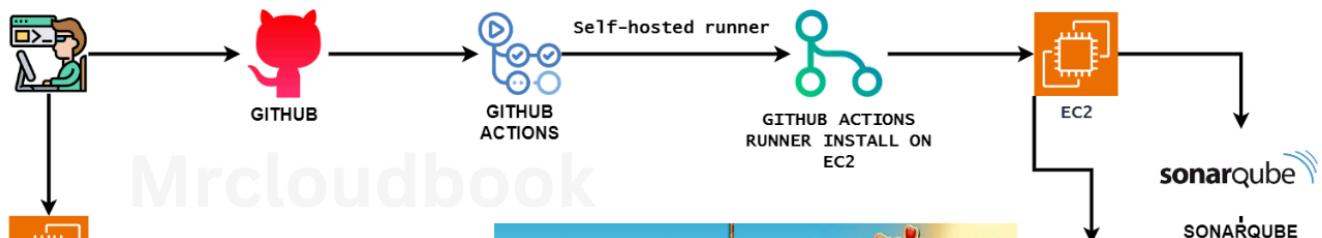


DevOps

# Candycrush Deployment on AWS EKS using GitHub Actions in DevSecOps Pipeline

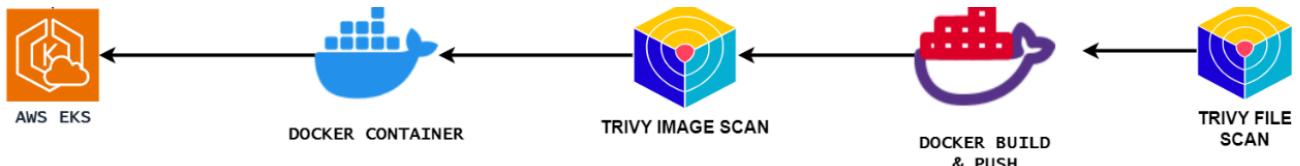


mrcloudbook.com · 31 December 2023



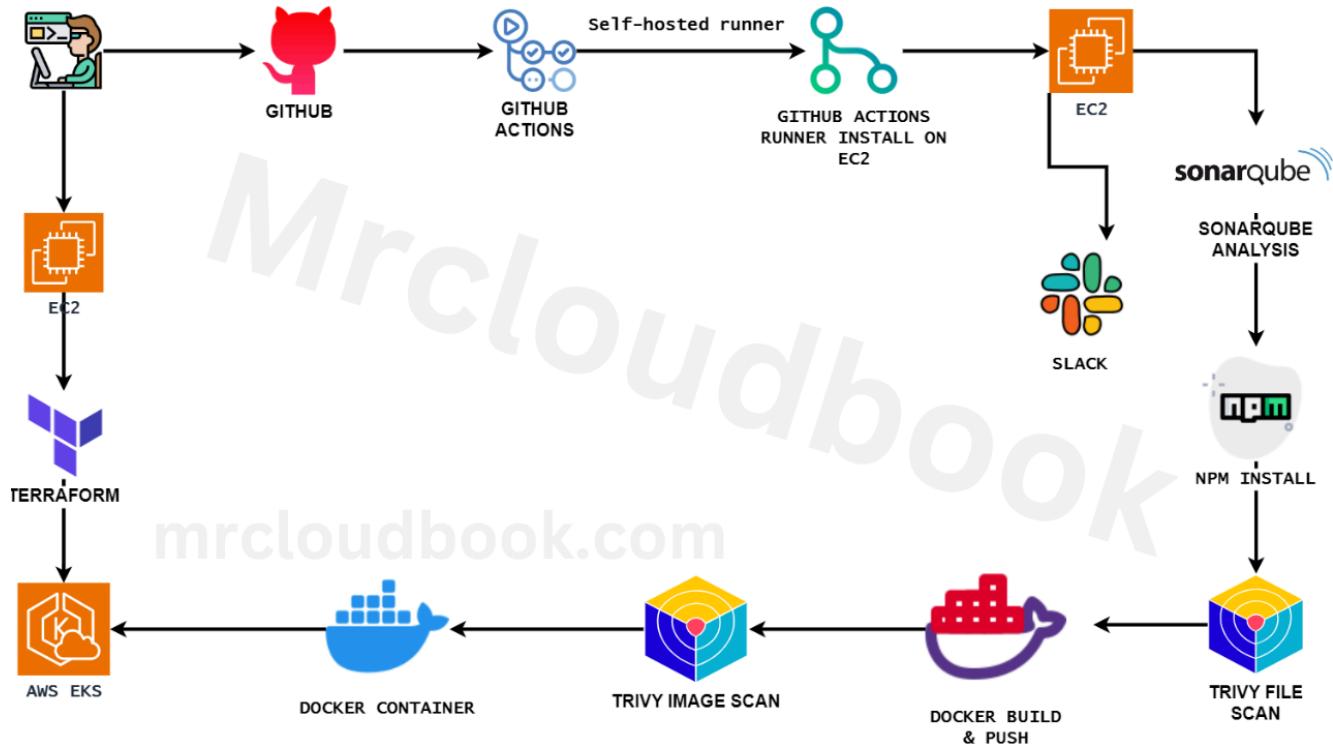
Mr Cloud Book

[Home](#) [Blog](#) [DevSecOps](#) [Contact](#) [About Me](#) [Testimonials](#)



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.



IF YOU HAVE DOUBTS HERE IS SIMILAR VIDEO

[https://youtu.be/FghkK2hFd1k?si=6HamuBu\\_IJPCbsqU](https://youtu.be/FghkK2hFd1k?si=6HamuBu_IJPCbsqU)

GitHub: <https://github.com/Aj7Ay/Candycrush.git>

EKS Video: <https://youtu.be/5-PZnYaoZUM>

## Contents [ hide ]

[Step1A: Launch an Ec2 Instance](#)

[Create an IAM ROLE](#)

[Step1B: Add a self-hosted runner to Ec2](#)

[Step2A: Install Docker and Run Sonarqube Container](#)

[Step2B: Integrating SonarQube with GitHub Actions](#)

[Step2C: INSTALLATION OF OTHER TOOLS](#)

[EKS provision](#)

[Deploy to EKS](#)

[SLACK](#)

[Complete Workflow](#)

[Destruction workflow](#)

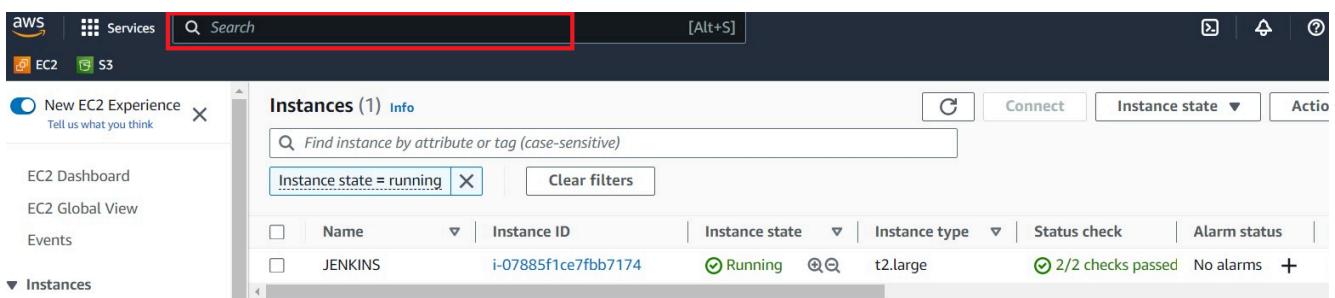
## Step1A: Launch an Ec2 Instance

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”

The screenshot shows the AWS IAM Roles page. On the left, there's a sidebar with 'Identity and Access Management (IAM)' selected. Under 'Access management', 'Roles' is highlighted with a red box. The main area shows a table of roles with columns for Role name, Trusted entities, and Last activity. Three roles are listed: 'AWSServiceRoleForSupport' (AWS Service: support), 'AWSServiceRoleForTrustedAdvisor' (AWS Service: trustedadvisor), and 'EC2' (AWS Service: ec2, last active 1 hour ago). At the top right of the table, there's a 'Create role' button, which is also highlighted with a red box.

Click “AWS service”

Click “Choose a service or use case”

The screenshot shows the 'Select trusted entity' step of a role creation wizard. On the left, there's a sidebar with 'Step 2: Add permissions' and 'Step 3: Name, review, and create'. The main area has a 'Trusted entity type' section with five options: 'AWS service' (selected and highlighted with a red box), 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy'. Below this is a 'Use case' section with a dropdown menu labeled 'Choose a service or use case', which is also highlighted with a red box.

Click “EC2”

Click “Next”

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

EC2

Choose a use case for the specified service.

