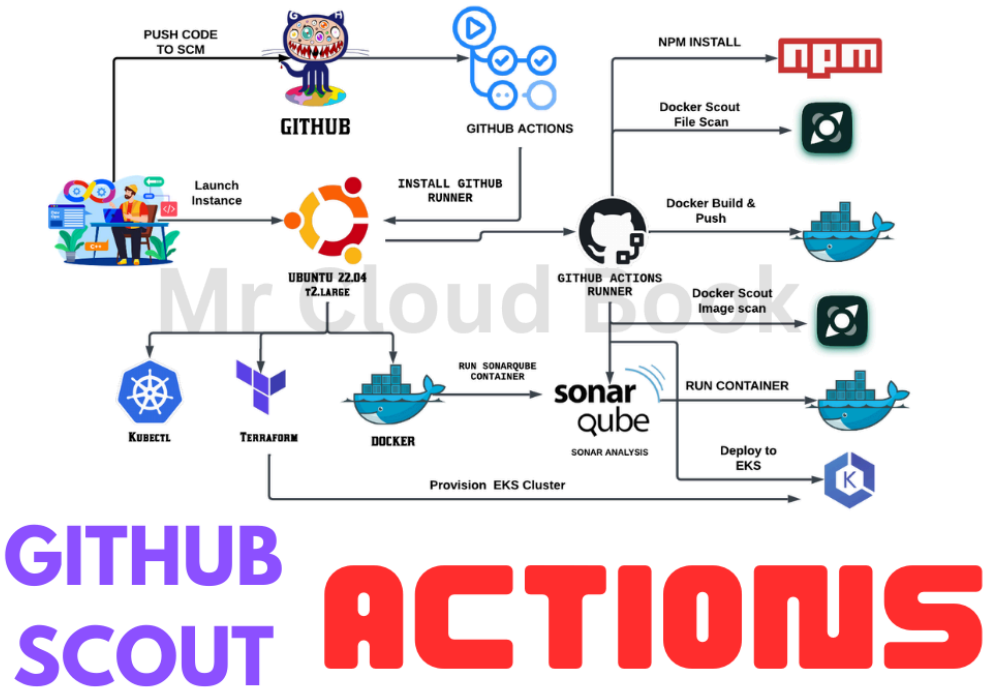
DevOps

GitHub Actions Scouting Myntra App | DevSecOps

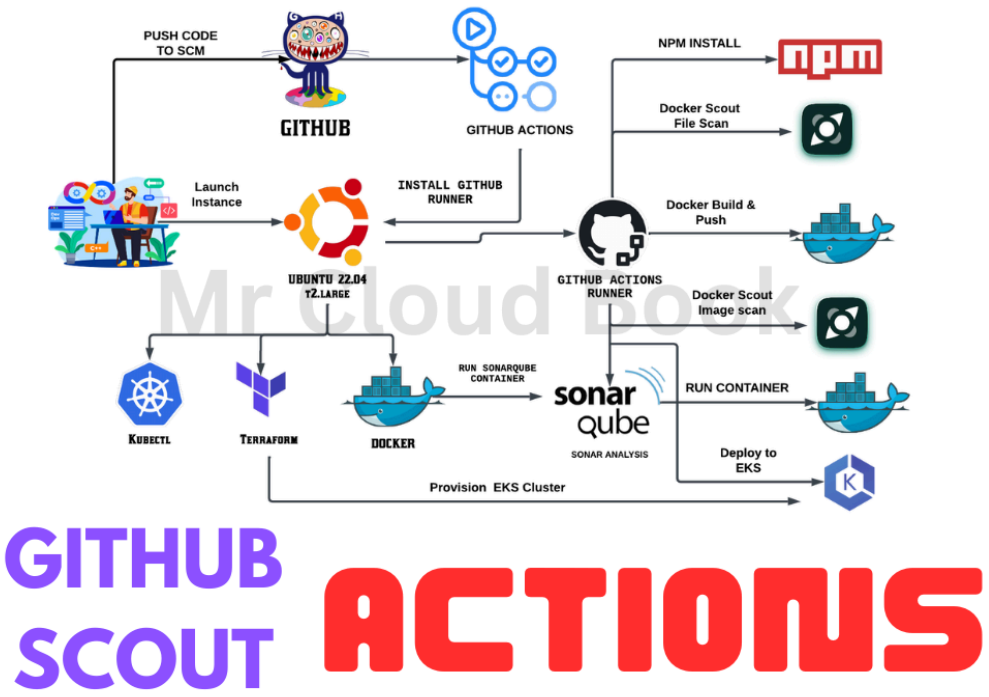
mrcloudbook.com · 21 January 2024



In today's fast-paced world of software development, automation is the name of the game. GitHub Actions is the ace up the sleeve of modern developers, enabling them to streamline their daily workflows in practical and impactful ways. In this article, we'll explore how GitHub Actions is making a real difference in real-life scenarios.

From Continuous Integration (CI) and Continuous Deployment (CD) to code quality assurance and security scanning, GitHub Actions brings automation to every

aspect of the development process. With custom workflows, enhanced collaboration, and release management, this tool empowers developers to be more efficient, reliable, and productive. Discover how GitHub Actions is not just a concept but a transformative solution in the daily lives of developers.



GitHub: https://github.com/Aj7Ay/Myntra-Clone.git

Contents [hide]

STEP 1A: Setting up AWS EC2 Instance and IAM Role
STEP 1B: IAM ROLE

STEP 1A: Setting up AWS EC2 Instance and IAM Role

1. **Sign in to the AWS Management Console:** Access the AWS Management Console using your credentials
2. **Navigate to the EC2 Dashboard:** Click on the “Services” menu at the top of the page and select “EC2” under the “Compute” section. This will take you to the EC2 Dashboard.
3. **Launch Instance:** Click on the “Instances” link on the left sidebar and then click the “Launch Instance” button.

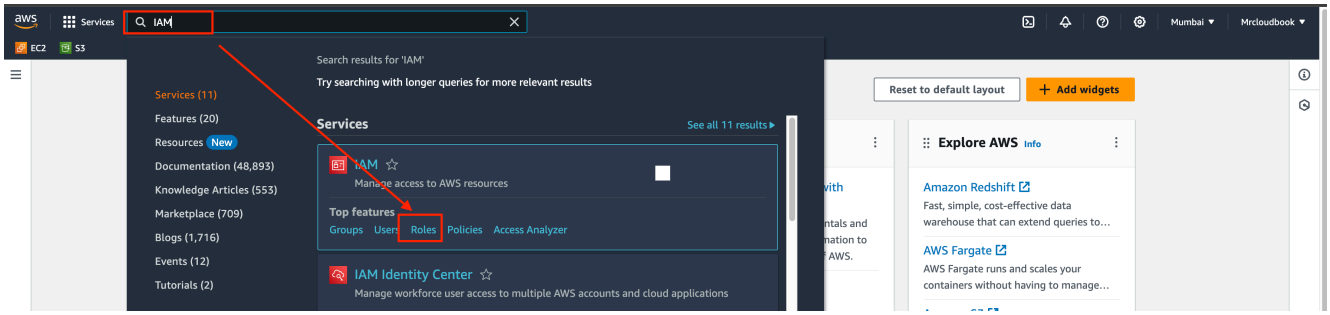
4. **Choose an Amazon Machine Image (AMI):** In the “Step 1: Choose an Amazon Machine Image (AMI)” section:
    - Select “AWS Marketplace” from the left-hand sidebar.
    - Search for “Ubuntu” in the search bar and choose the desired Ubuntu AMI (e.g., Ubuntu Server 22.04 LTS).
    - Click on “Select” to proceed.
5. **Choose an Instance Type:** In the “Step 2: Choose an Instance Type” section:
    - Scroll through the instance types and select “t2.large” from the list.
    - Click on “Next: Configure Instance Details” at the bottom.
6. **Configure Instance Details:** In the “Step 3: Configure Instance Details” section, you can leave most settings as default for now. However, you can configure settings like the network, subnet, IAM role, etc., according to your requirements.
    - Once done, click on “Next: Add Storage.”
7. **Add Storage:** In the “Step 4: Add Storage” section:
    - You can set the size of the root volume (usually /dev/sda1) to 30 GB by specifying the desired size in the “Size (GiB)” field.
    - Customize other storage settings if needed.
    - Click on “Next: Add Tags” when finished.
8. **Add Tags (Optional):** In the “Step 5: Add Tags” section, you can add tags to your instance for better identification and management. This step is optional but recommended for organizational purposes.
    - Click on “Next: Configure Security Group” when done.
9. **Configure Security Group:** In the “Step 6: Configure Security Group” section:
    - Create a new security group or select an existing one.
    - Ensure that at least SSH (port 22) is open for inbound traffic to allow remote access.
    - You might also want to open other ports as needed for your application’s requirements.

- Click on “Review and Launch” when finished.

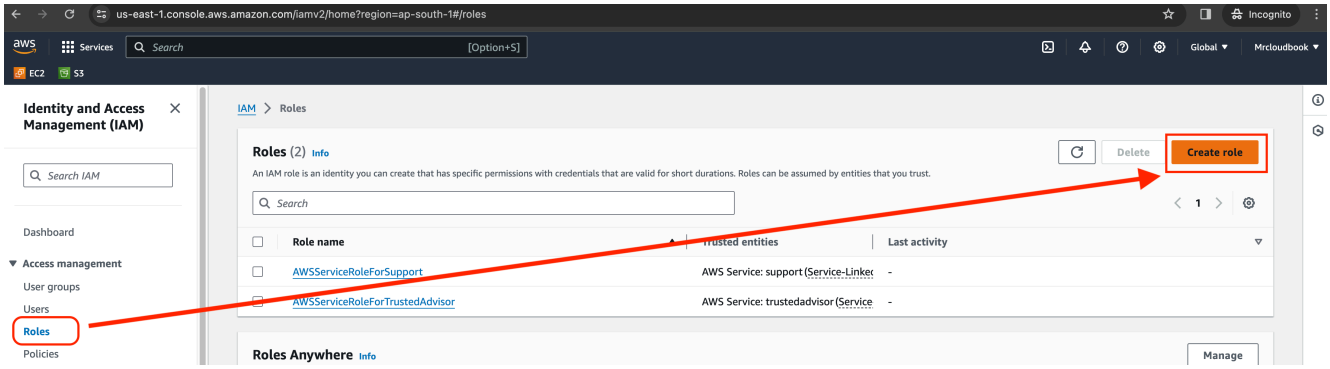
10. **Review and Launch:** Review the configuration details of your instance. If everything looks good:
    - Click on “Launch” to proceed.
    - A pop-up will prompt you to select or create a key pair. Choose an existing key pair or create a new one.
    - Finally, click on “Launch Instances.”
11. **Accessing the Instance:** Once the instance is launched, you can connect to it using SSH. Use the private key associated with the selected key pair to connect to the instance’s public IP or DNS address.

STEP 1B: IAM ROLE

Search for IAM in the search bar of AWS and click on roles.

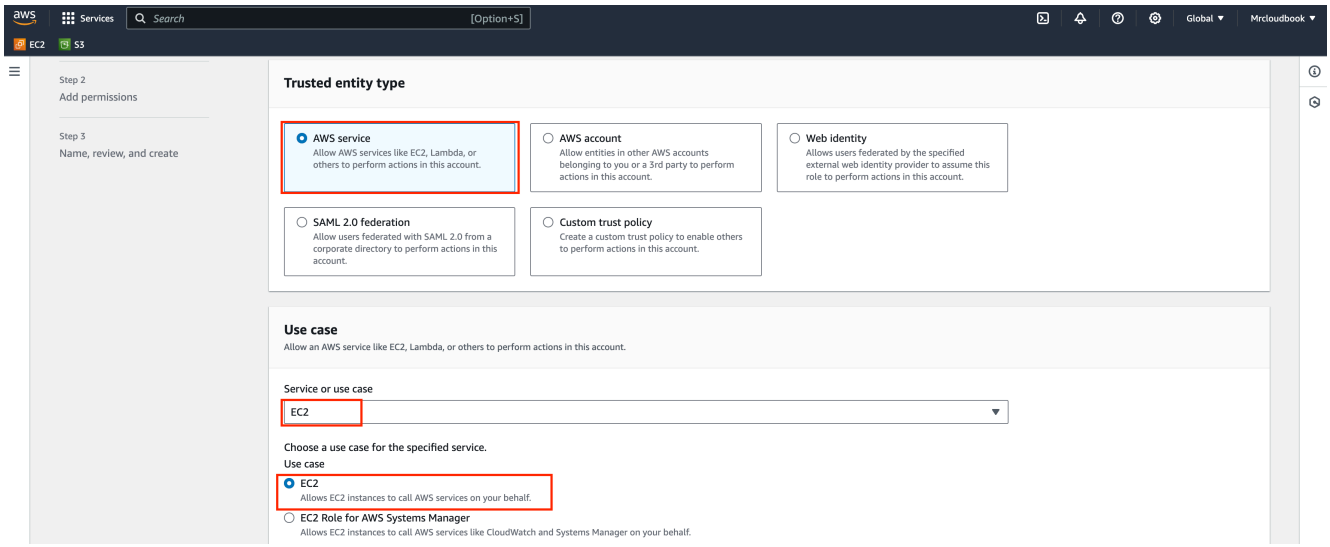


Click on Create Role

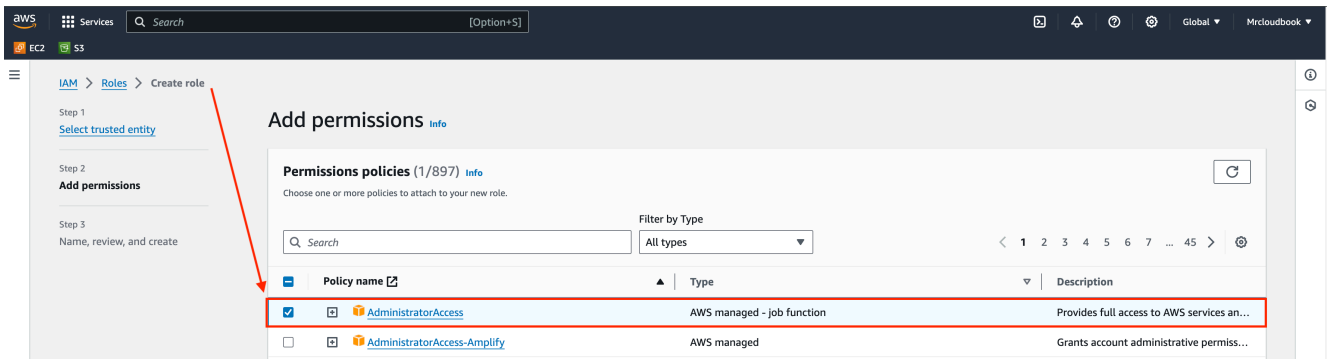


Select entity type as AWS service

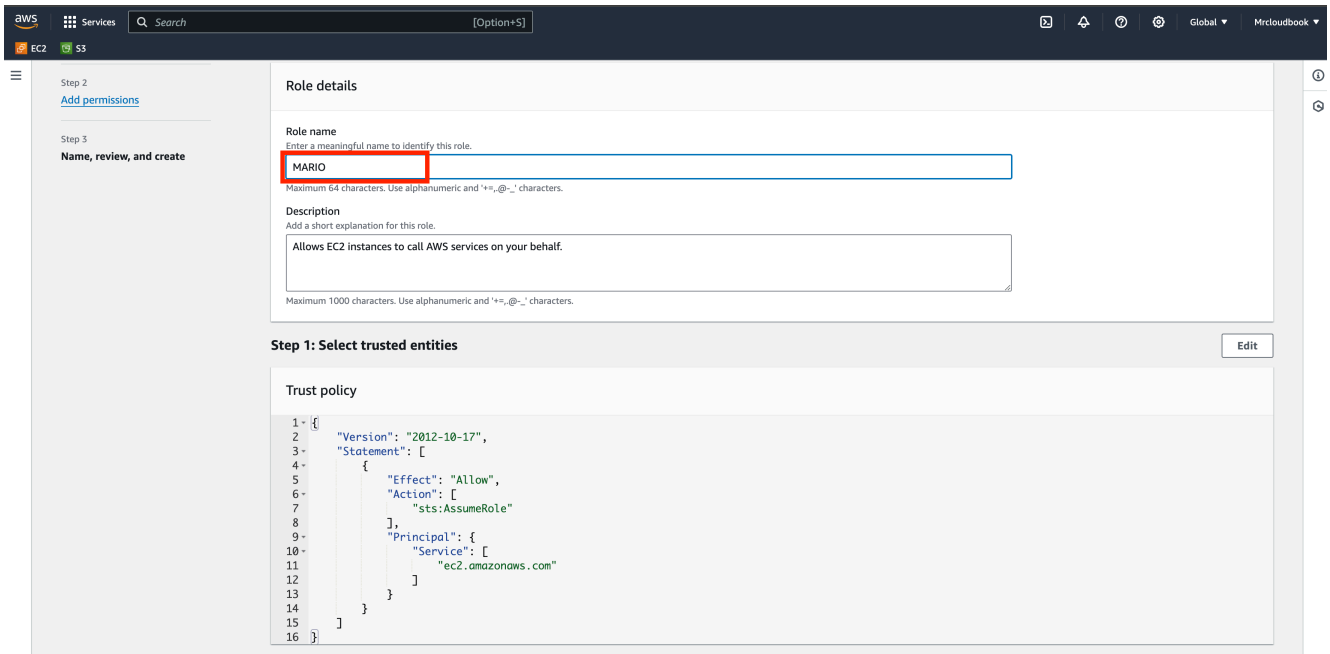
Use case as EC2 and click on Next.



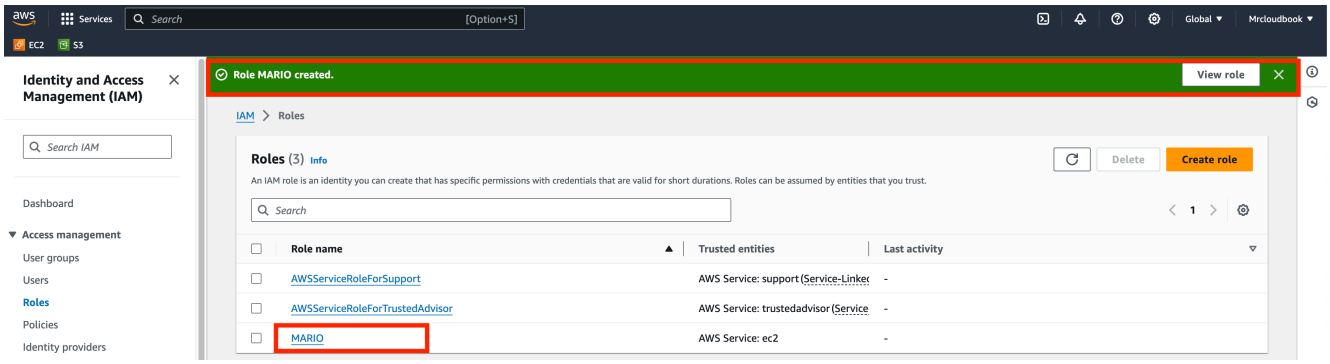
For permission policy select Administrator Access (Just for learning purpose), click Next.



Provide a Name for Role and click on Create role.



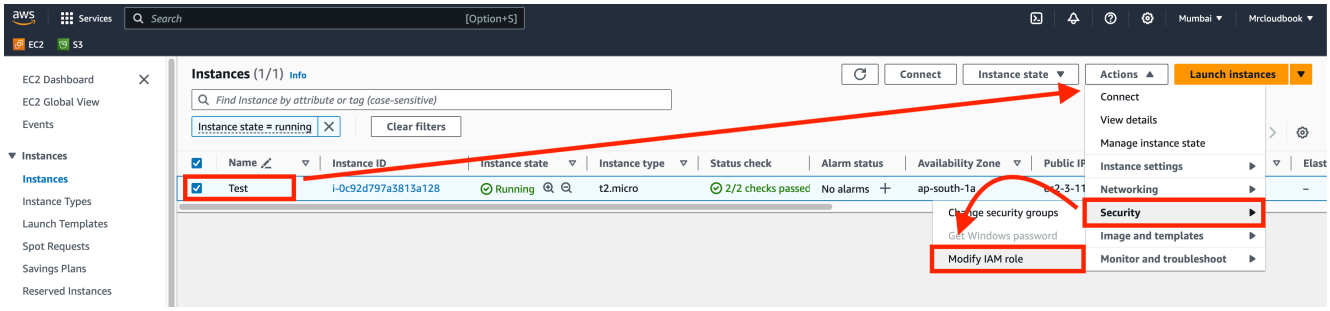
Role is created.



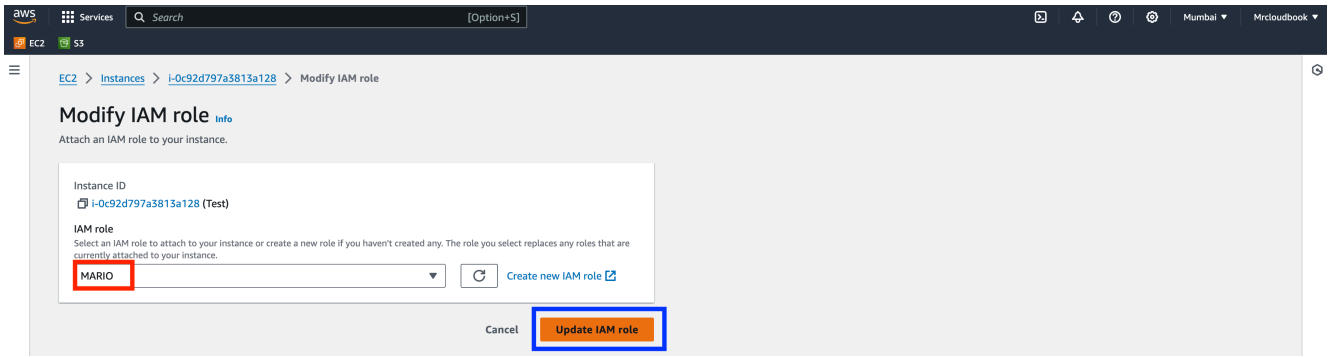
Now Attach this role to Ec2 instance that we created earlier, so we can provision cluster from that instance.

Go to EC2 Dashboard and select the instance.

Click on Actions –> Security –> Modify IAM role.



Select the Role that created earlier and click on Update IAM role.



Connect the instance to Mobaxtreme or Putty.

