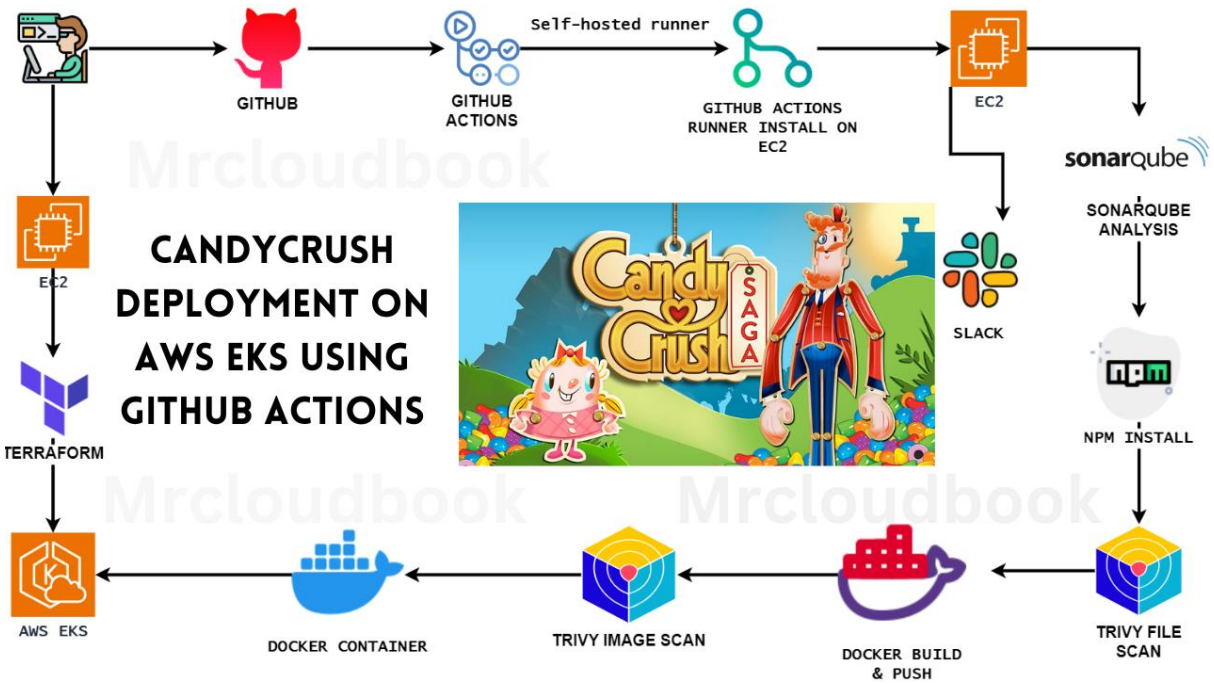
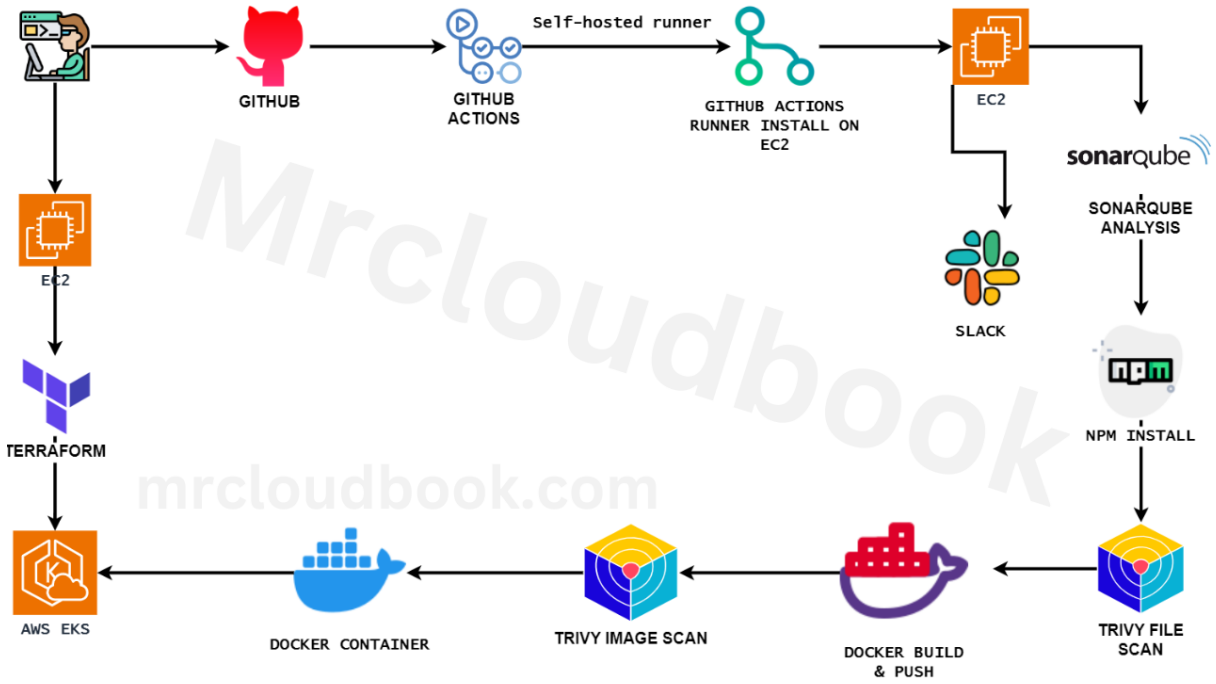
Candycrush Deployment on AWS EKS using GitHub Actions in DevSecOps Pipeline



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/Candycrush.git

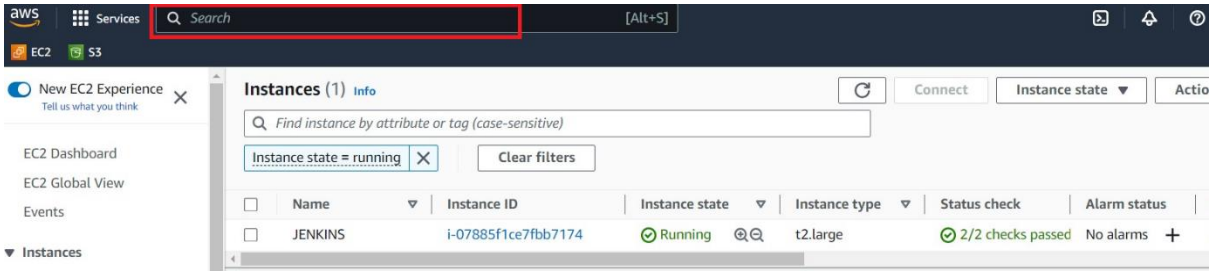
Step1A: Launch an Ec2 Instance

To launch an AWS EC2 instance with Ubuntu 22.04 using the AWS Management Console, sign in to your AWS account, access the EC2 dashboard, and click "Launch Instances." In "Step 1," select "Ubuntu 22.04" as the AMI, and in "Step 2," choose "t2.medium" as the instance type. Configure the instance details, storage, tags, and security group settings according to your requirements. Review the settings, create or select a key pair for secure access, and launch the instance. Once launched, you can connect to it via SSH using the associated key pair.

Create an IAM ROLE

Navigate to **AWS CONSOLE**

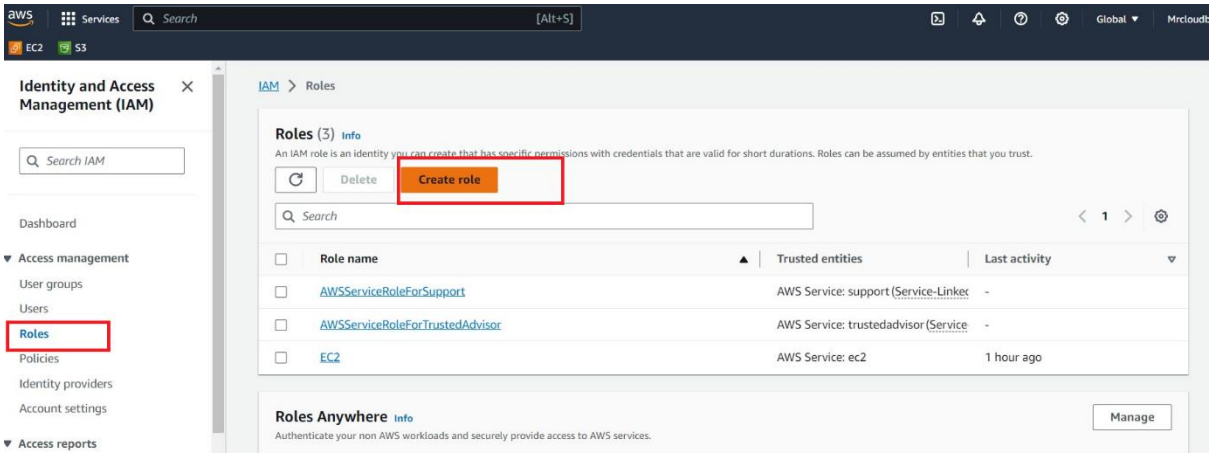
Click the "Search" field.



Type "IAM **enter**"

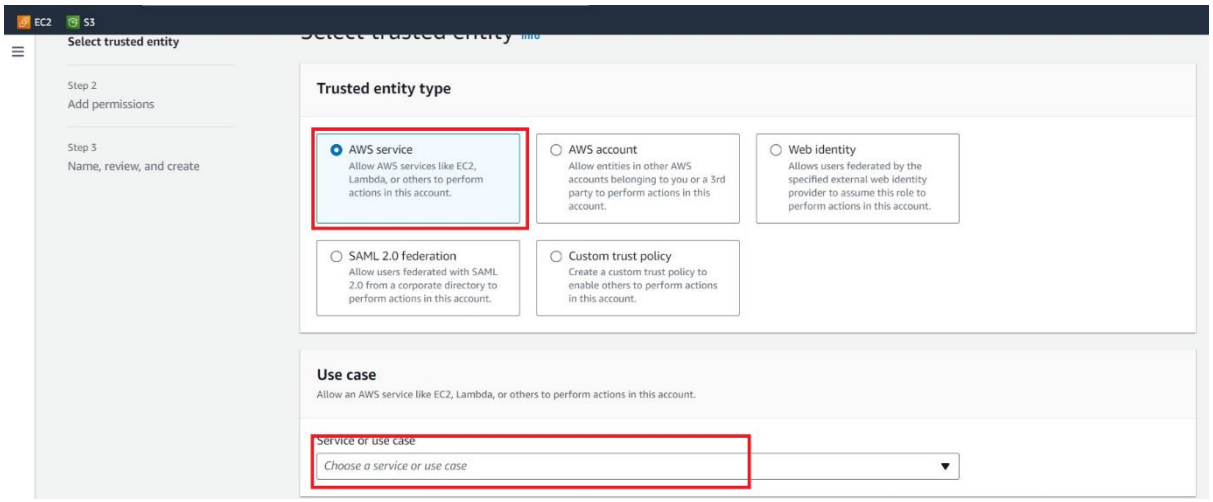
Click "Roles"

Click "Create role"



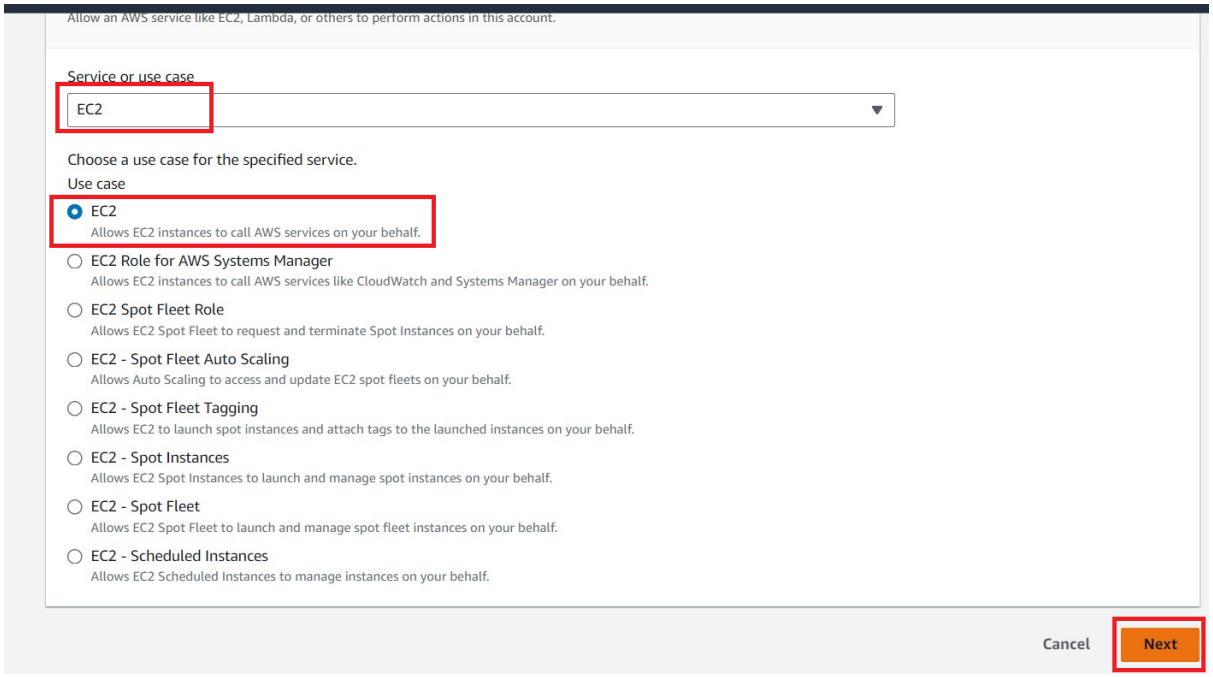
Click “AWS service”

Click “Choose a service or use case”



Click “EC2”

Click “Next”



Click the “Search” field.

Add permissions policies

Administrator Access (or) EC2 full access

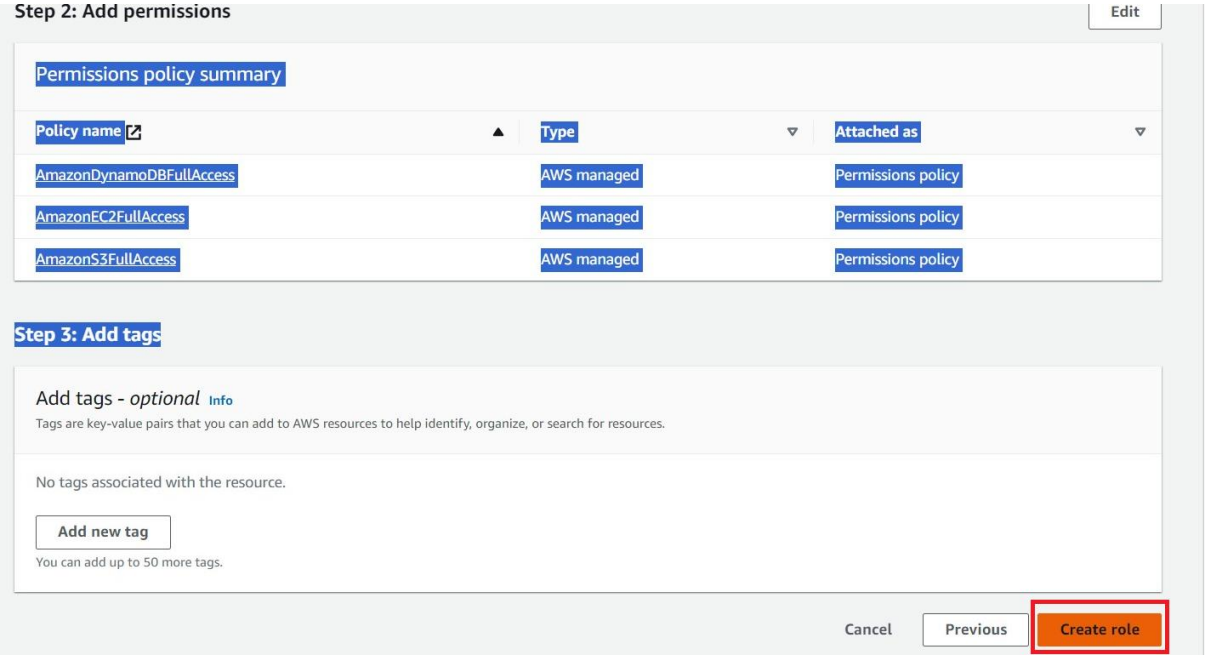
AmazonS3FullAccess and EKS Full access

click Next

Click the “Role name” field.

Type “Jenkins-cicd”

Click “Create role” (JUST SAMPLE IMAGE BELOW ONE)

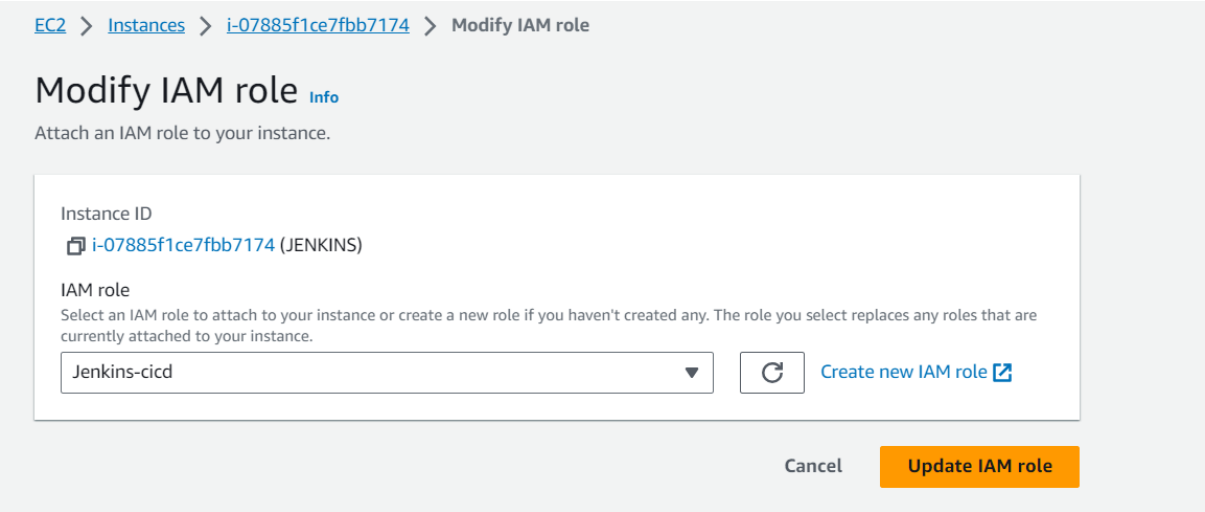


Click “EC2”

Go to the instance and add this role to the Ec2 instance.

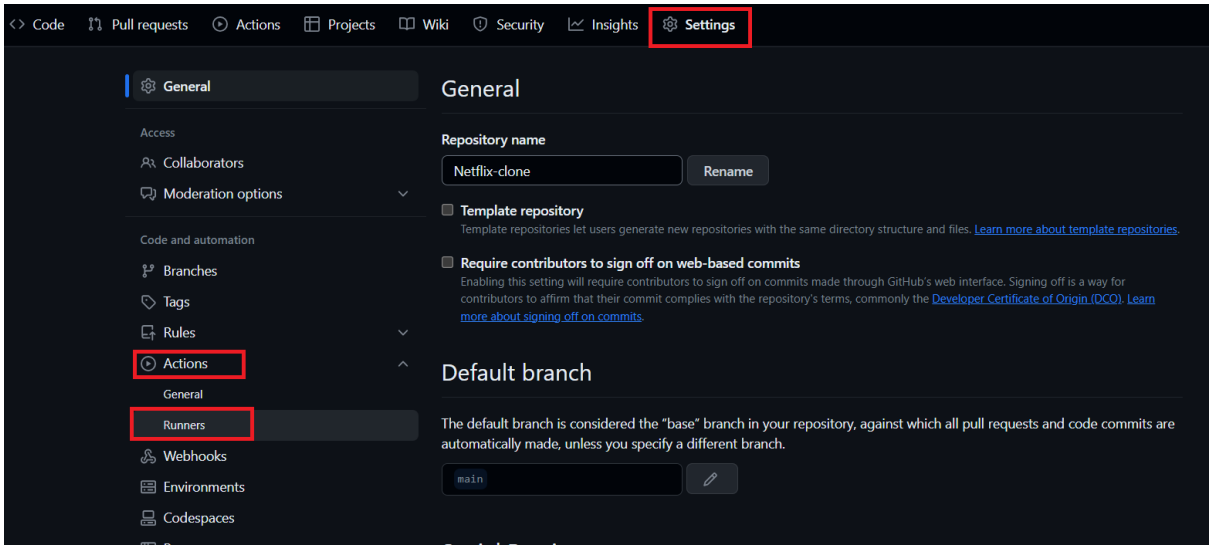
Select instance –> Actions –> Security –> Modify IAM role

Add a newly created Role and click on Update IAM role.