Use case

**EC2**  
Allows EC2 instances to call AWS services on your behalf.

**EC2 Role for AWS Systems Manager**  
Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.

**EC2 Spot Fleet Role**  
Allows EC2 Spot Fleet to request and terminate Spot instances on your behalf.

**EC2 - Spot Fleet Auto Scaling**  
Allows Auto Scaling to access and update EC2 spot fleets on your behalf.

**EC2 - Spot Fleet Tagging**  
Allows EC2 to launch spot instances and attach tags to the launched instances on your behalf.

**EC2 - Spot Instances**  
Allows EC2 Spot Instances to launch and manage spot instances on your behalf.

**EC2 - Spot Fleet**  
Allows EC2 Spot Fleet to launch and manage spot fleet instances on your behalf.

**EC2 - Scheduled Instances**  
Allows EC2 Scheduled Instances to manage instances on your behalf.

Cancel **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)

**Step 2: Add permissions**

Permissions policy summary		
Policy name	Type	Attached as
AmazonDynamoDBFullAccess	AWS managed	Permissions policy
AmazonEC2FullAccess	AWS managed	Permissions policy
AmazonS3FullAccess	AWS managed	Permissions policy

**Step 3: Add tags**

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

No tags associated with the resource.

[Add new tag](#)  
You can add up to 50 more tags.

Cancel [Previous](#) **Create role**

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.

[EC2](#) > [Instances](#) > [i-07885f1ce7fbb7174](#) > [Modify IAM role](#)

### Modify IAM role [Info](#)

Attach an IAM role to your instance.

Instance ID  
 i-07885f1ce7fbb7174 (JENKINS)

IAM role  
Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.  
 Jenkins-cicd [▼](#) [Create new IAM role](#)

Cancel **Update IAM role**

## Step1B: Add a self-hosted runner to Ec2

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

The screenshot shows the GitHub repository settings for 'Netflix-clone'. The 'Actions' tab is highlighted with a red box. Under the 'Default branch' section, there is a 'main' branch listed.

Click on New self-hosted runner

The screenshot shows the GitHub repository settings for 'Netflix-clone'. The 'Runners' tab is highlighted with a red box. A green button labeled 'New self-hosted runner' is also highlighted with a red box.

Now select Linux and Architecture X64

The screenshot shows the 'Runners / Add new self-hosted runner' configuration page for the 'Aj7Ay/Netflix-clone' repository. It displays the 'Runner image' (Linux selected) and 'Architecture' (x64 selected) dropdown menus, both highlighted with red boxes.

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 "fb28a1c3715e0a6c5051af0e6eff9c255009e2eec6fb08bc2708277fbb49f93" 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/ac
```



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$ 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  
0.2.tar.gz  
% Total    % Received % Xferd  Average Speed   Time     Time  Current  
          Dload  Upload Total Spent   Left Speed  
0     0    0     0      0      0 0:00:02  0:00:02      0 119M  
100 178M 100 178M 0     0 82.7M 0 0:00:02 0:00:02      0 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 "fb28a1c3715e0a6c5051af0e6eff9c255009e2eec6fb08bc2708277fbb49f93
```



```
ubuntu@ip-172-31-32-28:~/actions-runner$  
ubuntu@ip-172-31-32-28:~/actions-runner$ echo "fb28a1c3715e0a6c5051af0e6eff9c255009e2eec6fb08bc2708277fbb49f93" | 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 A2MXW45MNGB3QV6SJ6D5LWTGCRPW
```



```
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 A2MXW45MNGB3QV6SJ6D5LWTGCRPW  
-----  
          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).

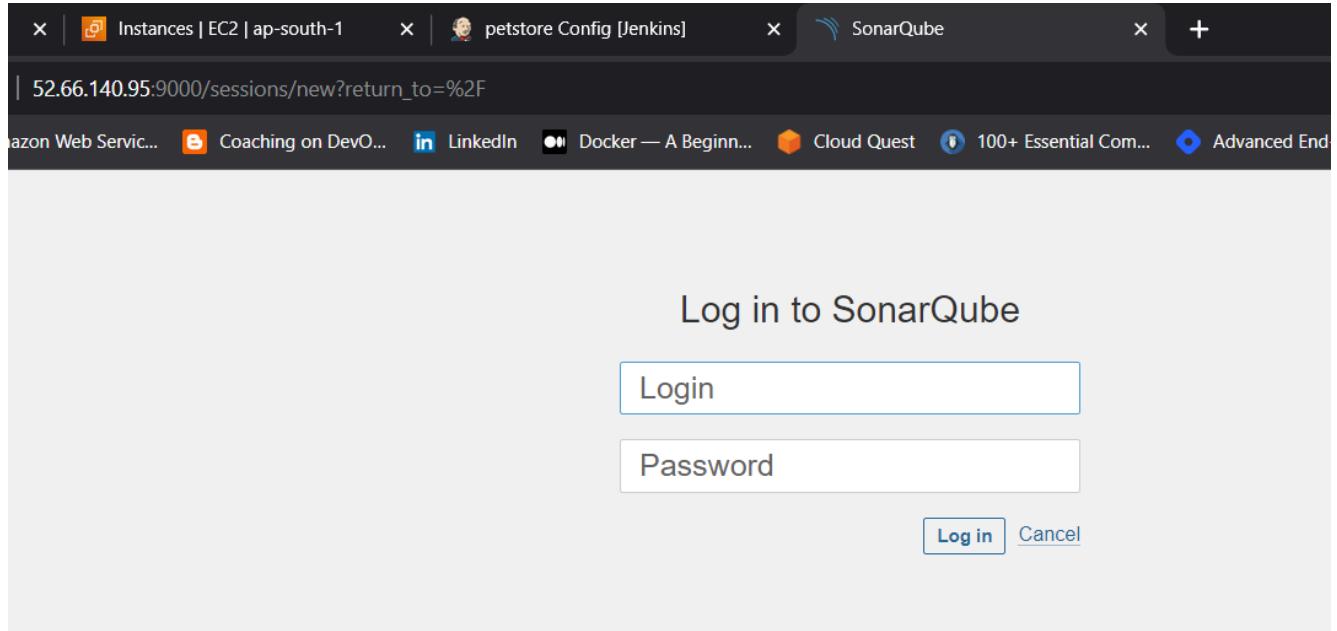


```
docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
```

```
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
44ba2882f9eb: Pull complete
2cabe57fa36: Pull complete
c20481384b6a: Pull complete
bf7b17ee74f8: Pull complete
38617faac714: Pull complete
706f20f58f5e: Pull complete
65a29566c257: Pull complete
Digest: sha256:1a118f8ab960d6c3d4ea8b4455a5a6560654511c88a6816f1603f764d5dcc77c
Status: Downloaded newer image for sonarqube:lts-community
4b66c96bf9ad3d62289436af7f752fdb04993092d0ca3065e2f2e32301b50139
ubuntu@ip-172-31-42-253:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
4b66c96bf9ad 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 password

This account should not use the default password.

Enter a new password

All fields marked with \* are required

Old Password \*

New Password \*

Confirm Password \*

Update

Update your Sonarqube password & This is the Sonarqube dashboard

How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform.

First, you need to set up a DevOps platform configuration.

From Azure DevOps Set up global configuration	From Bitbucket Server Set up global configuration	From Bitbucket Cloud Set up global configuration	From GitHub Set up global configuration	From GitLab Set up global configuration
--	--	---	--	--

## 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

The screenshot shows the SonarQube 'Create a project' interface. At the top, there's a header bar with various links like 'Projects', 'Issues', 'Rules', etc. Below the header, a section asks 'How do you want to create your project?'. It lists five options: 'From Azure DevOps', 'From Bitbucket Server', 'From Bitbucket Cloud', 'From GitHub', and 'From GitLab', each with a 'Set up global configuration' link. Below these, a note says 'Are you just testing or have an advanced use-case? Create a project manually.' A red box highlights the 'Manually' button, which has a double arrow icon and the text 'Manually' below it.

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

The screenshot shows the 'Create a project' form. It includes fields for 'Project display name' (containing 'Netflix'), 'Project key' (containing 'Netflix'), and 'Main branch name' (containing 'main'). A note below the main branch field says 'The name of your project's default branch [Learn More](#)'. At the bottom right of the form is a blue 'Set Up' button, which is highlighted with a red box.

On the next page click on With GitHub actions

How do you want to analyze your repository?

Do you want to integrate with your favorite CI? Choose one of the following tutorials.

- With Jenkins
- With GitHub Actions**
- With Bitbucket Pipelines
- With GitLab CI
- With Azure Pipelines
- Other CI

Are you just testing or have an advanced use-case? Analyze your project locally.

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

**1 Create GitHub Secrets**

In your GitHub repository, go to **Settings > Secrets** and create two new secrets:

- Click on **New repository secret**.
- In the **Name** field, enter `SONAR_TOKEN`.
- In the **Value** field, enter `http://13.126.202.56:9000`.
- Click on **Add secret**.

- Click on **New repository secret**.
- In the **Name** field, enter `SONAR_HOST_URL`.
- In the **Value** field, enter `http://13.126.202.56:9000`.
- Click on **Add secret**.