STEP 2: INSTALL REQUIRED PACKAGES ON INSTANCE

Create Docker & Docker scout Installation script

```
vi docker-setup.sh
```

Script

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dea
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/do
  $(. /etc/os-release &amp;&amp; echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list &gt; /dev/null
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx
curl -sSfL https://raw.githubusercontent.com/docker/scout-cli/main/insta
sudo usermod -aG docker ubuntu
newgrp docker
sudo chmod 777 /var/run/docker.sock
```

Provide executable versions

```
sudo chmod 777 docker-setup.sh
sh docker-setup.sh
```

Create Script for other packages

```
vi script-packages.sh
```

Script

```
#!/bin/bash
sudo apt update -y
sudo touch /etc/apt/keyrings/adoptium.asc
sudo wget -O /etc/apt/keyrings/adoptium.asc https://packages.adoptium.ne
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.ad
sudo apt update -y
sudo apt install temurin-17-jdk -y
/usr/bin/java --version

# Install Terraform
sudo apt install wget -y
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]
sudo apt update &amp;&amp; sudo apt install terraform

# Install kubectl
sudo apt update
sudo apt install curl -y
curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/releas
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
kubectl version --client

# Install AWS CLI
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awsc
sudo apt-get install unzip -y
unzip awscliv2.zip
sudo ./aws/install

# Install Node.js 16 and npm
curl -fsSL https://deb.nodesource.com/gpgkey/nodesource.gpg.key | sudo g
echo "deb [signed-by=/usr/share/keyrings/nodesource-archive-keyring.gpg]
```

```
sudo apt update
sudo apt install -y nodejs
```

```
#!/bin/bash
sudo apt update -y
sudo touch /etc/apt/keyrings/adoptium.asc
sudo wget -O /etc/apt/keyrings/adoptium.asc https://packages.adoptium.net/artifactory/api/gpg/key/public
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/artifactory/deb $(awk -F= '/^VERSION_CODENAME/{print$2}' /etc/os-release) main" | sudo tee /etc
/apt/sources.list.d/adoptium.list
sudo apt update -y
sudo apt install temurin-17-jdk -y
/usr/bin/java --version
# Install Trivy
sudo apt-get install wget apt-transport-https gnupg lsb-release -y
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | sudo tee -a /etc/apt/sources.list.d/trivy.list
sudo apt-get update
sudo apt-get install trivy -y
# Install Terraform
sudo apt install wget -y
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.l
ist
sudo apt update && sudo apt install terraform
# Install kubectl
sudo apt update
sudo apt install curl -y
curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
kubectl version --client
# Install AWS CLI
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
sudo apt-get install unzip -y
unzip awscliv2.zip
sudo ./aws/install
# Install Node.js 16 and npm
curl -fsSL https://deb.nodesource.com/gpgkey/nodesource.gpg.key | sudo gpg --dearmor -o /usr/share/keyrings/nodesource-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/nodesource-archive-keyring.gpg] https://deb.nodesource.com/node_16.x focal main" | sudo tee /etc/apt/sources.list.d/nodesource.list
sudo apt update
sudo apt install -y nodejs
~
```

Give executable permissions for script and run it

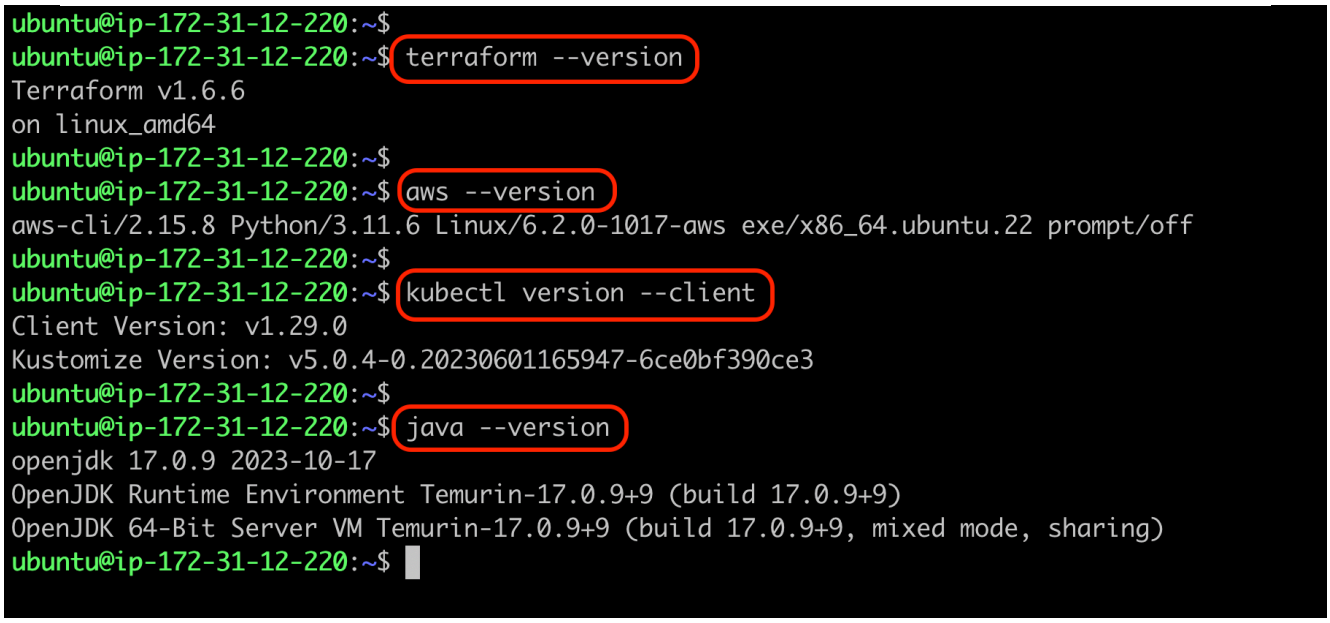
```
sudo chmod 777 script-packages.sh
sh script-packages.sh
```

```
ubuntu@ip-172-31-12-220:~$
ubuntu@ip-172-31-12-220:~$ vi script.sh
ubuntu@ip-172-31-12-220:~$
ubuntu@ip-172-31-12-220:~$ sudo chmod 777 script.sh
ubuntu@ip-172-31-12-220:~$
```

Check package versions

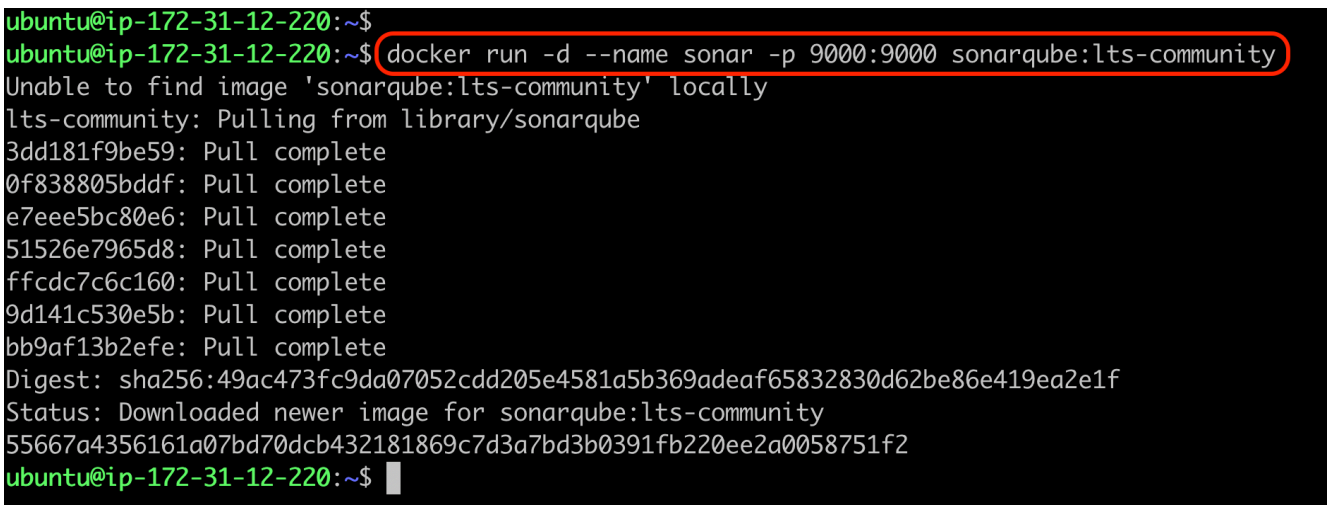
```
docker --version
terraform --version
```

```
aws --version
kubectl version
node -v
java --version
```



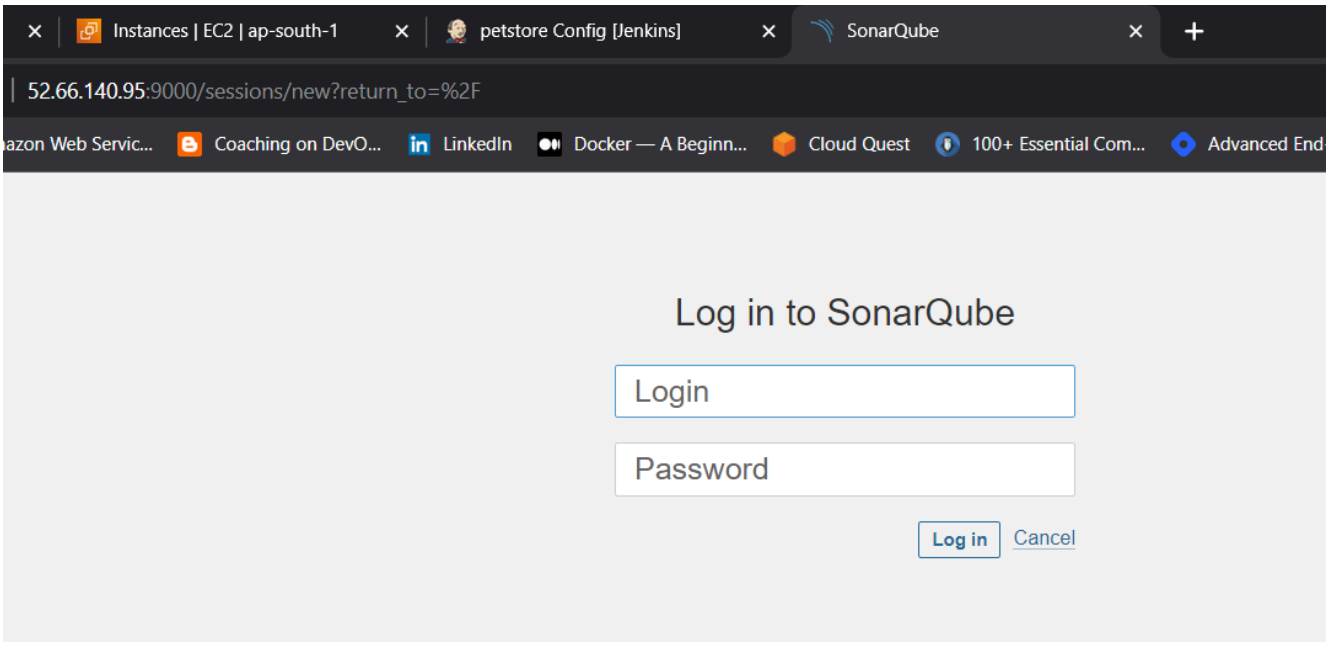
Run the Sonarqube container

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



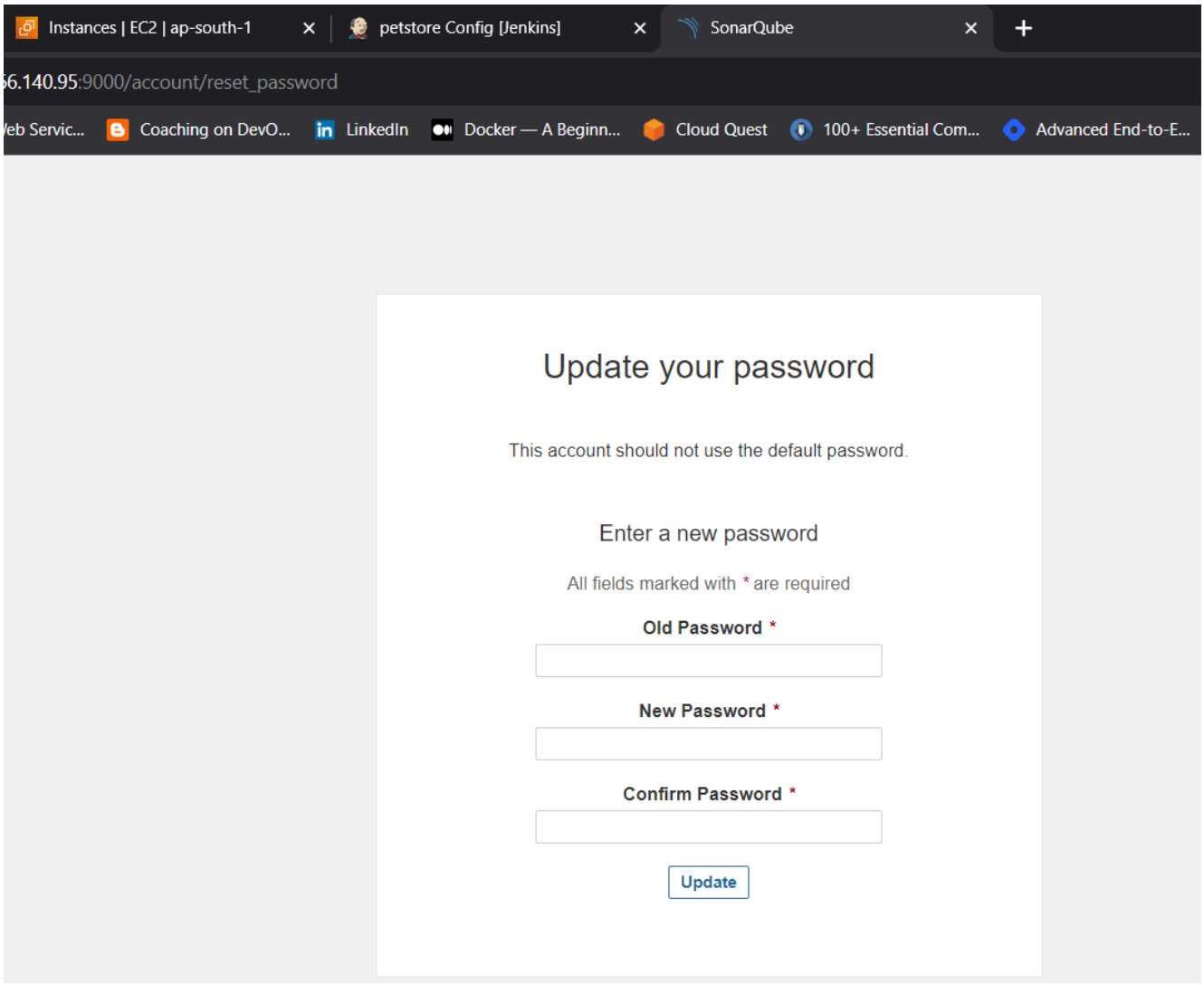
Now copy the IP address of the ec2 instance

```
ec2-public-ip:9000
```

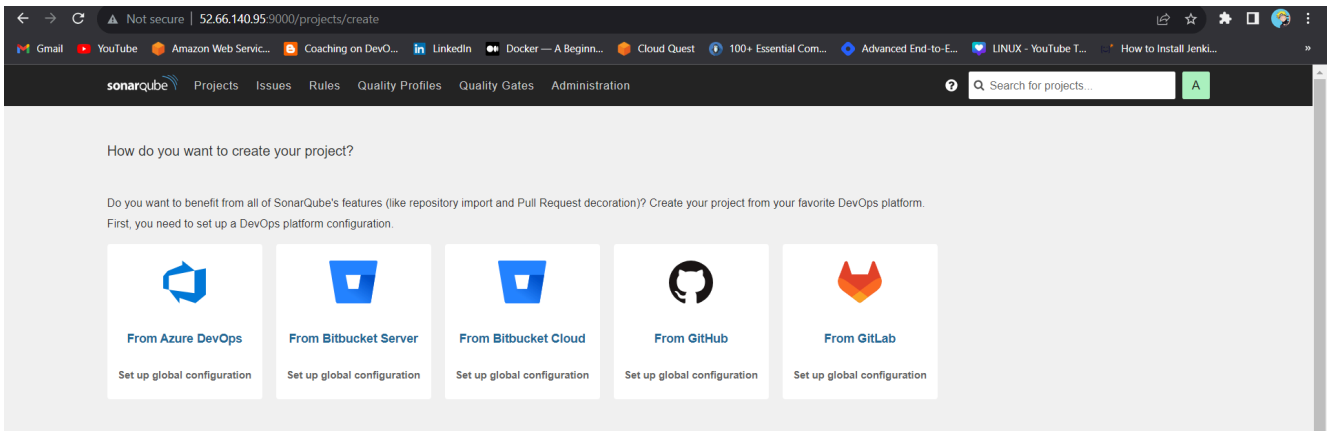


Provide Login and password