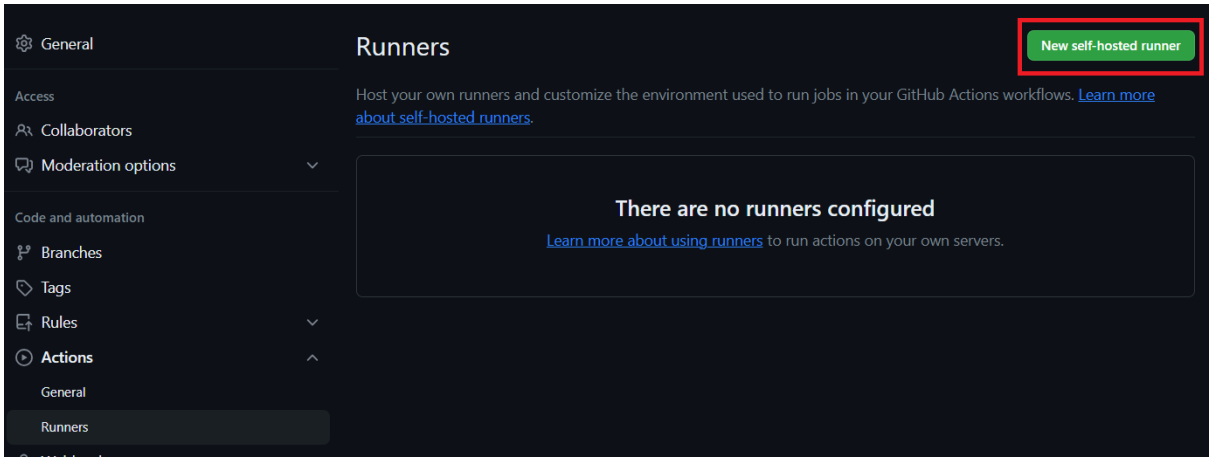


Step1B: Add a self-hosted runner to Ec2

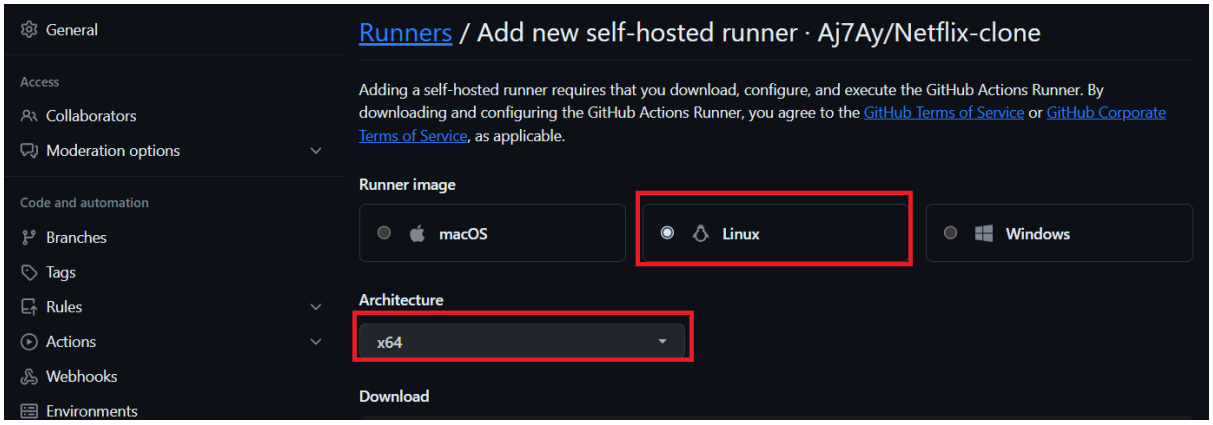
Go to GitHub and click on Settings –> Actions –> Runners



Click on New self-hosted runner



Now select Linux and Architecture X64



Use the below commands to add a self-hosted runner

Download

```
# Create a folder
$ mkdir actions-runner && cd actions-runner
# Download the latest runner package
$ curl -o actions-runner-linux-x64-2.310.2.tar.gz -L
https://github.com/actions/runner/releases/download/v2.310.2/actions-runner-linux-x64-2.310.2.tar.gz
# Optional: Validate the hash
$ echo "fb28a1c3715e0a6c5051af0e6eeff9c255009e2eec6fb08bc2708277fbb49f93  actions-runner-linux-x64-
2.310.2.tar.gz" | shasum -a 256 -c
# Extract the installer
$ tar xzf ./actions-runner-linux-x64-2.310.2.tar.gz
```

Configure

```
# Create the runner and start the configuration experience
$ ./config.sh --url https://github.com/Aj7Ay/Netflix-clone --token A2MXW45MNGB3QV6SJ6D5LWTFGCRPW
# Last step, run it!
$ ./run.sh
```

Using your self-hosted runner

```
# Use this YAML in your workflow file for each job
runs-on: self-hosted
```

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

```
ubuntu@ip-172-31-32-28:~$
ubuntu@ip-172-31-32-28:~$
ubuntu@ip-172-31-32-28:~$ mkdir actions-runner && cd actions-runner
ubuntu@ip-172-31-32-28:~/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/actions/runner/releases/download/v2.310.2/actions-runner-linux-x64-2.310.2.tar.gz

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.

```
ubuntu@ip-172-31-32-28:~/actions-runner$
ubuntu@ip-172-31-32-28:~/actions-runner$
ubuntu@ip-172-31-32-28:~/actions-runner$ curl -o actions-runner-linux-x64-2.310.2.tar.gz -L https://github.com/actions/runner/releases/download/v2.310.2/actions-runner-linux-x64-2.31
2.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--    0
100  178M  100  178M    0     0  82.7M      0  0:00:02  0:00:02 --:--:--  119M
ubuntu@ip-172-31-32-28:~/actions-runner$ ls -l
total 183028
-rw-rw-r-- 1 ubuntu ubuntu 187416718 Oct 19 02:33 actions-runner-linux-x64-2.310.2.tar.gz
ubuntu@ip-172-31-32-28:~/actions-runner$
```

Let’s validate the hash installation

echo "fb28a1c3715e0a6c5051af0e6eeff9c255009e2eec6fb08bc2708277fbb49f93 actions-runner-linux-x64-2.310.2.tar.gz" | shasum -a 256 -c

```
ubuntu@ip-172-31-32-28:~/actions-runner$
ubuntu@ip-172-31-32-28:~/actions-runner$
ubuntu@ip-172-31-32-28:~/actions-runner$ echo "fb28a1c3715e0a6c5051af0e6eeff9c255009e2eec6fb08bc2708277fbb49f93  actions-runner-linux-x64-2.310.2.tar.gz" | shasum -a 256 -c
actions-runner-linux-x64-2.310.2.tar.gz: OK
ubuntu@ip-172-31-32-28:~/actions-runner$
```

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 A2MXW4323ALGB72GGLH34NLFGI2T4

```
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

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

√ Connected to GitHub

Current runner version: '2.310.2'
2023-10-19 02:35:20Z: Listening for Jobs
```

Let's close Runner for now.

ctrl + c #to close

Step2A: Install Docker and Run Sonarqube Container

Connect to your Ec2 instance using Putty, Mobaxtreme or Git bash and install docker on it.

sudo apt-get update

sudo apt install docker.io -y

sudo usermod -aG docker ubuntu

newgrp docker

sudo chmod 777 /var/run/docker.sock

Pull the SonarQube Docker image and run it.

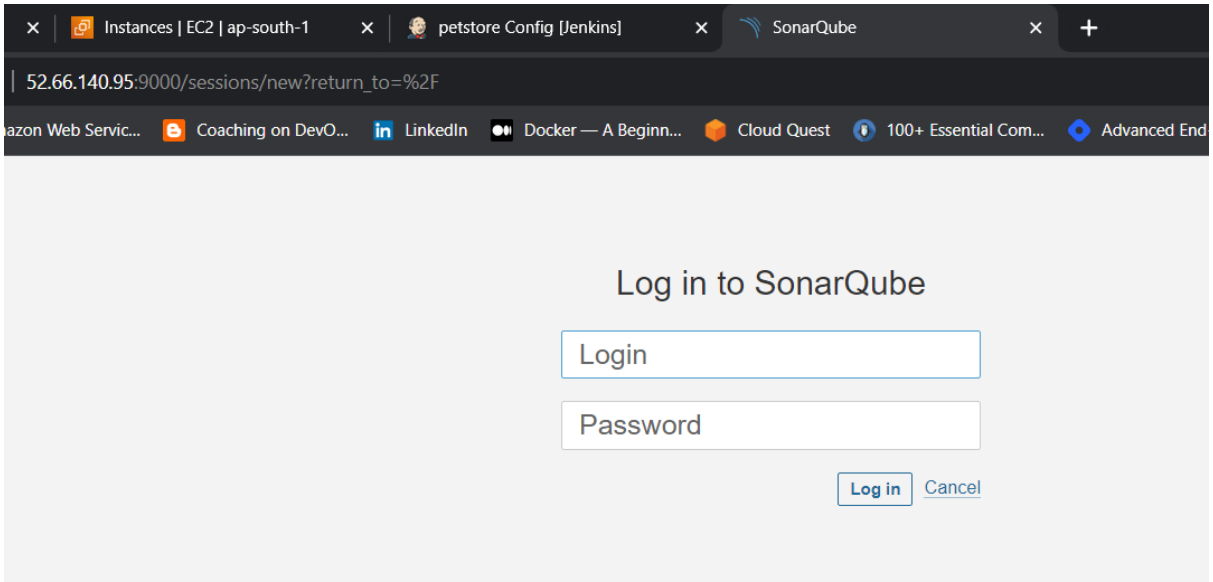
After the docker installation, we will create a Sonarqube container (Remember to add 9000 ports in the security group).

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