**Continue**

Let's Open your GitHub and select your Repository

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

Aj7Ay / Netflix-clone

Code Pull requests Actions Projects Security Insights Settings

Netflix-clone Public forked from [gsbarure/netflix-clone-react-typescript](#)

main 1 branch 0 tags

This branch is 16 commits ahead of [gsbarure/main](#).

Go to file Add file Code

Contribute Sync fork

**About**

Netflix Clone using React, Typescript, Material UI

[netflix-clone-react-typescript.vercel.app](#)

Readme Activity 7 stars 0 watching 120 forks

**Releases**

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

Netflix-clone

**Template repository**  
Template repositories let users generate new repositories with the same directory structure and files. [Learn more about template repositories](#).

**Require contributors to sign off on web-based commits**  
Enabling this setting will require contributors to sign off on commits made through GitHub's web interface. Signing off is a way for contributors to affirm that their commit complies with the repository's terms, commonly the [Developer Certificate of Origin \(DCO\)](#). [Learn more about signing off on commits](#)

**Default branch**

The default branch is considered the "base" branch in your repository, against which all pull requests and code commits are automatically made, unless you specify a different branch.

main

**Social Preview**

Upload an image to customize your repository's social media preview.  
Images should be at least 640×320px (1280×640px for best display).  
[Download template](#)

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

**General**

**Actions secrets and variables**

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. [Learn more about encrypted secrets](#). Variables are shown as plain text and are used for **non-sensitive** data. [Learn more about variables](#).

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

**Environment secrets**

There are no secrets for this repository's environments.

**Repository secrets**

There are no secrets for this repository.

Now go back to Your Sonarqube Dashboard

Copy SONAR\_TOKEN and click on Generate Token

**1 Create GitHub Secrets**

In your GitHub repository, go to **Settings > Secrets** and create two new secrets:

- Click on **New repository secret**
- In the **Name** field, enter `SONAR_TOKEN` **COPY THIS ONE**
- In the **Value** field, enter an existing token, or a newly generated one. **Generate a token** **CLICK HERE**
- Click on **Add secret**

1. Click on **New repository secret**  
 2. In the **Name** field, enter `SONAR_HOST_URL`  
 3. In the **Value** field, enter `http://13.126.202.56:9000`  
 4. Click on **Add secret**

**Continue**

Click on Generate

## Generate a project token

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

Token name	Expires in
Analyze "Netflix"	30 days <b>Generate</b>

**i** Please note that this token will only allow you to analyze the current project. If you want to use the same token to analyze multiple projects, you need to generate a global token in your [user account](#). See the [documentation](#) for more information.

**Continue**

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

## Generate a project token

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

Analyze "Netflix":

sqp\_0fcfd59dfc0eef5378e95d8aebae06134b34bbe55 +

! New token "sqp\_0fcfd59dfc0eef5378e95d8aebae06134b34bbe55" has been created. Make sure you copy it now, you won't be able to see it again!

Continue

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

Actions secrets / New secret

Name \* SONAR\_TOKEN

Secret \* sqp\_0fcfd59dfc0eef5378e95d8aebae06134b34bbe55

Add secret

General  
Access  
Collaborators  
Moderation options  
Code and automation  
Branches  
Tags  
Rules  
Actions  
Webhooks  
Environments  
Codespaces  
Pages

Now go back to the Sonarqube Dashboard

Copy the Name and Value

## 1 Create GitHub Secrets

In your GitHub repository, go to **Settings > Secrets** and create two new secrets:

1. Click on **New repository secret**.

2. In the **Name** field, enter `SONAR_TOKEN` 

3. In the **Value** field, enter an existing token, or a newly generated one: 

4. Click on **Add secret**.

1. Click on **New repository secret**.

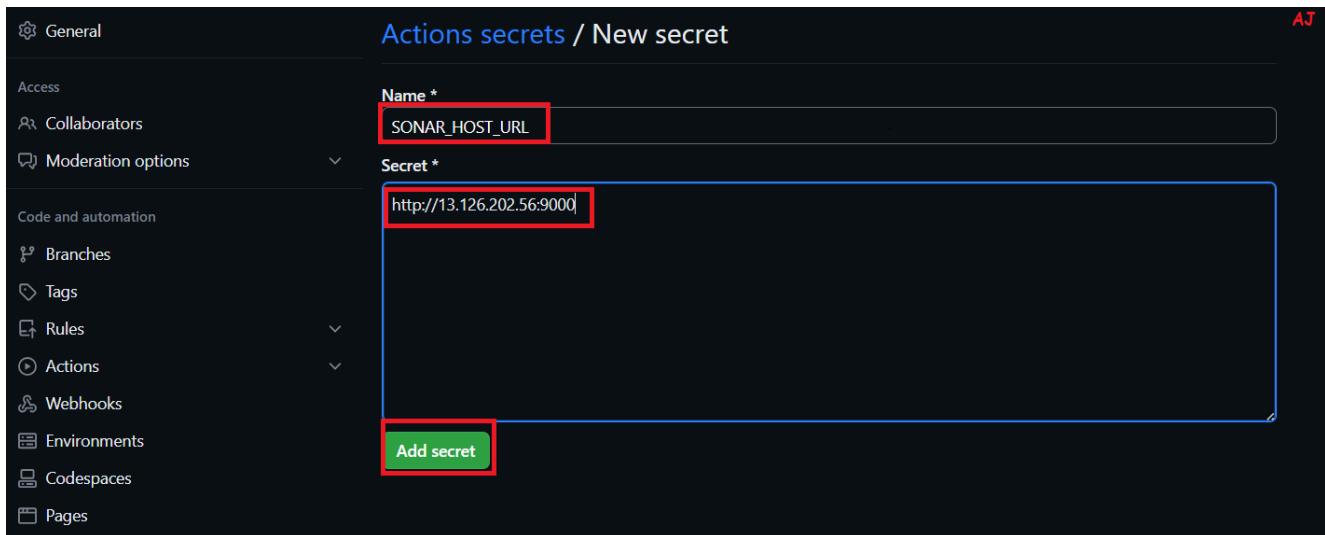
2. In the **Name** field, enter `SONAR_HOST_URL` 

3. In the **Value** field, enter `http://13.126.202.56:9000` 

4. Click on **Add secret**.

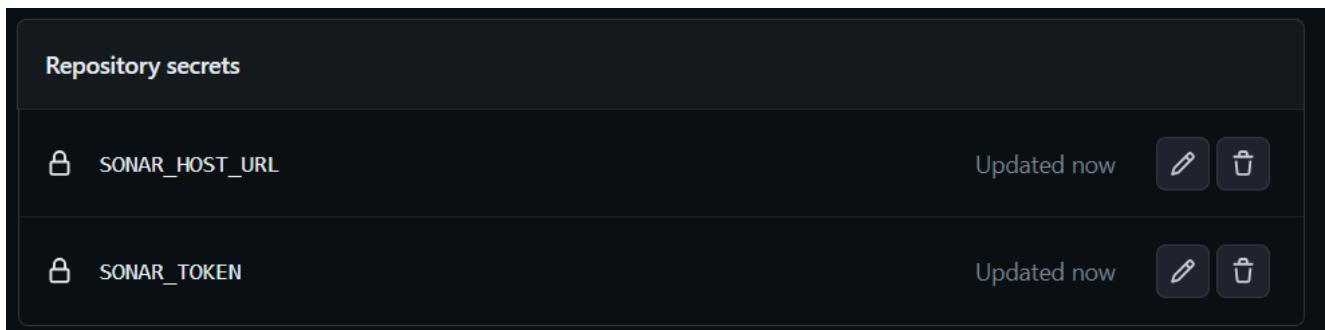


Go to GitHub now and paste-like this and click on add secret



The screenshot shows the 'Actions secrets / New secret' form. On the left is a sidebar with sections like General, Access, Collaborators, Moderation options, Code and automation, Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, and Pages. The 'Actions' section is expanded. The main area has 'Name \*' set to 'SONAR\_HOST\_URL' and 'Secret \*' set to 'http://13.126.202.56:9000'. A green 'Add secret' button at the bottom is highlighted with a red box.

Our Sonarqube secrets are added and you can see



The screenshot shows the 'Repository secrets' list. It contains two entries: 'SONAR\_HOST\_URL' and 'SONAR\_TOKEN', both updated 'now'. Each entry has edit and delete icons.

Secret	Last Updated	Action
SONAR_HOST_URL	Updated now	 
SONAR_TOKEN	Updated now	 

## Go to Sonarqube Dashboard and click on continue

The screenshot shows the Sonarqube dashboard for the Netflix project. Step 1, 'Create GitHub Secrets', is displayed. It provides instructions to create two new secrets in the GitHub repository settings. The 'Continue' button at the bottom is highlighted with a red box.

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

The screenshot shows the Sonarqube dashboard for the Netflix project. Step 2, 'Create Workflow YAML File', is displayed. It asks what option best describes the build. The 'Other (for JS, TS, Go, Python, PHP, ...)' option is selected and highlighted with a red box. A note 'IN MY CASE IAM USING REACT JS' is written next to it.