```
login admin
password admin
```



Update your Sonarqube password & This is the Sonarqube dashboard

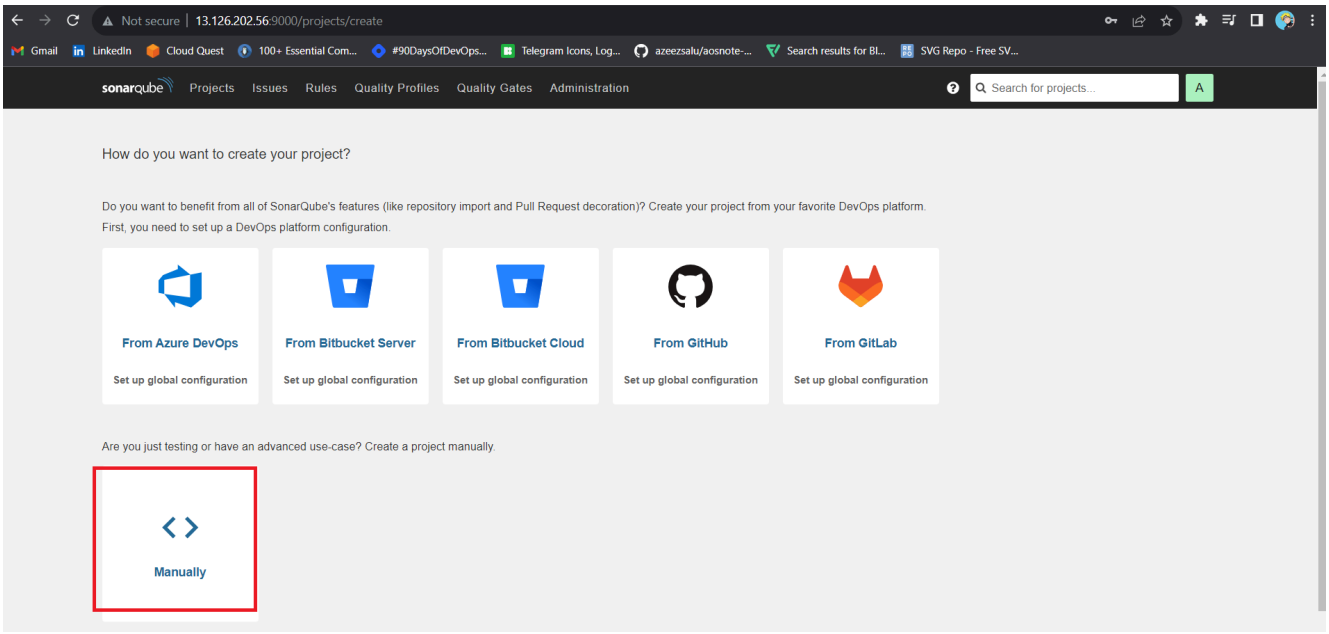


Step 3: Integrating SonarQube with GitHub Actions

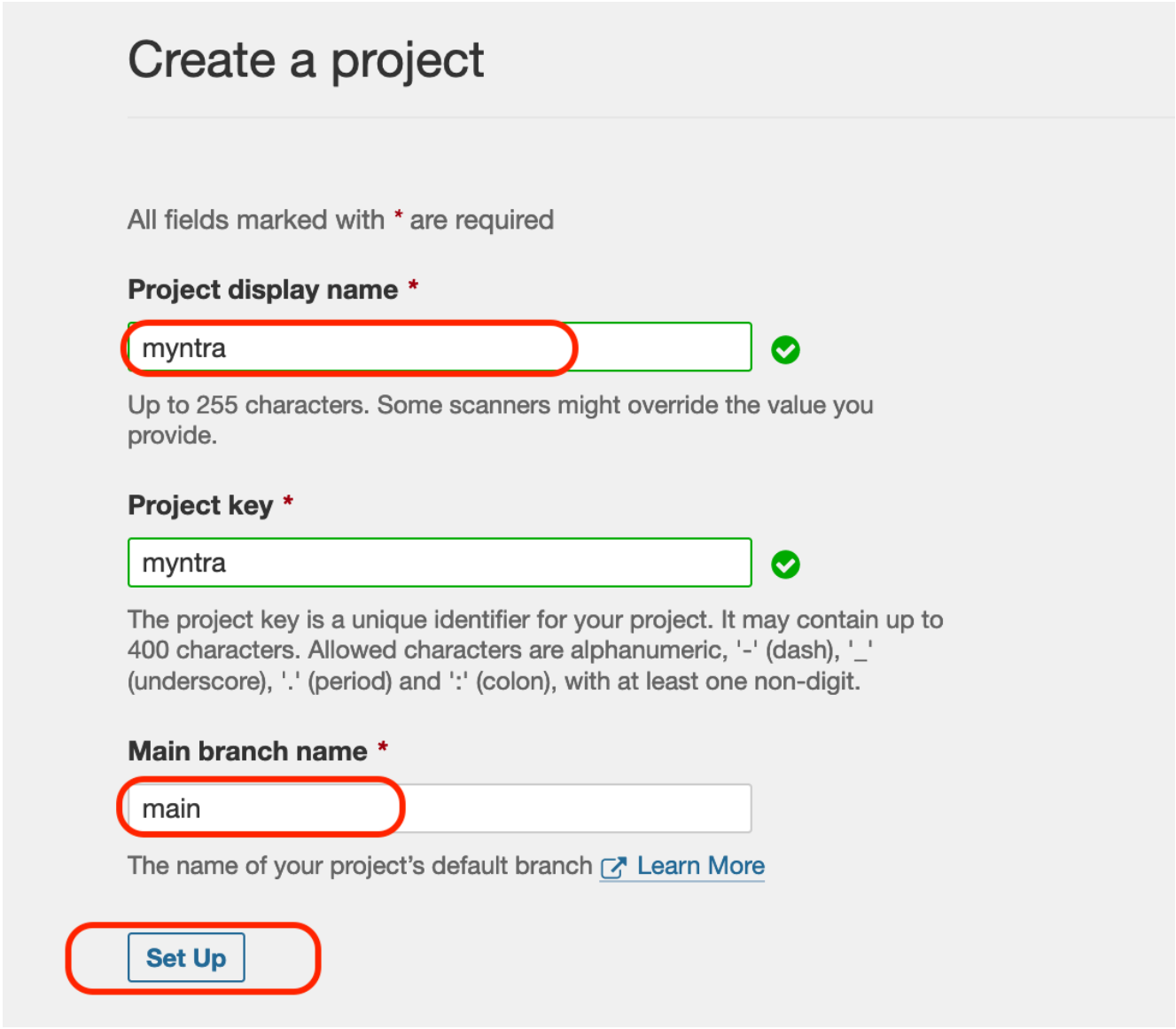
Integrating SonarQube with GitHub Actions allows you to automatically analyze your code for quality and security as part of your continuous integration pipeline.

We already have Sonarqube up and running

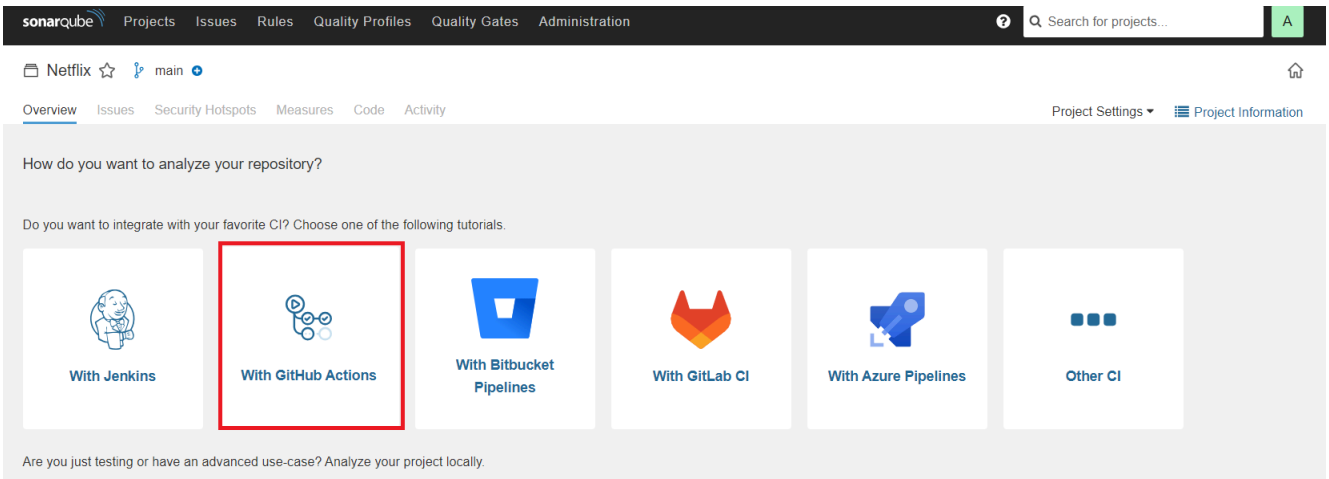
On Sonarqube Dashboard click on Manually



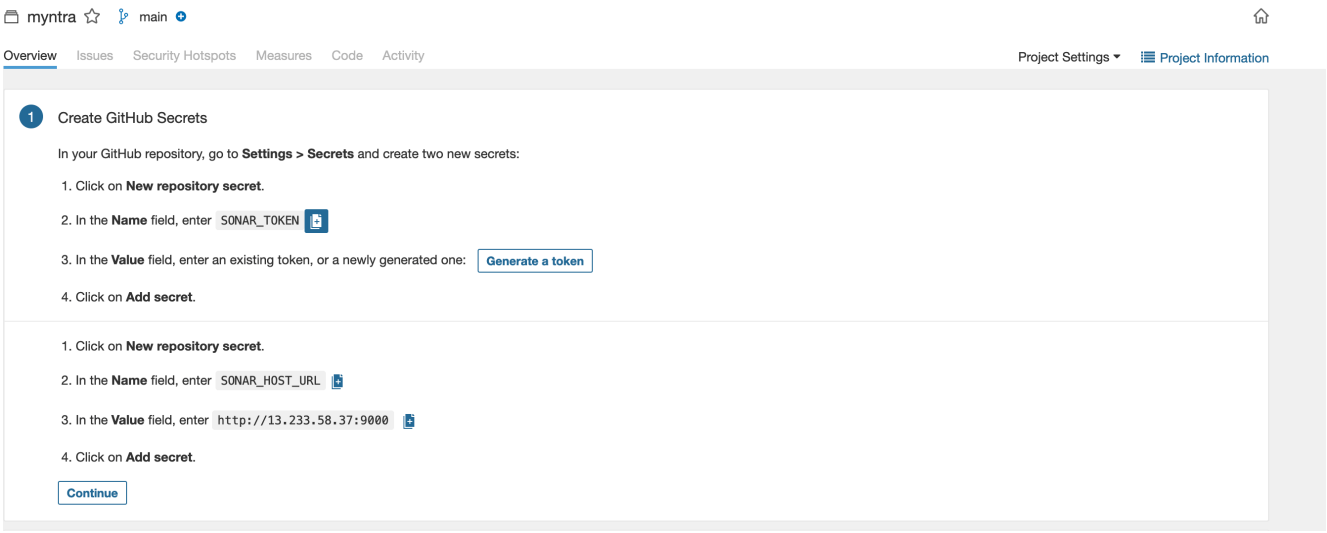
Next, provide a name for your project and provide a Branch name and click on setup



On the next page click on With GitHub actions

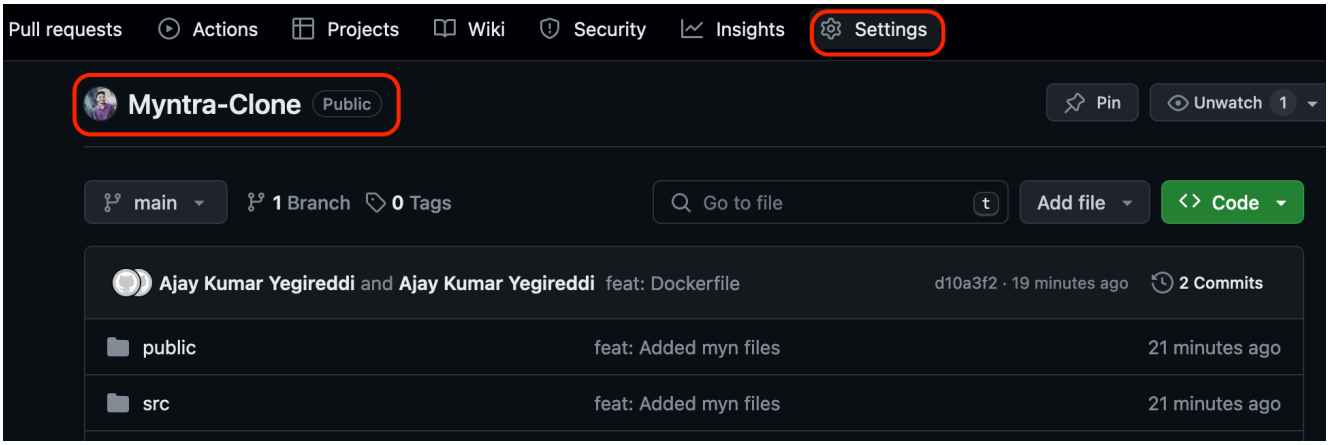


This will Generate an overview of the Project and provide some instructions to integrate

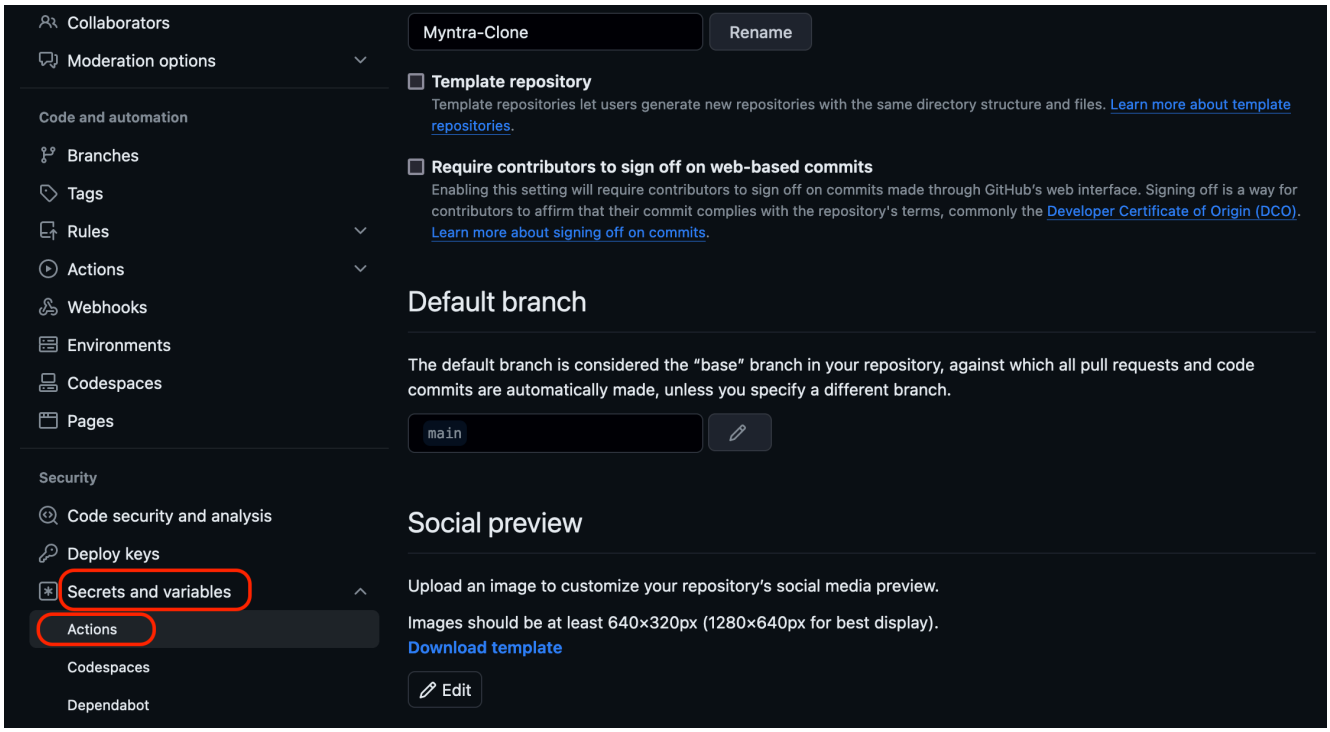


Let's Open your GitHub and select your Repository

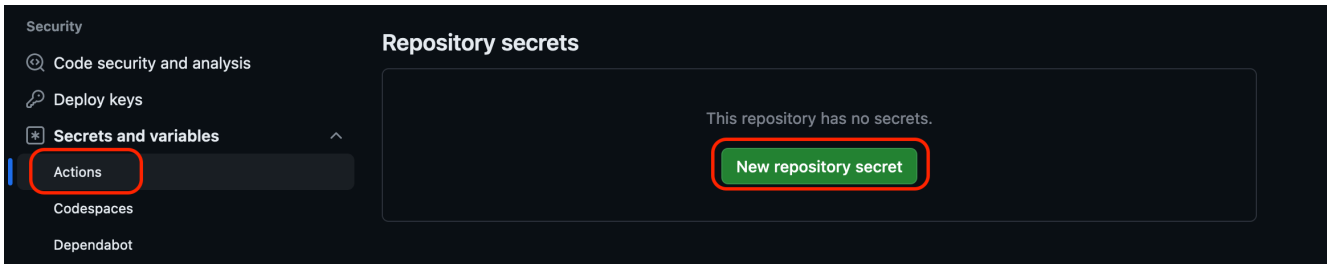
In my case it is Myntra-clone and Click on Settings



Search for Secrets and variables and click on and again click on actions

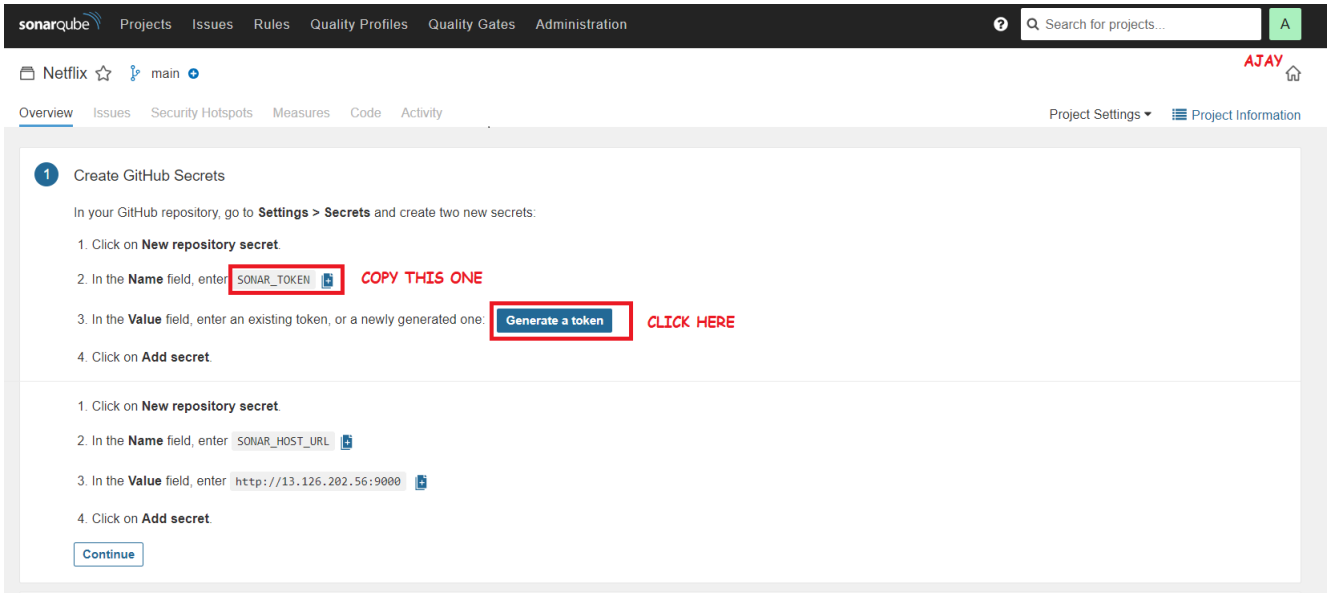


It will open a page like this click on New Repository secret

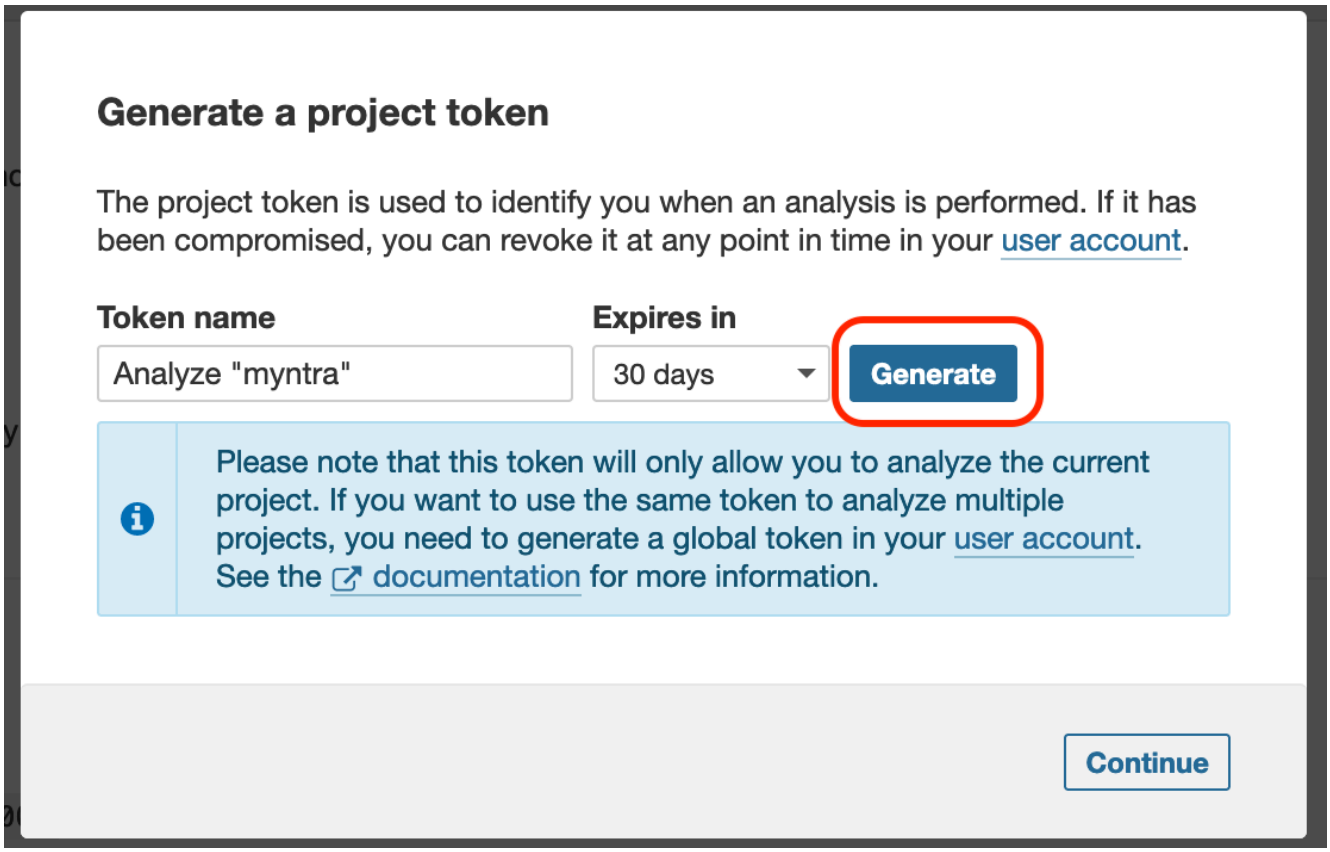


Now go back to Your Sonarqube Dashboard

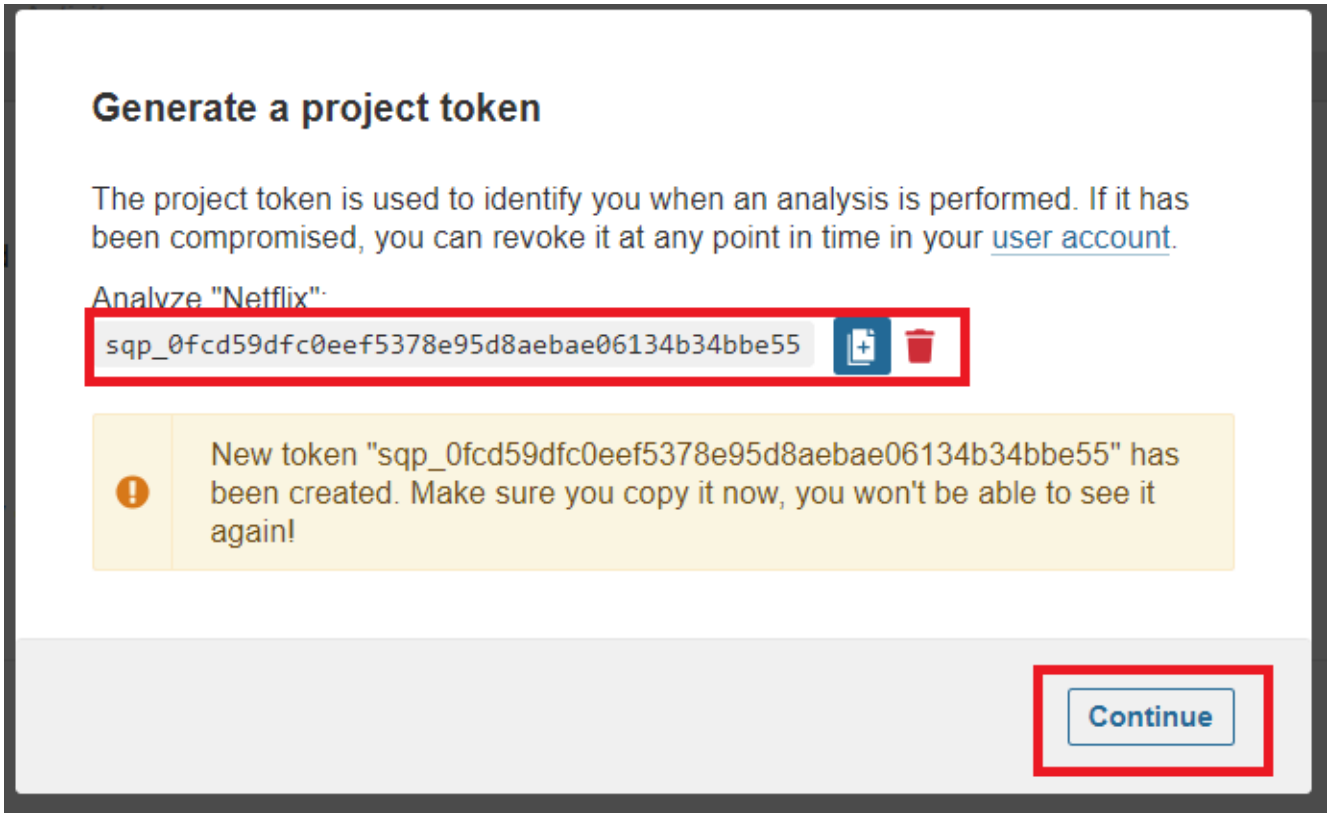
Copy SONAR_TOKEN and click on Generate Token



Click on Generate



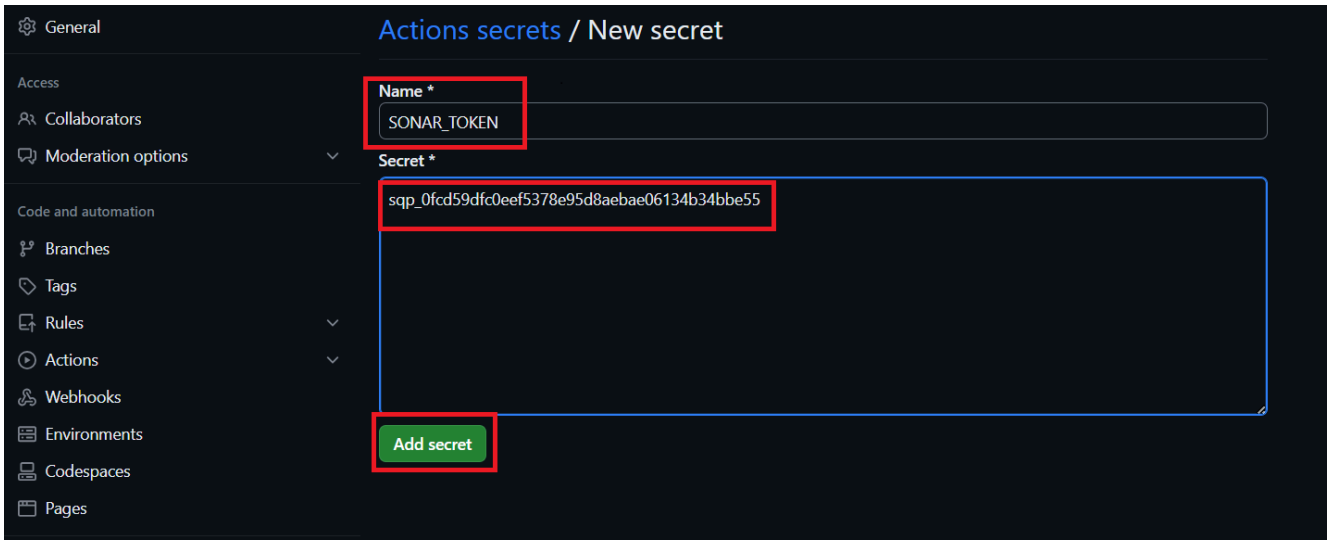
Let's copy the Token and add it to GitHub secrets



Now go back to GitHub and Paste the copied name for the secret and token

Name: SONAR_TOKEN

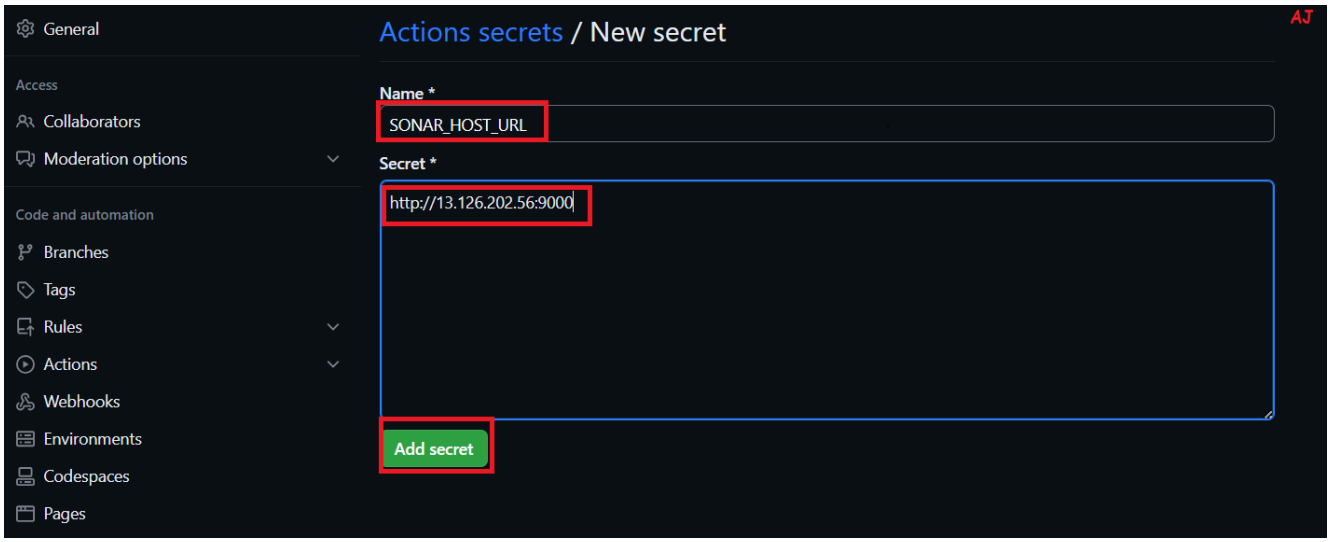
Secret: Paste Your Token and click on Add secret



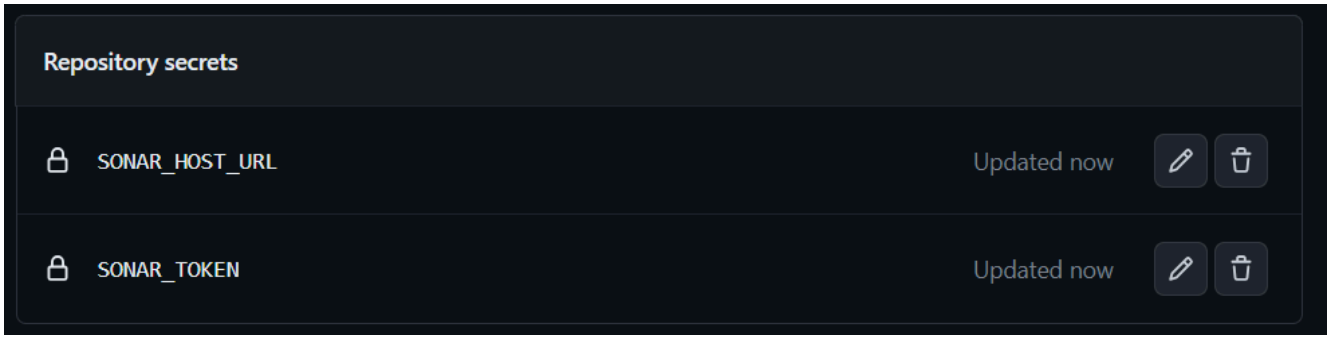
Now go back to the Sonarqube Dashboard

Copy the Name and Value

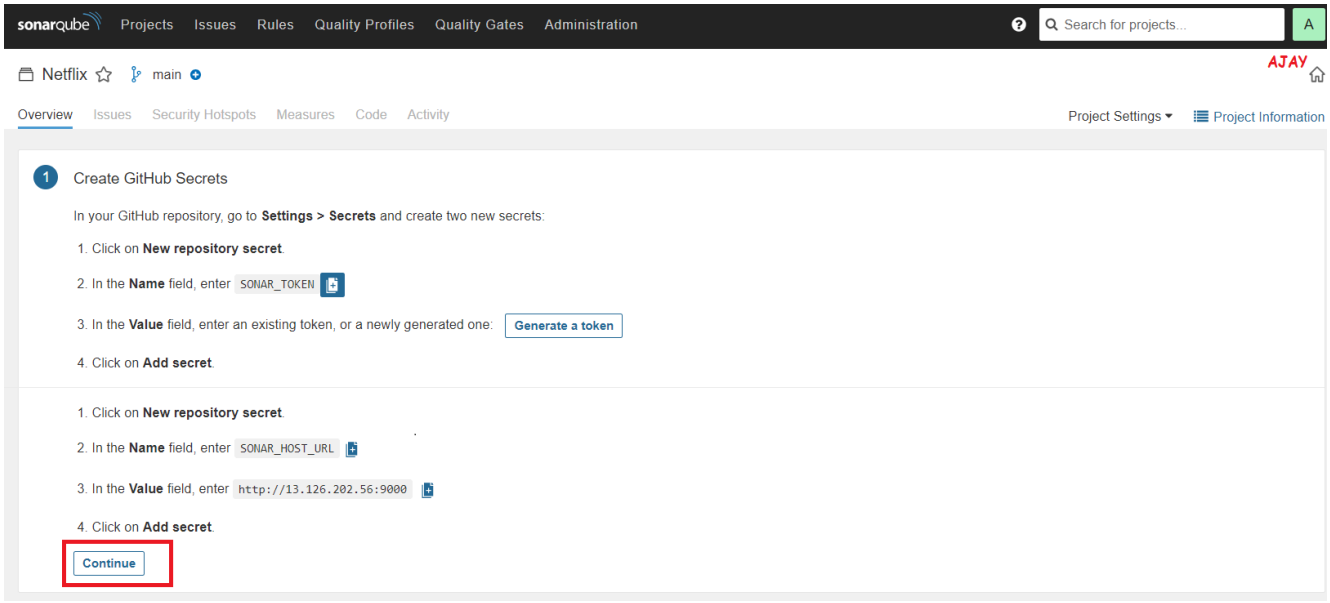
Go to GitHub now and paste-like this and click on add secret



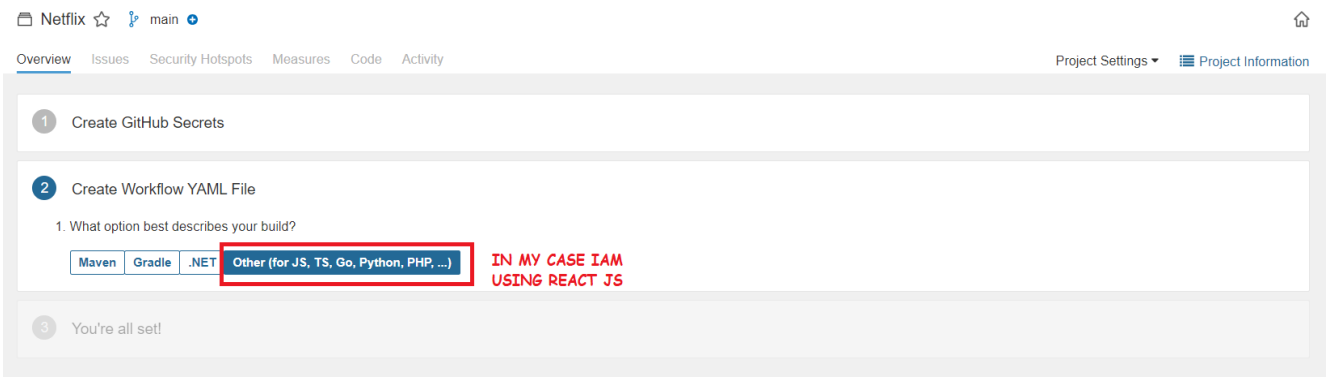
Our Sonarqube secrets are added and you can see



Go to Sonarqube Dashboard and click on continue

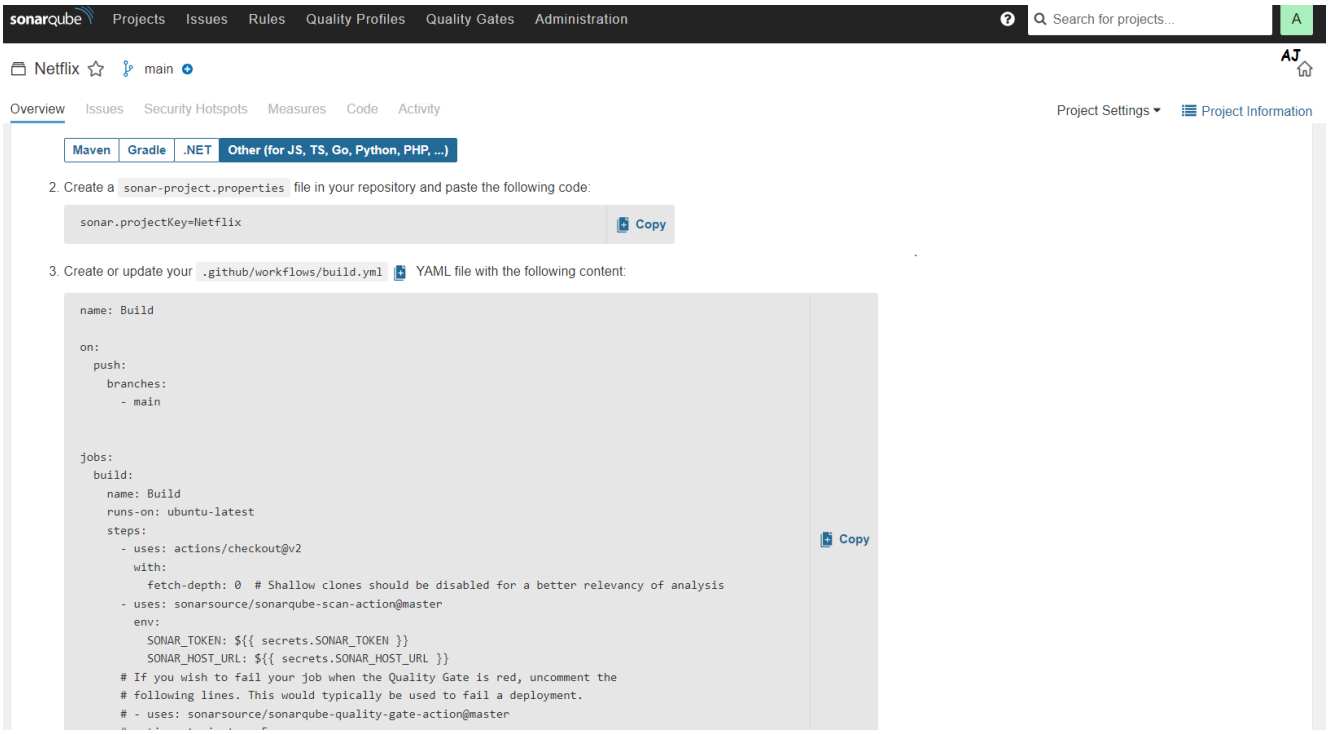


Now create your Workflow for your Project. In my case, the Netflix project is built using React Js. That's why I am selecting Other

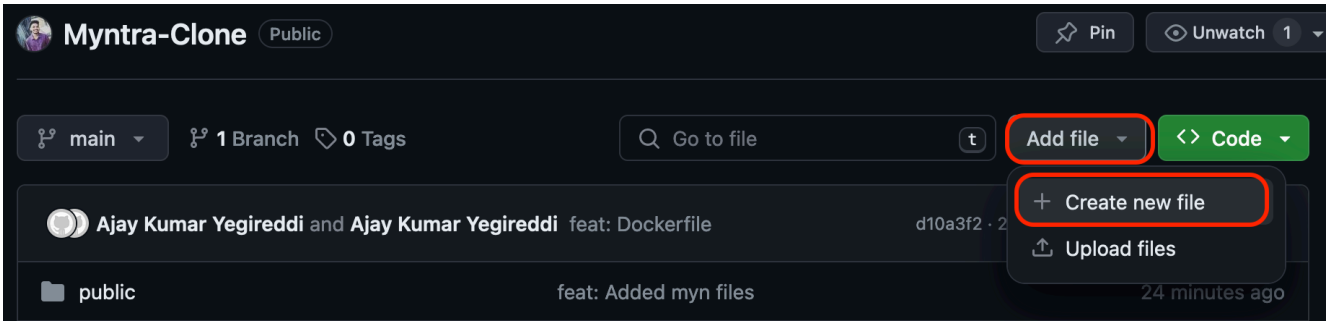


Now it Generates and workflow for my Project

(Use your files for this block please)



Go back to GitHub. click on Add file and then create a new file



Go back to the Sonarqube dashboard and copy the file name and content

2. Create a `sonar-project.properties` file in your repository and paste the following code:

