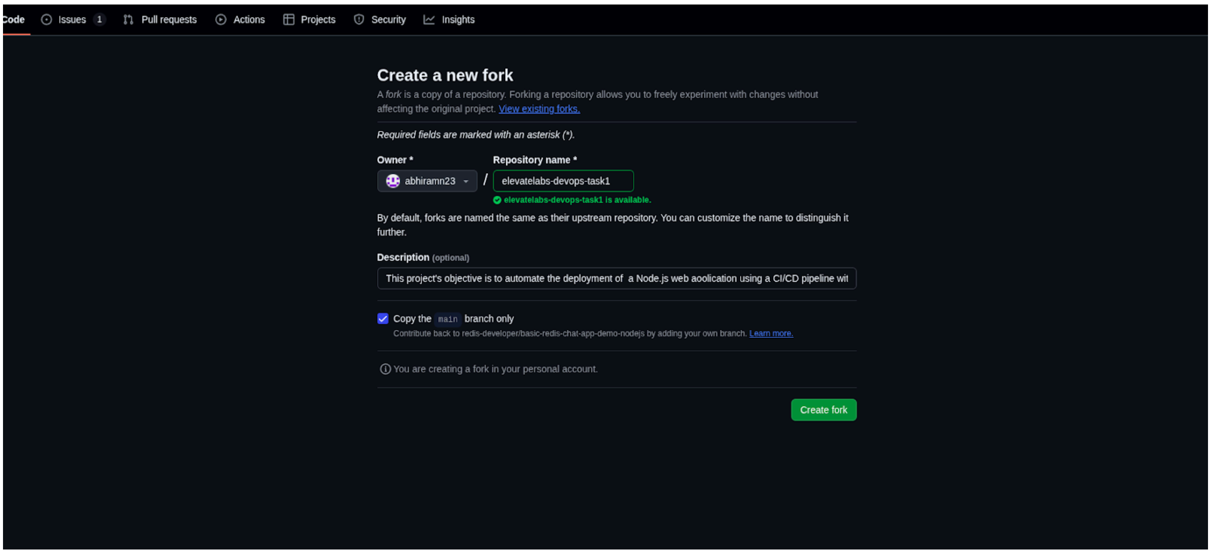
6️⃣ Start Docker service

```
sudo systemctl start docker
sudo systemctl enable docker
```

Check status:

```
sudo systemctl status docker
```

Step3- Forked a NodeJS webapplication (in GItHub)



Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. View existing forks.

Required fields are marked with an asterisk (*).

Owner * abhiramn23 / Repository name * elevatelabs-devops-task1

elevatelabs-devops-task1 is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

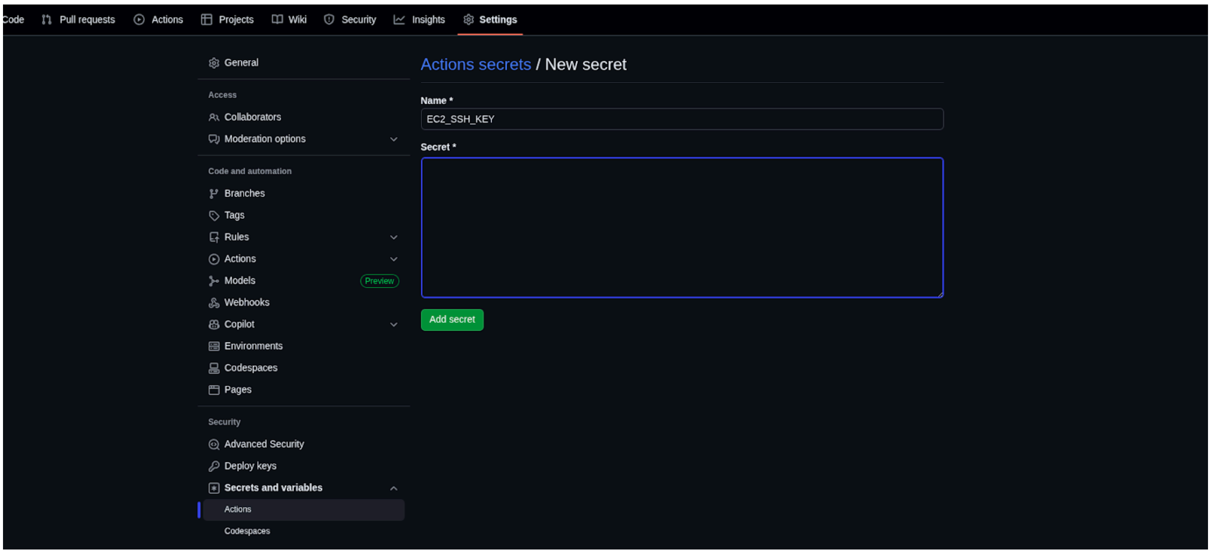
Description (optional)

This project's objective is to automate the deployment of a Node.js web aoolication using a CI/CD pipeline wit

☑ Copy the `main` branch only

Contribute back to redis-developer/basic-redis-chat-app-demo-nodejs by adding your own branch. Learn more.

You are creating a fork in your personal account.



Actions secrets / New secret

Name * EC2_SSH_KEY

Secret *

Added secrets, like ec2 instance public ip, hostname, ssh key value,docker username and docker token.

Step3-
Added .github/workflows/main.yaml
For automating the push, build and notification tasks

Step4- Go to GitHub Actions for checking the progress, once the workflow completes successfully open the EC2-instance.s public IP to see the web application is working propely

Redis chat demo

Welcome Back !

Sign in to continue

Name

Joe ▾

Password

•••••••••••

Sign in