```
ubuntu@ip-172-31-42-253:~$ sudo chmod 777 /var/run/docker.sock
ubuntu@ip-172-31-42-253:~$ docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
44ba2882f8eb: Pull complete
2cabec57fa36: Pull complete
c20481384b6a: Pull complete
bf7b17ee74f8: Pull complete
38617faac714: Pull complete
706f20f58f5e: Pull complete
65a29568c257: Pull complete
Digest: sha256:1a118f8ab960d6c3d4ea8b4455a5a6560654511c88a6816f1603f764d5dcc77c
Status: Downloaded newer image for sonarqube:lts-community
4b60c96bf9ad3d62289436af7f752fdb04993092d0ca3065e2f2e32301b50139
ubuntu@ip-172-31-42-253:~$ docker ps
CONTAINER ID   IMAGE                   COMMAND                  CREATED         STATUS         PORTS                                       NAMES
4b60c96bf9ad   sonarqube:lts-community   "/opt/sonarqube/dock…"   9 seconds ago   Up 5 seconds   0.0.0.0:9000->9000/tcp, :::9000->9000/tcp   sonar
ubuntu@ip-172-31-42-253:~$
```

Now copy the IP address of the ec2 instance

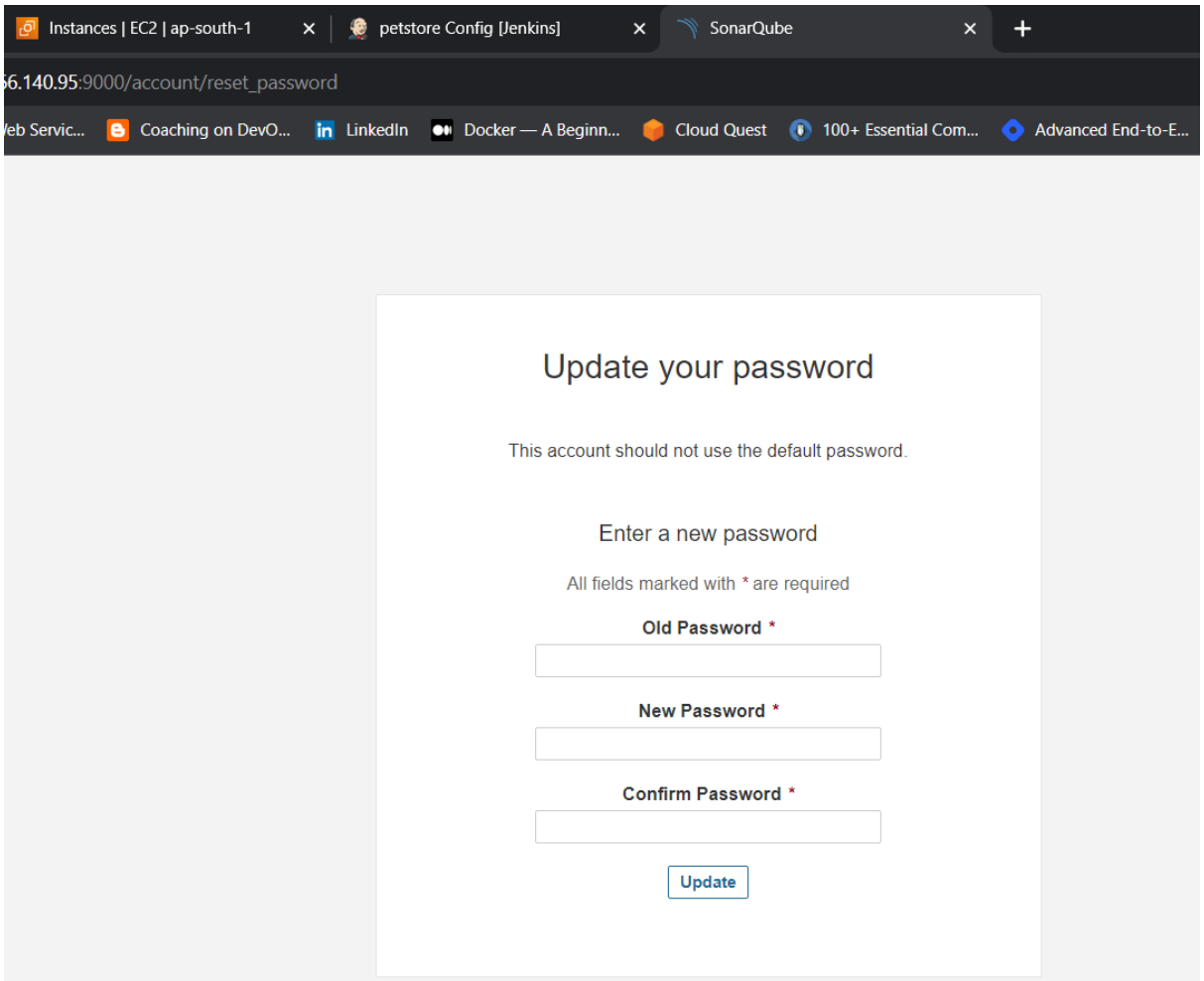
<ec2-public-ip:9000>



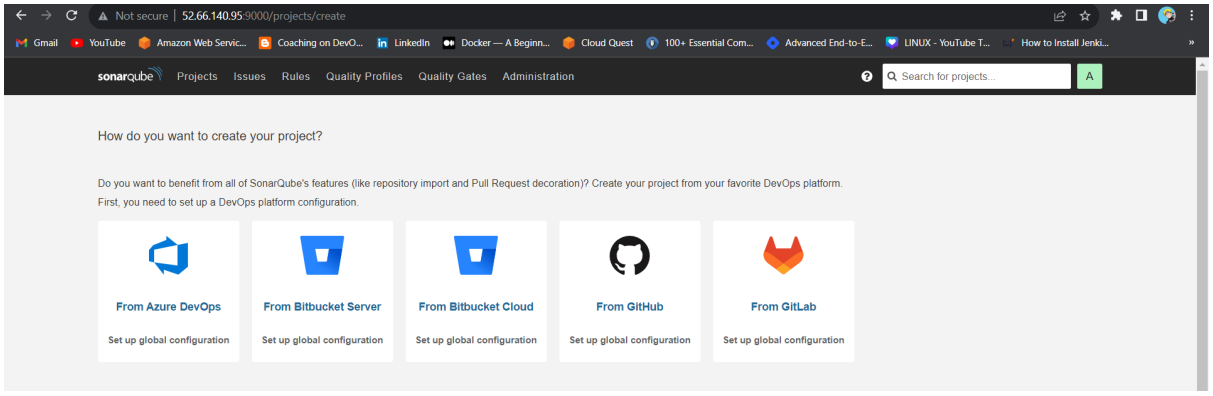
Provide Login and password

login admin

password admin



Update your Sonarqube password & This is the Sonarqube dashboard

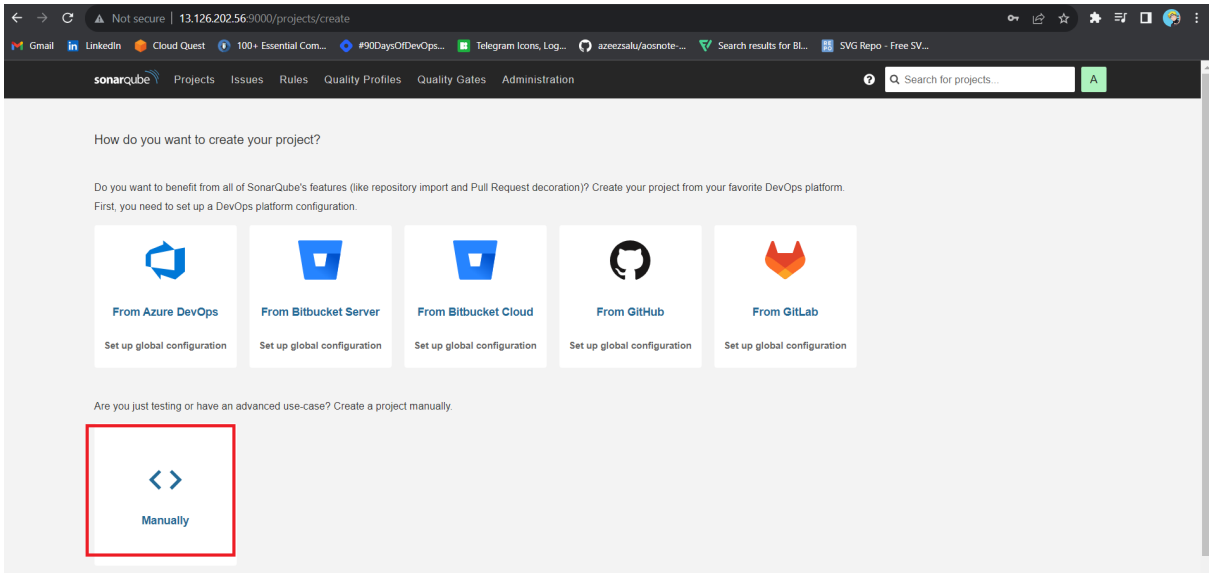


Step2B: Integrating SonarQube with GitHub Actions

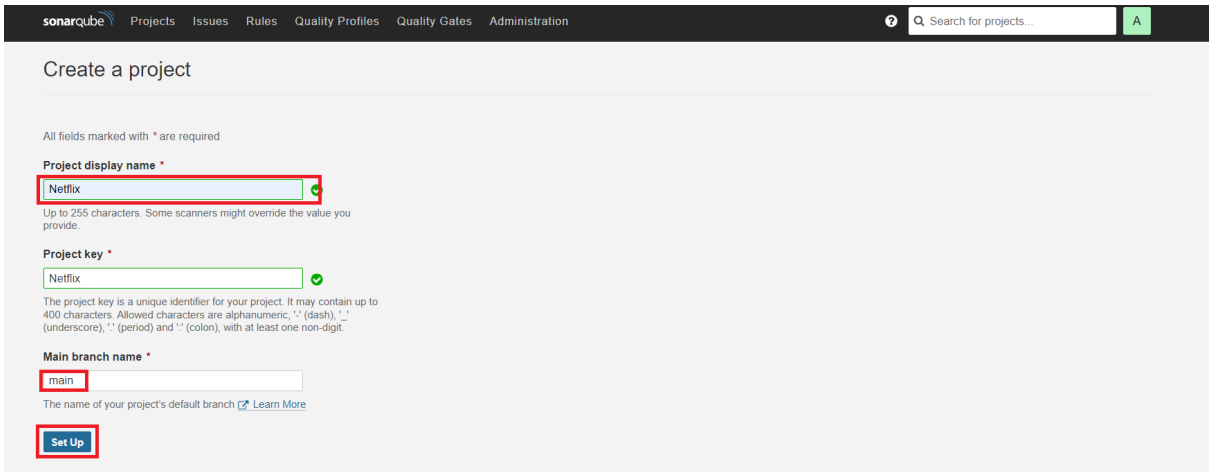
Integrating SonarQube with GitHub Actions allows you to automatically analyze your code for quality and security as part of your continuous integration pipeline.

We already have Sonarqube up and running

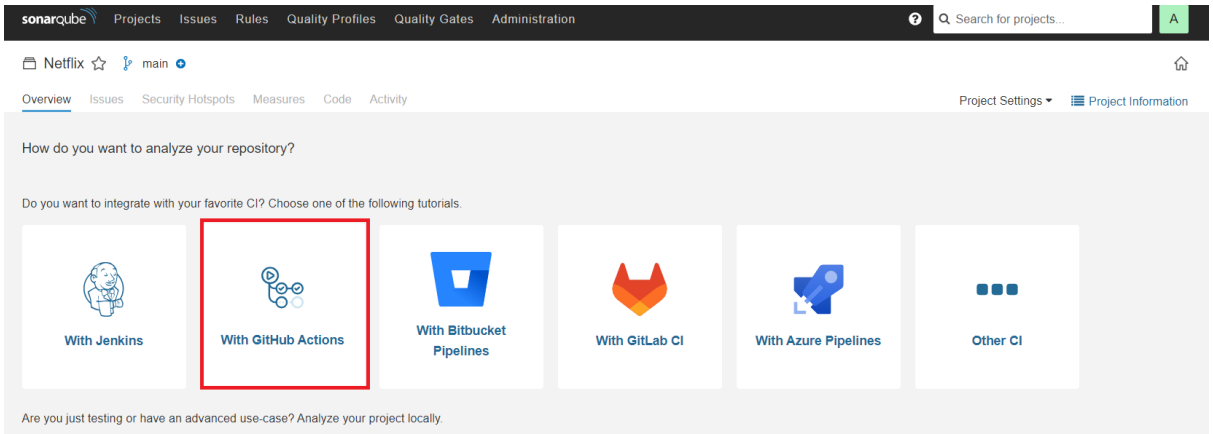
On Sonarqube Dashboard click on Manually



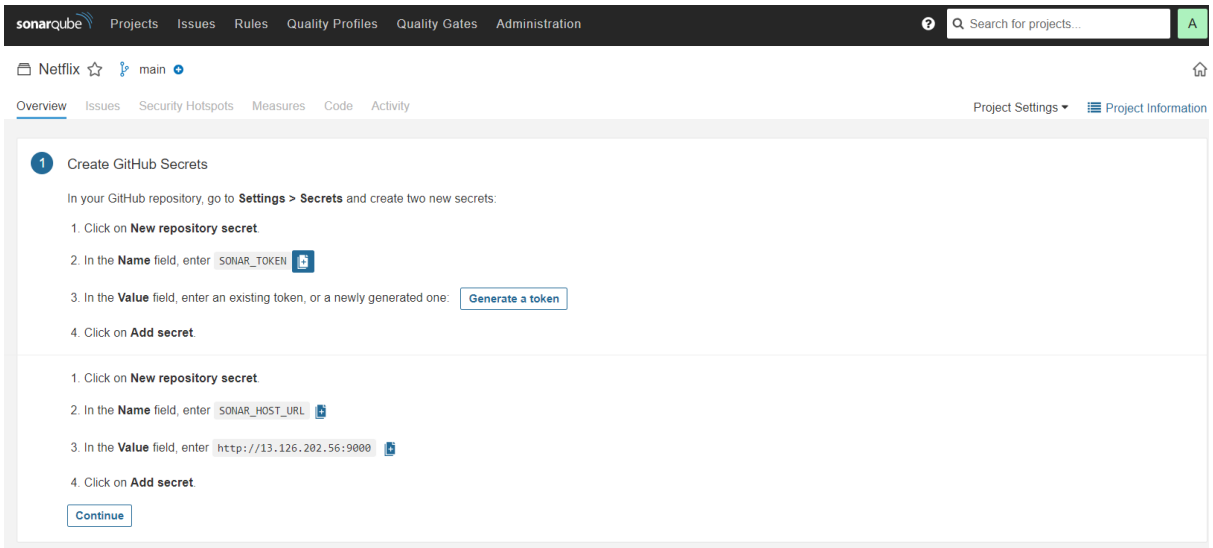
Next, provide a name for your project and provide a Branch name and click on setup



On the next page click on With GitHub actions

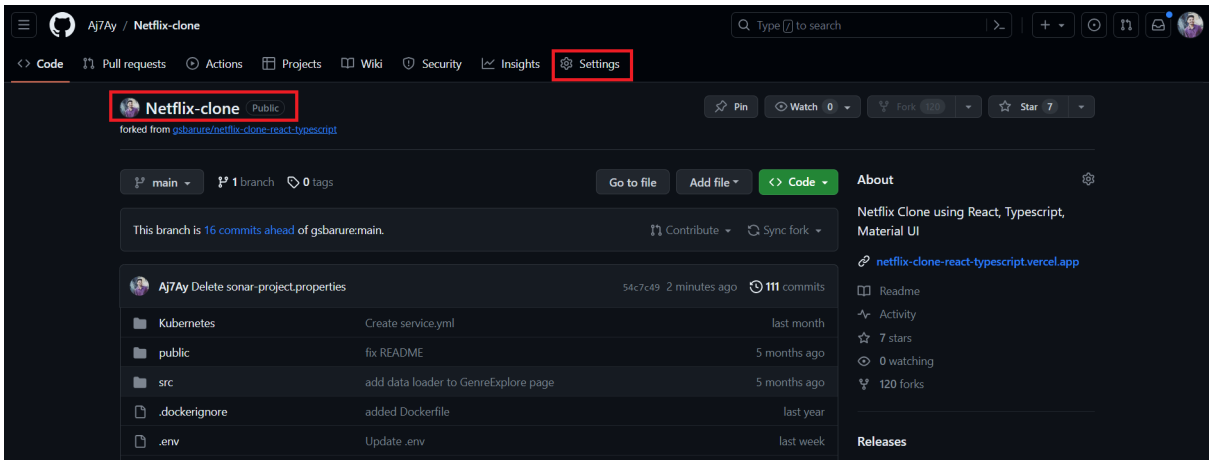


This will Generate an overview of the Project and provide some instructions to integrate

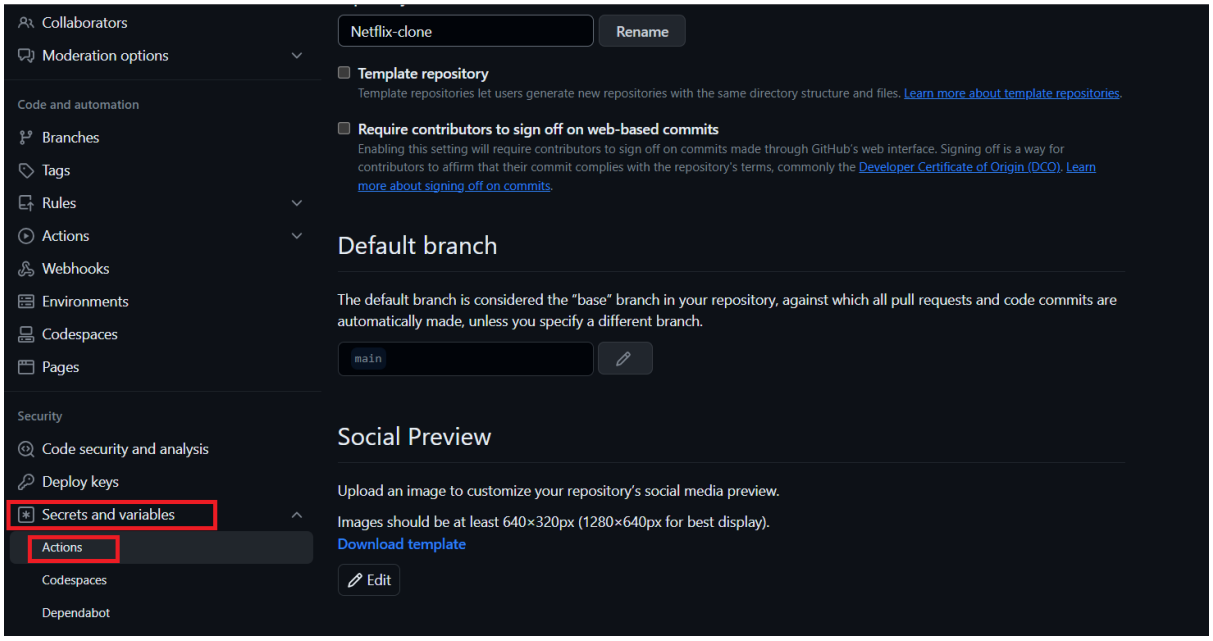


Let's Open your GitHub and select your Repository

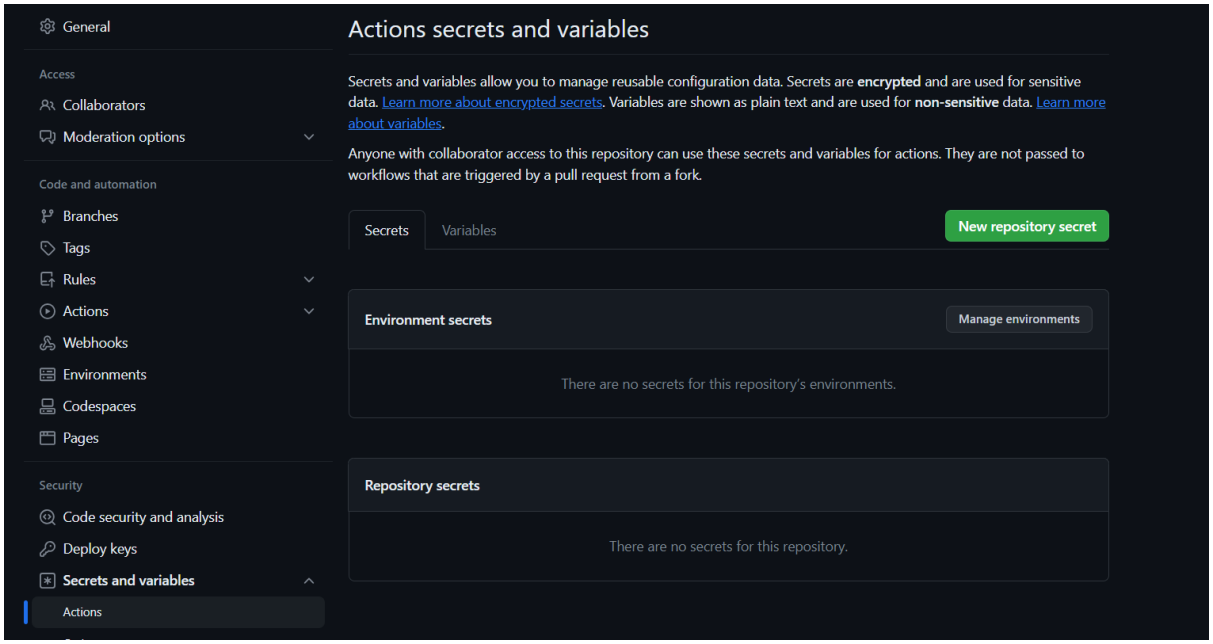
In my case it is Netflix-clone and Click on Settings



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

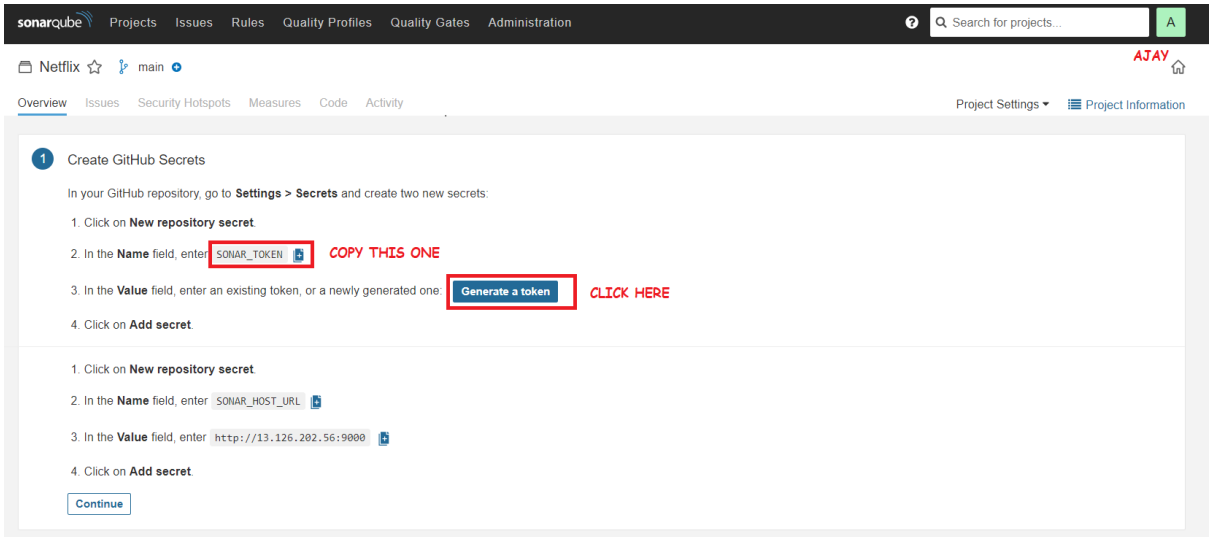


It will open a page like this click on New Repository secret

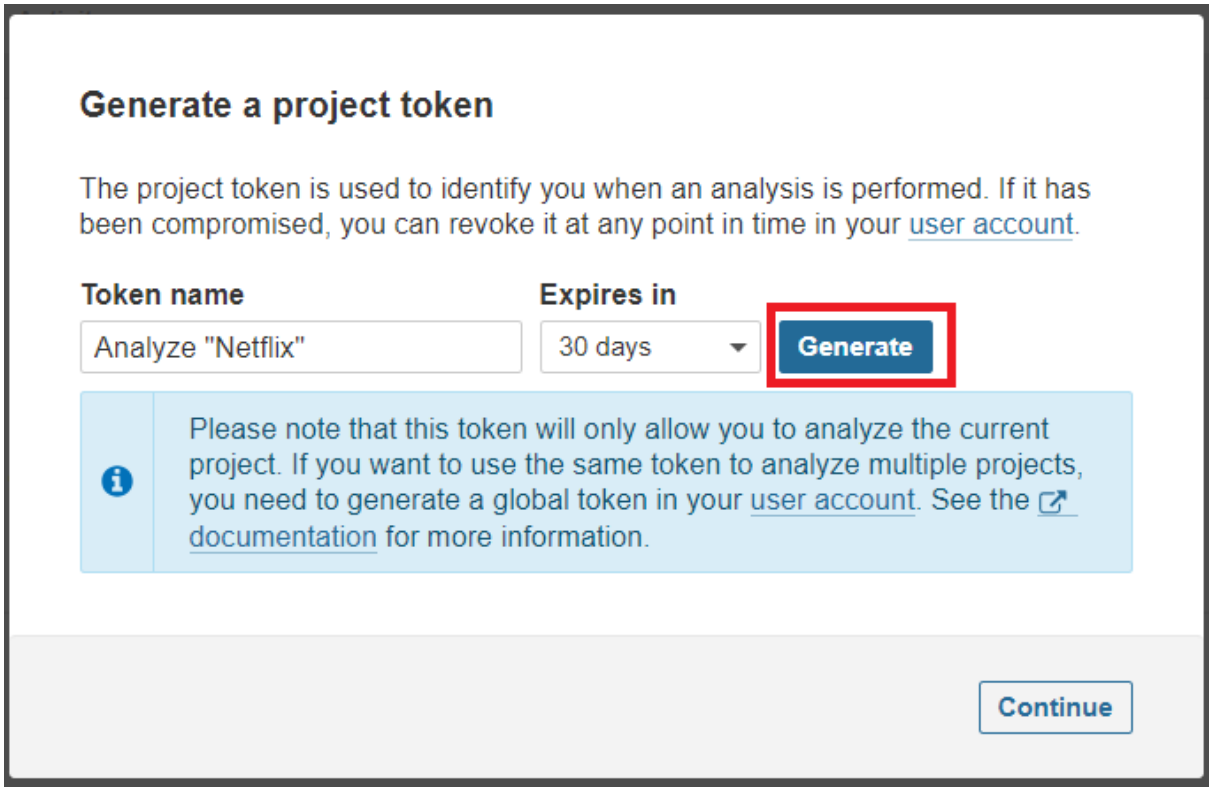


Now go back to Your Sonarqube Dashboard

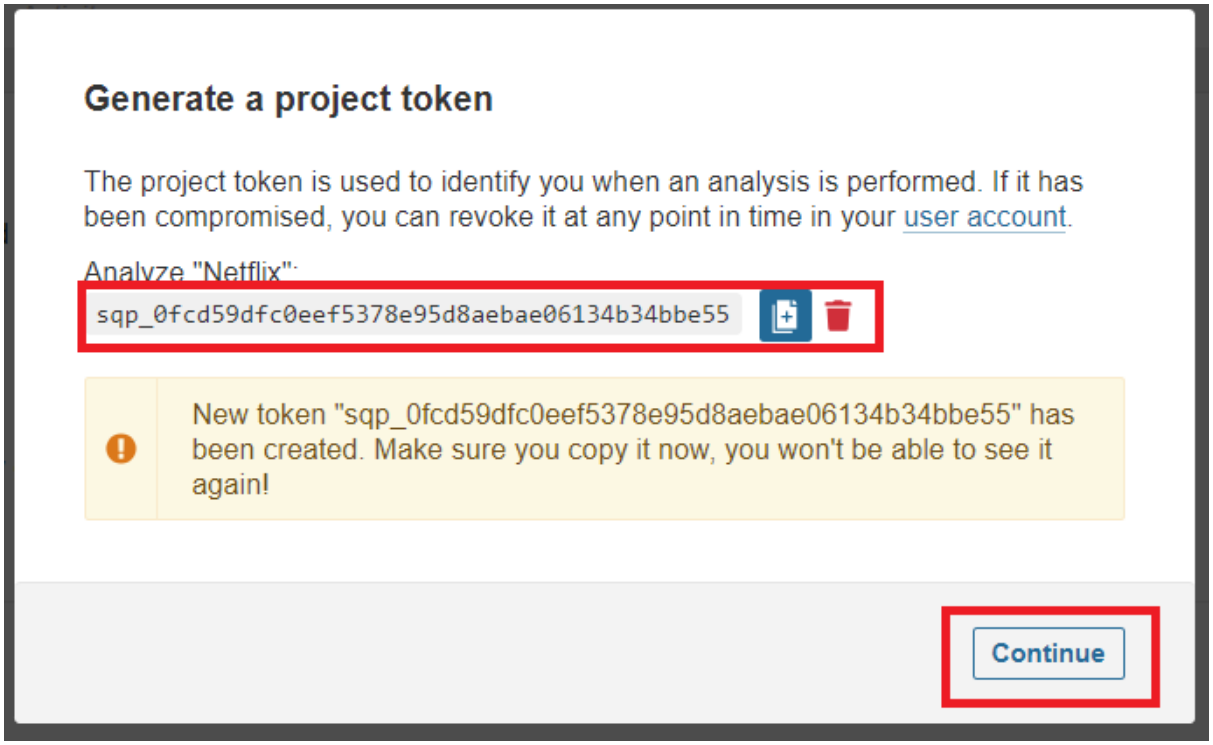
Copy SONAR_TOKEN and click on Generate Token



Click on Generate



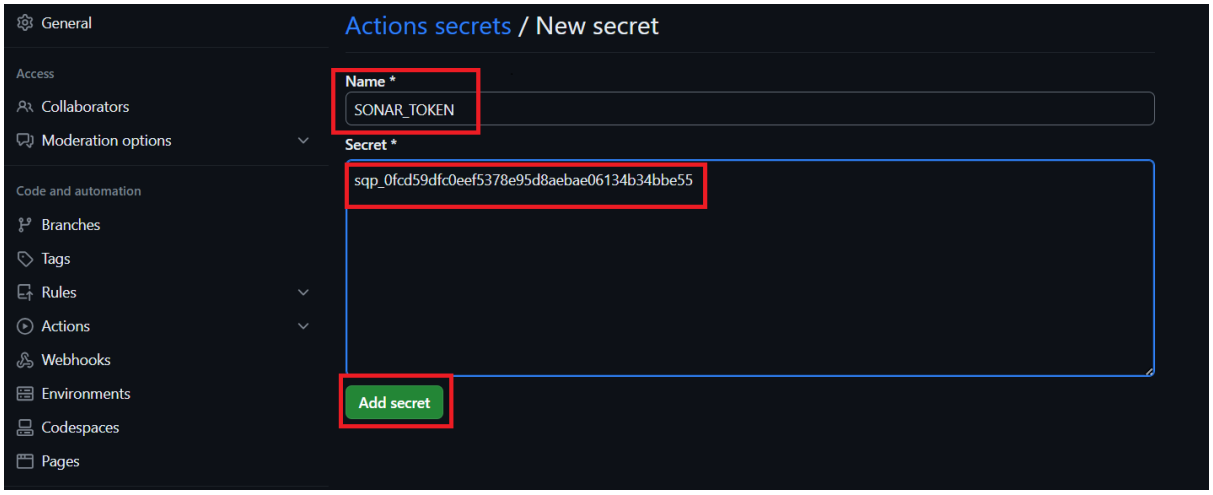
Let’s copy the Token and add it to GitHub secrets



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

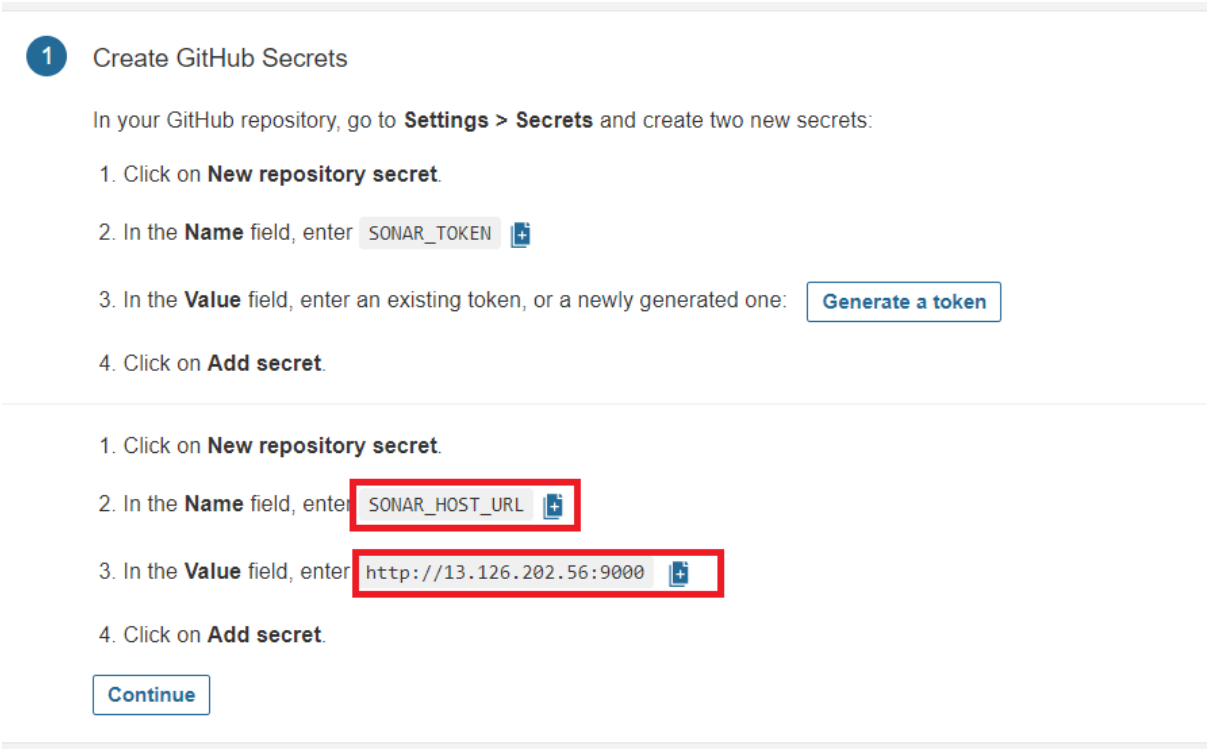
Name: SONAR_TOKEN

Secret: Paste Your Token and click on Add secret

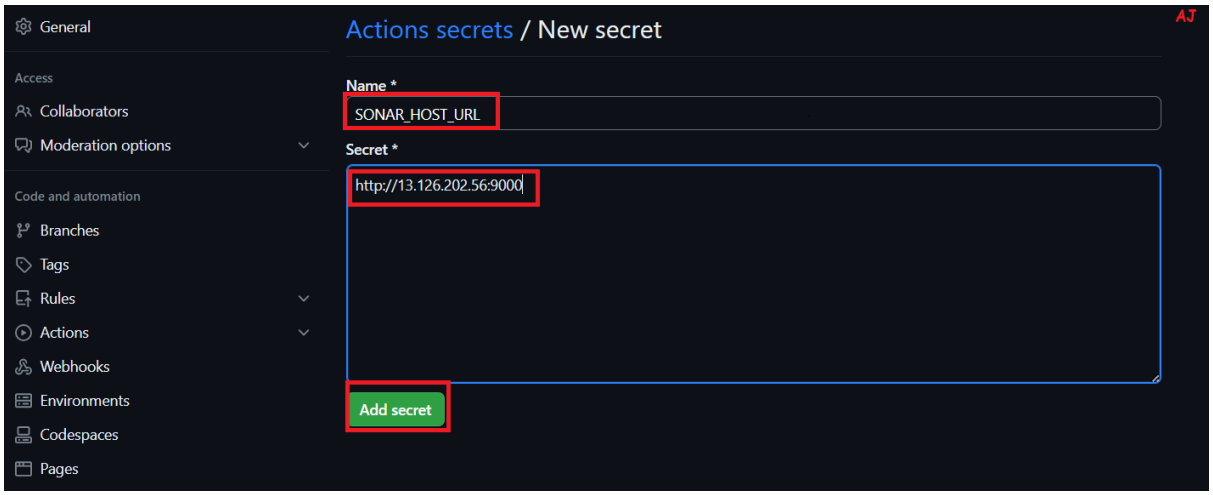


Now go back to the Sonarqube Dashboard

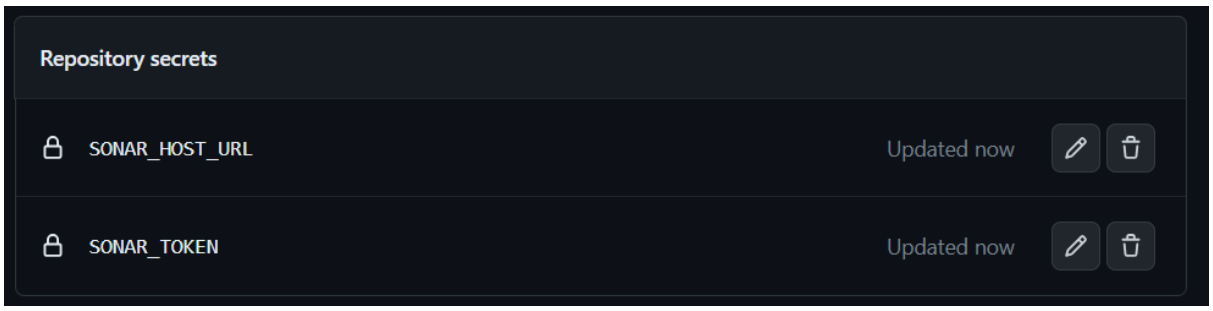
Copy the Name and Value



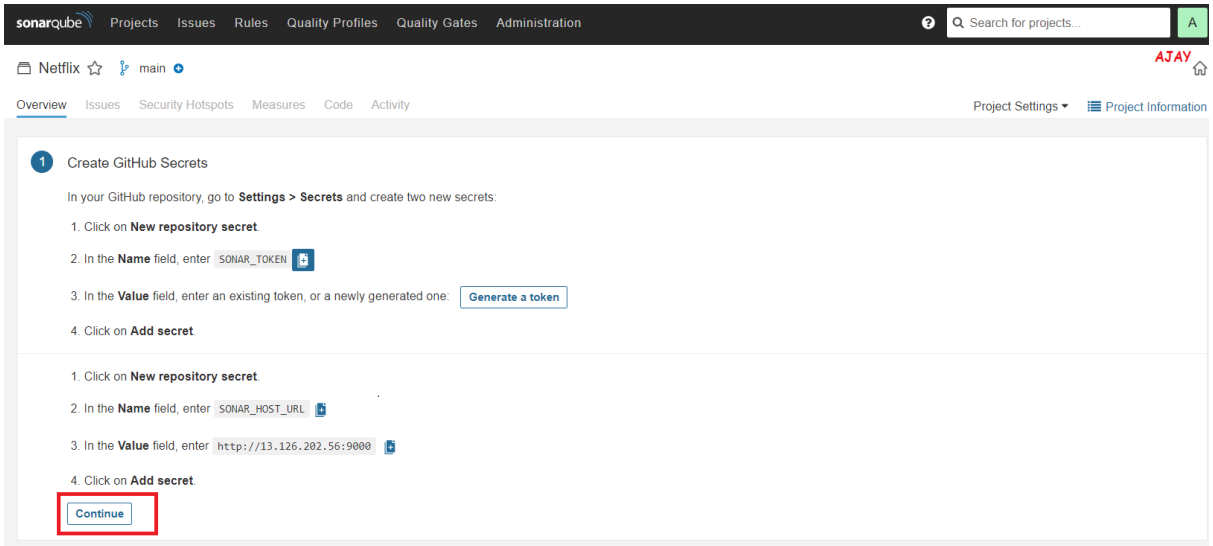
Go to GitHub now and paste-like this and click on add secret



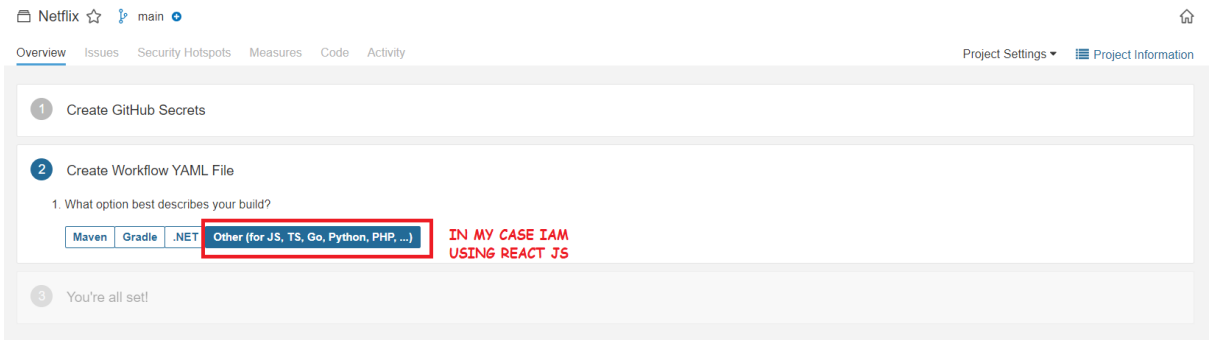
Our Sonarqube secrets are added and you can see



Go to Sonarqube Dashboard and click on continue

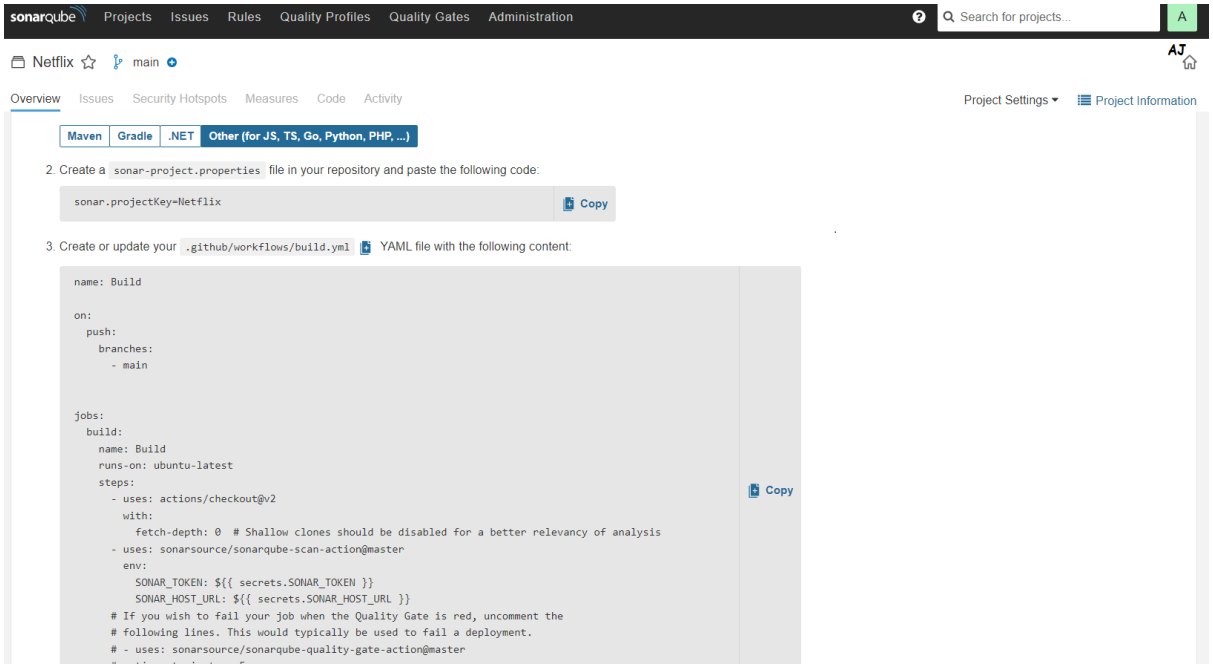


Now create your Workflow for your Project. In my case, the Netflix project is built using React Js. That's why I am selecting Other

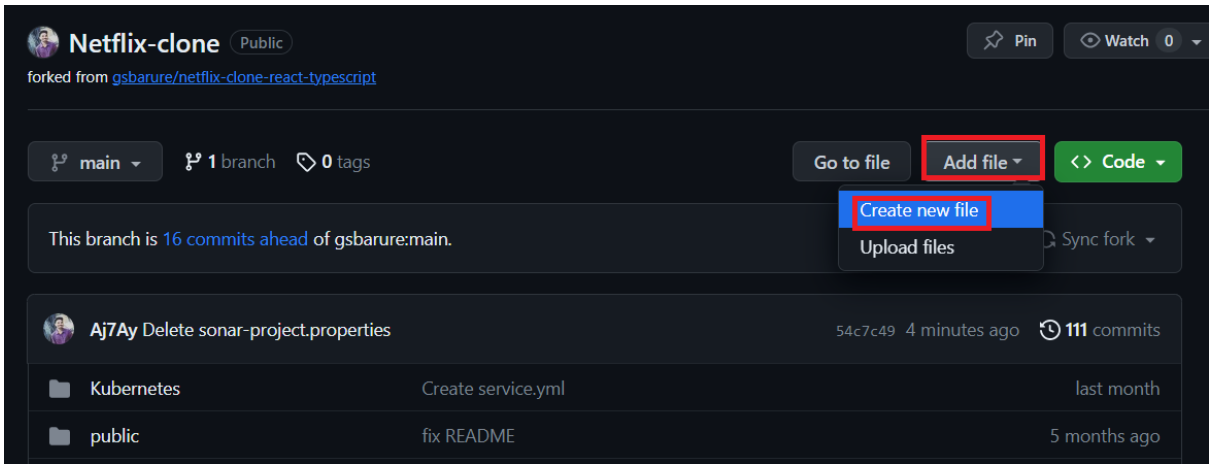


Now it Generates and workflow for my Project

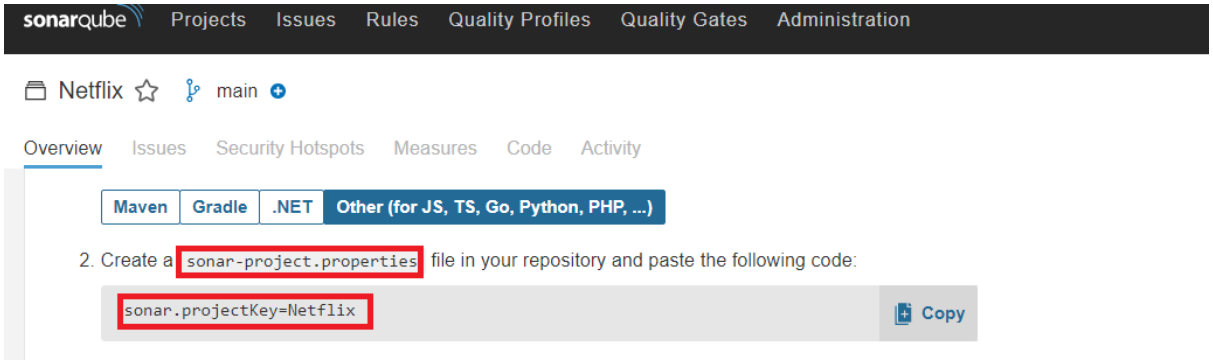
(Use your files for this block please)



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



Here file name (in my case only)

sonar-project.properties

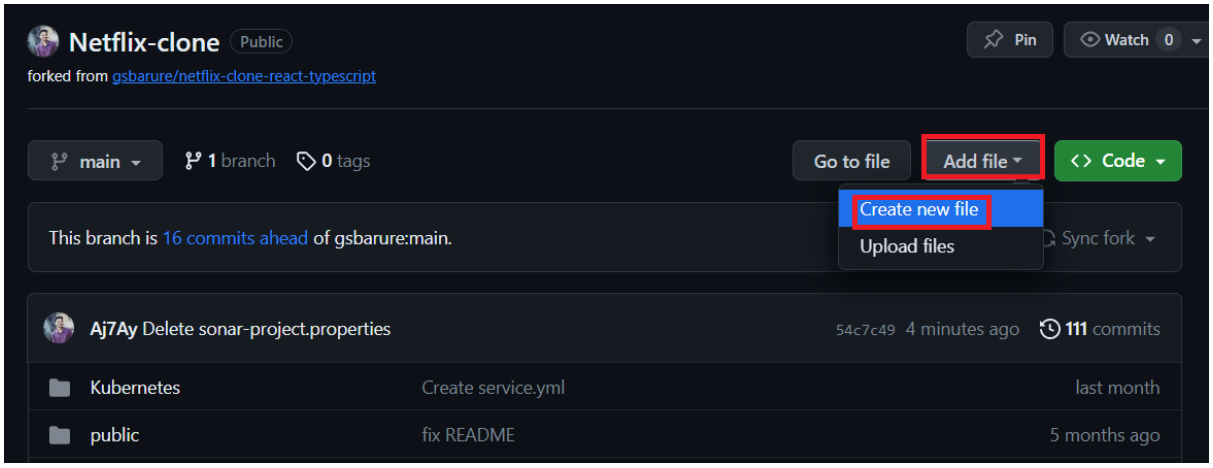
The content to add to the file is (copied from the above image)

sonar.projectKey=Tic-game

Add in GitHub like this (sample images)

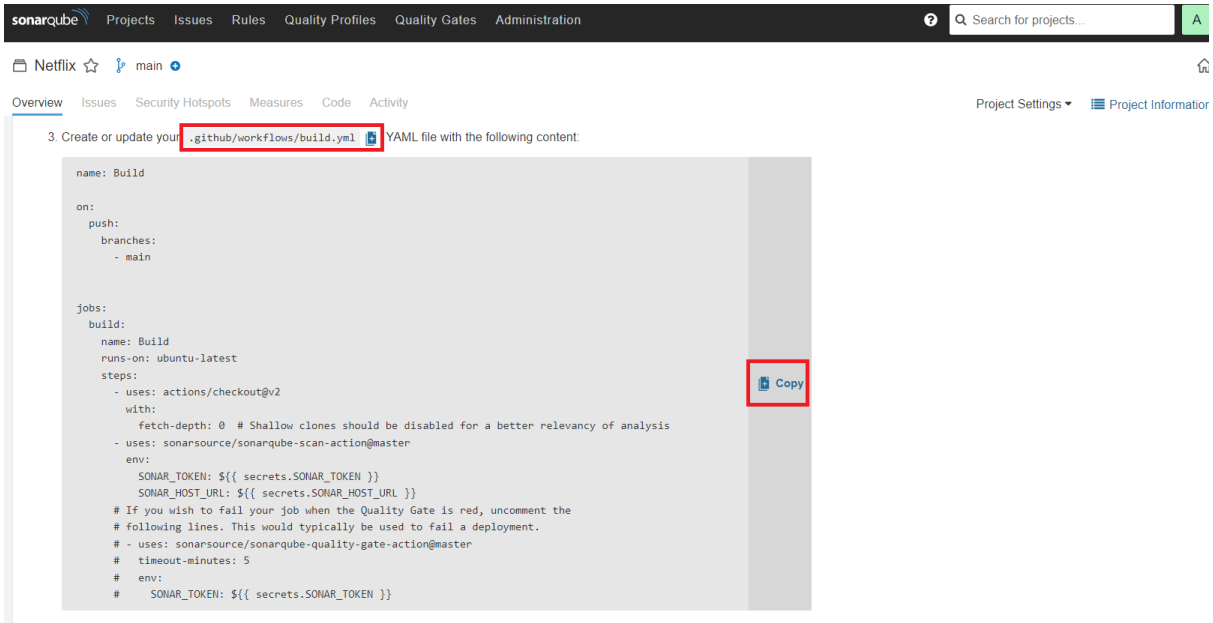
Let's add our workflow

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



Here is the file name

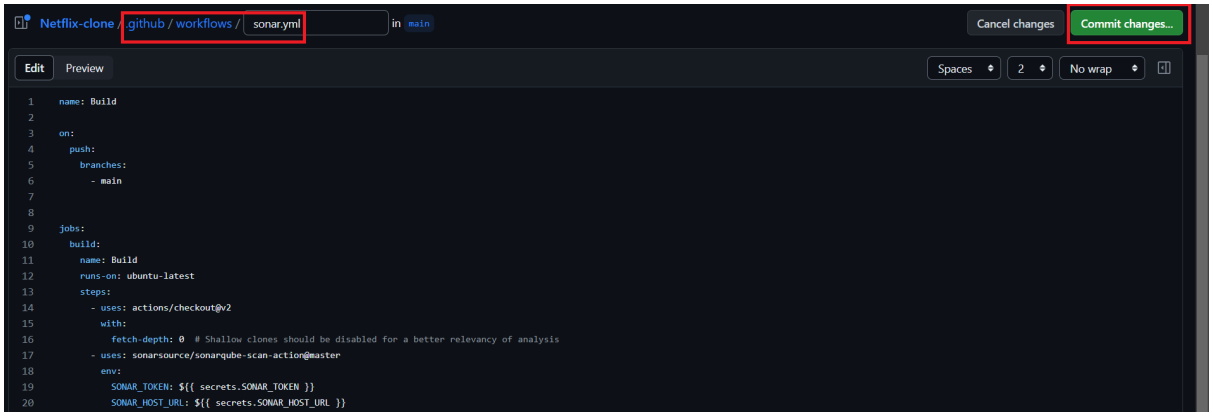
.github/workflows/build.yml #you can use any name iam using sonar.yml



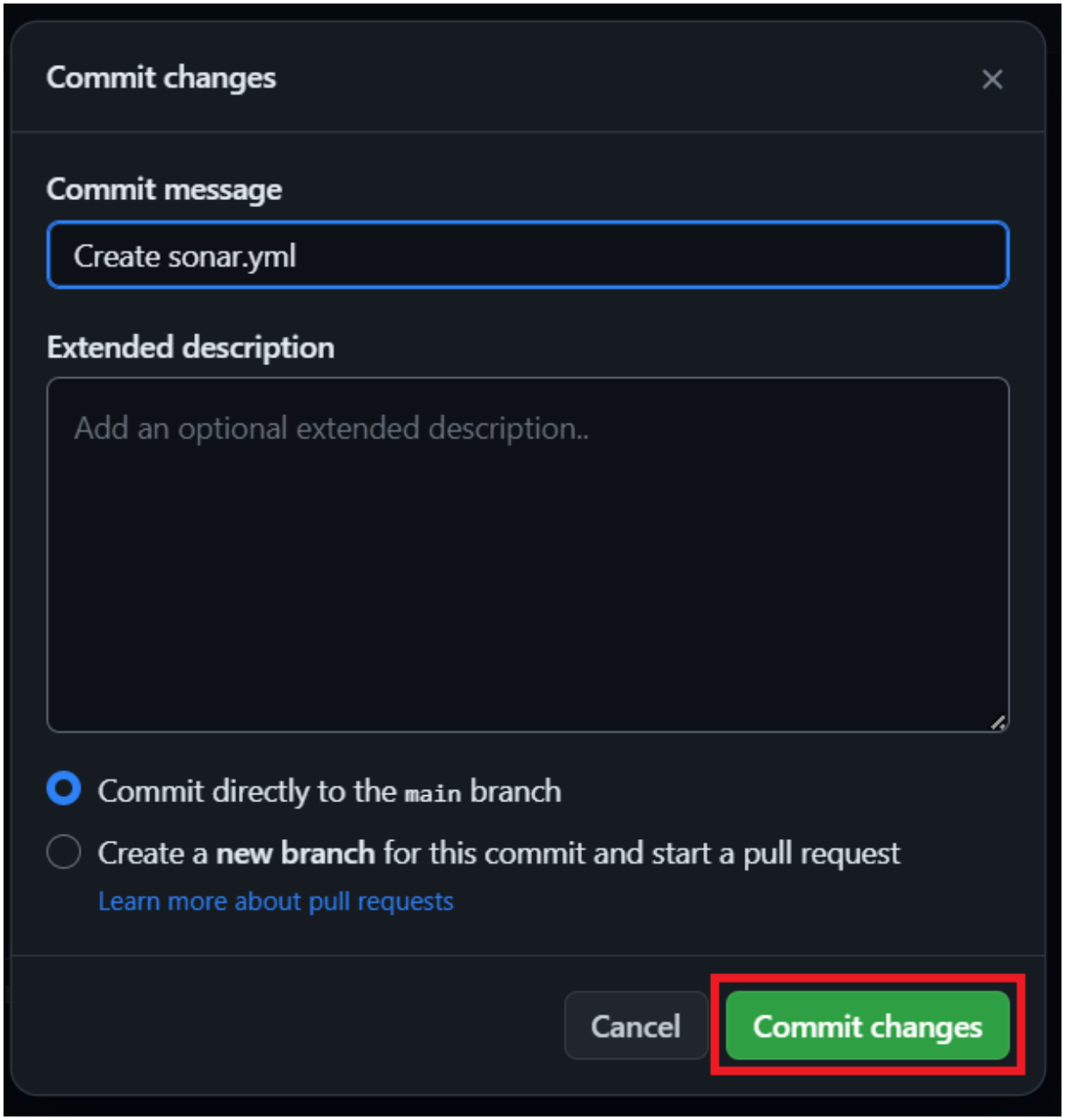
Copy content and add it to the file