```
sonar.projectKey=myntra
```

Copy

Here file name (in my case only)

```
sonar-project.properties
```

The content to add to the file is (copied from the above image)

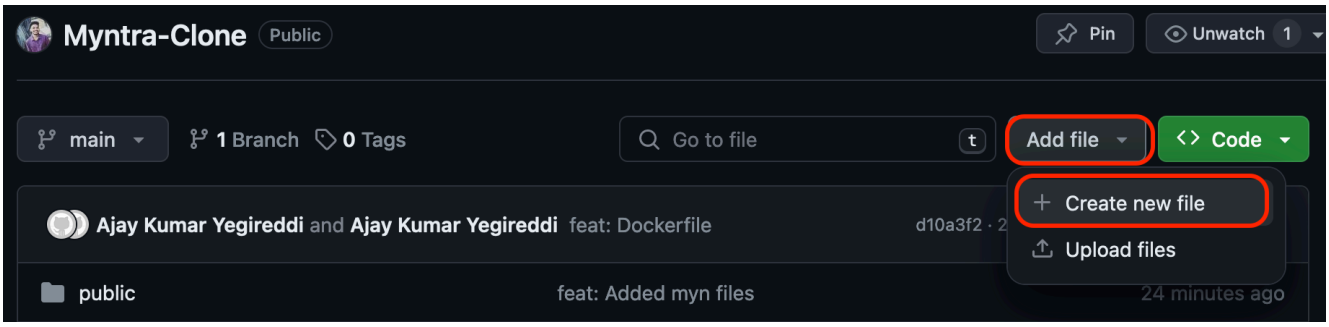
```
sonar.projectKey=myntra
```

Add in GitHub like this (sample images)

Commit changes

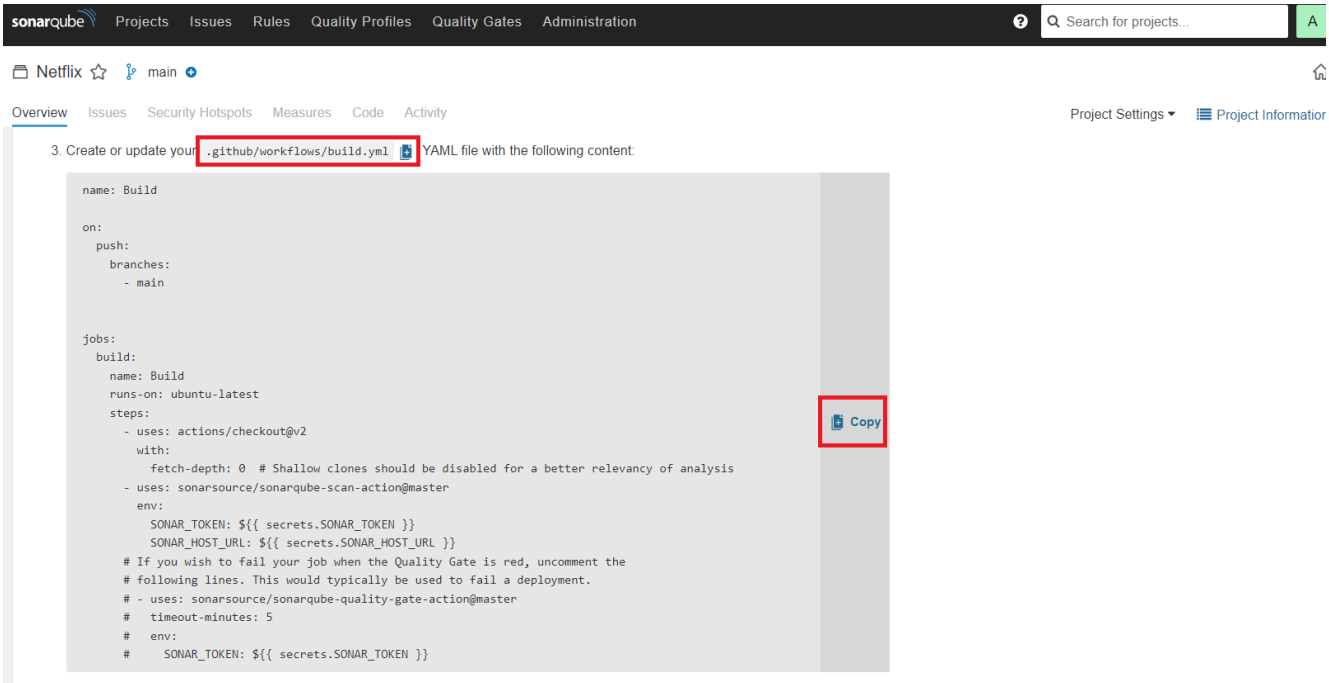
Let's add our workflow

To do that click on Add file and then click on Create a new file



Here is the file name

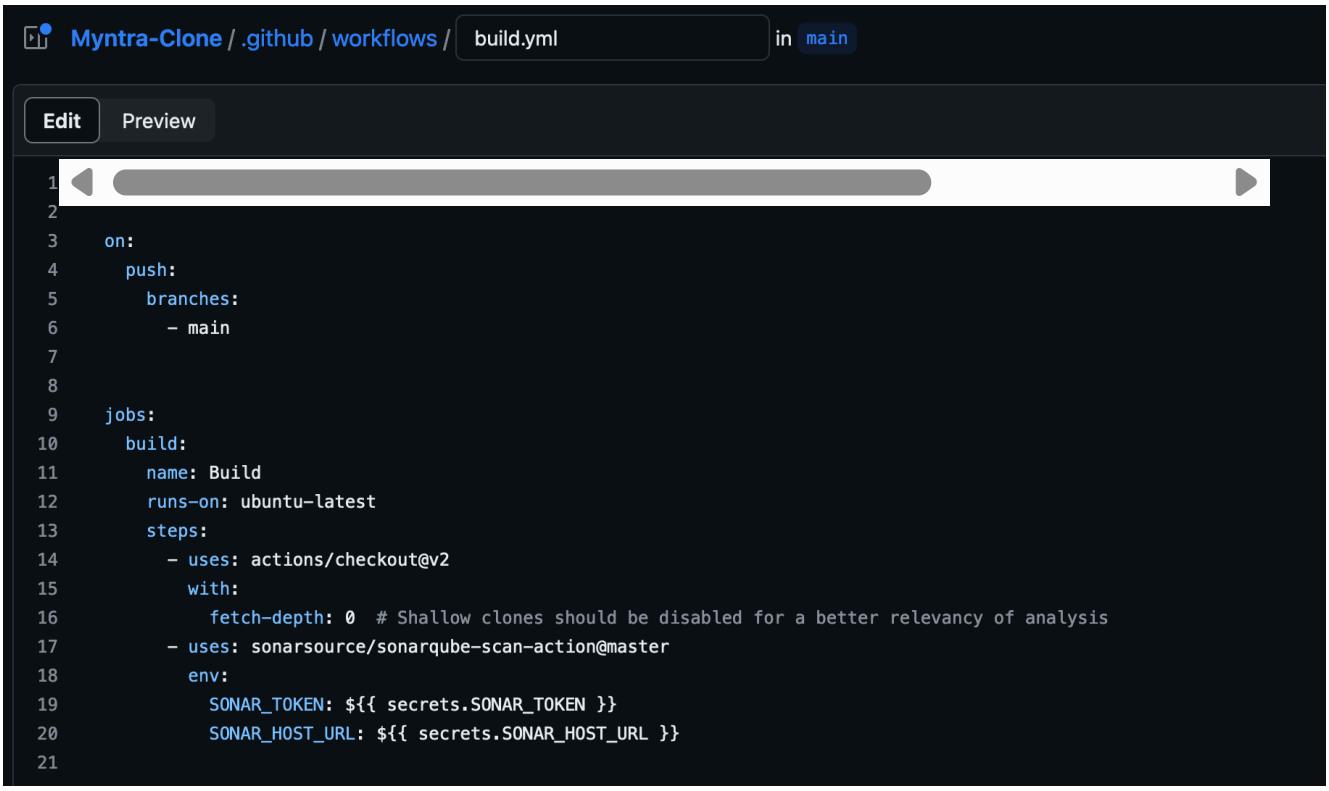
```
.github/workflows/build.yml  #you can use any name iam using sonar.yml
```



Copy content and add it to the file

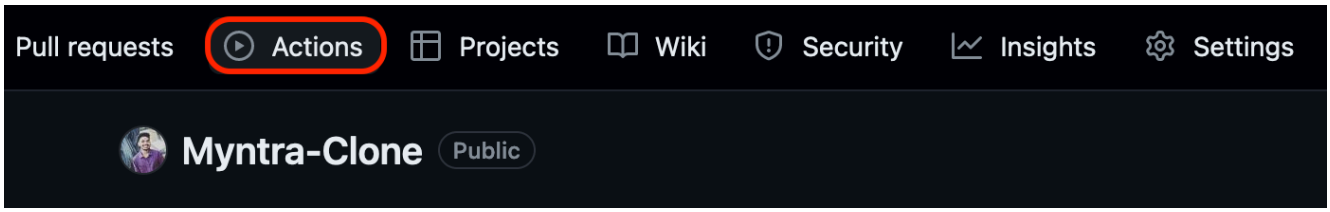
```
name: Build,Analyze,scan

on:
  push:
    branches:
      - main
```

```
jobs:
  build-analyze-scan:
    name: Build
    runs-on: [self-hosted]
    steps:
      - name: Checkout code
        uses: actions/checkout@v2
        with:
          fetch-depth: 0  # Shallow clones should be disabled for a bet