Now it Generates and workflow for my Project

(Use your files for this block please)

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

```
sonar.projectKey=Netflix
```

3. Create or update your `.github/workflows/build.yml` YAML file with the following content:

```
name: Build

on:
  push:
    branches:
      - main

jobs:
  build:
    name: Build
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
        with:
          fetch-depth: 0 # Shallow clones should be disabled for a better relevancy of analysis
      - uses: sonarsource/sonarqube-scan-action@master
        env:
          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
          SONAR_HOST_URL: ${{ secrets.SONAR_HOST_URL }}
      # If you wish to fail your job when the Quality Gate is red, uncomment the
      # following lines. This would typically be used to fail a deployment.
      # - uses: sonarsource/sonarqube-quality-gate-action@master
```

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

This branch is 16 commits ahead of gsbarure:main.

Aj7Ay Delete sonar-project.properties

- Kubernetes Create service.yml last month
- public fix README 5 months ago

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=Netflix
```

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)

```
sonar.projectKey=Netflix
```

Let's add our workflow

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

This branch is 16 commits ahead of gsbarure:main.

Author	Commit Message	Date	Commits
Aj7Ay	Delete sonar-project.properties	54c7c49 4 minutes ago	111 commits
	Kubernetes	Create service.yml	last month
	public	fix README	5 months ago

Here is the file name



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



SonarQube Projects Issues Rules Quality Profiles Quality Gates Administration ? Search for projects... A

Netflix main Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

3. Create or update your `.github/workflows/build.yml` YAML file with the following content:

```
name: Build

on:
  push:
    branches:
      - main

jobs:
  build:
    name: Build
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
        with:
          fetch-depth: 0 # Shallow clones should be disabled for a better relevancy of analysis
      - uses: sonarsource/sonarqube-scan-action@master
        env:
          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
          SONAR_HOST_URL: ${{ secrets.SONAR_HOST_URL }}
    # If you wish to fail your job when the Quality Gate is red, uncomment the
    # following lines. This would typically be used to fail a deployment.
    # - uses: sonarsource/sonarqube-quality-gate-action@master
    #   timeout-minutes: 5
    #   env:
    #     SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
```

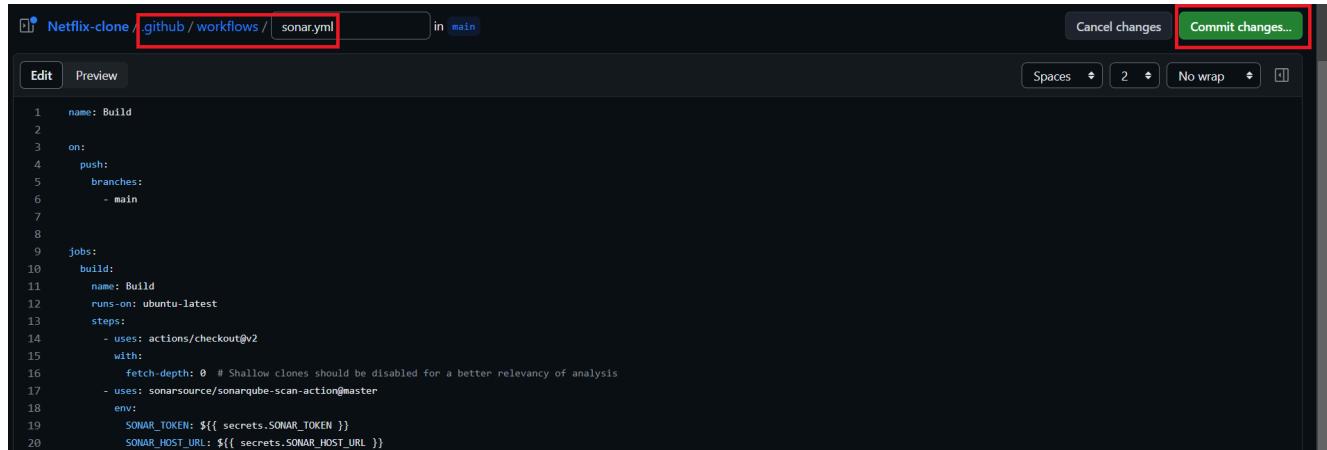
**Copy**

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 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 }}
```

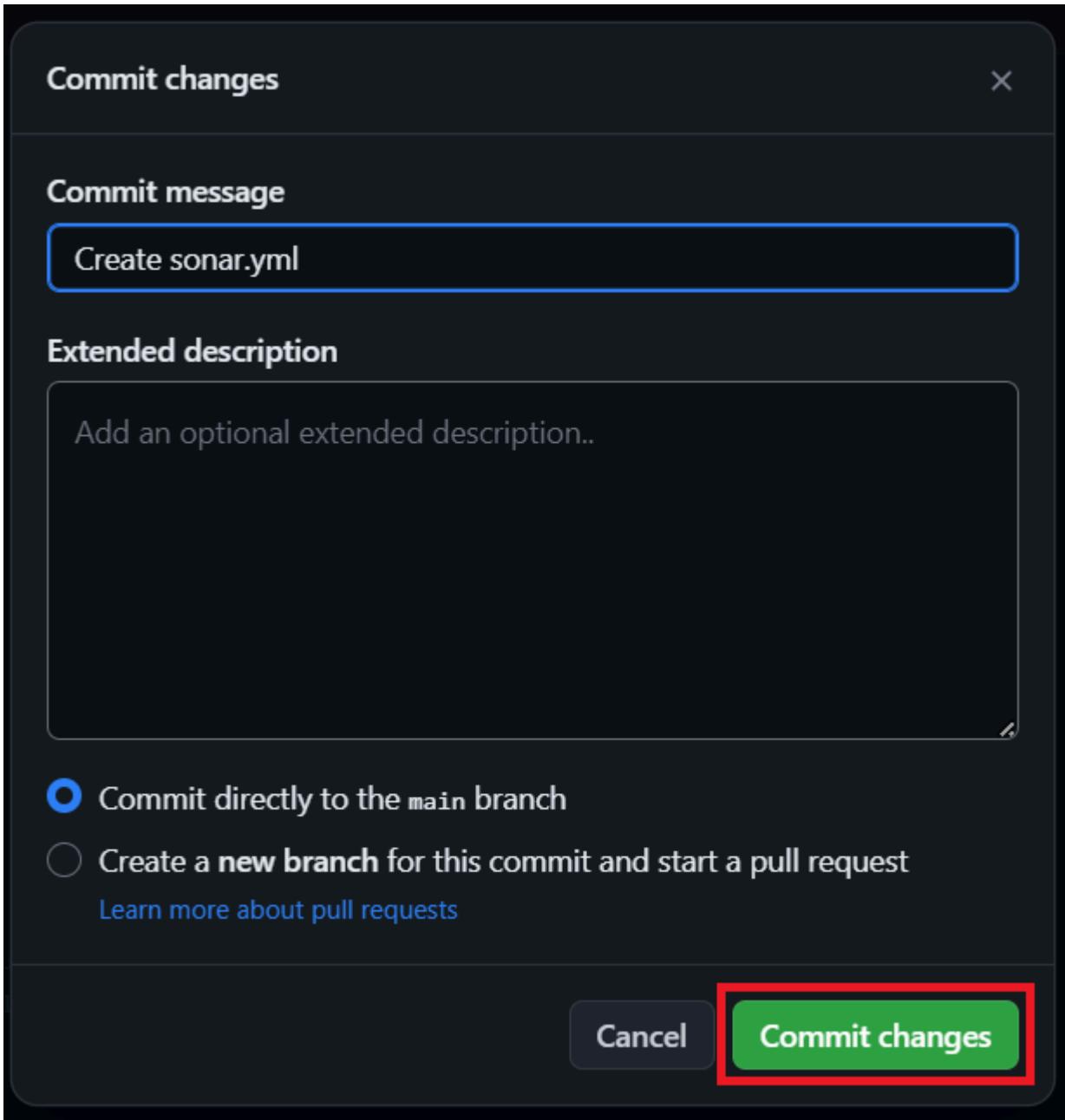


A screenshot of a GitHub workflow editor. The URL in the address bar is "Netflix-clone / github / workflows / sonaryml" in the "main" branch. The editor shows a YAML configuration file:

```
name: Build
on:
  push:
    branches:
      - main
jobs:
  build:
    name: Build
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
        with:
          fetch-depth: 0 # Shallow clones should be disabled for a better relevancy of analysis
      - uses: sonarsource/sonarqube-scan-action@master
        env:
          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
          SONAR_HOST_URL: ${{ secrets.SONAR_HOST_URL }}
```

The "Commit changes..." button in the top right corner is highlighted with a red box.

Click on Commit changes



Now workflow is created.

Start again GitHub actions runner from instance



```
cd actions-runner  
.run.sh
```



Click on Actions now

This branch is 18 commits ahead of gsbarure:main.