```
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 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 }}
```



Click on commit changes



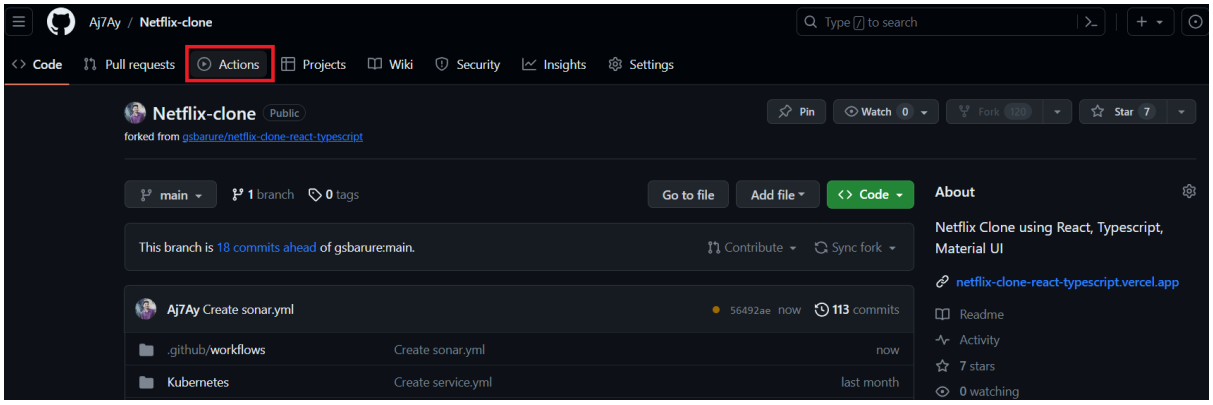
Now workflow is created.

Start again GitHub actions runner from instance

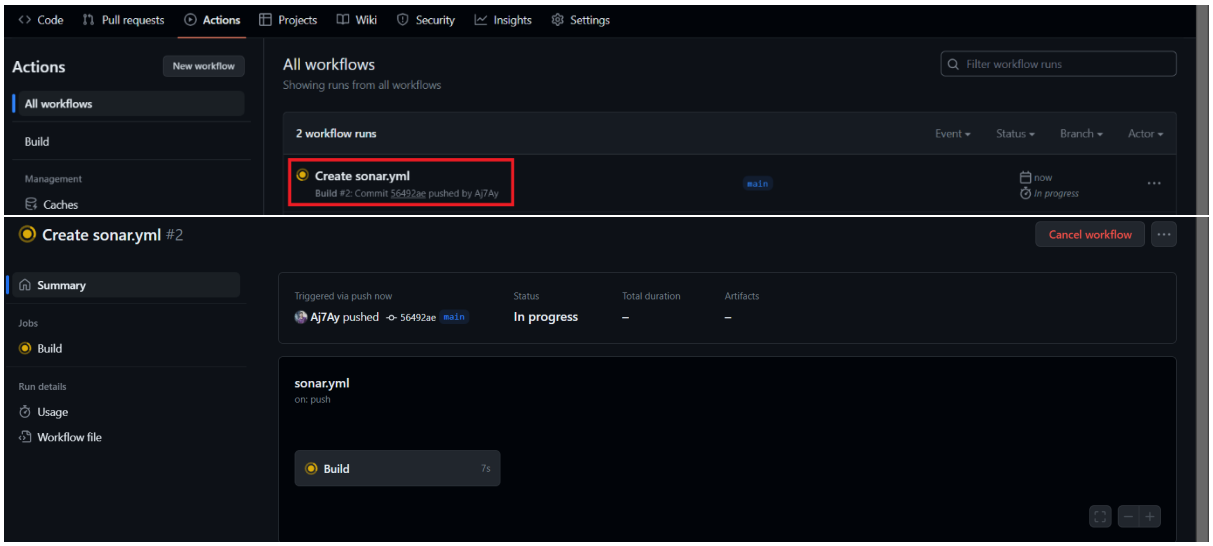
cd actions-runner

./run.sh

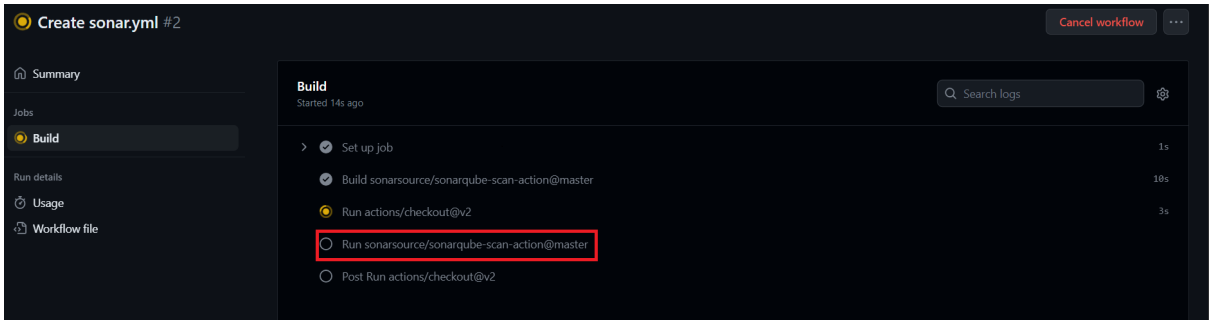
Click on Actions now



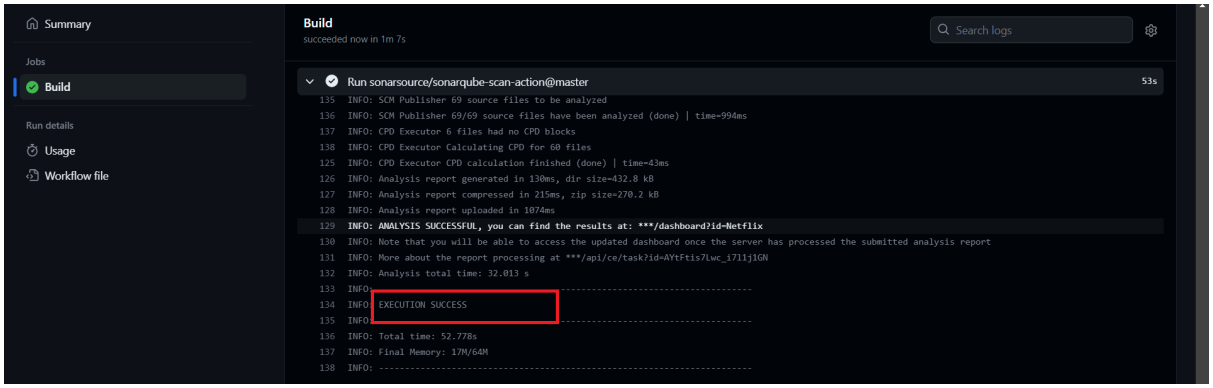
Now it’s automatically started the workflow



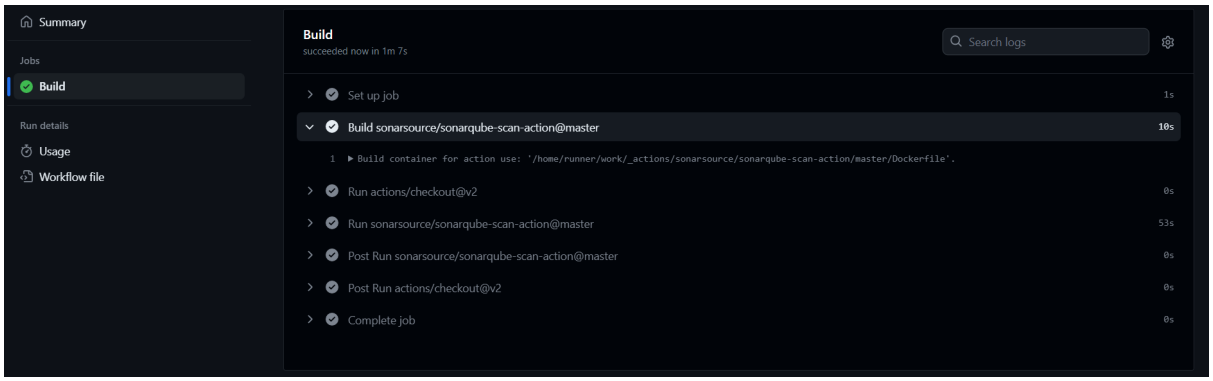
Let’s click on Build and see what are the steps involved



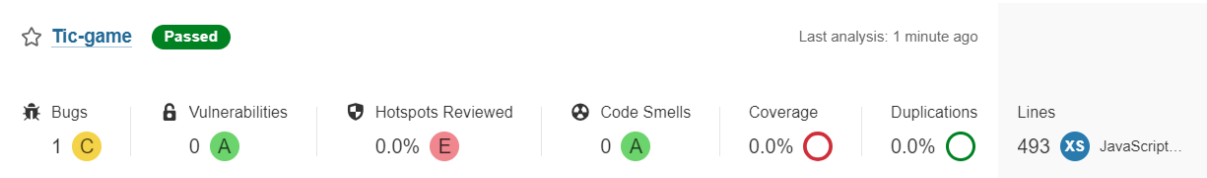
Click on Run Sonarsource and you can do this after the build completion



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.

Step2C: INSTALLATION OF OTHER TOOLS

1. **Install Java 17**:
    - Install Temurin (formerly Adoptium) JDK 17.
2. **Install Trivy** (Container Vulnerability Scanner).
3. **Install Terraform**.
4. **Install kubectl** (Kubernetes command-line tool).
5. **Install AWS CLI** (Amazon Web Services Command Line Interface).
6. **Install Node.js 16 and npm**.

The script automates the installation of these software tools commonly used for development and deployment.

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.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.list

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
```