      - name: Build and analyze with SonarQube
        uses: sonarsource/sonarqube-scan-action@master
        env:
          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
          SONAR_HOST_URL: ${{ secrets.SONAR_HOST_URL }}
```

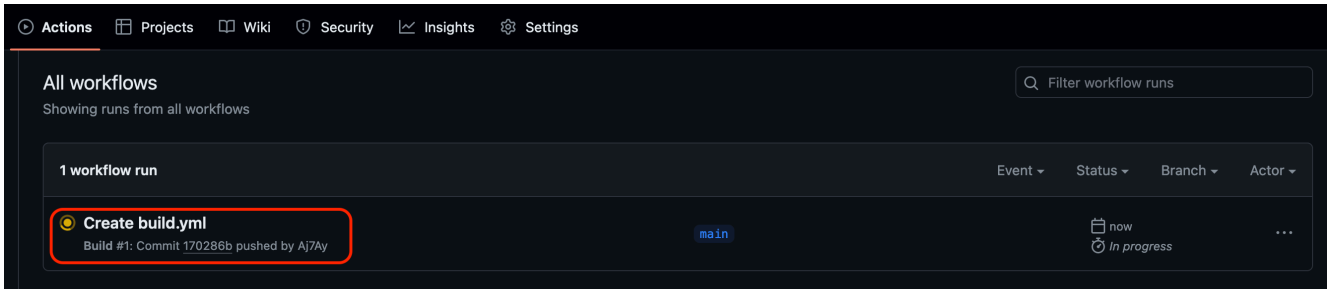


Commit changes

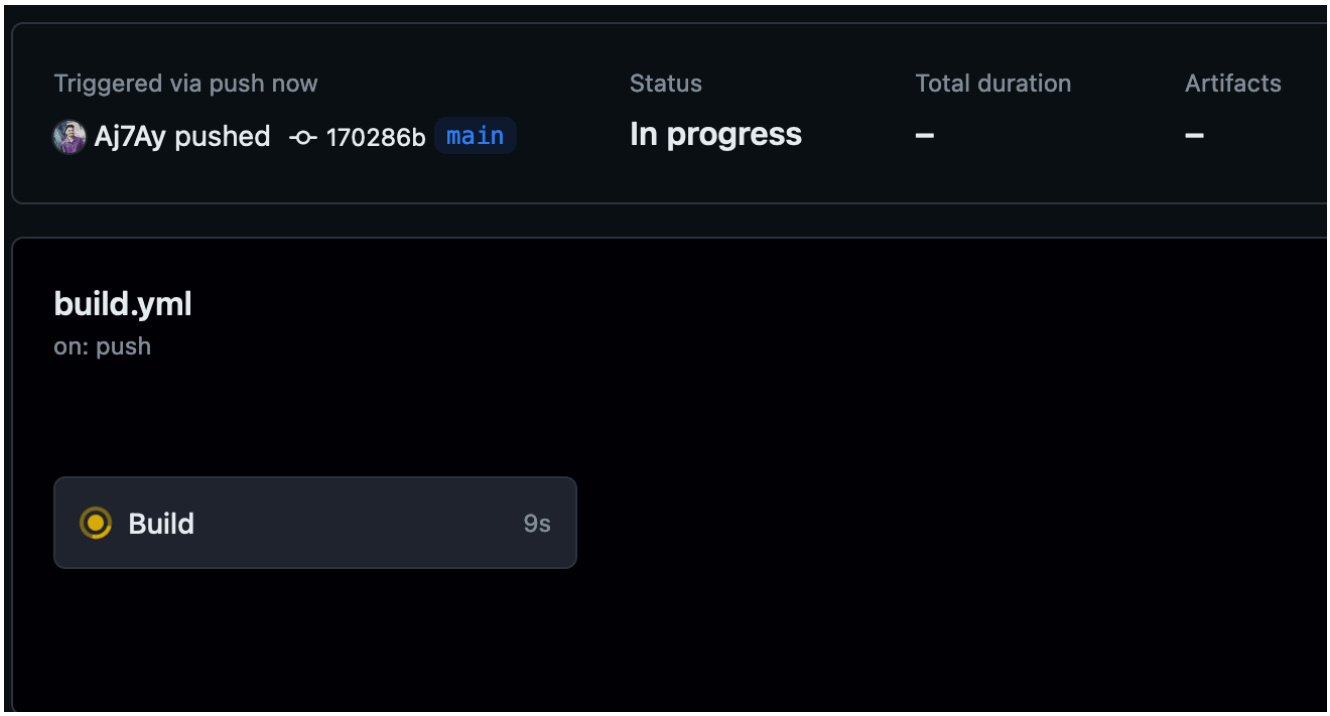
Now click on Actions



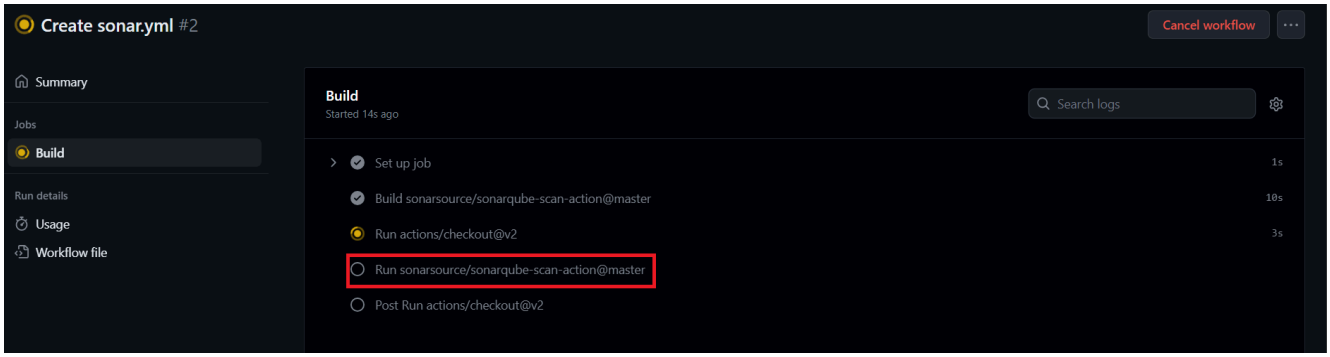
Click on Workflow



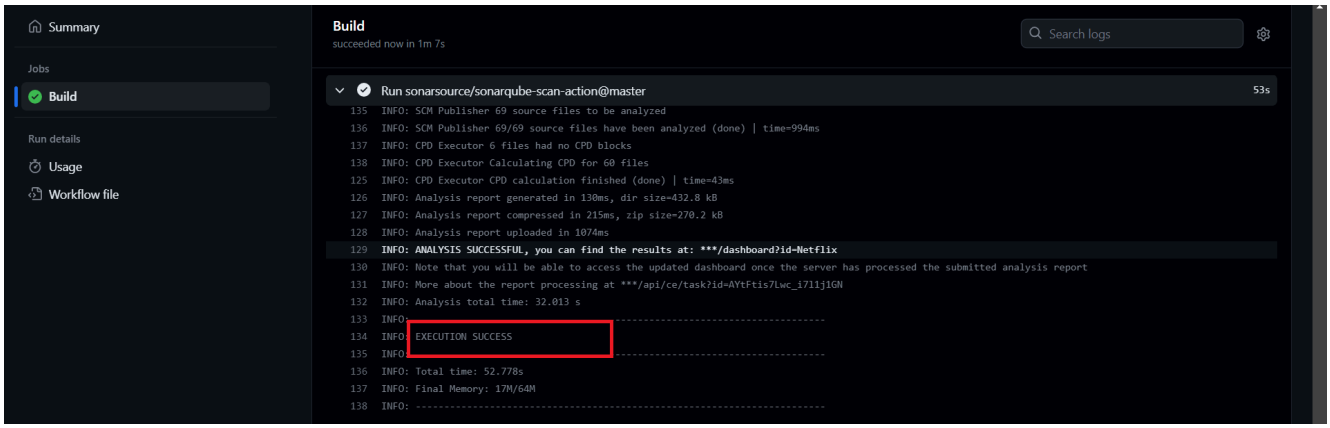
Click on build



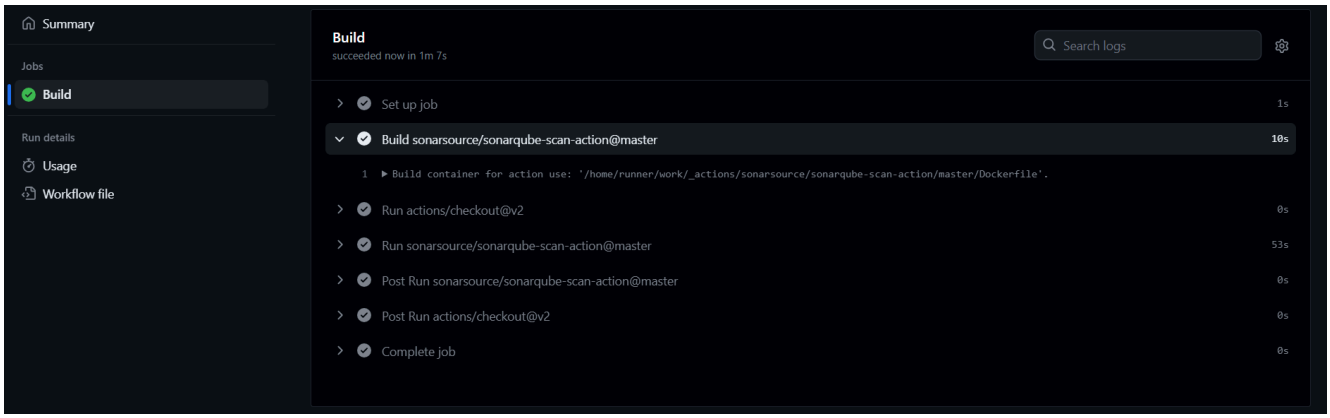
Let's click on Build and see what are the steps involved



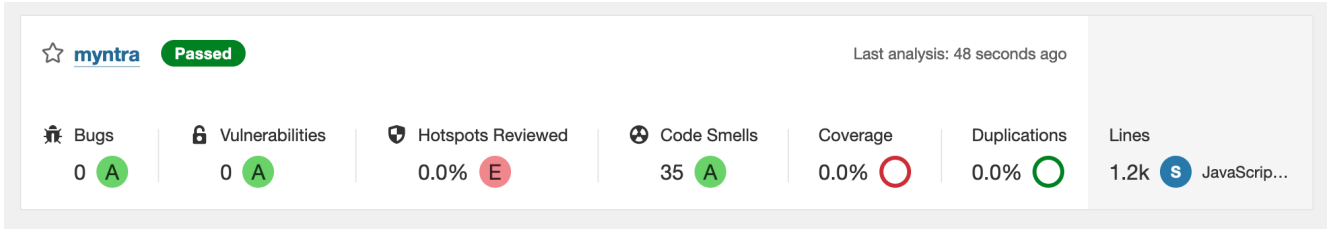
Click on Run Sonarsource and you can do this after the build completion



Build complete.



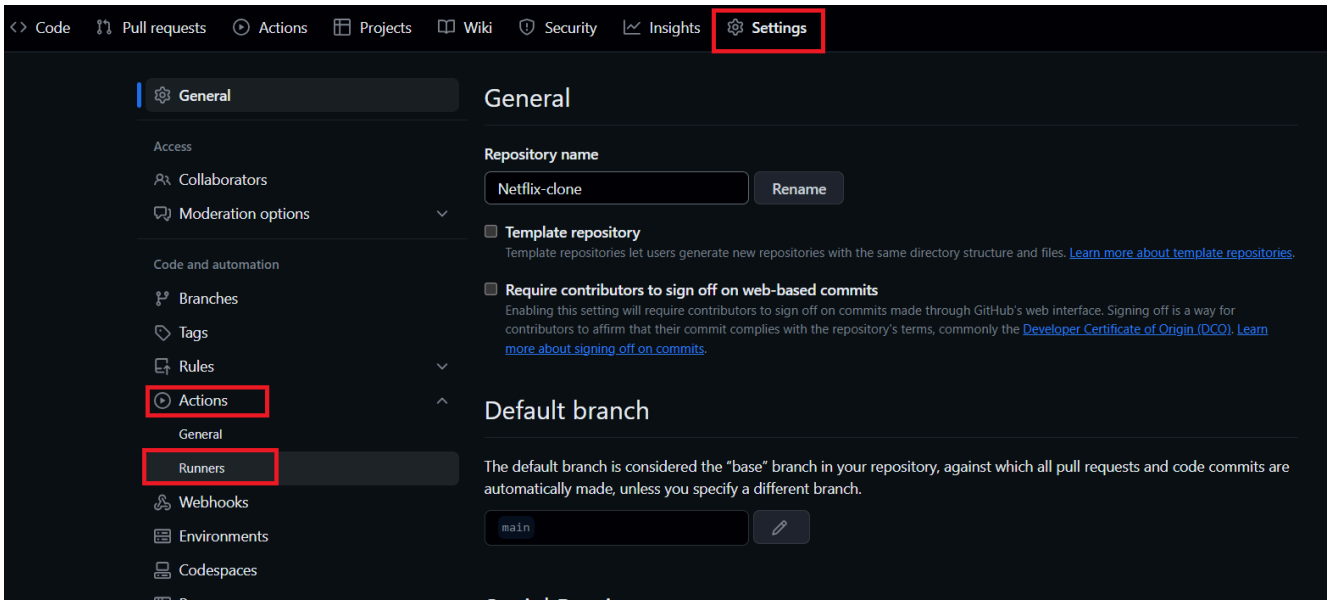
Go to the Sonarqube dashboard and click on projects and you can see the analysis



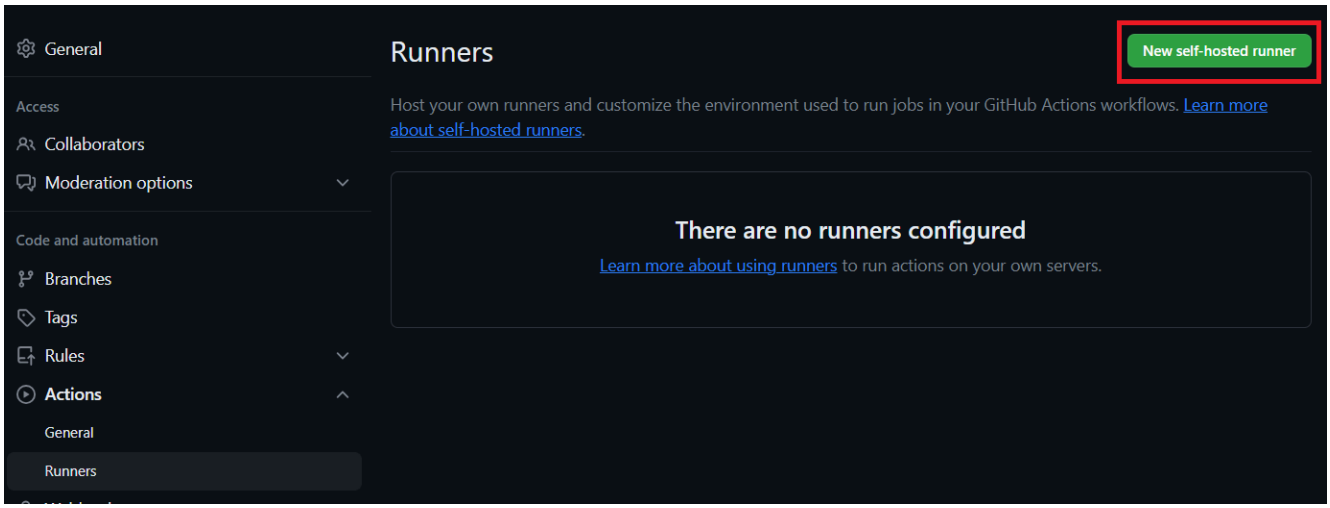
If you want to see the full report, click on issues.

Step 4: Add GitHub Runner

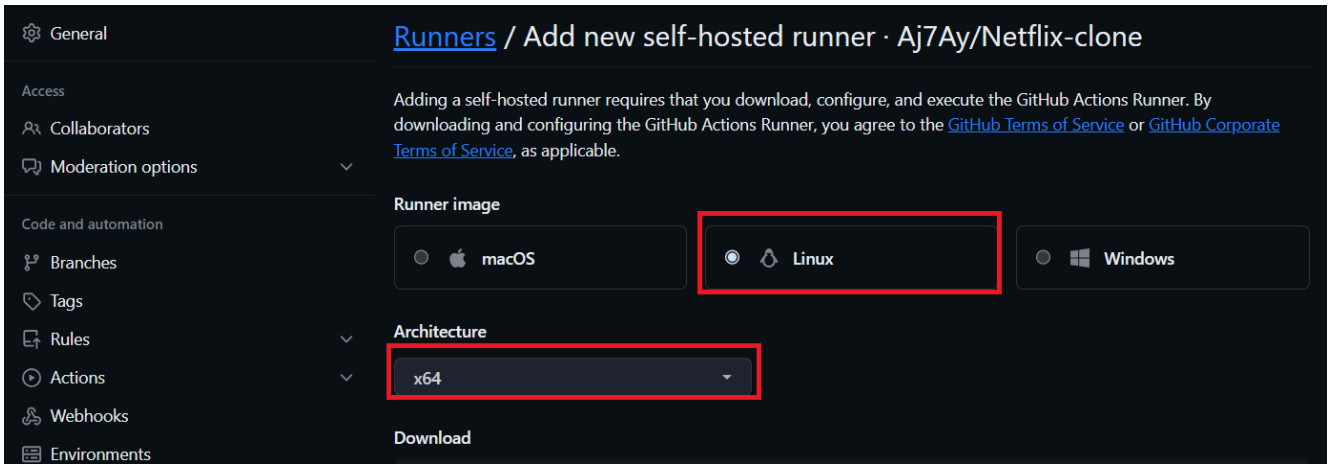
Go to GitHub and click on Settings –> Actions –> Runners



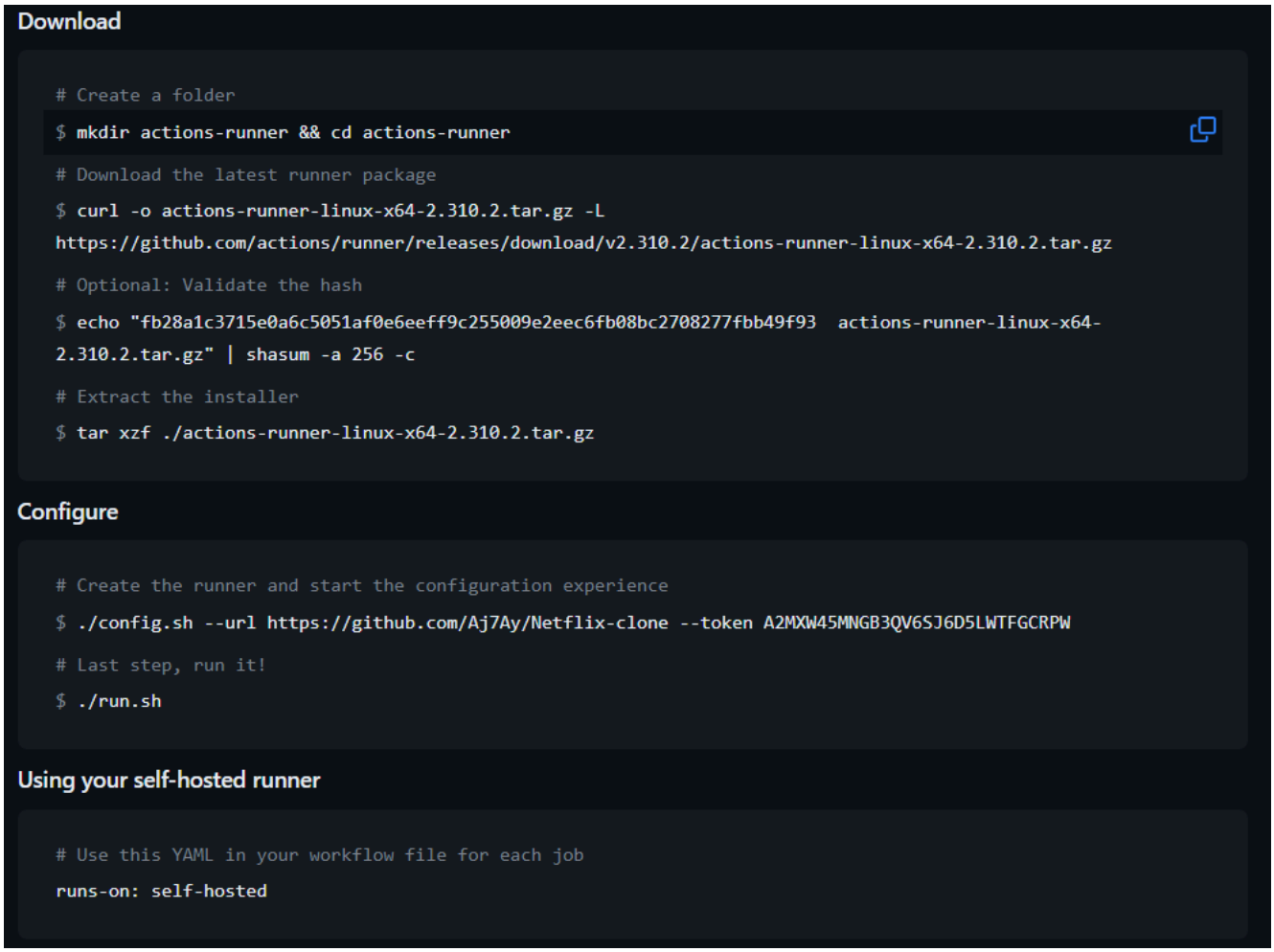
Click on New self-hosted runner



Now select Linux and Architecture X64



Use the below commands to add a self-hosted runner

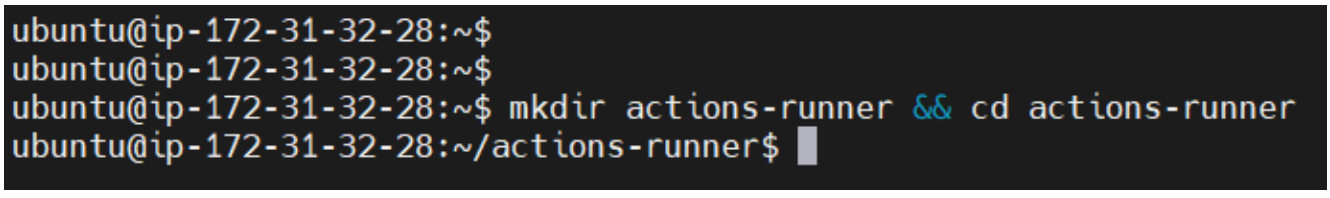


Go to Putty or Mobaxtreme and connect to your ec2 instance

And paste the commands

NOTE: USE YOUR RUNNER COMMANDS (EXAMPLE CASE IAM USING MINE)

```
mkdir actions-runner && cd actions-runner
```



The command “mkdir actions-runner && cd actions-runner” is used to create a new directory called “actions-runner” in the current working directory and then immediately change the current working directory to the newly created “actions-

runner" directory. This allows you to organize your files and perform subsequent actions within the newly created directory without having to navigate to it separately.

```
curl -o actions-runner-linux-x64-2.310.2.tar.gz -L https://github.com/a
```

This command downloads a file called "actions-runner-linux-x64-2.310.2.tar.gz" from a specific web address on GitHub and saves it in your current directory.

Let's validate the hash installation

```
echo "fb28a1c3715e0a6c5051af0e6eeff9c255009e2eec6fb08bc2708277fbb49f93
```

Now Extract the installer

```
tar xzf ./actions-runner-linux-x64-2.310.2.tar.gz
```

```
ubuntu@ip-172-31-32-28:~/actions-runner$
ubuntu@ip-172-31-32-28:~/actions-runner$
ubuntu@ip-172-31-32-28:~/actions-runner$ tar xzf ./actions-runner-linux-x64-2.310.2.tar.gz
ubuntu@ip-172-31-32-28:~/actions-runner$
```

Let's configure the runner

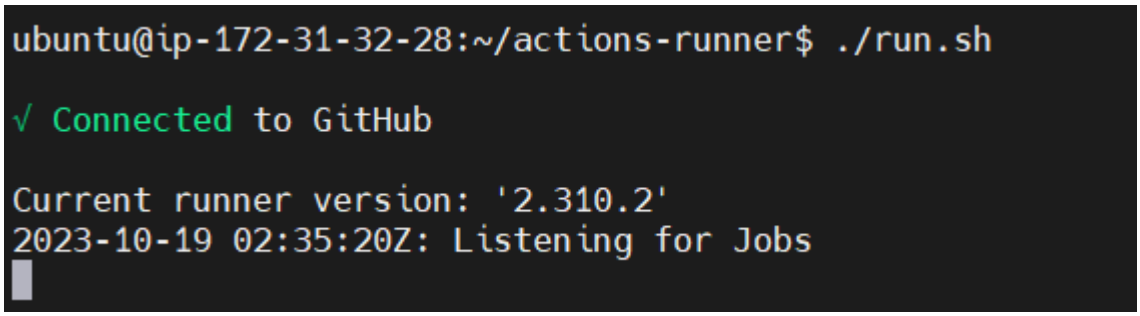
```
./config.sh --url https://github.com/Aj7Ay/Netflix-clone --token A2MXW4
```

```
ubuntu@ip-172-31-32-28:~/actions-runner$
ubuntu@ip-172-31-32-28:~/actions-runner$
ubuntu@ip-172-31-32-28:~/actions-runner$ ./config.sh --url https://github.com/Aj7Ay/Netflix-clone --token A2MXW45MNGB3QV6SJ6D5LWTFGCRPW

--------------------------------------------------------------------------------
|                              AJAY                                            |
|        ____ _ _   _ _   _  _ _   _ _  _                                      |
|       / ___(_) |_| | |_| || |__    / \   ___| |_(_) ___  _ __  ___          |
|      | |  _| | __| |_| | | | '_ \  / _ \ / __| __| |/ _ \| '_ \/ __|         |
|      | |_| | | |_|  _  | |_| |_) |/ ___ \ (__| |_| | (_) | | | \__ \         |
|       \____|_|\__|_| |_|\__,_|_.__/ /_/   \_\___|\__|_|\___/|_| |_|___/      |
|                                                                              |
|                       Self-hosted runner registration                        |
|                                                                              |
--------------------------------------------------------------------------------

# Authentication


√ Connected to GitHub

# Runner Registration

Enter the name of the runner group to add this runner to: [press Enter for Default]  CLICK ENTER

Enter the name of runner: [press Enter for ip-172-31-32-28] aws-netflix   PROVIDE A RUNNER NAME HERE

This runner will have the following labels: 'self-hosted', 'Linux', 'X64'
Enter any additional labels (ex. label-1,label-2): [press Enter to skip] aws-netflix  LABEL NAME