Aj7Ay Create sonar.yml

- .github/workflows Create sonar.yml now
- Kubernetes Create service.yml last month

Now it's automatically started the workflow

All workflows

2 workflow runs

Create sonar.yml

Build #2: Commit 56492ae pushed by Aj7Ay

Status: In progress

7s

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

Build

Started 14s ago

Set up job

Build sonarsource/sonarqube-scan-action@master

Run actions/checkout@v2

Run sonarsource/sonarqube-scan-action@master

Post Run actions/checkout@v2

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

The screenshot shows the GitHub Actions build logs for a job named "Run sonarsource/sonarqube-scan-action@master". The log output includes several INFO messages from the SonarQube Publisher and Executor. A red box highlights the message "INFO: EXECUTION SUCCESS" at line 134.

```

Build succeeded now in 1m 7s
Run sonarsource/sonarqube-scan-action@master
135 INFO: SCM Publisher 69 source files to be analyzed
136 INFO: SCM Publisher 69/69 source files have been analyzed (done) | time=994ms
137 INFO: CPD Executor 6 files had no CPD blocks
138 INFO: CPD Executor Calculating CPD for 60 files
125 INFO: CPD Executor CPD calculation finished (done) | time=43ms
126 INFO: Analysis report generated in 130ms, dir size=432.8 kB
127 INFO: Analysis report compressed in 215ms, zip size=270.2 kB
128 INFO: Analysis report uploaded in 1074ms
129 INFO: ANALYSIS SUCCESSFUL, you can find the results at: ****/dashboard?id=Netflix
130 INFO: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
131 INFO: More about the report processing at ****/api/ce/task?id=AyTftis7LwC_1711jGN
132 INFO: Analysis total time: 32.013 s
133 INFO:
134 INFO: EXECUTION SUCCESS
135 INFO:
136 INFO: Total time: 52.778s
137 INFO: Final Memory: 17M/64M
138 INFO:

```

Build complete.

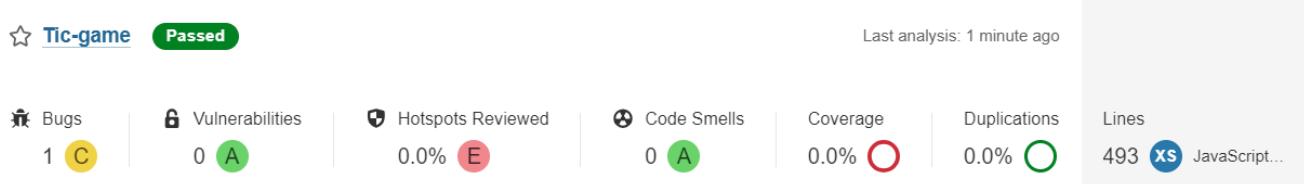
The screenshot shows the GitHub Actions build logs for a job named "Build sonarsource/sonarqube-scan-action@master". The log output details the execution of various steps: Set up job, Build container for action, Run actions/checkout@v2, Run sonarsource/sonarqube-scan-action@master, Post Run sonarsource/sonarqube-scan-action@master, Post Run actions/checkout@v2, and Complete job. A red box highlights the "Build sonarsource/sonarqube-scan-action@master" step.

```

Build succeeded now in 1m 7s
Set up job
Build sonarsource/sonarqube-scan-action@master
1 ▶ Build container for action use: '/home/runn.../actions/sonarsource/sonarqube-scan-action/master/Dockerfile'.
> ✓ Run actions/checkout@v2
> ✓ Run sonarsource/sonarqube-scan-action@master
> ✓ Post Run sonarsource/sonarqube-scan-action@master
> ✓ Post Run actions/checkout@v2
> ✓ Complete job

```

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/adoptium/jdk/temurin/17/jdk/17.0.1+12/2023-07-18T14-00Z/adoptium-jdk-17.0.1-17.0.1+12-2023-07-18T14-00Z-amd64.deb
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/adoptium/jdk/temurin/17/jdk/17.0.1+12/2023-07-18T14-00Z/ adoptium-jdk" | sudo tee /etc/apt/sources.list.d/adoptium.list > /dev/null
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 -o /etc/apt/trivy.gpg
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb/ /" | sudo tee /etc/apt/sources.list.d/trivy.list > /dev/null
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 /etc/apt/travis-backdoor.gpg
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com/ /" | sudo tee /etc/apt/sources.list.d/hashicorp.list > /dev/null
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)
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 /etc/apt/travis-backdoor.gpg
curl -fsSL https://deb.nodesource.com/nodesource.list | sudo tee /etc/apt/sources.list.d/nodesource.list > /dev/null
sudo apt update
sudo apt install nodejs -y
```

```
echo "deb [signed-by=/usr/share/keyrings/nodesource-archive-keyring.gpg]
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:~$ trivy --version
Version: 0.46.0
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:~$ terraform --version
Terraform v1.6.2
on linux_amd64
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
<!--
-->
```

```
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

The screenshot shows the AWS EKS Clusters page. At the top, there is a navigation bar with 'EKS > Clusters'. Below it is a search bar labeled 'Filter clusters'. A table lists one cluster:

Cluster name	Status	Kubernetes version	Provider
EKS_CLOUD	Active	1.28	EKS

Node group ec2 instance

The screenshot shows the AWS Instances page. At the top, there is a search bar labeled 'Find Instance by attribute or tag (case-sensitive)'. A table lists two instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Put
Jenkins-ARGO	i-0323f37f837248e53	Running	t2.large	2/2 checks passed	No alarms	ap-south-1b	ec2
	i-049634a401c64808b	Running	t2.medium	2/2 checks passed	No alarms	ap-south-1b	ec2

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.yaml 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 <a href="https://avd.aquasec.com/nvd/cve-2023-26364">https://avd.aquasec.com/nvd/cve-2023-26364</a>
@babel/traverse	CVE-2023-45133	CRITICAL		7.18.13	7.23.2, 8.0.0-alpha.4	arbitrary code execution <a href="https://avd.aquasec.com/nvd/cve-2023-45133">https://avd.aquasec.com/nvd/cve-2023-45133</a>
json5	CVE-2022-46175	HIGH		1.0.1	2.2.2, 1.0.2	Prototype Pollution in JSON5 via Parse Method <a href="https://avd.aquasec.com/nvd/cve-2022-46175">https://avd.aquasec.com/nvd/cve-2022-46175</a>
loader-utils	CVE-2022-37601	CRITICAL		2.0.2	2.0.3, 1.4.1	prototype pollution in function parseQuery in parseQuery.js <a href="https://avd.aquasec.com/nvd/cve-2022-37601">https://avd.aquasec.com/nvd/cve-2022-37601</a>
	CVE-2022-37599	HIGH			1.4.2, 2.0.4, 3.2.1	regular expression denial of service in interpolateName.js <a href="https://avd.aquasec.com/nvd/cve-2022-37599">https://avd.aquasec.com/nvd/cve-2022-37599</a>
	CVE-2022-37603					Regular expression denial of service <a href="https://avd.aquasec.com/nvd/cve-2022-37603">https://avd.aquasec.com/nvd/cve-2022-37603</a>
	CVE-2022-37599					regular expression denial of service in interpolateName.js <a href="https://avd.aquasec.com/nvd/cve-2022-37599">https://avd.aquasec.com/nvd/cve-2022-37599</a>
	CVE-2022-37603					Regular expression denial of service <a href="https://avd.aquasec.com/nvd/cve-2022-37603">https://avd.aquasec.com/nvd/cve-2022-37603</a>
minimatch	CVE-2022-3517			3.0.4	3.0.5	ReDoS via the braceExpand function <a href="https://avd.aquasec.com/nvd/cve-2022-3517">https://avd.aquasec.com/nvd/cve-2022-3517</a>

Create a Personal Access token for your Dockerhub account

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

The screenshot shows the Docker Hub user interface. At the top, there's a navigation bar with links for Explore, Repositories, Organizations, Help, and Upgrade. On the right, a user profile is shown with the name 'sevenajay'. A dropdown menu is open next to the profile, with 'Account Settings' highlighted by a red box and a red number '2' indicating it's the second step. Below the profile, the user's name 'sevenajay' is displayed, along with their status as a 'User' and the date 'Joined August 10, 2022'. The main content area has a sidebar on the left with options like General, Security (which is selected and highlighted with a red box and a red number '3'), Default Privacy, Notifications, Convert Account, and Deactivate Account. The main panel shows 'Access Tokens' with a table listing one token: 'Ansible' (Scope: Read, Write, Delete), Last Used on Sep 16, 2023 at 08:07:09, Created on Sep 15, 2023 at 18:08:25, and Active status. A red box highlights the 'New Access Token' button at the bottom right of the tokens table.