Check whether the versions are also installed or not.

```
trivy --version

terraform --version

aws --version

kubectl version

node -v

java --version
```

```
ubuntu@ip-172-31-11-71:~$
ubuntu@ip-172-31-11-71:~$
ubuntu@ip-172-31-11-71:~$ trivy --version
Version: 0.46.0
ubuntu@ip-172-31-11-71:~$
ubuntu@ip-172-31-11-71:~$
ubuntu@ip-172-31-11-71:~$ aws --version
aws-cli/2.13.29 Python/3.11.6 Linux/5.19.0-1025-aws exe/x86_64.ubuntu.22 prompt/off
ubuntu@ip-172-31-11-71:~$
ubuntu@ip-172-31-11-71:~$
ubuntu@ip-172-31-11-71:~$ terraform --version
Terraform v1.6.2
on linux_amd64
ubuntu@ip-172-31-11-71:~$
ubuntu@ip-172-31-11-71:~$
ubuntu@ip-172-31-11-71:~$ kubectl --version
error: unknown flag: --version
See 'kubectl --help' for usage.
ubuntu@ip-172-31-11-71:~$ kubectl version
Client Version: v1.28.3
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Error from server (Forbidden): <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fvers
timeout%3D32s');</script></head><body style='background-color:white; color:white;'>


Authentication required
<!--
-->

</body></html>
ubuntu@ip-172-31-11-71:~$
```

```
ubuntu@ip-172-31-36-122:~$
ubuntu@ip-172-31-36-122:~$ node -v
v16.20.2
ubuntu@ip-172-31-36-122:~$
```

```
ubuntu@ip-172-31-36-122:~$
ubuntu@ip-172-31-36-122:~$ java --version
openjdk 17.0.9 2023-10-17
OpenJDK Runtime Environment Temurin-17.0.9+9 (build 17.0.9+9)
OpenJDK 64-Bit Server VM Temurin-17.0.9+9 (build 17.0.9+9, mixed mode, sharing)
ubuntu@ip-172-31-36-122:~$
```

EKS provision

Clone the repo onto your instance

git clone https://github.com/Aj7Ay/Candycrush.git

cd Candycrush

cd Eks-terraform

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.
```

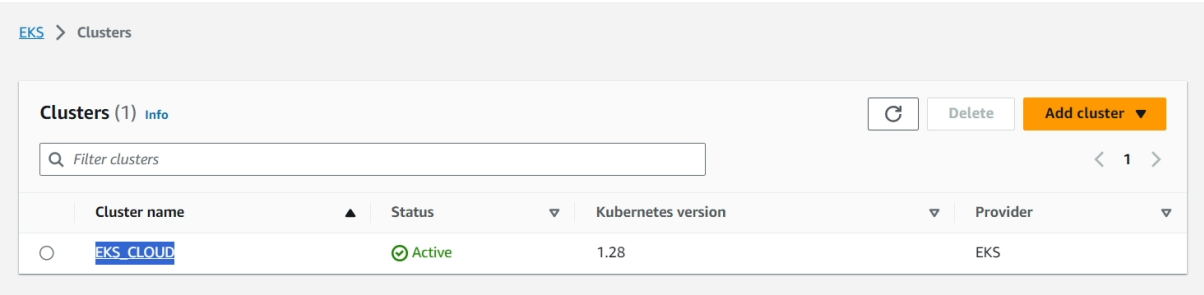
Plan and apply

terraform plan

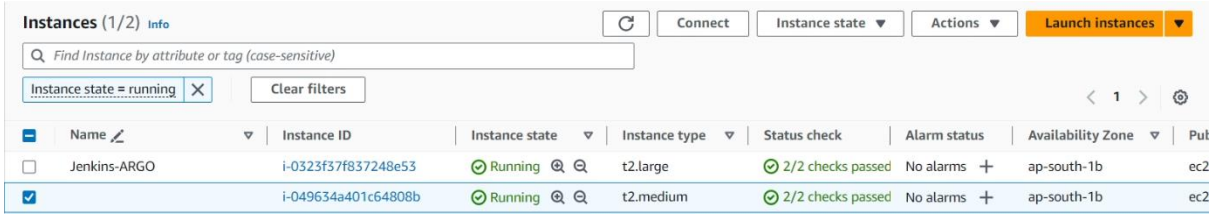
terraform apply --auto-approve

```
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.

- name: Install Trivy

  run: |

    # Scanning files

    trivy fs . > trivyfs.txt

This step runs Trivy to scan files. It scans the current directory (denoted by .) and redirects the output to a file named trivyfs.txt.

If you add this to the workflow, you will get below output

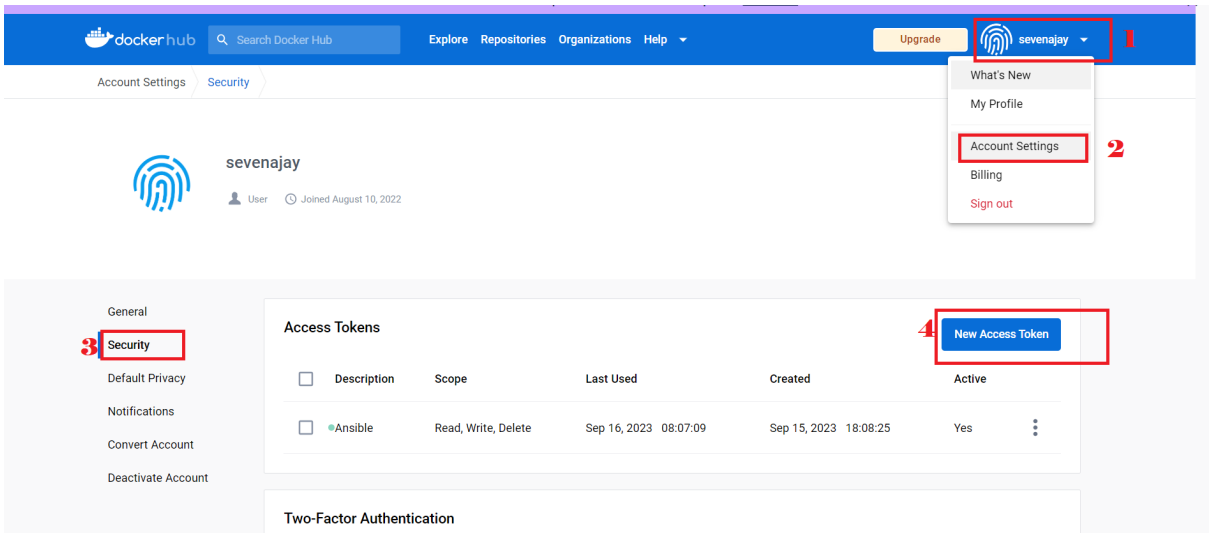
```
Trivy file scan                                                                    2s
1  ▶ Run trivy fs . > trivyfs.txt
4
4  2023-10-29T07:01:03.800Z    INFO    Vulnerability scanning is enabled
5  2023-10-29T07:01:03.800Z    INFO    Secret scanning is enabled
6  2023-10-29T07:01:03.800Z    INFO    If your scanning is slow, please try '--scanners vuln' to disable secret scanning
7  2023-10-29T07:01:03.800Z    INFO    Please see also https://aquasecurity.github.io/trivy/v0.46/docs/scanner/secret/#recommendation for faster secret
   detection
8  2023-10-29T07:01:04.584Z    INFO    Number of language-specific files: 1
9  2023-10-29T07:01:04.584Z    INFO    Detecting npm vulnerabilities...
```

```
ubuntu@ip-172-31-36-122:~/actions-runner/_work/TIC-TAC-TOE$
ubuntu@ip-172-31-36-122:~/actions-runner/_work/TIC-TAC-TOE$ cd TIC-TAC-TOE/
ubuntu@ip-172-31-36-122:~/actions-runner/_work/TIC-TAC-TOE/TIC-TAC-TOE$ ls
Dockerfile  Eks-terraform  Eks.yml  README.md  deployment-service.yml  node_modules  package-lock.json  package.json  public  script.sh  sonar-project.properties  src  trivyfs.txt
ubuntu@ip-172-31-36-122:~/actions-runner/_work/TIC-TAC-TOE/TIC-TAC-TOE$ cat trivyfs.txt

package-lock.json (npm)
=======================
Total: 19 (UNKNOWN: 0, LOW: 0, MEDIUM: 8, HIGH: 9, CRITICAL: 2)
```