√ Runner successfully added
√ Runner connection is good

# Runner settings

Enter name of work folder: [press Enter for _work]  Enter

√ Settings Saved.

ubuntu@ip-172-31-32-28:~/actions-runner$
```

If you provide multiple labels use commas for each label

Let's start runner

```
./run.sh
```



EKS provision

Clone the repo onto your instance

```
git clone https://github.com/Aj7Ay/Myntra-Clone.git
cd Myntra-Clone
cd EKS-TF
```

This changes the directory to EKS terraform files

Change your S3 bucket in the backend file

Initialize the terraform

```
terraform init
```

```
ubuntu@ip-172-31-36-122:~/TIC-TAC-TOE/Eks-terraform$ terraform init

Initializing the backend...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.23.1...
- Installed hashicorp/aws v5.23.1 (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.
```

Validate the configuration and syntax of files

```
terraform validate
```

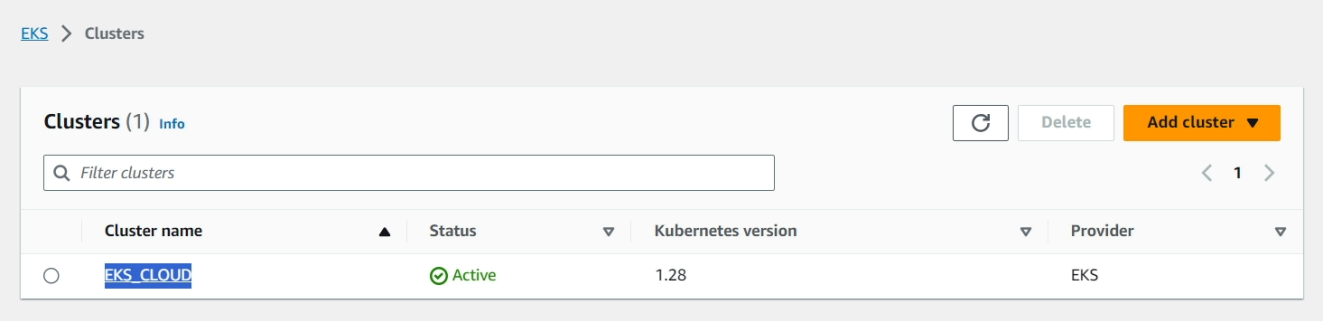
```
ubuntu@ip-172-31-36-122:~/TIC-TAC-TOE/Eks-terraform$
ubuntu@ip-172-31-36-122:~/TIC-TAC-TOE/Eks-terraform$ terraform validate
Success! The configuration is valid.
```

Plan and apply

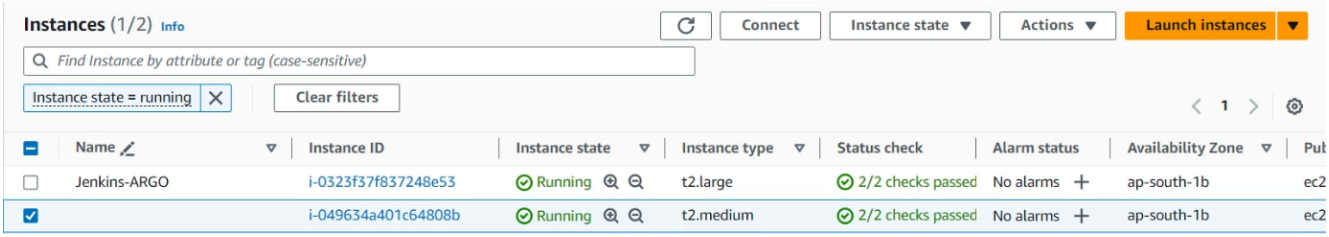
```
terraform plan
terraform apply --auto-approve
```

```
ubuntu@ip-172-31-36-122:~/TIC-TAC-TOE/Eks-terraform$
ubuntu@ip-172-31-36-122:~/TIC-TAC-TOE/Eks-terraform$ terraform apply
data.aws_iam_policy_document.assume_role: Reading...
data.aws_vpc.default: Reading...
data.aws_iam_policy_document.assume_role: Read complete after 0s [id=3552664922]
data.aws_vpc.default: Read complete after 1s [id=vpc-0f6bdd74ced5c07c0]
data.aws_subnets.public: Reading...
data.aws_subnets.public: Read complete after 0s [id=ap-south-1]

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_eks_cluster.example will be created
  + resource "aws_eks_cluster" "example" {
      + arn                   = (known after apply)
      + certificate_authority = (known after apply)
      + cluster_id            = (known after apply)
      + created_at            = (known after apply)
      + endpoint              = (known after apply)
      + id                    = (known after apply)
      + identity              = (known after apply)
      + name                  = "EKS_CLOUD"
      + platform_version      = (known after apply)
      + role_arn              = (known after apply)
      + status                = (known after apply)
      + tags_all              = (known after apply)
      + version               = (known after apply)
```

It will take 10 minutes to create the cluster



Node group ec2 instance



Now add the remaining steps

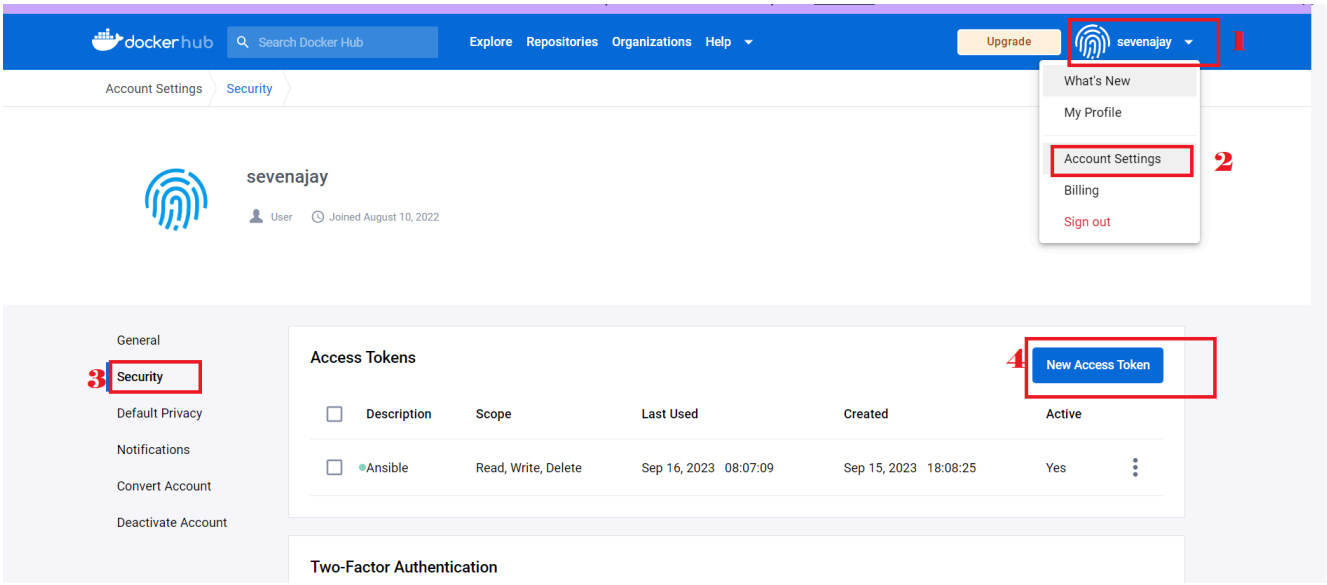
Next, install npm dependencies

```
    - name: NPM Install
      run: npm install # Add your specific npm install command
```

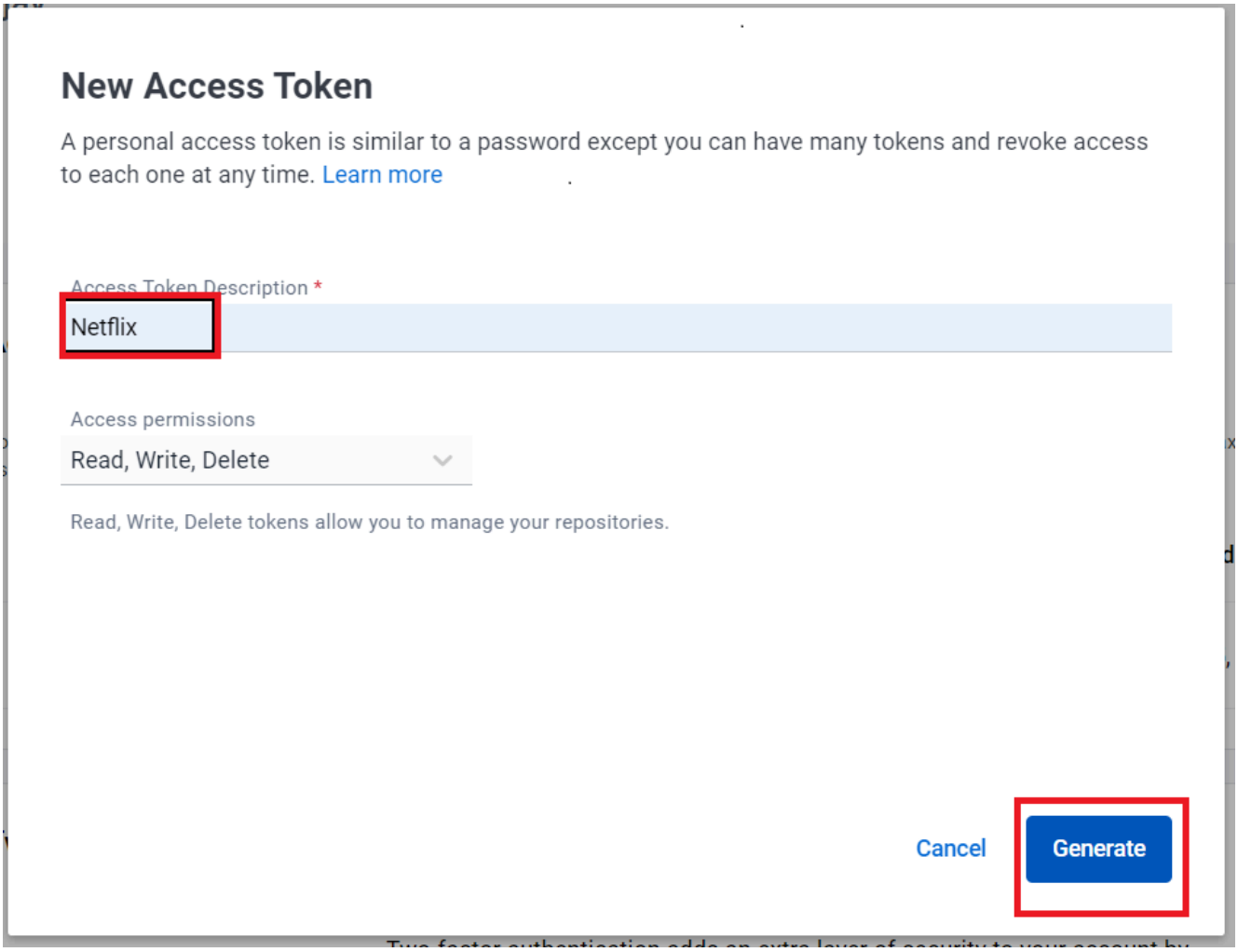
This step runs `npm install` to install Node.js dependencies. You can replace this with your specific npm install command.

Create a Personal Access token for your Dockerhub account

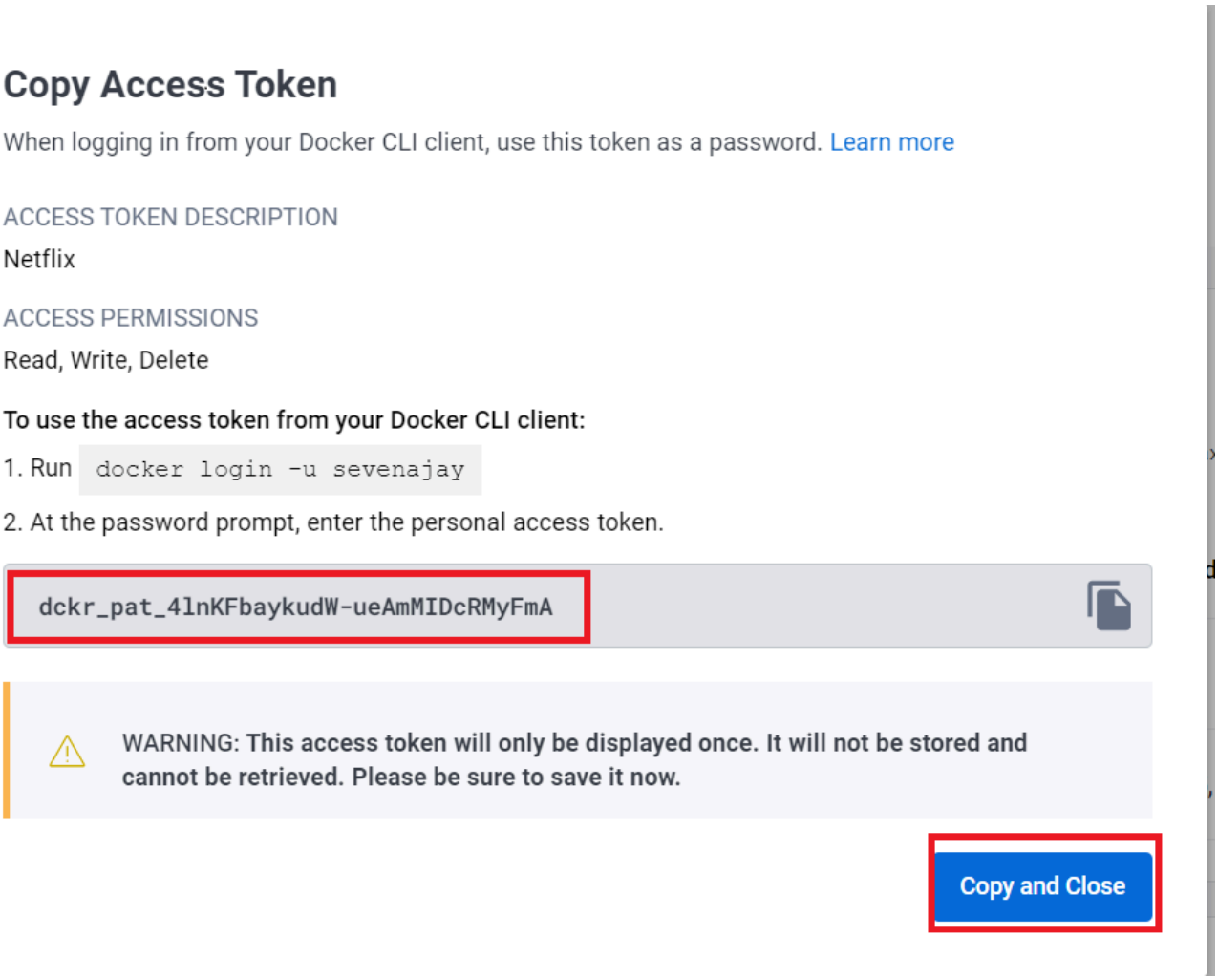
Go to docker hub and click on your profile –> Account settings –> security –> New access token



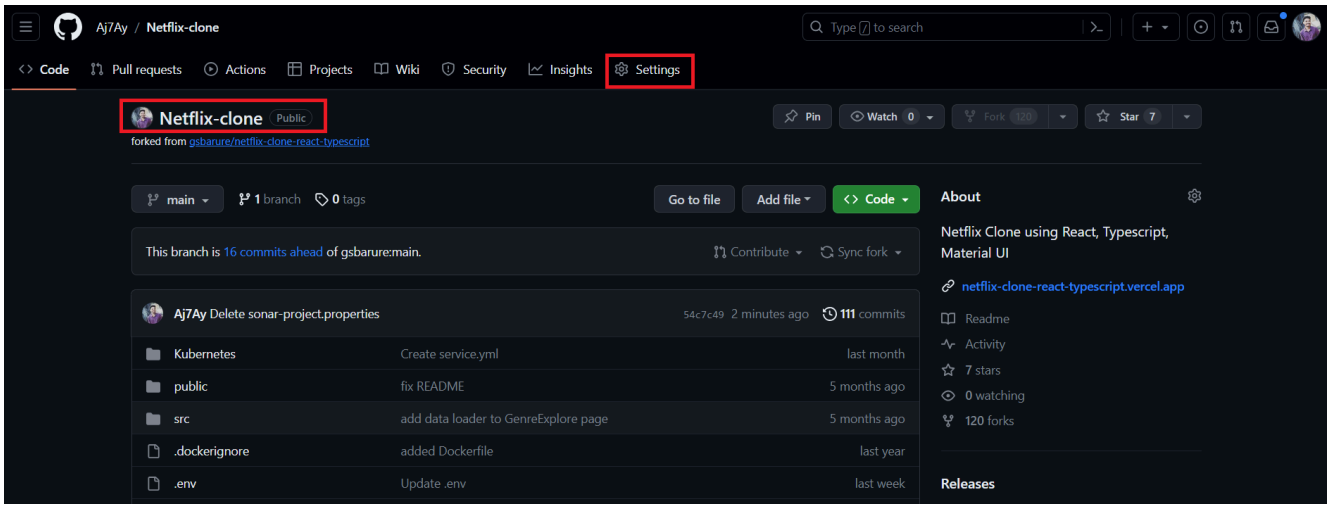
It asks for a name Provide a name and click on generate token



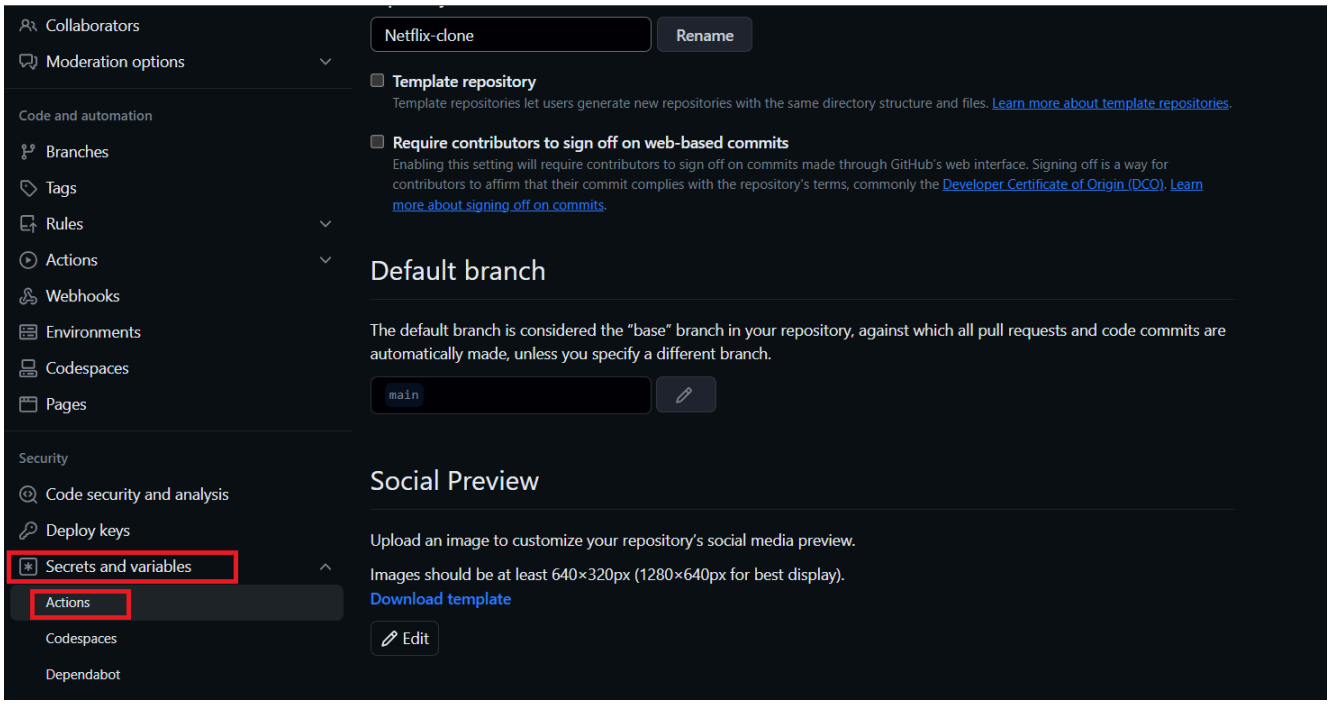
Copy the token save it in a safe place, and close



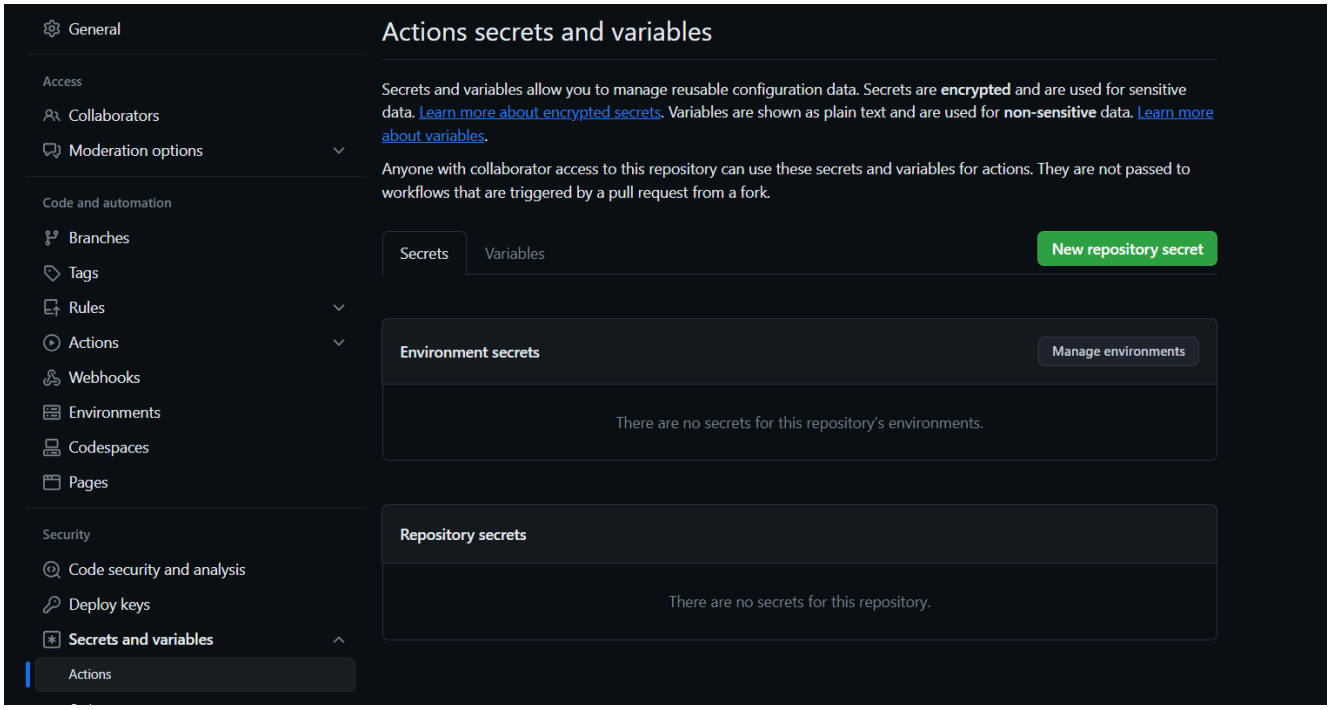
Now Go to GitHub again and click on settings



Search for Secrets and variables and click on and again click on actions

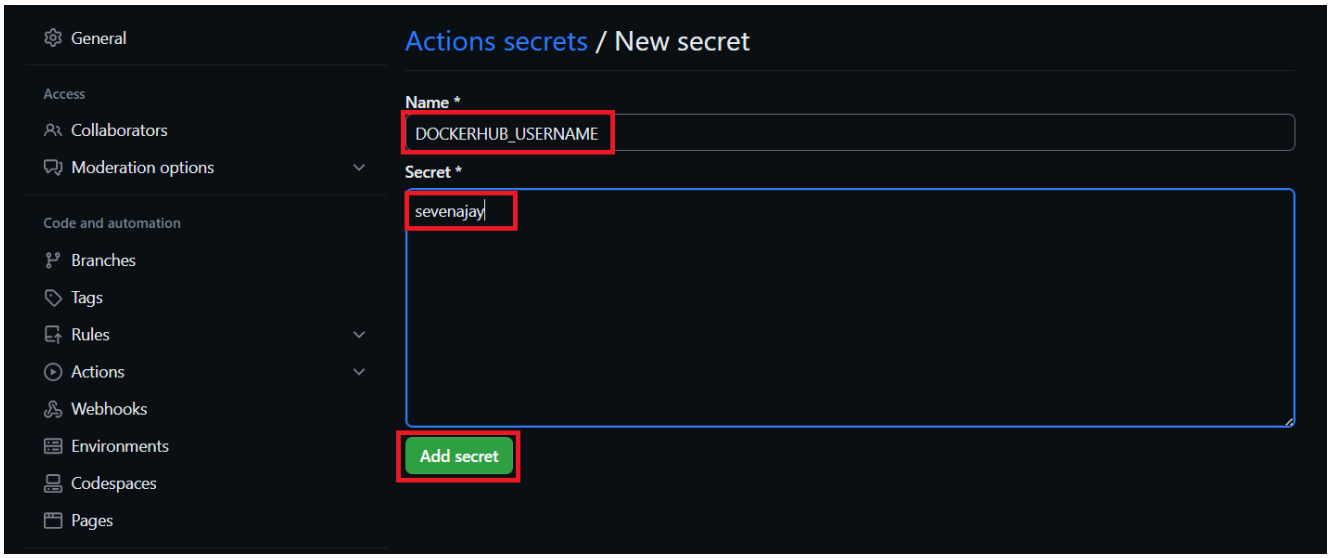


It will open a page like this click on New Repository secret



Add your Dockerhub username with the secret name as