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

The screenshot shows a modal dialog titled 'New Access Token'. It explains that a personal access token is similar to a password but can have many tokens and be revoked. The 'Access Token Description' field is filled with 'Netflix' and is highlighted with a red box. The 'Access permissions' dropdown is set to 'Read, Write, Delete' and is also highlighted with a red box. Below the permissions, a note says 'Read, Write, Delete tokens allow you to manage your repositories.' At the bottom right of the dialog, there are 'Cancel' and 'Generate' buttons, with 'Generate' highlighted with a red box.

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

## Copy Access Token

When logging in from your Docker CLI client, use this token as a password. [Learn more](#)

### ACCESS TOKEN DESCRIPTION

Netflix

### ACCESS PERMISSIONS

Read, Write, Delete

To use the access token from your Docker CLI client:

1. Run `docker login -u sevenajay`

2. At the password prompt, enter the personal access token.

`dckr_pat_4lnKFbaykudW-ueAmMIDcRMyFmA`



WARNING: This access token will only be displayed once. It will not be stored and cannot be retrieved. Please be sure to save it now.

[Copy and Close](#)

Now Go to GitHub again and click on settings

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

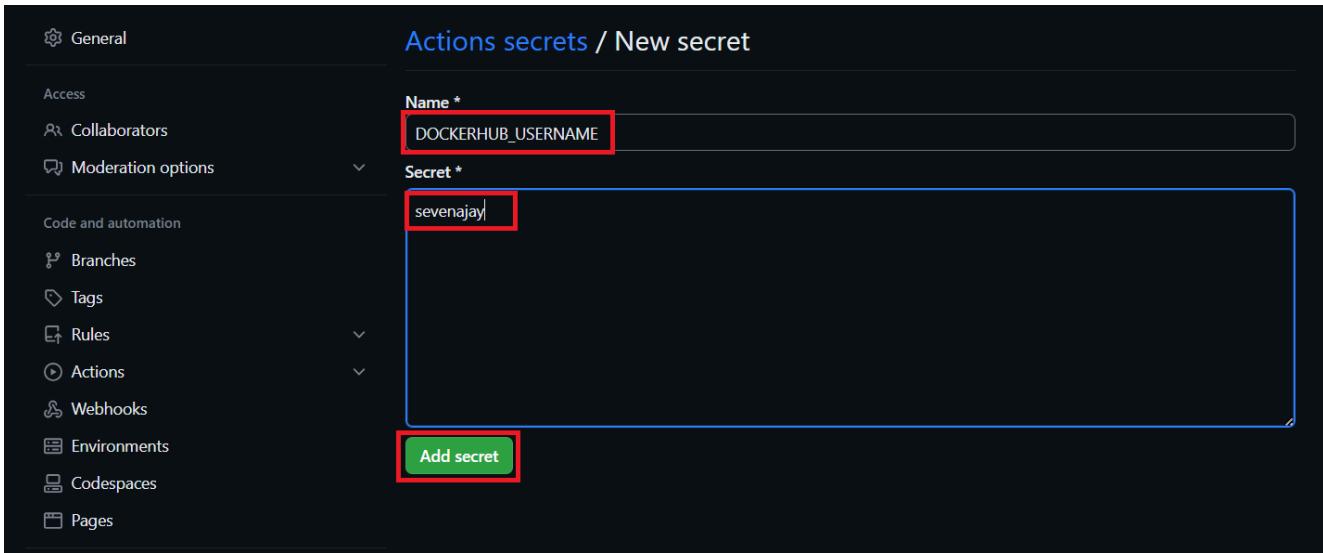
The screenshot shows the GitHub repository settings for 'Netflix-clone'. The left sidebar has sections like Collaborators, Moderation options, Code and automation (Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, Pages), Security (Code security and analysis, Deploy keys, Secrets and variables, Actions), Codespaces, and Dependabot. The 'Secrets and variables' and 'Actions' sections are highlighted with red boxes. The main area shows a 'Default branch' set to 'main' with an edit button. Below it is a 'Social Preview' section with a placeholder for an image and a 'Download template' button.

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

The screenshot shows the 'Actions secrets and variables' page. The left sidebar is identical to the previous one. The main area has tabs for 'Secrets' (which is selected) and 'Variables'. A green button labeled 'New repository secret' is visible. Below are two sections: 'Environment secrets' (with a 'Manage environments' button) and 'Repository secrets', both of which show a message: 'There are no secrets for this repository's environments.' and 'There are no secrets for this repository.'

Add your Dockerhub username with the secret name as

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



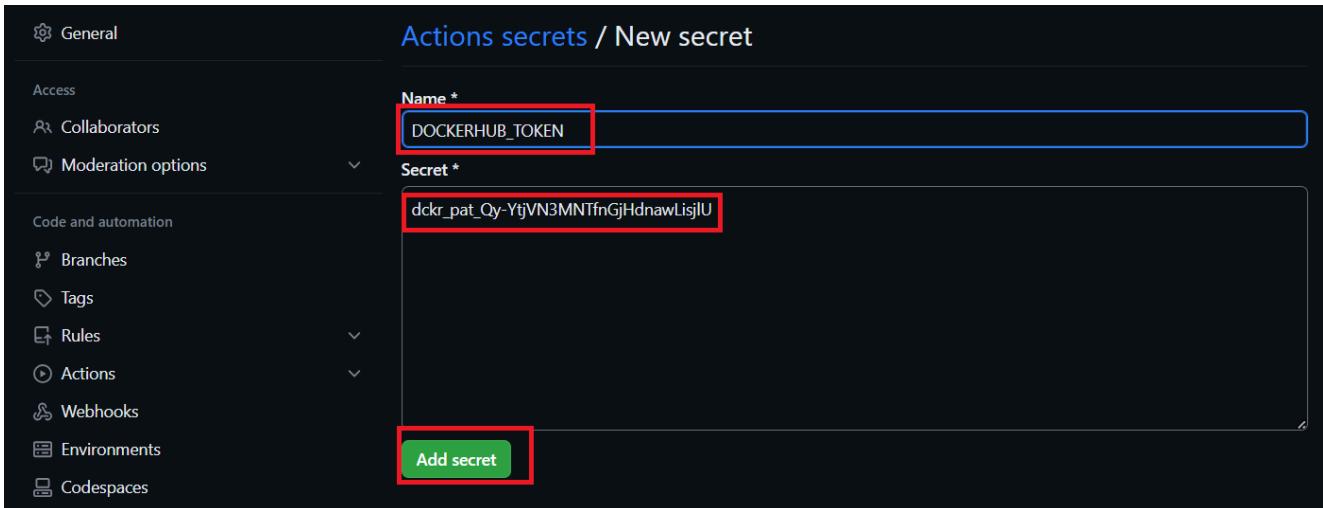
Click on Add Secret.

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

```
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	Docker commands
Description	To push a new tag to this repository:
This repository does not have a description	<code>docker push sevenajay/tic-tac-toe:tagname</code>
Last pushed: a few seconds ago	

## 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



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

The screenshot shows a GitHub Actions pipeline interface. At the top, there's a header with 'build.yml' and 'on: push'. Below it is a horizontal flowchart with two nodes: 'Build' (yellow circle) and 'deploy' (grey circle). An arrow points from 'Build' to 'deploy', labeled '26s'. To the right of the flowchart are three small icons: a square, a minus sign, and a plus sign.

**Docker Build and push**

```

98 af1982f6d133: Layer already exists
99 c5b325bdd721: Layer already exists
100 be322b479aee: Layer already exists
101 d41bcd3a037b: Layer already exists
102 fe0dd845e767b: Layer already exists
103 f25ec1d93a58: Layer already exists
104 3220beed9b06: Layer already exists
105 794ce8b1b516: Layer already exists
106 684fb2921421: Layer already exists
107 9af5f53e8f62: Layer already exists
108 42642eab0757: Pushed
109 9fc715bef52e: Pushed
110 latest: digest: sha256:0295f011d6645ab0e1c5fc00f37d90b6f2d624c0be4df5a593675bfb32dd6c3 size: 3055
  
```

**Image scan**

```

1 ► Run trivy image ***/tic-tac-toe:latest > trivyimage.txt
4
5 2023-10-29T07:10:08.686Z      INFO    Vulnerability scanning is enabled
6 2023-10-29T07:10:08.686Z      INFO    Secret scanning is enabled
7 2023-10-29T07:10:08.686Z      INFO    If your scanning is slow, please try '--scanners vuln' to disable secret scanning
8 2023-10-29T07:10:08.686Z      INFO    Please see also https://aquasecurity.github.io/trivy/v0.46/docs/scanner/secret/#recommendation for faster secret detection
  
```

## Image scan report

sevenajay/tic-tac-toe:latest (debian 10.13)							
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...	
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	
cowin	CVE-2018-1000876					integer overflow leads to heap-based buffer overflow objdump https://avd.aquasec.com/nvd/cve-2018-1000876	

```
deploy
succeeded 2 minutes ago in 1m 51s

> ✓ 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 529004a51c3519f99d6017a007df1b9938b77b1d3e26e22a51f69aaee3a2a0b