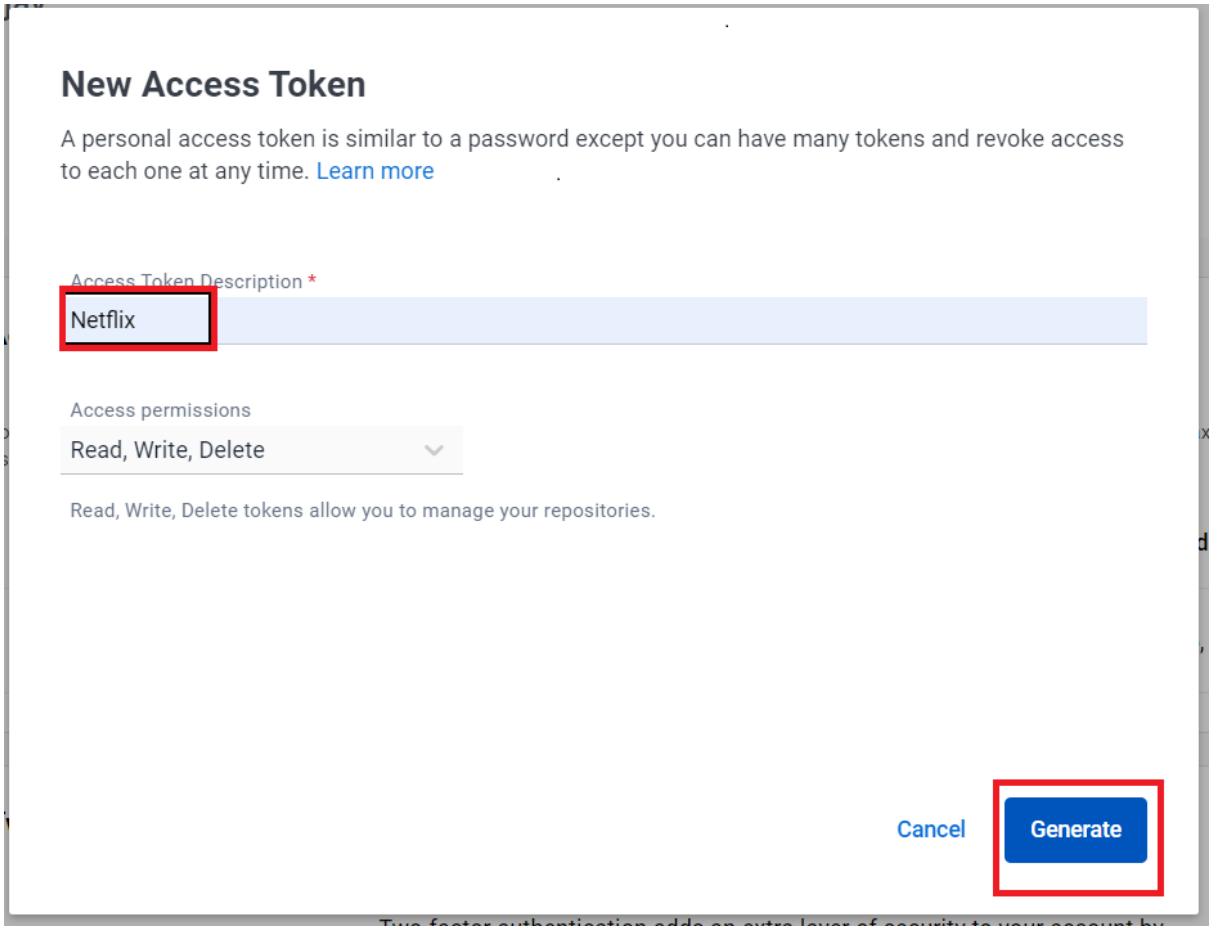
Library	Vulnerability	Severity	Status	Installed Version	Fixed Version	Title
@adobe/css-tools	CVE-2023-26364	MEDIUM	fixed	4.0.1	4.3.1	@adobe/css-tools Regular Expression Denial of Service (ReDOS) while Parsing CSS https://avd.aquasec.com/nvd/cve-2023-26364
@babel/traverse	CVE-2023-45133	CRITICAL		7.18.13	7.23.2, 8.0.0-alpha.4	arbitrary code execution https://avd.aquasec.com/nvd/cve-2023-45133
json5	CVE-2022-46175	HIGH		1.0.1	2.2.2, 1.0.2	Prototype Pollution in JSON5 via Parse Method https://avd.aquasec.com/nvd/cve-2022-46175
				2.2.1		
loader-utils	CVE-2022-37601	CRITICAL		2.0.2	2.0.3, 1.4.1	prototype pollution in function parseQuery in parseQuery.js https://avd.aquasec.com/nvd/cve-2022-37601
	CVE-2022-37599	HIGH			1.4.2, 2.0.4, 3.2.1	regular expression denial of service in interpolateName.js https://avd.aquasec.com/nvd/cve-2022-37599
	CVE-2022-37603					Regular expression denial of service https://avd.aquasec.com/nvd/cve-2022-37603
	CVE-2022-37599			3.2.0		regular expression denial of service in interpolateName.js https://avd.aquasec.com/nvd/cve-2022-37599
	CVE-2022-37603					Regular expression denial of service https://avd.aquasec.com/nvd/cve-2022-37603
minimatch	CVE-2022-3517			3.0.4	3.0.5	ReDoS via the braceExpand function https://avd.aquasec.com/nvd/cve-2022-3517

Create a Personal Access token for your Dockerhub account

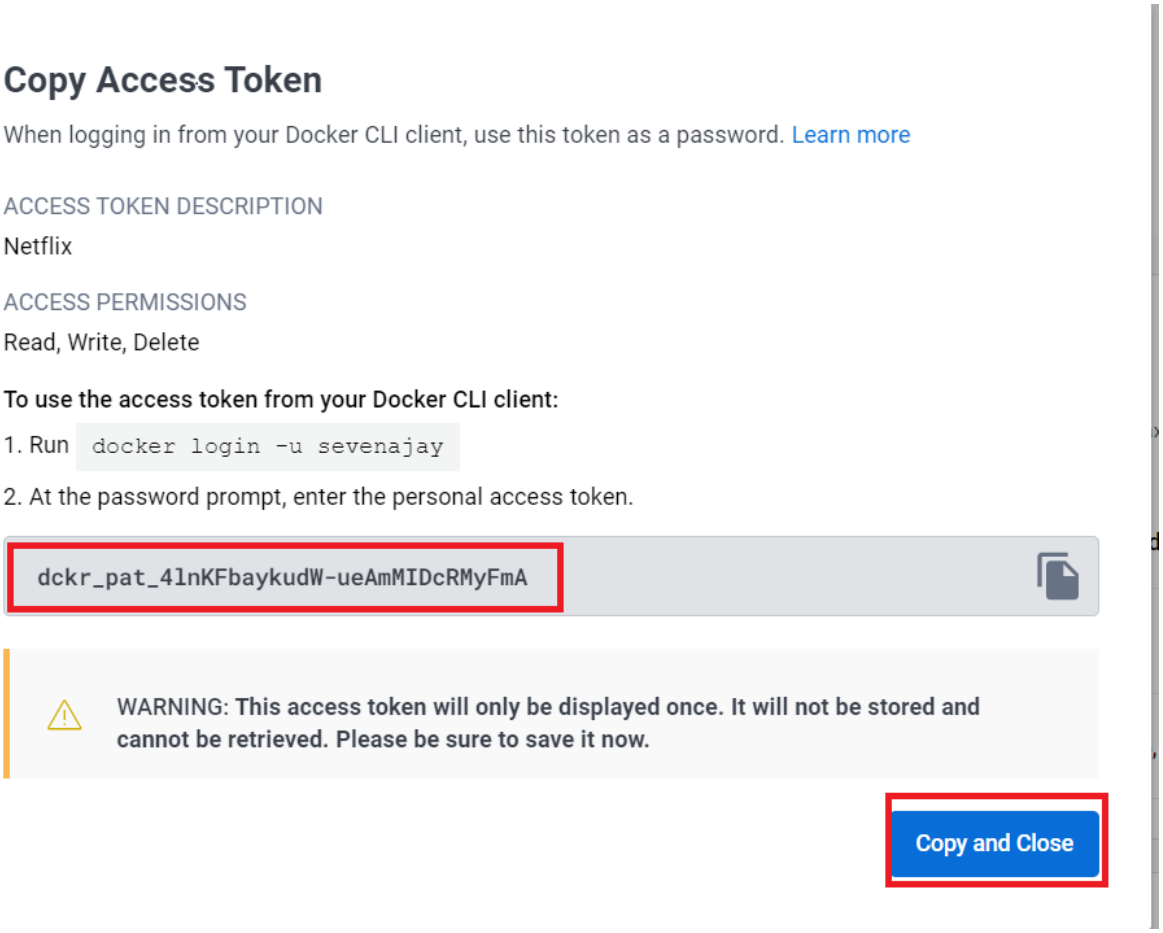
Go to docker hub and click on your profile –> Account settings –> security –> New access token



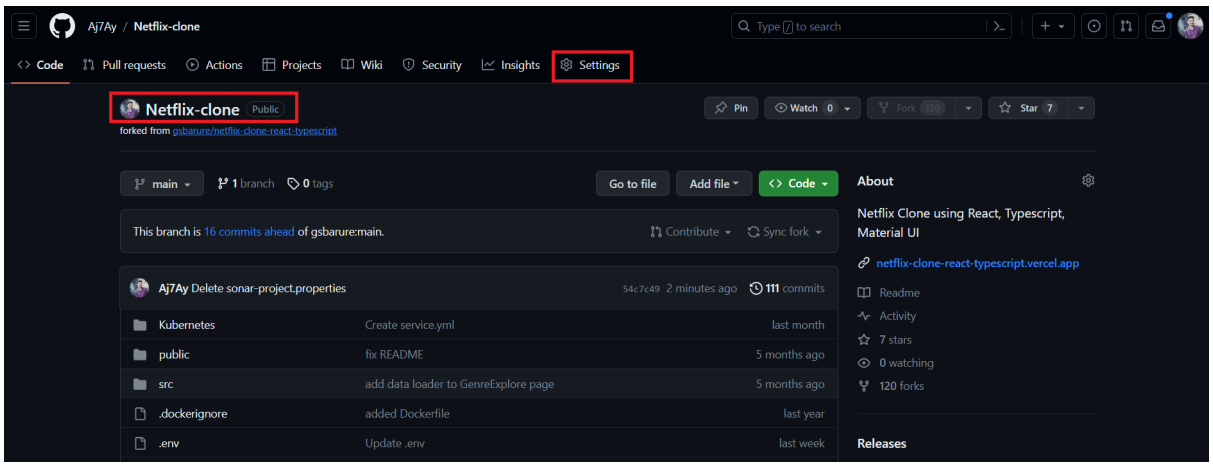
It asks for a name Provide a name and click on generate token



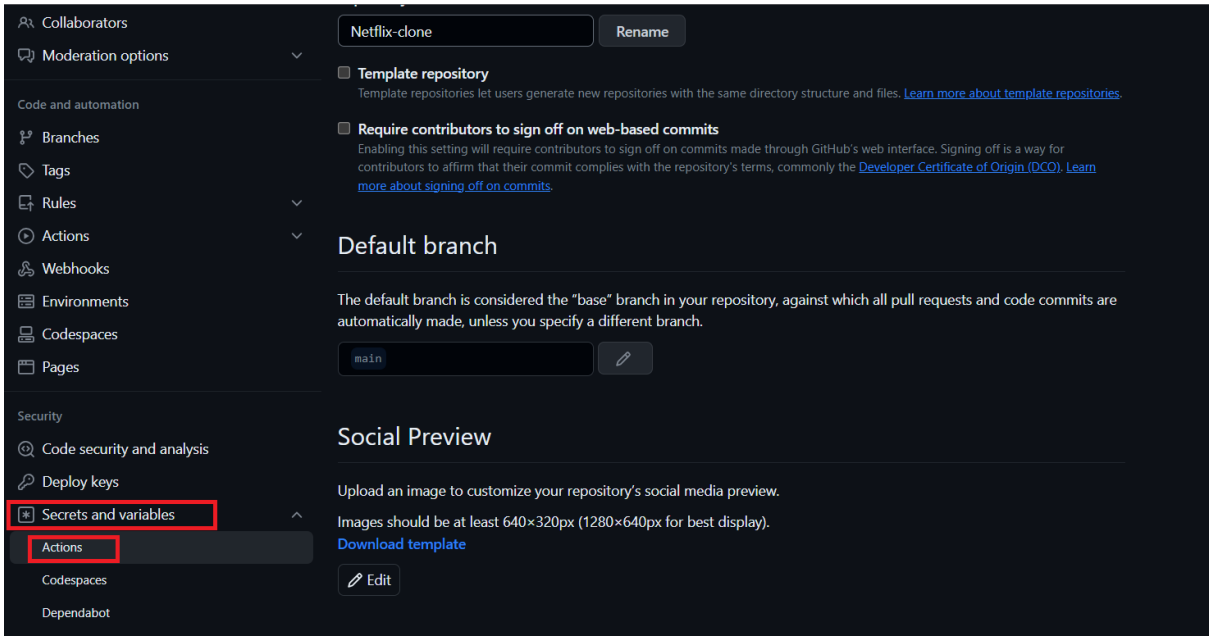
Copy the token save it in a safe place, and close



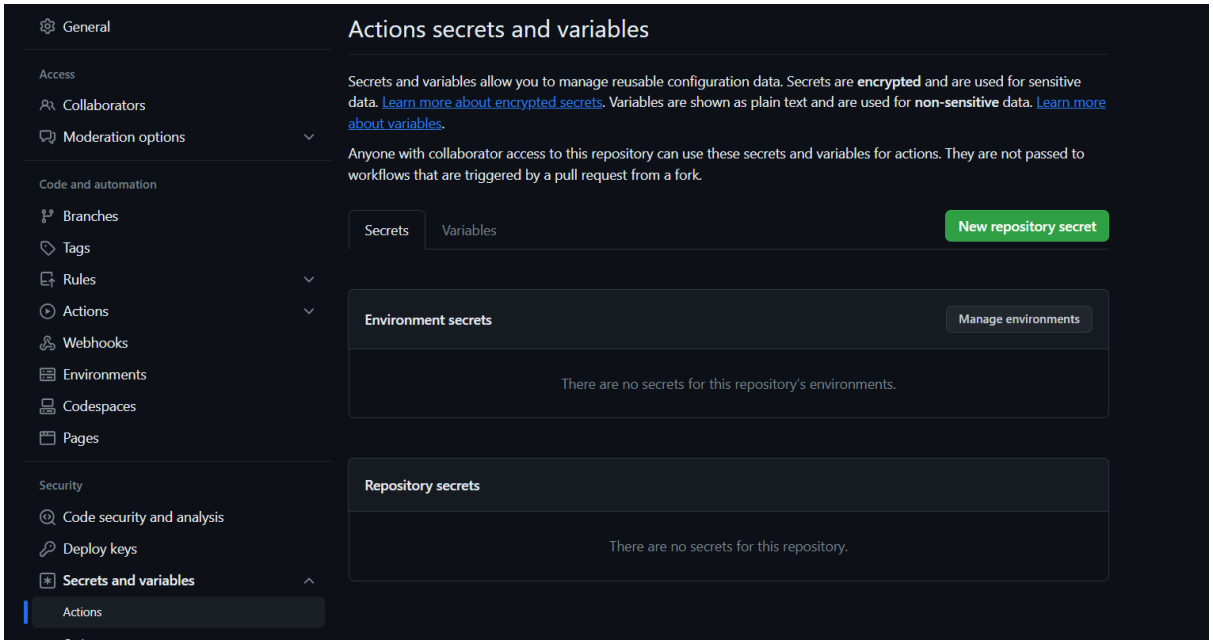
Now Go to GitHub again and click on settings



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

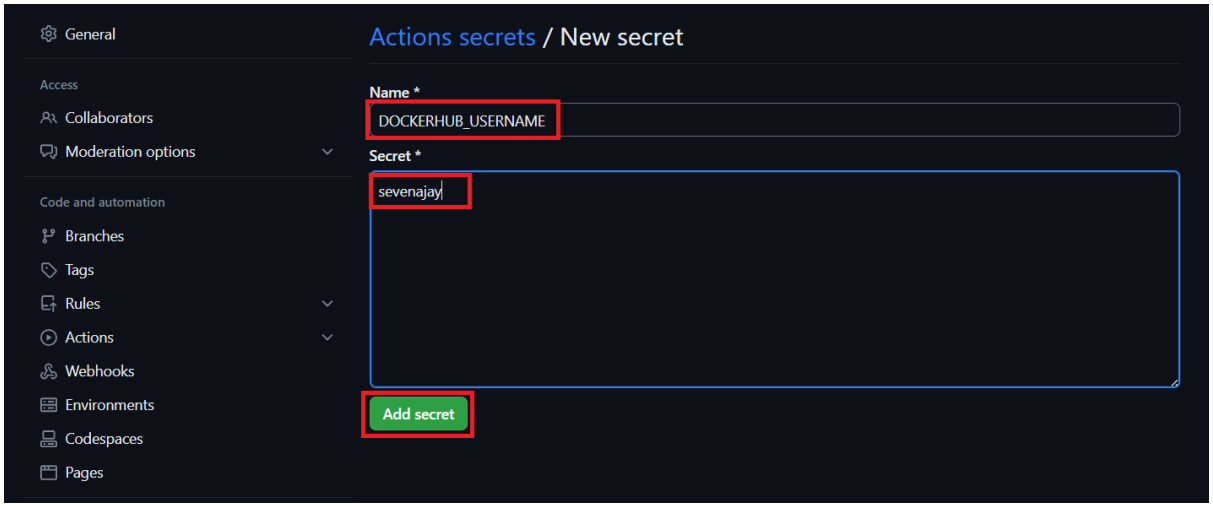


It will open a page like this click on New Repository secret



Add your Dockerhub username with the secret name as

DOCKERHUB_USERNAME #use your dockerhub username

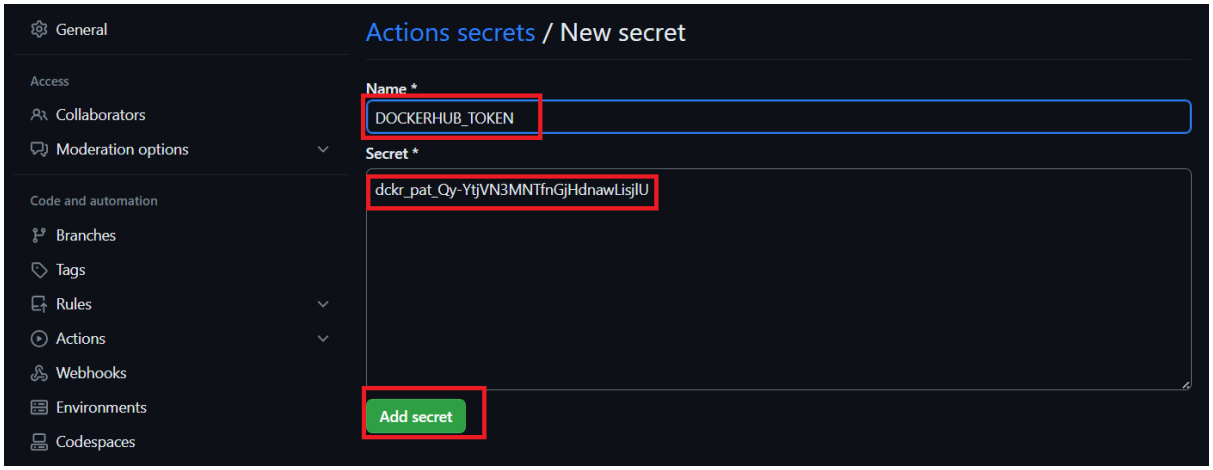


Click on Add Secret.

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

This step builds a Docker image with specific build arguments and tags it. It also logs in to Docker Hub using the provided credentials stored in secrets and pushes the Docker image.

If you run this job now you will get below output

```
Docker Build and push                                                    3m 30s
1  ▶ Run docker build -t tic-tac-toe .
9  DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
10             Install the buildx component to build images with BuildKit:
11             https://docs.docker.com/go/buildx/
13
15 Sending build context to Docker daemon  266.5MB
16
17 Step 1/8 : FROM node:16
18 16: Pulling from library/node
19 311da6c465ea: Pulling fs layer
20 7e9bf114588c: Pulling fs layer
21 ffd9397e94b7: Pulling fs layer
22 513d77925604: Pulling fs layer
23 ae3b95bbaa61: Pulling fs layer
24 0e421f66aff4: Pulling fs layer
25 ca266fd61921: Pulling fs layer
26 ee7d78be1eb9: Pulling fs layer
27 513d77925604: Waiting
28 ae3b95bbaa61: Waiting
29 0e421f66aff4: Waiting
30 ca266fd61921: Waiting
31 ee7d78be1eb9: Waiting
32 7e9bf114588c: Verifying Checksum
33 7e9bf114588c: Download complete
34 311da6c465ea: Verifying Checksum
35 311da6c465ea: Download complete
```

Image is pushed to Dockerhub

sevenajay / **tic-tac-toe**

Description

This repository does not have a description

Last pushed: a few seconds ago

Docker commands

Public View

To push a new tag to this repository:

```
docker push sevenajay/tic-tac-toe:tagname
```

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.

steps:

- name: Pull the Docker image

run: docker pull sevenajay/candycrush:latest

This step pulls the Docker image from Docker Hub, specified by sevenajay/tic-tac-toe:latest, which was built and pushed in the previous “build-analyze-scan” job

- name: Trivy image scan

run: trivy image sevenajay/candycrush:latest # Add Trivy scan command here

This step runs Trivy to scan the Docker image tagged as sevenajay/tic-tac-toe:latest. You should add the Trivy scan command here.

- name: Run the container

run: docker run -d --name ticgame -p 3000:3000 sevenajay/candycrush: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/tic-tac-toe:latest.