```
DOCKERHUB_USERNAME    #use your dockerhub username
```

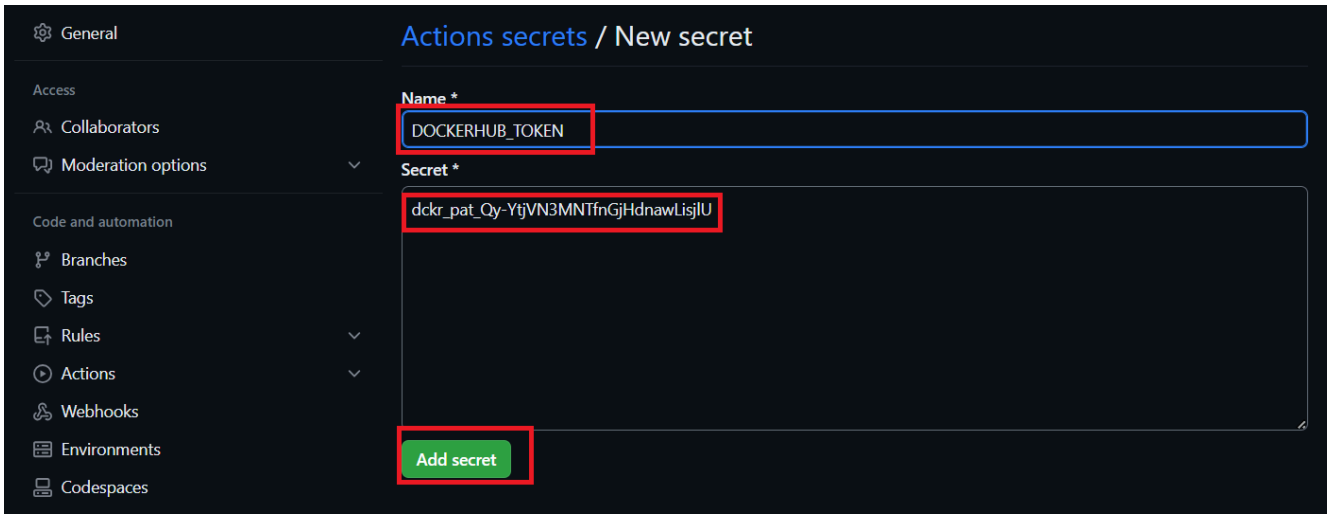


Click on Add Secret.

Let's add our token also and click on the new repository secret again

Name

```
DOCKERHUB_TOKEN
```



Paste the token that you generated and click on Add secret.

```
- name: Docker Login
  run: docker login -u ${{ secrets.DOCKERHUB_USERNAME }} -p ${{ secrets.

- name: Docker Scout Scan
  run: |
    docker-scout quickview fs://.
    docker-scout cves fs://.

- name: Docker build and push
  run: |
    # Run commands to build and push Docker images
    docker build -t myntra .
    docker tag myntra sevenajay/myntra:latest
    docker login -u ${{ secrets.DOCKERHUB_USERNAME }} -p ${{ secrets.DO(
    docker push sevenajay/myntra:latest
   env:
     DOCKER_CLI_ACI: 1

- name: Docker Scout Image Scan
  run: |
    docker-scout quickview sevenajay/myntra:latest
    docker-scout cves sevenajay/myntra:latest
```

1. **Docker Login:**
   - This step is logging into Docker Hub using the provided Docker Hub username and token.
   - The Docker Hub username and token are stored as secrets (`DOCKERHUB_USERNAME` and `DOCKERHUB_TOKEN`), which is a best practice for keeping sensitive information secure.
2. **Docker Scout Scan:**
   - This step is using a tool called `docker-scout` to perform scans on the Docker image.
   - The `quickview` command is used to perform a quick review of the Docker image file system.

- The `cves` command is used to check for Common Vulnerabilities and Exposures (CVEs) in the Docker image.

3. **Docker build and push:**
   - This step involves building a Docker image using the `docker build` command.
   - It then tags the built image with the name "sevenajay/myntra:latest" using the `docker tag` command.
   - Subsequently, it logs into Docker Hub again and pushes the built image to Docker Hub using the `docker push` command.
   - The `env` section sets an environment variable `DOCKER_CLI_ACI` to 1.
4. **Docker Scout Image Scan:**
   - Similar to the second step, this step performs scans on the Docker image that was just built and pushed to Docker Hub.
   - It uses `docker-scout` with the `quickview` and `cves` commands to inspect the image's file system and check for vulnerabilities.

In summary, this workflow automates the process of building, tagging, and pushing a Docker image to Docker Hub. Additionally, it performs security scans on both the local Docker image and the pushed image on Docker Hub using the `docker-scout` tool.

Commit changes after adding the code.

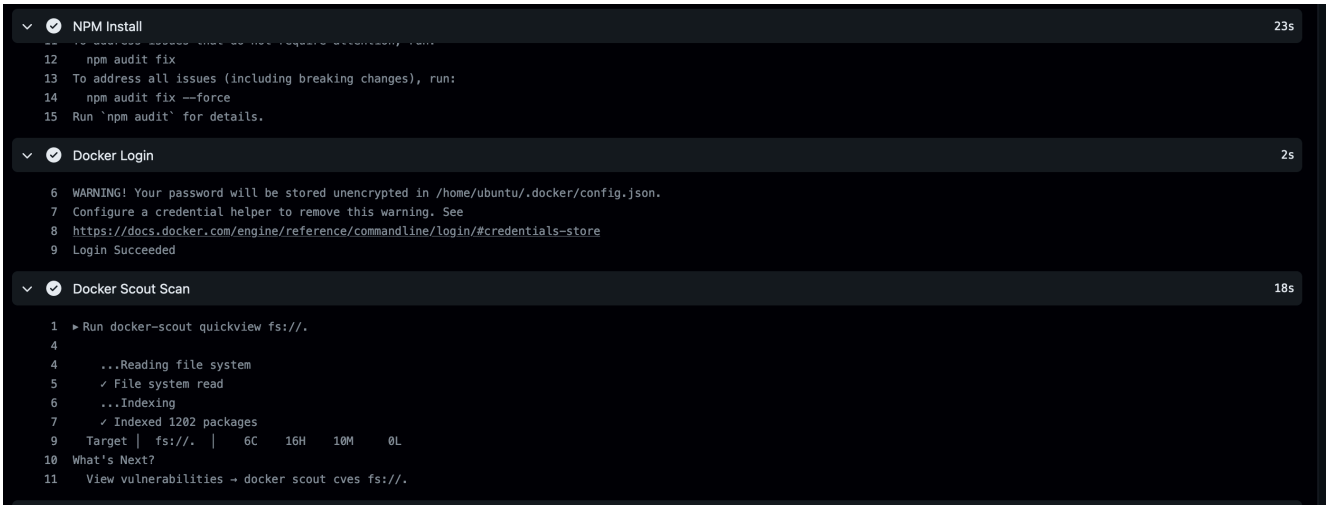
Myntra-Clone / .github / workflows / build.yml in main

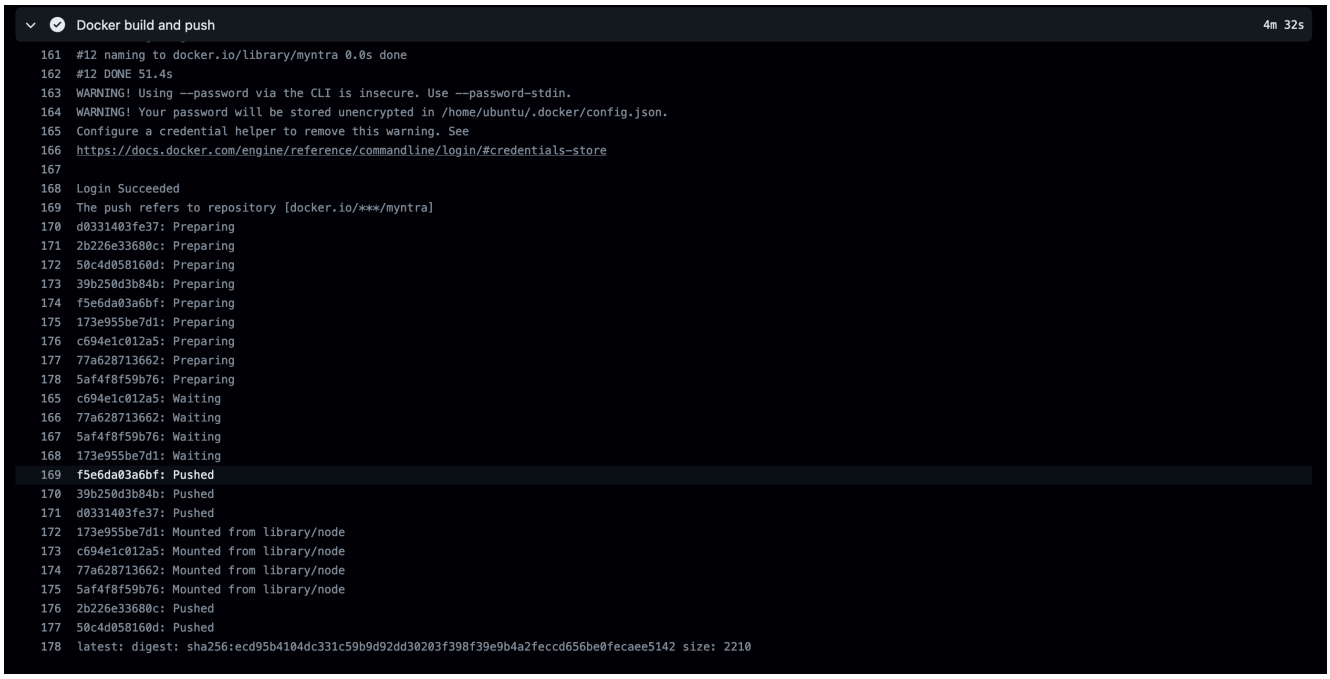
Edit Preview Spaces 2 No wrap

```yaml
jobs:
  build-analyze-scan:
    name: Build, Analyze, and Scan
    runs-on: [self-hosted]
    steps:
      - name: Checkout code
        uses: actions/checkout@v2
        with:
          fetch-depth: 0  # Shallow clones should be disabled for a better relevancy of analysis

      - name: Build and analyze with SonarQube
        uses: sonarsource/sonarqube-scan-action@master
        env:
          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
          SONAR_HOST_URL: ${{ secrets.SONAR_HOST_URL }}

      - name: NPM Install
        run: npm install # Add your specific npm install command

      - name: Docker Login
        run: docker login -u ${{ secrets.DOCKERHUB_USERNAME }} -p ${{ secrets.DOCKERHUB_TOKEN }}

      - name: Docker Scout Scan
        run: |
          docker-scout quickview fs://.
          docker-scout cves fs://.

      - name: Docker build and push
        run: |
          # Run commands to build and push Docker images
          docker build -t myntra .
          docker tag myntra sevenajay/myntra:latest
          docker login -u ${{ secrets.DOCKERHUB_USERNAME }} -p ${{ secrets.DOCKERHUB_TOKEN }}
          docker push sevenajay/myntra:latest
        env:
          DOCKER_CLI_ACI: 1
```

Use Control + Shift + m to toggle the tab key moving focus. Alternatively, use esc then tab to move to the next interactive element on the page.

Use Control + Space or Option + Space to trigger autocomplete in most situations.





Docker Image scan

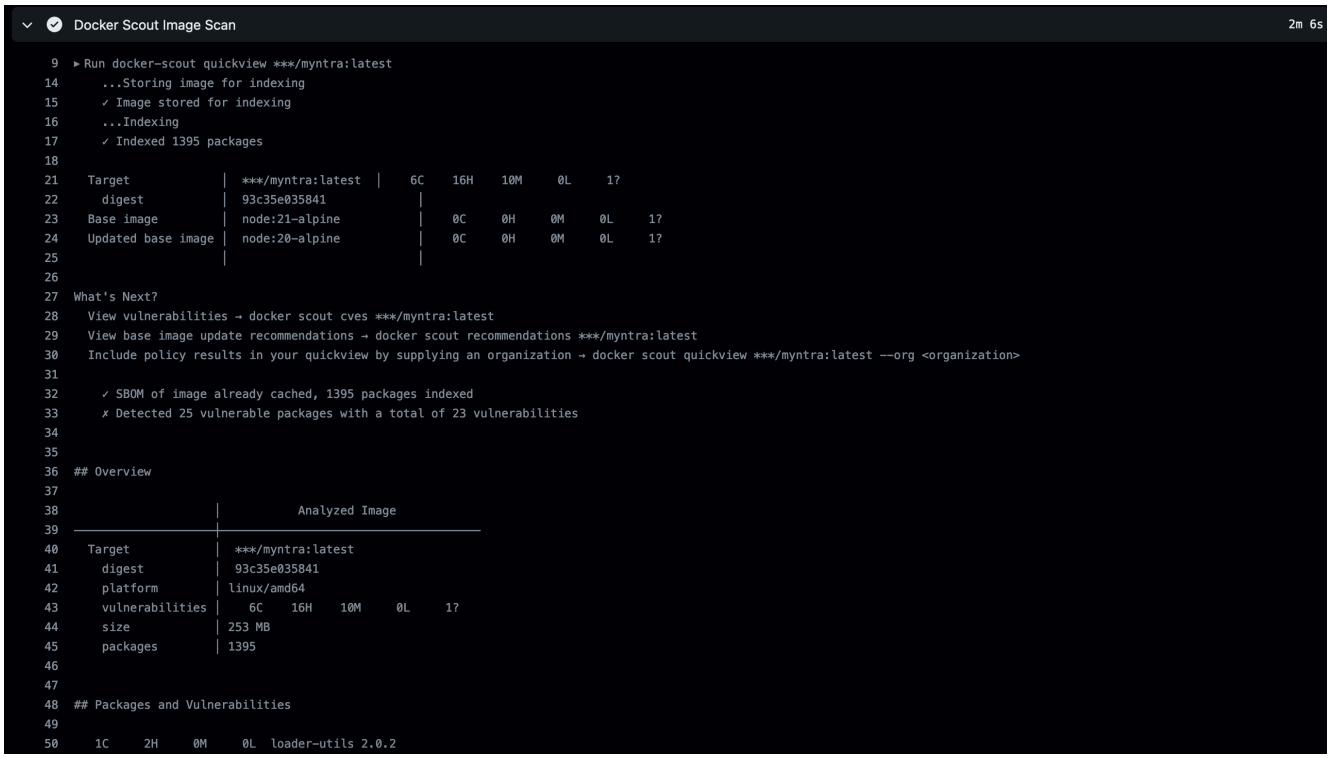
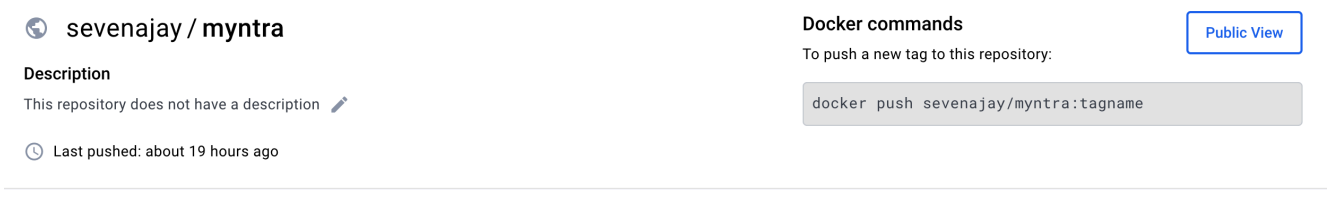


Image is pushed to Dockerhub



DEPLOY

```
deploy:
  needs: build-analyze-scan
  runs-on: self-hosted # Use your self-hosted runner label here
```

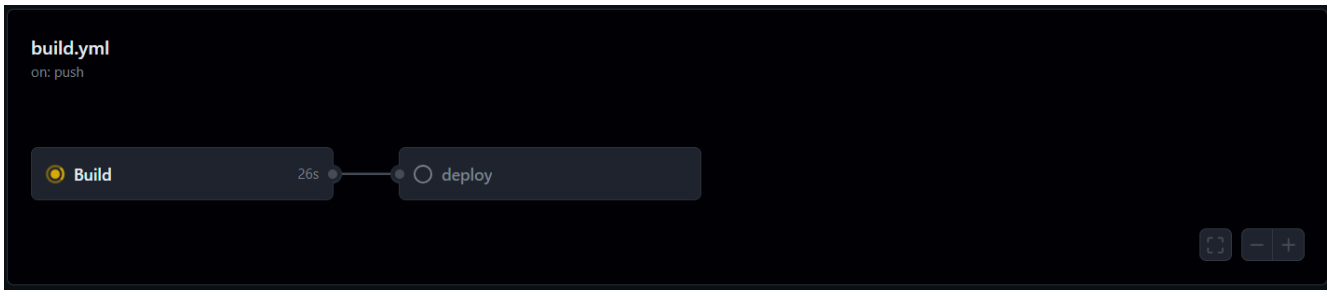
This section defines another job named “deploy.” It specifies that this job depends on the successful completion of the “build-analyze-scan” job. It also runs on a self-hosted runner. You should replace `self-hosted` with the label of your self-hosted runner.