> ✓ 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-
```



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

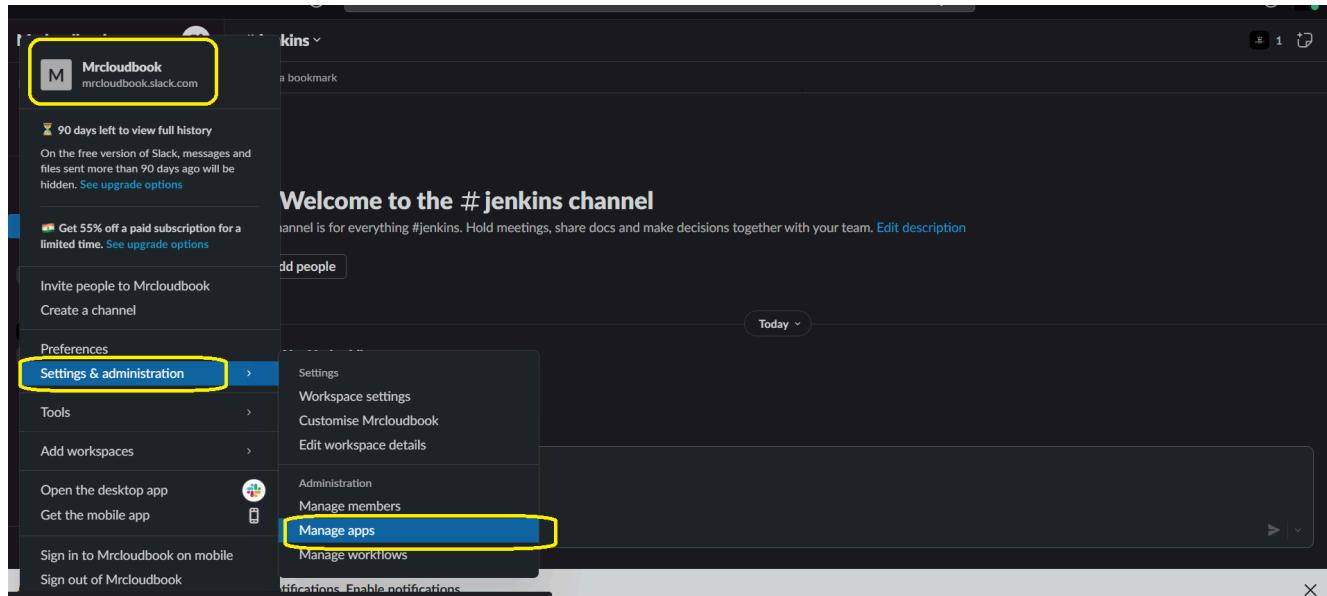
Go to your Slack channel, if you don't have create one

Go to Slack channel and create a channel for notifications

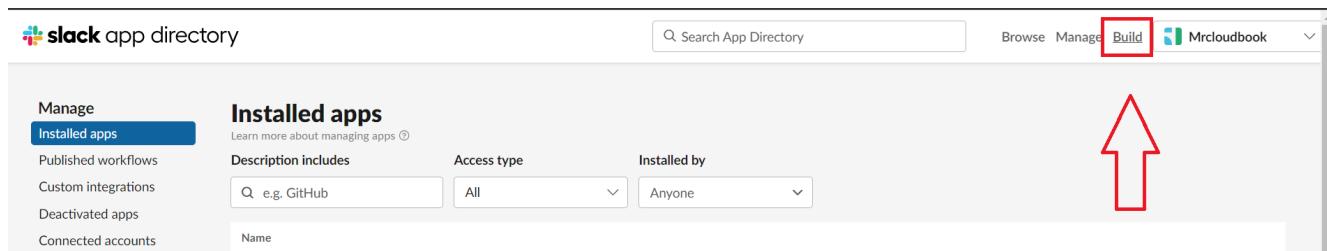
click on your name

Select Settings and Administration

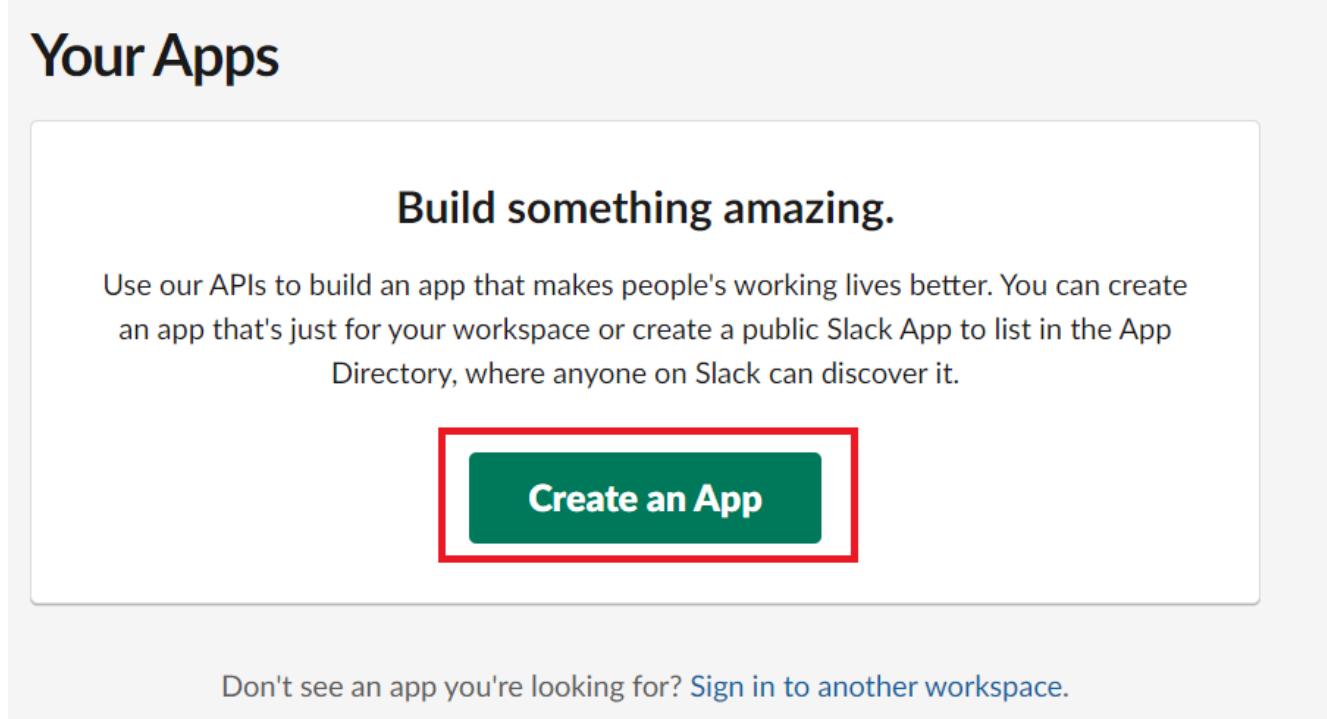
Click on Manage apps



It will open a new tab, select build now



Click on create an app



Select from scratch

# Your Apps

## Create an app

X

Choose how you'd like to configure your app's scopes and settings.

### From scratch

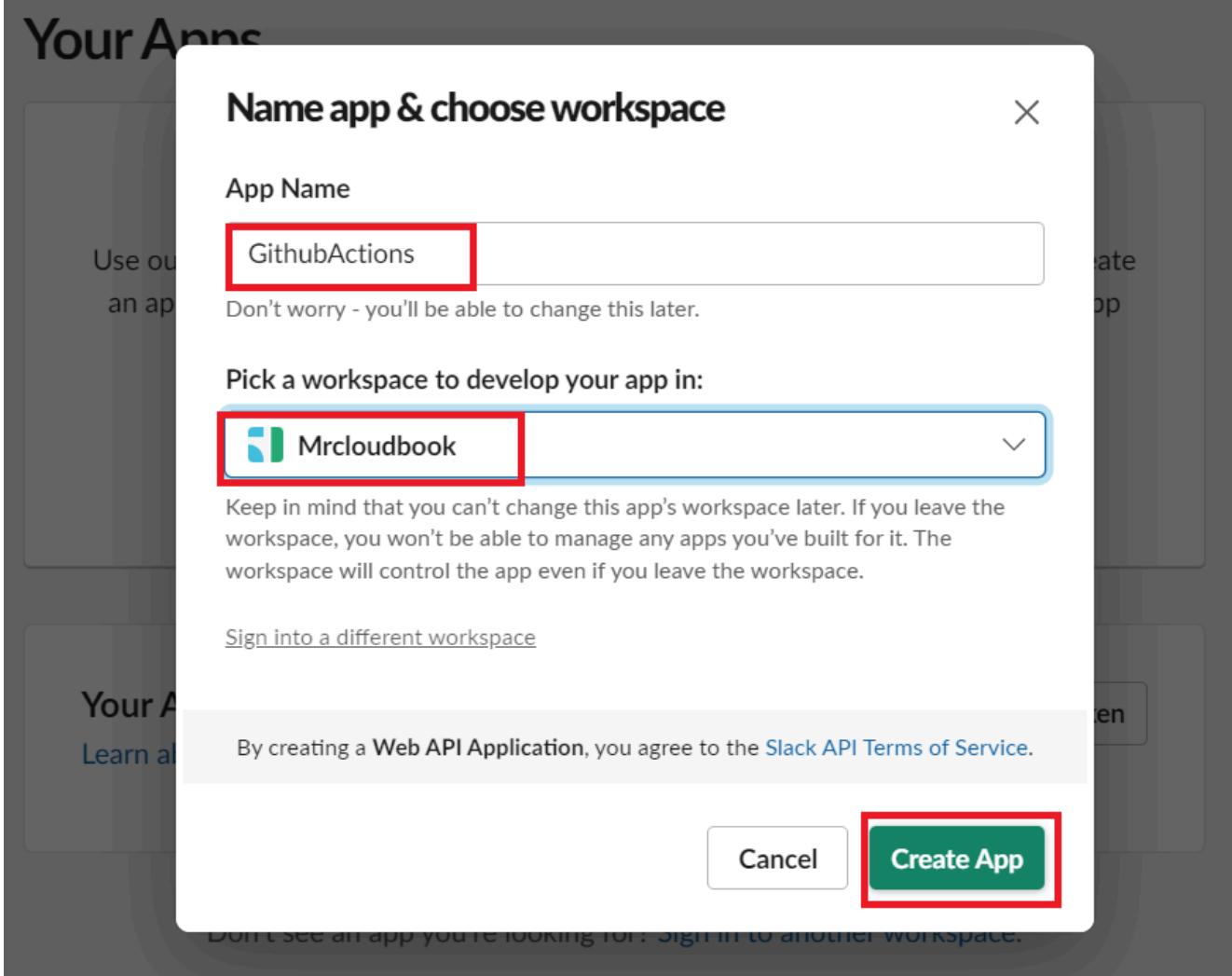
Use our configuration UI to manually add basic info, scopes, & features to your app. >

### From an app manifest

Use a manifest file to add your app's basic info, scopes, & features to your app. >

Need help? Check our [documentation](#), or [see an example](#)

Provide a name for the app and select workspace and create



Select Incoming webhooks

# Building Apps for Slack

Create an app that's just for your workspace (or build one that can be used by any workspace) by following the steps below.

## Add features and functionality

Choose and configure the tools you'll need to create your app (or review all our documentation).

### Building an internal app locally or behind a firewall?

To receive your app's payloads over a WebSockets connection, enable [Socket Mode](#) for your app.

#### Incoming Webhooks

Post messages from external sources into Slack.

#### Interactive Components

Add components like buttons and select menus to your app's interface, and create an interactive experience for users.

#### Slash Commands

Allow users to perform app actions by typing commands in Slack.

#### Event Subscriptions

Make it easy for your app to respond to activity in Slack.

Set incoming webhooks to on

# Incoming Webhooks

## Activate Incoming Webhooks



[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
<hr/>		
No webhooks have been added yet.		

[Add New Webhook to Workspace](#)

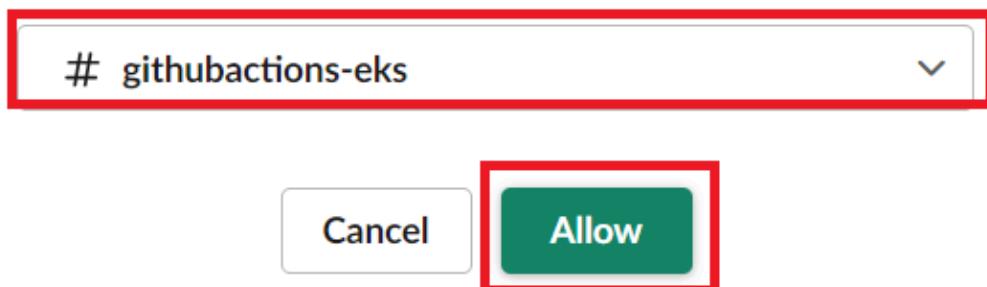
Select Your channel that created for notifications and allow



## GithubActions is requesting permission to access the Mrcloudbook Slack workspace

### Where should GithubActions post?

- # GithubActions requires a channel to post to as an app



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

Actions secrets / New secret

Name \*

 (The input field is highlighted with a red border)

Secret \*

 (The input field is highlighted with a blue border)

Add secret

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 beta
      - name: Build and analyze with SonarQube
        uses: sonarsource/sonarqube-scan-action@master
```