If you run this workflow.

Output

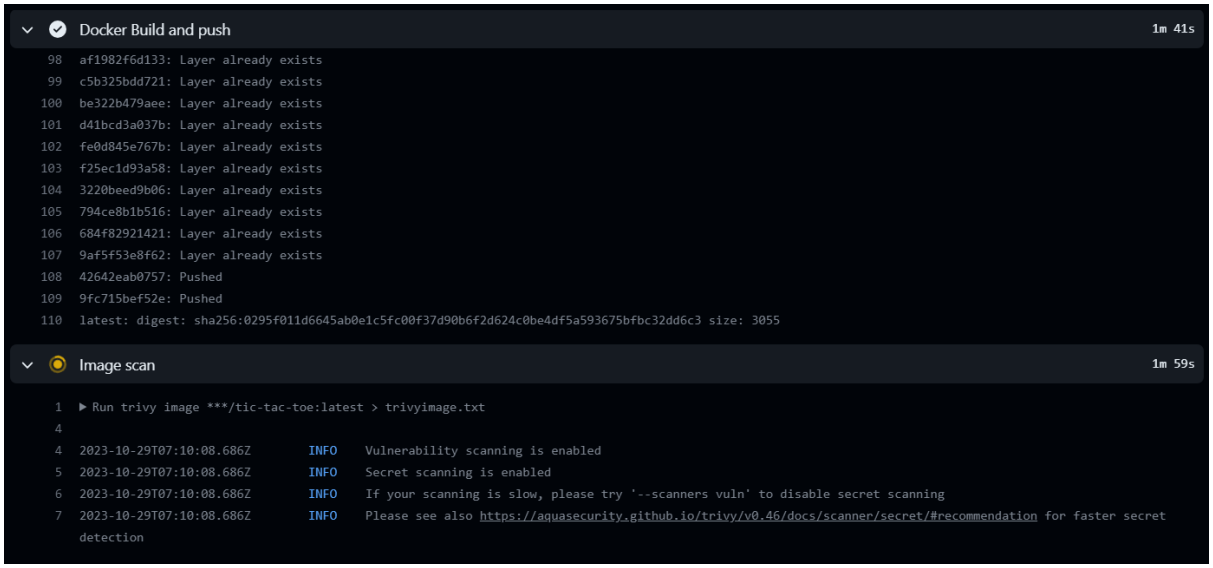


Image scan report

```
ubuntu@ip-172-31-36-122:~/actions-runner/_work/TIC-TAC-TOE/TIC-TAC-TOE$
ubuntu@ip-172-31-36-122:~/actions-runner/_work/TIC-TAC-TOE/TIC-TAC-TOE$
ubuntu@ip-172-31-36-122:~/actions-runner/_work/TIC-TAC-TOE/TIC-TAC-TOE$ cat trivyimage.txt

sevenajay/tic-tac-toe:latest (debian 10.13)
==========================================
Total: 1658 (UNKNOWN: 1, LOW: 1066, MEDIUM: 351, HIGH: 227, CRITICAL: 13)

┌────────────────────────┬────────────────┬──────────┬────────────┬───────────────────────┬───────────────────┬──────────────────────────────────────────────────────────────┐
│        Library         │ Vulnerability  │ Severity │   Status   │   Installed Version   │   Fixed Version   │                              Title                           │
├────────────────────────┼────────────────┼──────────┼────────────┼───────────────────────┼───────────────────┼──────────────────────────────────────────────────────────────┤
│ apt                    │ CVE-2011-3374  │ LOW      │ affected   │ 1.8.2.3               │                   │ It was found that apt-key in apt, all versions, do           │
│                        │                │          │            │                       │                   │ correctly...                                                 │
│                        │                │          │            │                       │                   │ https://avd.aquasec.com/nvd/cve-2011-3374                    │
├────────────────────────┼────────────────┼──────────┼────────────┼───────────────────────┼───────────────────┼──────────────────────────────────────────────────────────────┤
│ bash                   │ CVE-2019-18276 │          │            │ 5.0-4                 │                   │ when effective UID is not equal to its real UID th           │
│                        │                │          │            │                       │                   │ https://avd.aquasec.com/nvd/cve-2019-18276                    │
├────────────────────────┼────────────────┼──────────┼────────────┼───────────────────────┼───────────────────┼──────────────────────────────────────────────────────────────┤
│ binutils               │ CVE-2017-13716 │          │            │ 2.31.1-16             │                   │ binutils: Memory leak with the C++ symbol demangle           │
│                        │                │          │            │                       │                   │ in libiberty                                                 │
│                        │                │          │            │                       │                   │ https://avd.aquasec.com/nvd/cve-2017-13716                    │
│                        ├────────────────┼──────────┼────────────┼───────────────────────┼───────────────────┼──────────────────────────────────────────────────────────────┤
│                        │ CVE-2018-1000876 │        │            │                       │                   │ integer overflow leads to heap-based buffer overflow in      │
│                        │                │          │            │                       │                   │ objdump                                                      │
│                        │                │          │            │                       │                   │ https://avd.aquasec.com/nvd/cve-2018-1000876                  │
```

deploy
succeeded 2 minutes ago in 1m 51s

Search logs

Set up job 0s

docker pull image 3s

```
1 ▶ Run docker pull sevenajay/tic-tac-toe:latest
4 latest: Pulling from sevenajay/tic-tac-toe
5 Digest: sha256:6a293a885184298ae304db17a4ba827f2ff0a12f8267c14f6cdef400ed349059
6 Status: Image is up to date for sevenajay/tic-tac-toe:latest
7 docker.io/sevenajay/tic-tac-toe:latest
```

Image scan 1m 42s

Deploy to container 1s

```
1 ▶ Run docker run -d --name game -p 3000:3000 sevenajay/tic-tac-toe:latest
4 529004a51c3519f99d6017a007df1b9938b77b1d3e26e22a51f69aaeae3a2a0b
```

Complete job 0s

Deployed to the container.

output

ec2-ip:3000

Deploy to EKS

- name: Update kubeconfig

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

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.

SLACK

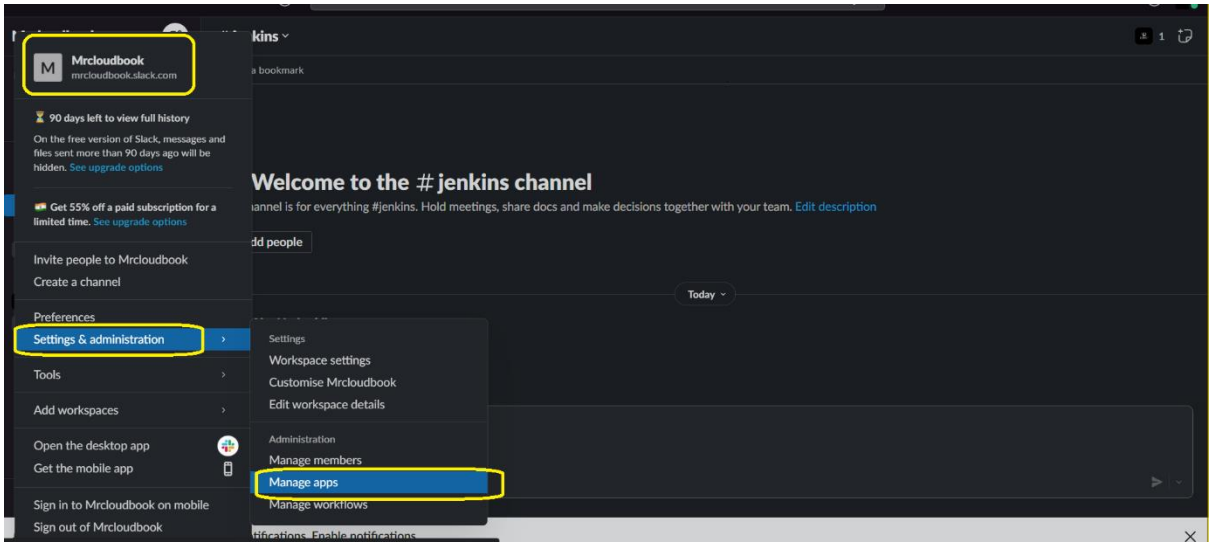
Go to your Slack channel, if you don't have create one

Go to Slack channel and create a channel for notifications

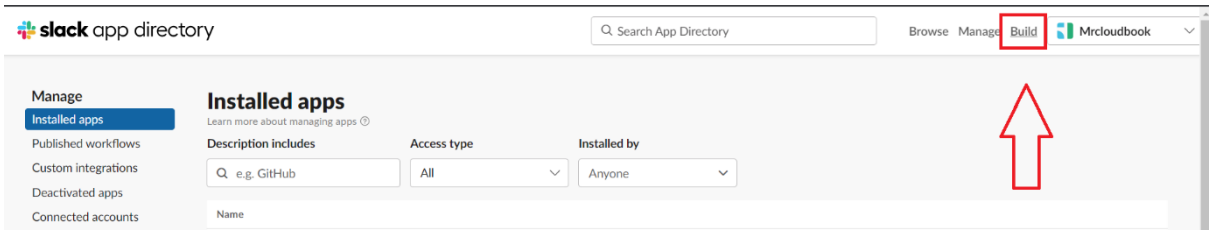
click on your name

Select Settings and Administration

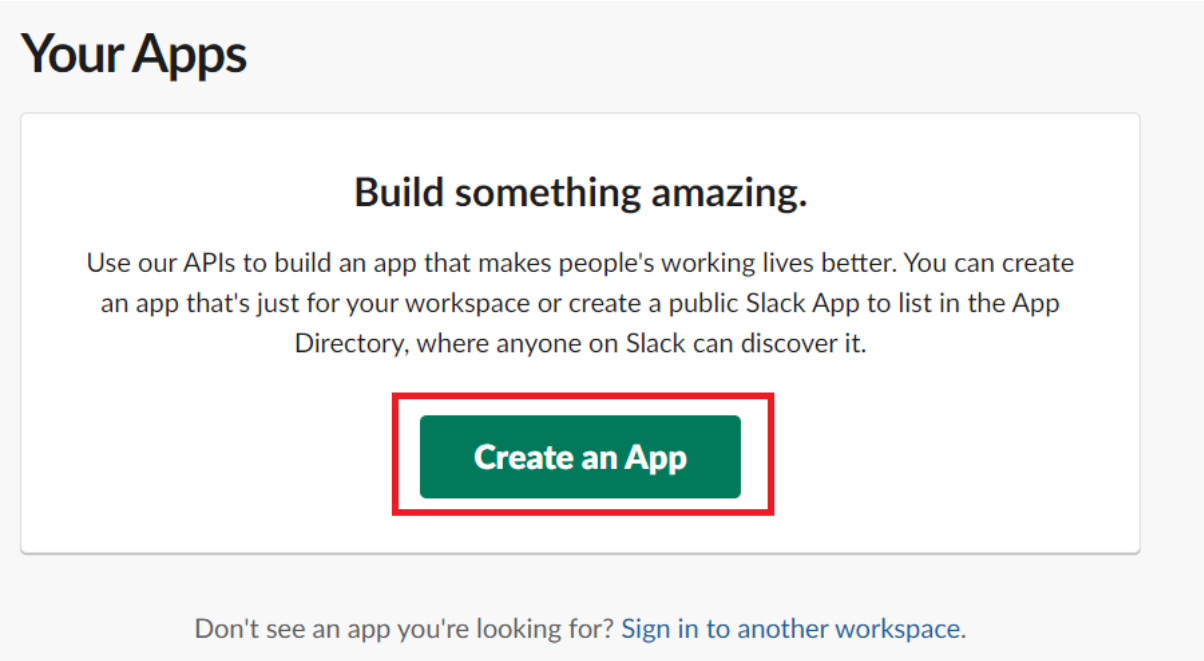
Click on Manage apps



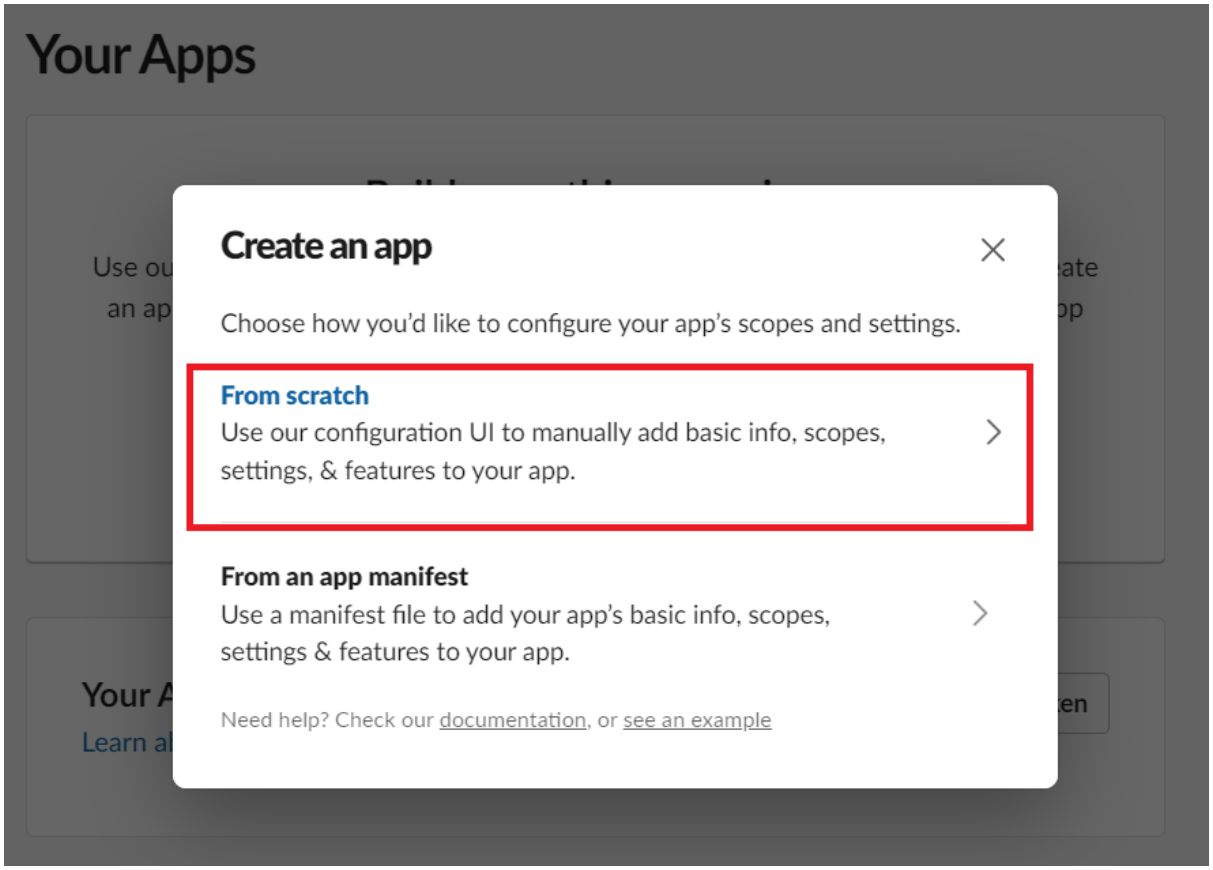
It will open a new tab, select build now



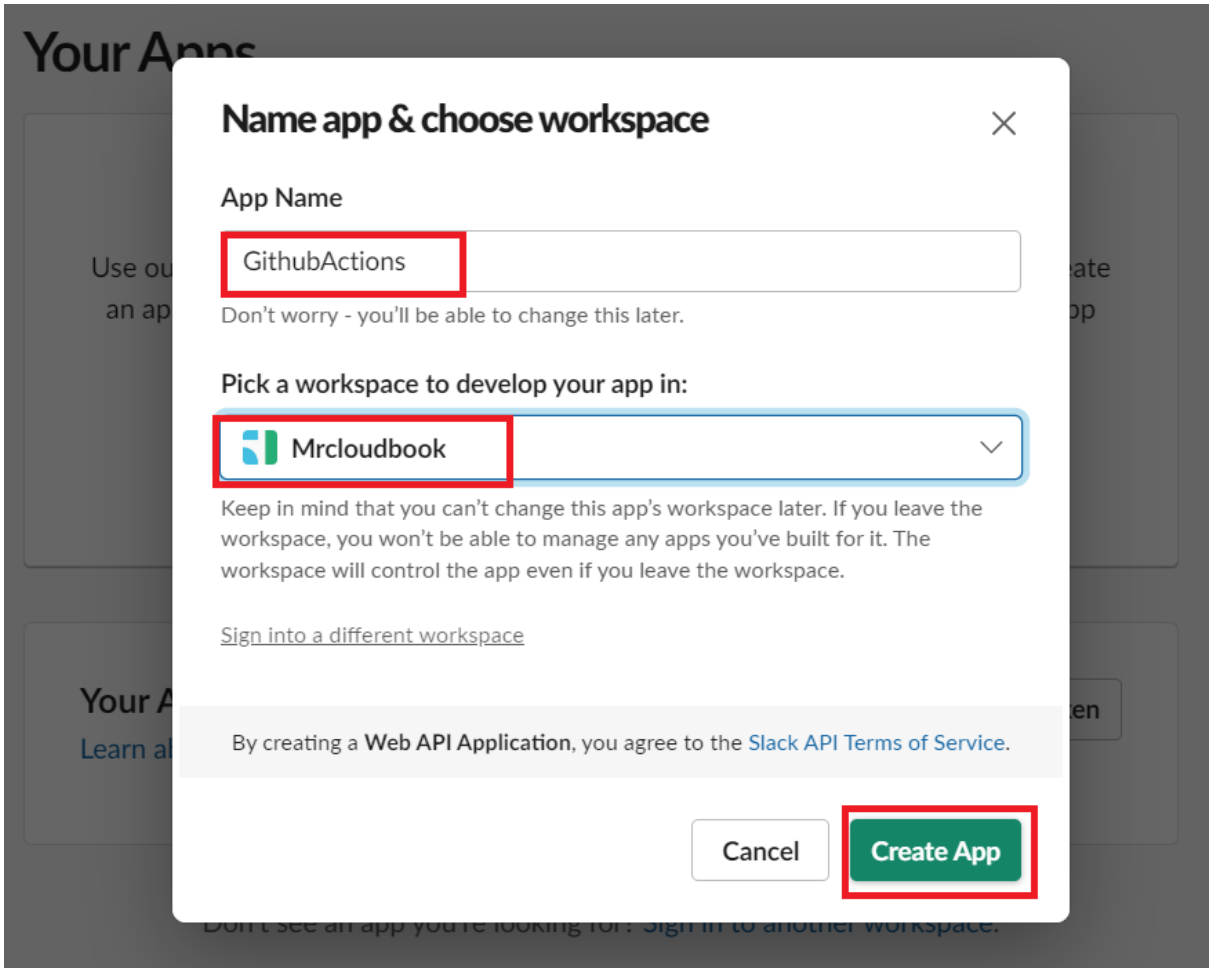
Click on create an app



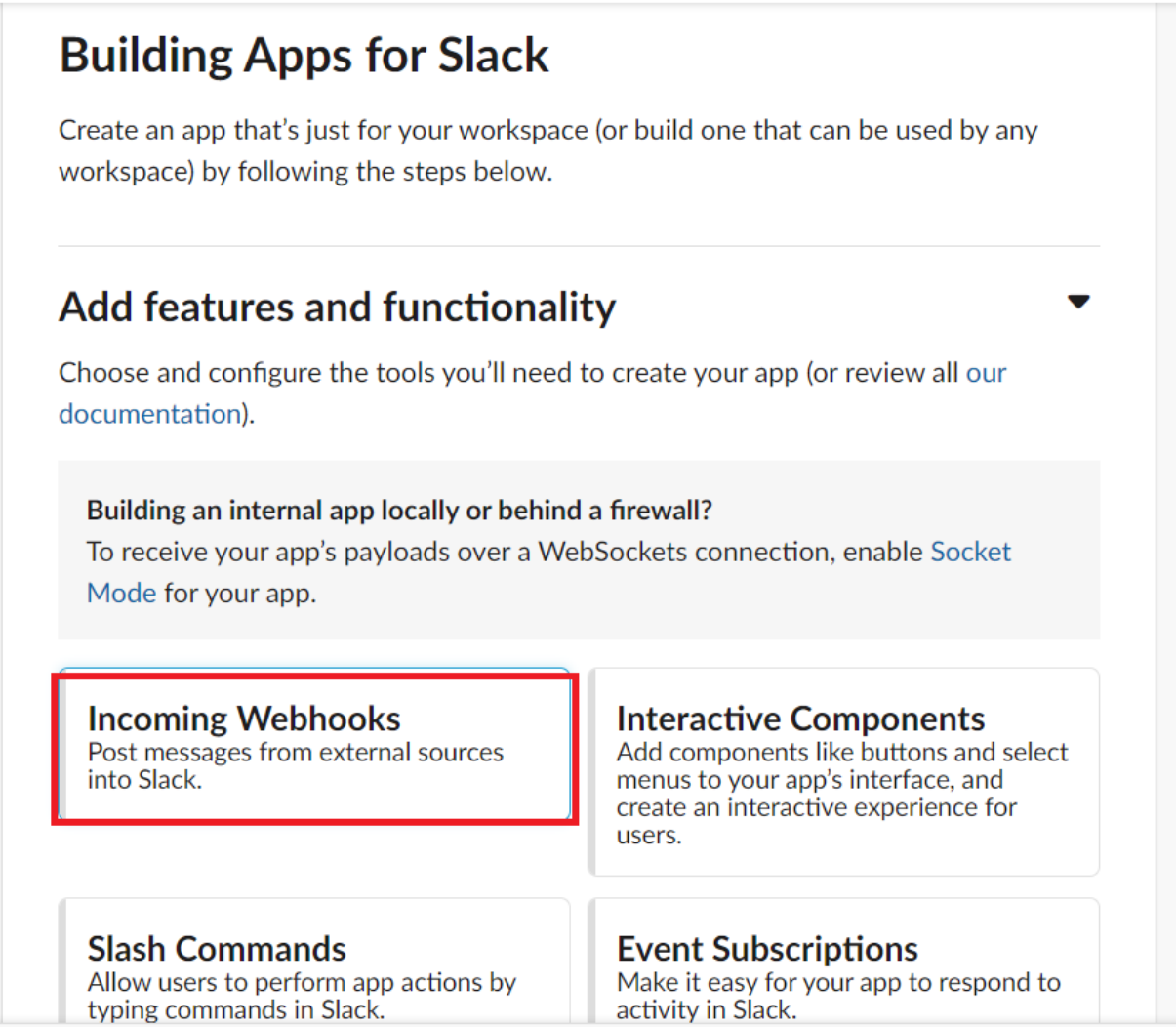
Select from scratch



Provide a name for the app and select workspace and create



Select Incoming webhooks



Set incoming webhooks to on

Incoming Webhooks

Activate Incoming Webhooks

On

Incoming webhooks are a simple way to post messages from external sources into Slack. They make use of normal HTTP requests with a JSON payload, which includes the message and a few other optional details. You can include message attachments to display richly-formatted messages.