```
- name: Run the container
  run: docker run -d --name myntra -p 3000:3000 sevenajay/myntra:latest
```

This step runs a Docker container named “ticgame” in detached mode (`-d`). It maps port 3000 on the host to port 3000 in the container. It uses the Docker image tagged as `sevenajay/myntra:latest` .

If you run this workflow.

Output



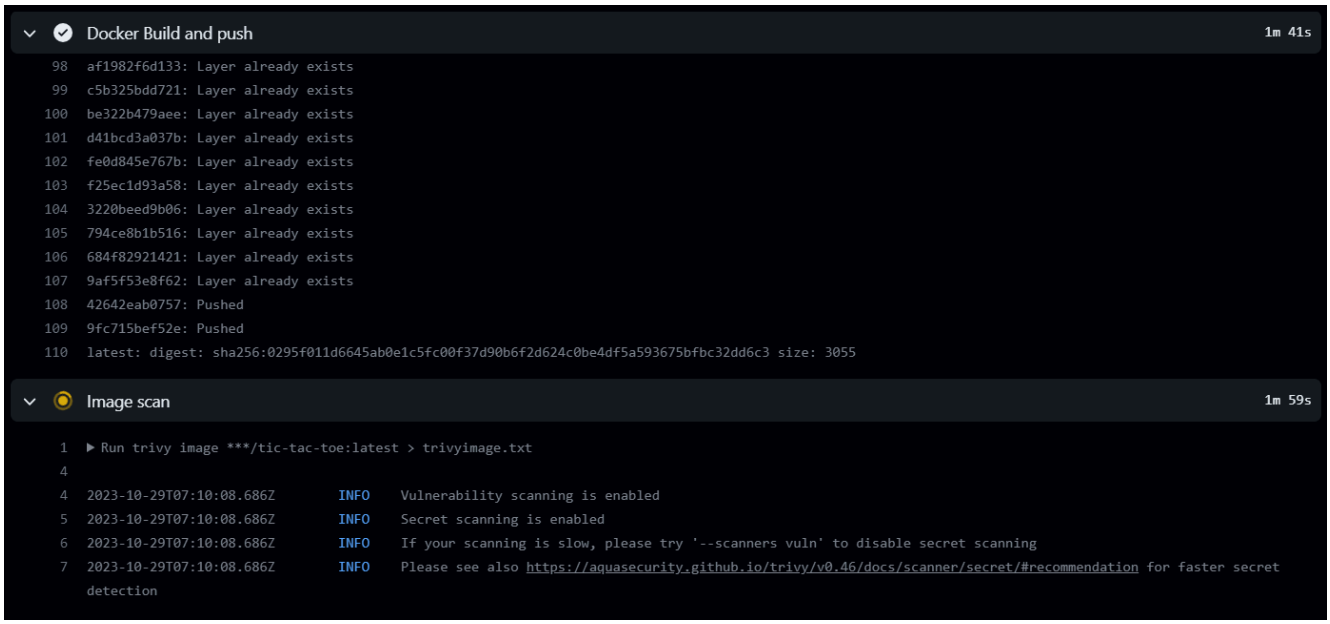
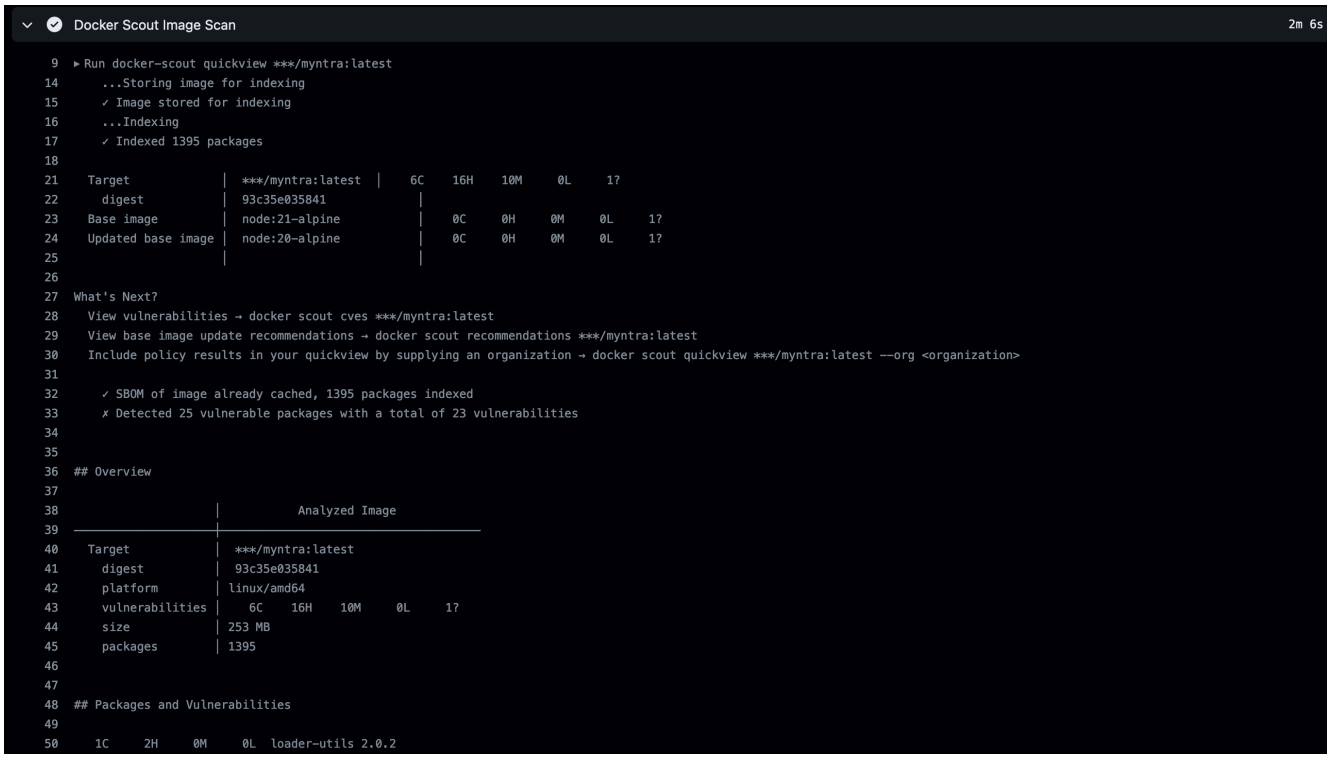


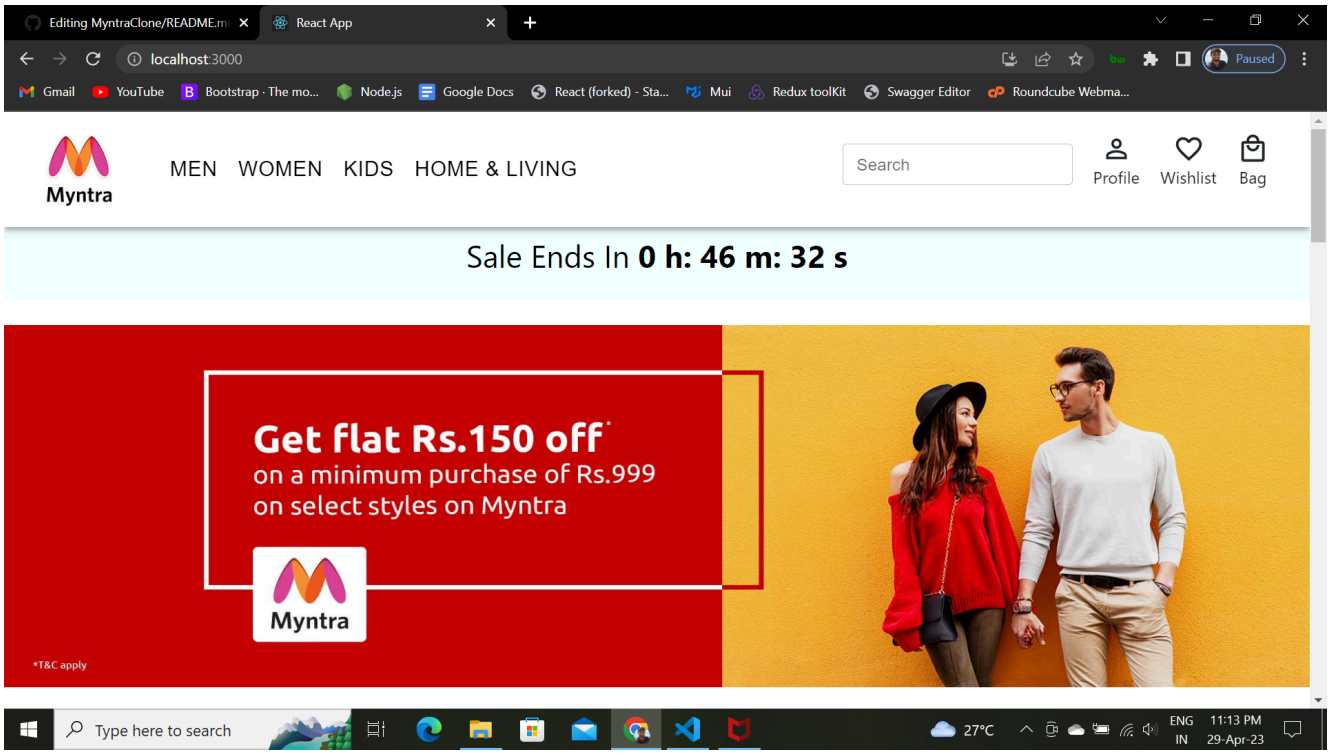
Image scan report



Deployed to the container.

output

```
ec2-ip:3000
```



Deploy to EKS

```
- name: Update kubeconfig
  run: aws eks --region cluster-region update-kubeconfig --name cluster
```

This step updates the kubeconfig to configure `kubectl` to work with an Amazon EKS cluster in the region with the name of your cluster.

```
- name: Deploy to EKS
  run: kubectl apply -f deployment-service.yml
```

This step deploys Kubernetes resources defined in the `deployment-service.yml` file to the Amazon EKS cluster using `kubectl apply`.

Complete Workflow

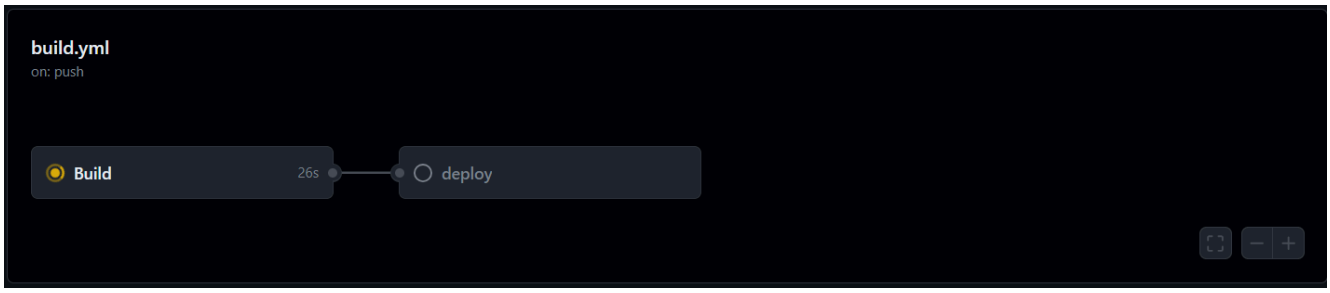
```
name: Build,Analyze,scan

on:
  push:
    branches:
      - main


jobs:
  build-analyze-scan:
    name: Build
    runs-on: [self-hosted]
    steps:
      - name: Checkout code
        uses: actions/checkout@v2
        with:
          fetch-depth: 0  # Shallow clones should be disabled for a beti
```

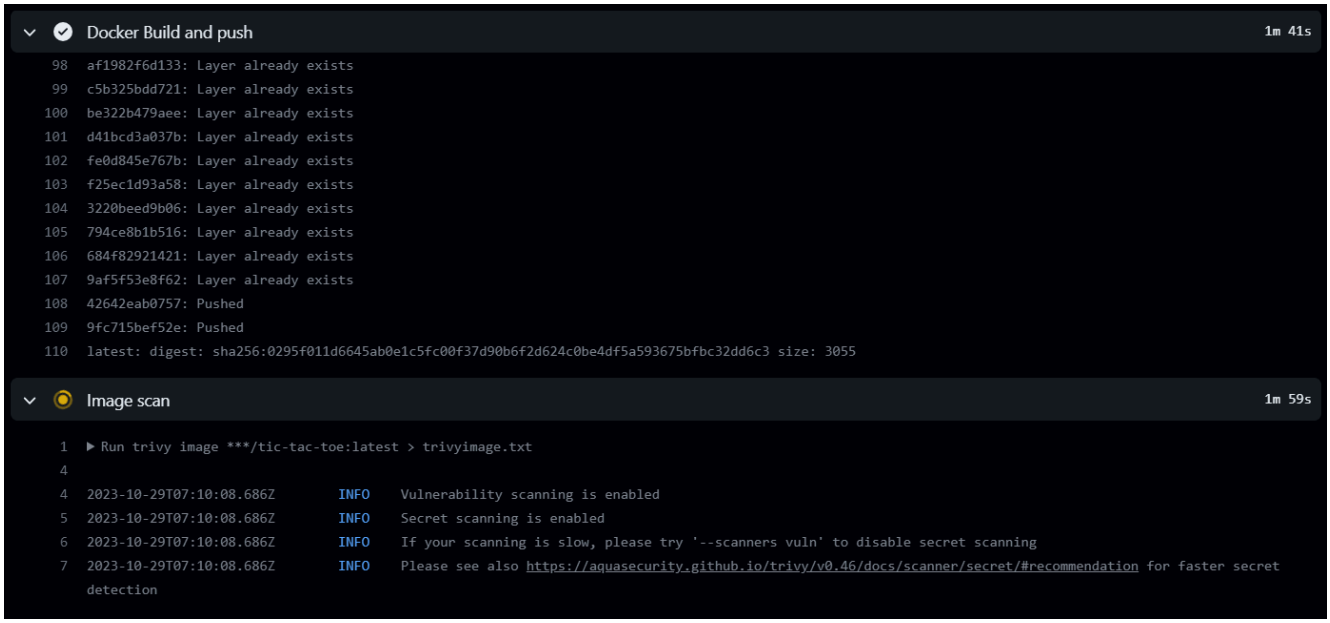
```
      - name: Build and analyze with SonarQube
        uses: sonarsource/sonarqube-scan-action@master
        env:
          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
          SONAR_HOST_URL: ${{ secrets.SONAR_HOST_URL }}

      - name: npm install dependency
        run: npm install

      - name: Docker Login
        run: docker login -u ${{ secrets.DOCKERHUB_USERNAME }} -p ${{ se

      - name: Docker Scout Scan
        run: |
          docker-scout quickview fs://.
          docker-scout cves fs://.

      - name: Docker build and push
        run: |
          # Run commands to build and push Docker images
          docker build -t myntra .
          docker tag myntra sevenajay/myntra:latest
          docker login -u ${{ secrets.DOCKERHUB_USERNAME }} -p ${{ secre
          docker push sevenajay/myntra:latest
        env:
          DOCKER_CLI_ACI: 1

      - name: Docker Scout Image Scan
        run: |
          docker-scout quickview sevenajay/myntra:latest
          docker-scout cves sevenajay/myntra:latest

  deploy:
   needs: build-analyze-scan
   runs-on: [self-hosted]
   steps:
      - name: docker pull image
        run: docker pull sevenajay/myntra:latest

      - name: Deploy to container
        run: docker run -d --name game -p 3000:3000 sevenajay/myntra:lat

      - name: Update kubeconfig
        run: aws eks --region ap-south-1 update-kubeconfig --name EKS_CI
```

```
      - name: Deploy to kubernetes
        run: kubectl apply -f deployment-service.yml
```

commit changes

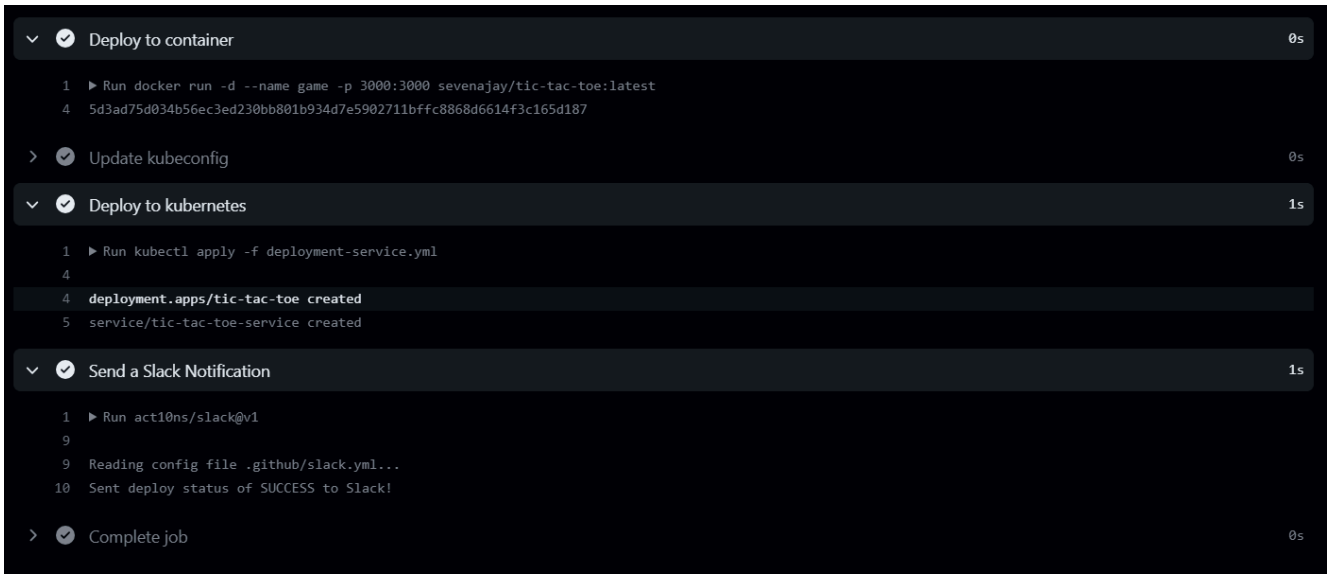
Run this workflow now

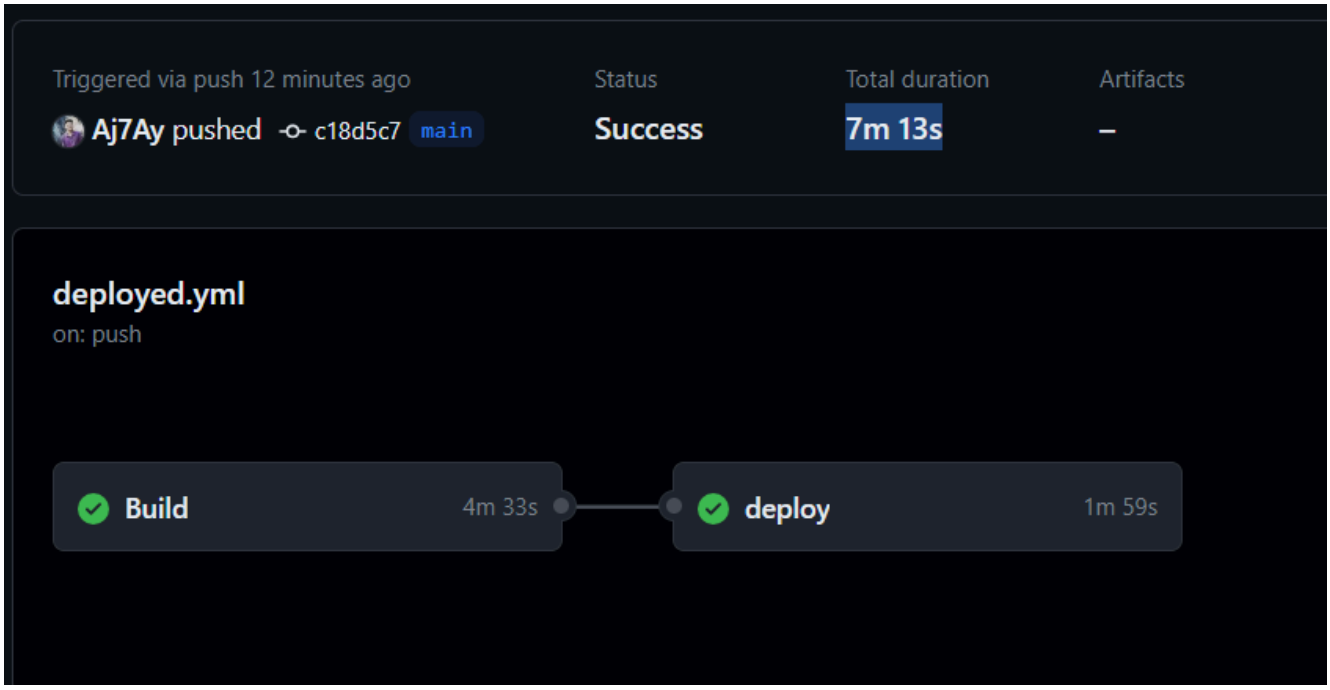




Deployed to the container.

Deployed to EKS



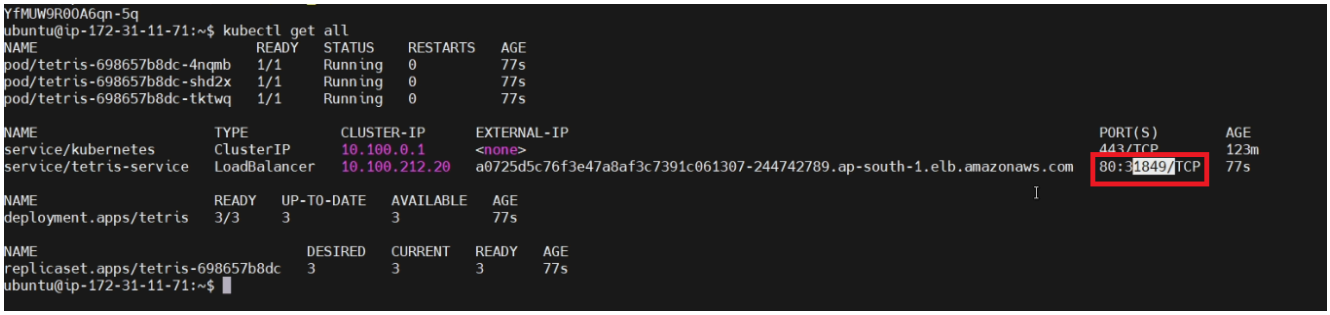


Job completed.

Let’s go to the Ec2 ssh connection

Provide this command

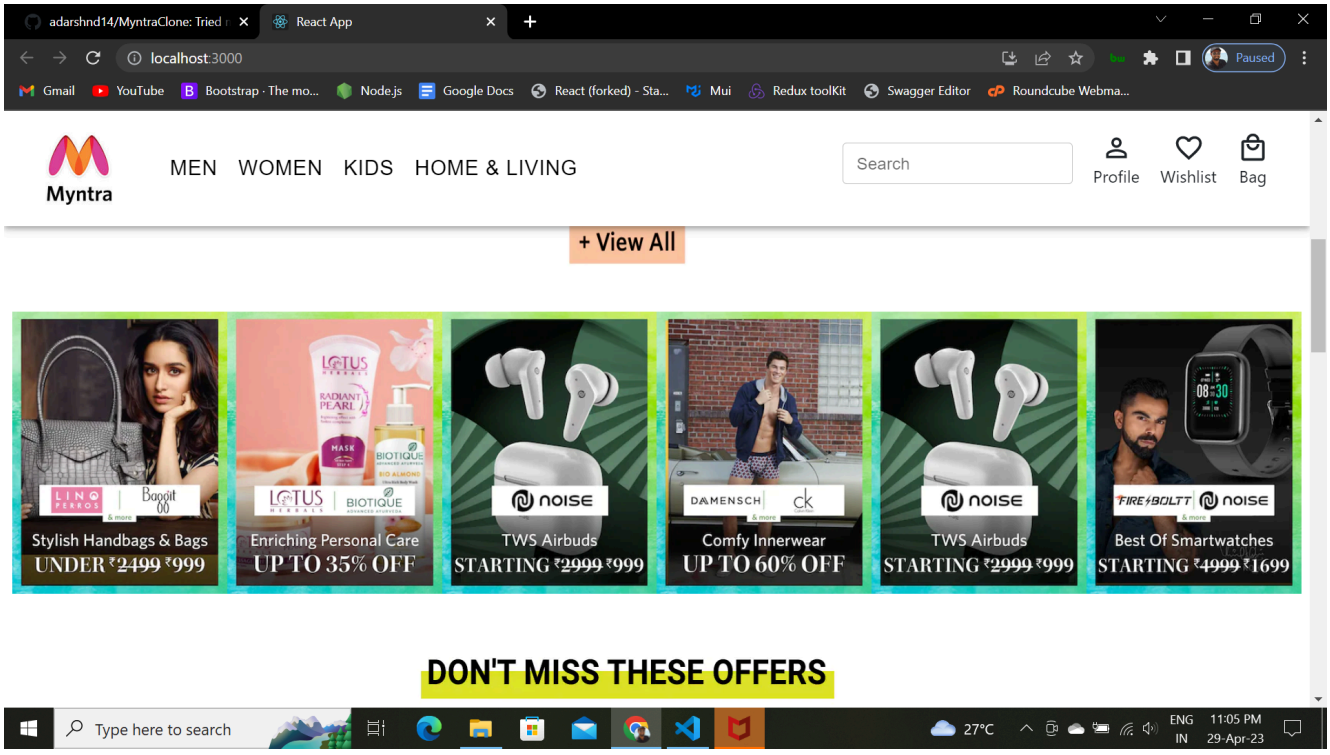
```
kubectl get all
```



Open the port in the security group for the Node group instance.

After that copy the external IP and paste it into the browser

output



Destruction workflow

```
name: Build,Analyze,scan
```

```
on:
  push:
    branches:
      - main


jobs:
  build-analyze-scan:
    name: Build
    runs-on: [self-hosted]
    steps:
      - name: Checkout code
        uses: actions/checkout@v2
        with:
          fetch-depth: 0  # Shallow clones should be disabled for a beti

      - name: Deploy to container
        run: |
          docker stop myntra
          docker rm myntra

      - name: Update kubeconfig
        run: aws eks --region ap-south-1 update-kubeconfig --name EKS_CI

      - name: Deploy to kubernetes
        run: kubectl delete -f deployment-service.yml
```

It will delete the container and delete the Kubernetes deployment.

Stop the self-hosted runner.

Now go inside the Myntra-clone

To delete the Eks cluster

```
cd /home/ubuntu
cd Myntra-clone
```

```
cd EKS-TF
terraform destroy --auto-approve
```

It w

Meanwhile, delete the Dockerhub Token

Once cluster destroys

Delete The ec2 instance and IAM role.

Delete the secrets from GitHub also.

Ajay Kumar Yegireddi is a DevSecOps Engineer and System Administrator, with a passion for sharing real-world DevSecOps projects and tasks. **Mr. Cloud Book**, provides hands-on tutorials and practical insights to help others master DevSecOps tools and workflows. Content is designed to bridge the gap between development, security, and operations, making complex concepts easy to understand for both beginners and professionals.

Comments

2 responses to “GitHub Actions Scouting Myntra App | DevSecOps”

Naveen
13 March 2024

Is it a multi-microservice based project or just a front end?

Reply

mrcloudbook.com
17 March 2024

Just FE

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

I'm not a robot
reCAPTCHA
Privacy - Terms

Post Comment

Uncategorized

How to Automate Incident Response : How Q Developer Helped Me Automate a Daily Pain Point

22 July 2025

AI

How to Run Docker Model Runner on Ubuntu 24.04

11 July 2025

AI, DevOps

How to Install docker-ai on Ubuntu 24.04

15 June 2025

Upskill with Ajay: DevSecOps Mastery

Join Mr Cloud book to master DevSecOps through real-world projects. Learn CI/CD, security integration, automation, and more, gaining hands-on skills for industry-level challenges.

Important Links

Privacy Policy

Terms & Conditions

Contact

Resources

Blog

YouTube Channel