Adding incoming webhooks requires a bot user. If your app doesn't have a bot user, we'll add one for you.

Each time your app is installed, a new Webhook URL will be generated.

If you deactivate incoming webhooks, new Webhook URLs will not be generated when your app is installed to your team. If you'd like to remove access to existing Webhook URLs, you will need to Revoke All OAuth Tokens.

Click on Add New webhook to workspace

Webhook URLs for Your Workspace

To dispatch messages with your webhook URL, send your message in JSON as the body of an `application/json` POST request.

Add this webhook to your workspace below to activate this curl example.

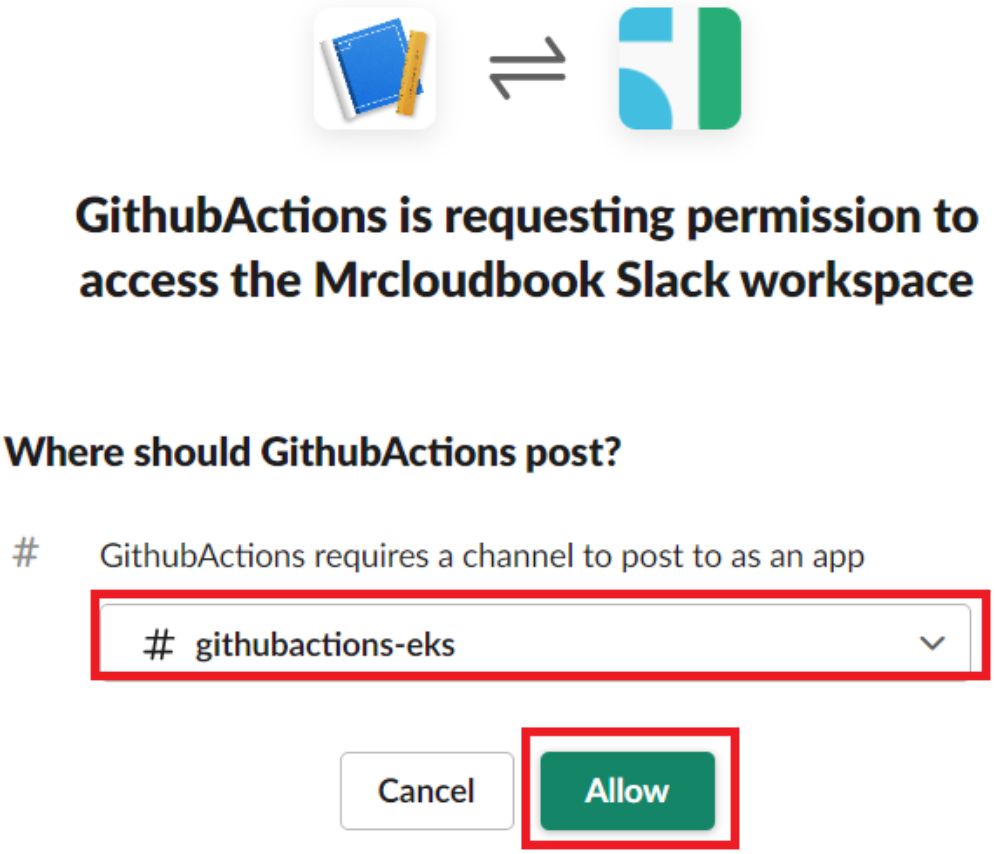
Sample curl request to post to a channel:

```
curl -X POST -H 'Content-type: application/json' --data '{"text":"Hello,
World!"}' YOUR_WEBHOOK_URL_HERE
```

Webhook URL	Channel	Added By
No webhooks have been added yet.		

Add New Webhook to Workspace

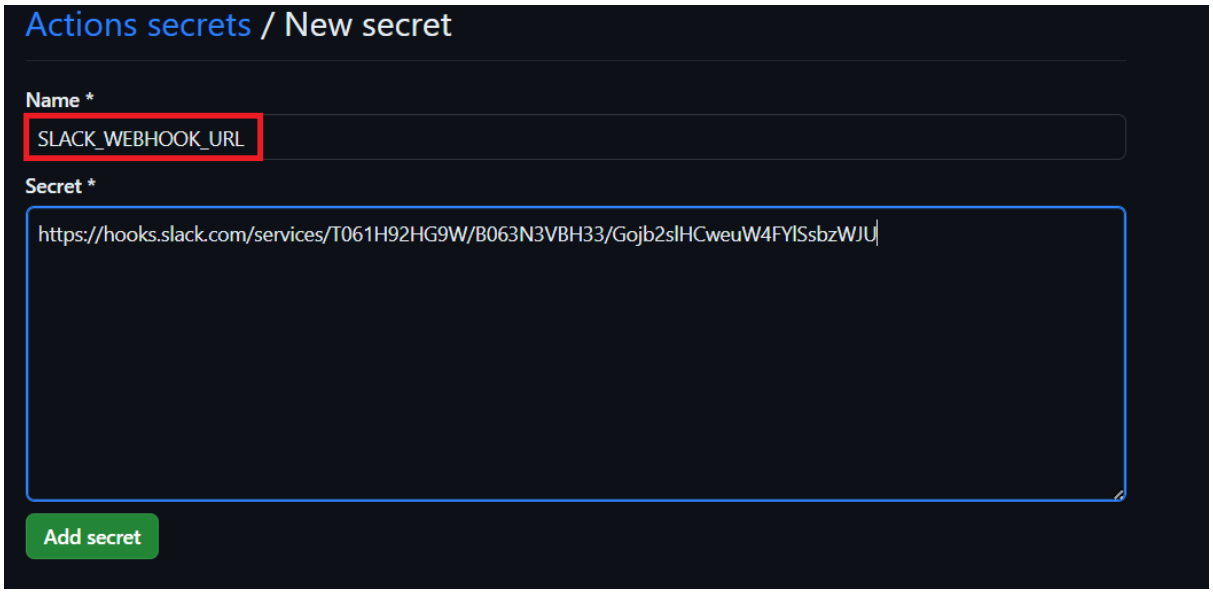
Select Your channel that created for notifications and allow



It will generate a webhook URL copy it

Now come back to GitHub and click on settings

Go to secrets –> actions –> new repository secret and add



Add the below code to the workflow and commit and the workflow will start.

```
- name: Send a Slack Notification
    if: always()
    uses: act10ns/slack@v1
    with:
      status: ${{ job.status }}
      steps: ${{ toJson(steps) }}
      channel: '#git'
    env:
      SLACK_WEBHOOK_URL: ${{ secrets.SLACK_WEBHOOK_URL }}
```

This step sends a Slack notification. It uses the act10ns/slack action and is configured to run “always,” which means it runs regardless of the job status. It sends the notification to the specified Slack channel using the webhook URL stored in secrets.

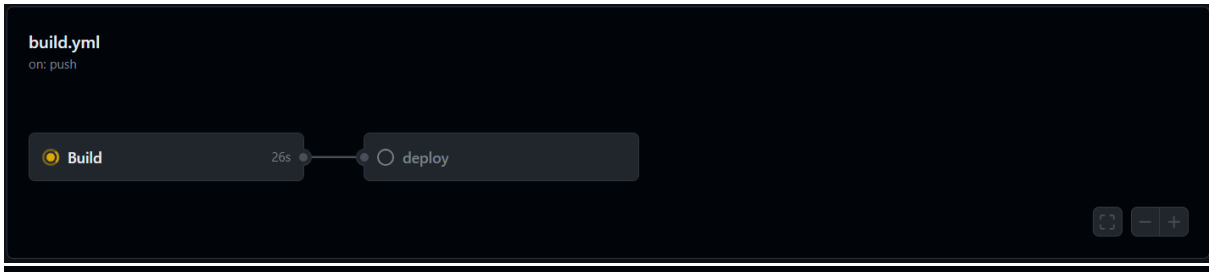
Complete Workflow

```
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 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 dependency
      run: npm install
    - name: Trivy file scan
      run: trivy fs . > trivyfs.txt
    - name: Docker Build and push
      run: |
        docker build -t candycrush .
        docker tag candycrush sevenajay/candycrush:latest
        docker login -u ${{ secrets.DOCKERHUB_USERNAME }} -p ${{ secrets.DOCKERHUB_TOKEN }}
        docker push sevenajay/candycrush:latest
      env:
        DOCKER_CLI_ACI: 1
    - name: Image scan
      run: trivy image sevenajay/candycrush:latest > trivyimage.txt
deploy:
 needs: build-analyze-scan
 runs-on: [self-hosted]
 steps:
    - name: docker pull image
      run: docker pull sevenajay/candycrush:latest
    - name: Image scan
      run: trivy image sevenajay/candycrush:latest > trivyimagedeploy.txt
    - name: Deploy to container
      run: docker run -d --name game -p 3000:3000 sevenajay/candycrush:latest
    - name: Update kubeconfig
      run: aws eks --region ap-south-1 update-kubeconfig --name EKS_CLOUD
    - name: Deploy to kubernetes
      run: kubectl apply -f deployment-service.yml
    - name: Send a Slack Notification
      if: always()
      uses: act10ns/slack@v1
```

```
    with:
      status: ${{ job.status }}
      steps: ${{ toJson(steps) }}
      channel: '#githubactions-eks'
    env:
      SLACK_WEBHOOK_URL: ${{ secrets.SLACK_WEBHOOK_URL }}
```

Run this workflow now



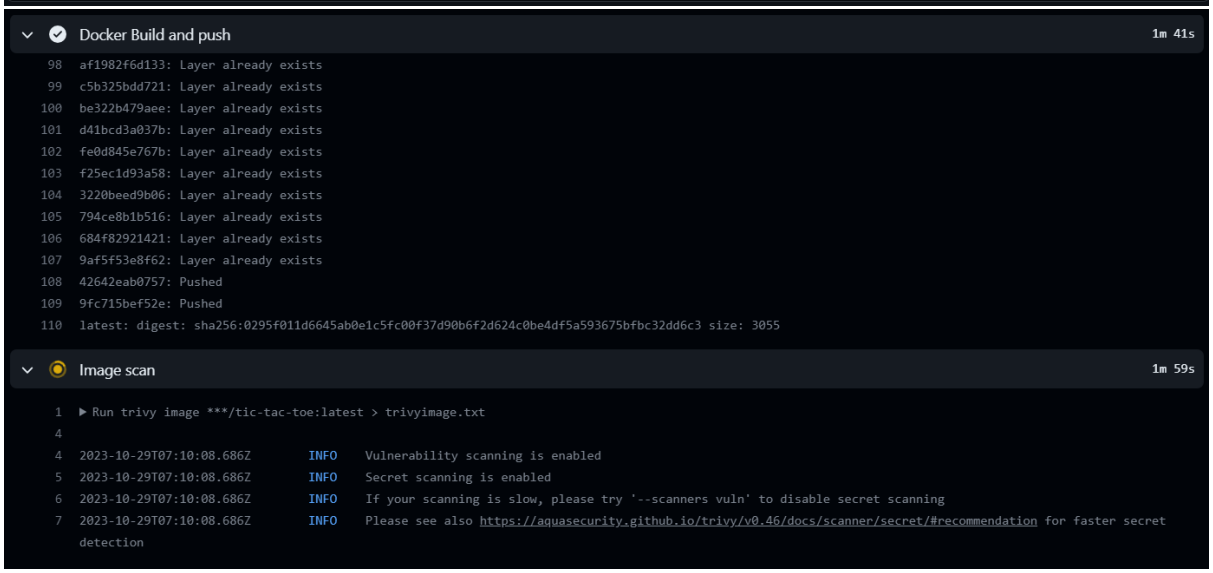


Image scan report

```
ubuntu@ip-172-31-36-122:~/actions-runner/_work/TIC-TAC-TOE/TIC-TAC-TOE$
ubuntu@ip-172-31-36-122:~/actions-runner/_work/TIC-TAC-TOE/TIC-TAC-TOE$
ubuntu@ip-172-31-36-122:~/actions-runner/_work/TIC-TAC-TOE/TIC-TAC-TOE$ cat trivyimage.txt

sevenajay/tic-tac-toe:latest (debian 10.13)
==========================================
Total: 1658 (UNKNOWN: 1, LOW: 1066, MEDIUM: 351, HIGH: 227, CRITICAL: 13)

┌────────────────────────┬────────────────┬──────────┬────────────┬───────────────────────┬──────────────────┬──────────────────────────────────────────────────────────────┐
│        Library         │ Vulnerability  │ Severity │   Status   │   Installed Version   │   Fixed Version  │                              Title                           │
├────────────────────────┼────────────────┼──────────┼────────────┼───────────────────────┼──────────────────┼──────────────────────────────────────────────────────────────┤
│ apt                    │ CVE-2011-3374  │ LOW      │ affected   │ 1.8.2.3               │                  │ It was found that apt-key in apt, all versions, do           │
│                        │                │          │            │                       │                  │ correctly...                                                 │
│                        │                │          │            │                       │                  │ https://avd.aquasec.com/nvd/cve-2011-3374                    │
├────────────────────────┼────────────────┼──────────┼────────────┼───────────────────────┼──────────────────┼──────────────────────────────────────────────────────────────┤
│ bash                   │ CVE-2019-18276 │          │            │ 5.0-4                 │                  │ when effective UID is not equal to its real UID th           │
│                        │                │          │            │                       │                  │ https://avd.aquasec.com/nvd/cve-2019-18276                    │
├────────────────────────┼────────────────┼──────────┼────────────┼───────────────────────┼──────────────────┼──────────────────────────────────────────────────────────────┤
│ binutils               │ CVE-2017-13716 │          │            │ 2.31.1-16             │                  │ binutils: Memory leak with the C++ symbol demangle           │
│                        │                │          │            │                       │                  │ in libiberty                                                 │
│                        │                │          │            │                       │                  │ https://avd.aquasec.com/nvd/cve-2017-13716                    │
│                        ├────────────────┼──────────┼────────────┼───────────────────────┼──────────────────┼──────────────────────────────────────────────────────────────┤
│                        │ CVE-2018-1000876 │        │            │                       │                  │ integer overflow leads to heap-based buffer overflow in      │
│                        │                │          │            │                       │                  │ objdump                                                      │
│                        │                │          │            │                       │                  │ https://avd.aquasec.com/nvd/cve-2018-1000876                  │
```

deploy
succeeded 2 minutes ago in 1m 51s

Search logs

Set up job 0s

docker pull image 3s

```
1 ▶ Run docker pull sevenajay/tic-tac-toe:latest
4 latest: Pulling from sevenajay/tic-tac-toe
5 Digest: sha256:6a293a885184298ae304db17a4ba827f2ff0a12f8267c14f6cdef400ed349059
6 Status: Image is up to date for sevenajay/tic-tac-toe:latest
7 docker.io/sevenajay/tic-tac-toe:latest
```

Image scan 1m 42s

Deploy to container 1s

```
1 ▶ Run docker run -d --name game -p 3000:3000 sevenajay/tic-tac-toe:latest
4 529004a51c3519f99d6017a007df1b9938b77b1d3e26e22a51f69aaeae3a2a0b
```

Complete job 0s

Deployed to the container.

Deployed to EKS

```
Deploy to container                                                    0s
1  ▶ Run docker run -d --name game -p 3000:3000 sevenajay/tic-tac-toe:latest
4  5d3ad75d034b56ec3ed230bb801b934d7e5902711bffc8868d6614f3c165d187

Update kubeconfig                                                      0s

Deploy to kubernetes                                                   1s
1  ▶ Run kubectl apply -f deployment-service.yml
4
4  deployment.apps/tic-tac-toe created
5  service/tic-tac-toe-service created

Send a Slack Notification                                              1s
1  ▶ Run act10ns/slack@v1
9
9  Reading config file .github/slack.yml...
10 Sent deploy status of SUCCESS to Slack!

Complete job                                                           0s
```

```
Triggered via push 12 minutes ago        Status      Total duration      Artifacts
Aj7Ay pushed  c18d5c7  main              Success     7m 13s              –

deployed.yml
on: push

Build   4m 33s ——— deploy   1m 59s
```

Job completed.

Let’s go to the Ec2 ssh connection

Provide this command

kubectl get all

```
YfMUW9R0OA6qn-5q
ubuntu@ip-172-31-11-71:~$ kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/tetris-698657b8dc-4nqmb         1/1     Running   0          77s
pod/tetris-698657b8dc-shd2x         1/1     Running   0          77s
pod/tetris-698657b8dc-tktwq         1/1     Running   0          77s

NAME                    TYPE           CLUSTER-IP      EXTERNAL-IP                                                          PORT(S)        AGE
service/kubernetes      ClusterIP      10.100.0.1      <none>                                                               443/TCP        123m
service/tetris-service  LoadBalancer   10.100.212.20   a0725d5c76f3e47a8af3c7391c061307-244742789.ap-south-1.elb.amazonaws.com  80:31849/TCP   77s

NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/tetris  3/3     3            3           77s

NAME                               DESIRED   CURRENT   READY   AGE
replicaset.apps/tetris-698657b8dc  3         3         3       77s
ubuntu@ip-172-31-11-71:~$
```

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

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

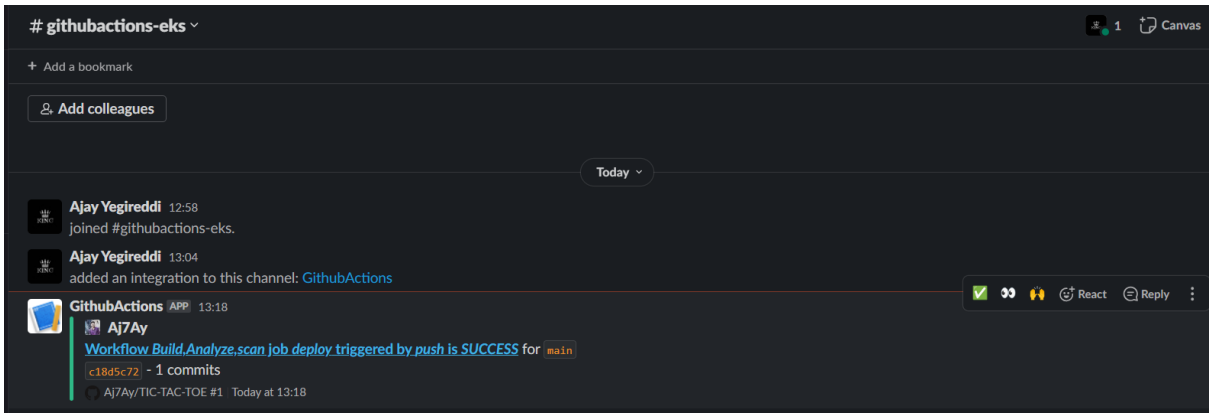
output

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 better relevancy of analysis
      - name: Deploy to container
        run: |
          docker stop game
          docker rm game
```

```
  - name: Update kubeconfig
    run: aws eks --region ap-south-1 update-kubeconfig --name EKS_CLOUD
  - name: Deploy to kubernetes
    run: kubectl delete -f deployment-service.yml
  - name: Send a Slack Notification
    if: always()
    uses: act10ns/slack@v1
    with:
      status: ${{ job.status }}
      steps: ${{ toJson(steps) }}
      channel: '#githubactions-eks'
    env:
      SLACK_WEBHOOK_URL: ${{ secrets.SLACK_WEBHOOK_URL }}
```

Slack Notification



It will delete the container and delete the Kubernetes deployment.

Stop the self-hosted runner.

Now go inside the candycrush

To delete the Eks cluster

cd /home/ubuntu

cd Candycrush

cd Eks-terraform

terraform destroy --auto-approve

It will take 10 minutes to destroy the EKS cluster

Meanwhile, delete the Dockerhub Token

Once cluster destroys

Delete The ec2 instance and IAM role.

Delete the secrets from GitHub also.