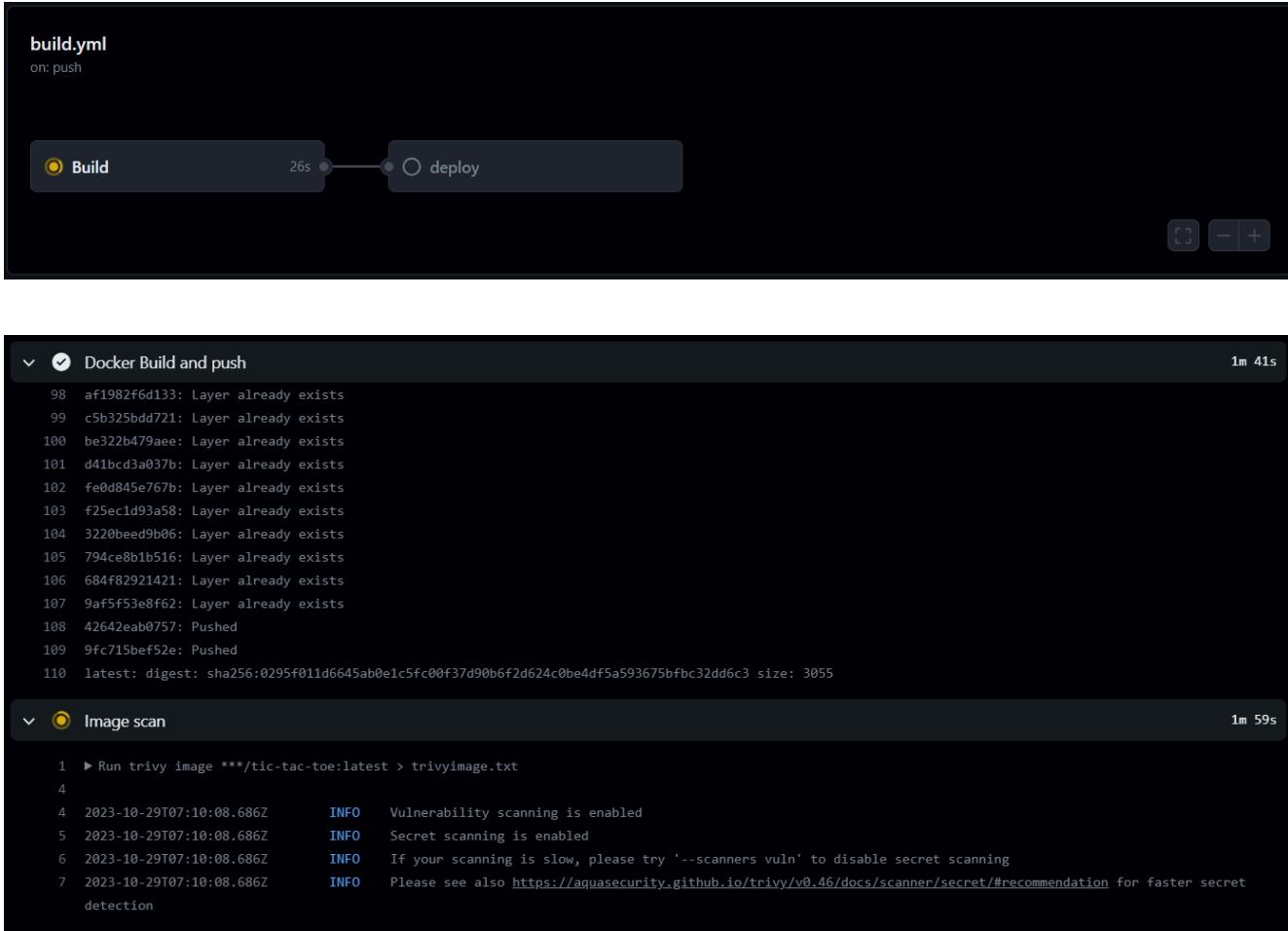


```

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_PASSWORD }}
    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
    - name: Deploy to container
      run: docker run -d --name game -p 3000:3000 sevenajay/candycrush
    - 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
    - name: Send a Slack Notification
      if: always()
      uses: actions/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

sevenatjay/tic-tac-toe:latest (debian 10.13)							
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	
not						correctly...	
						<a href="https://avd.aquasec.com/nvd/cve-2011-3374">https://avd.aquasec.com/nvd/cve-2011-3374</a>	
bash	CVE-2019-18276			5.0-4		when effective UID is not equal to its real UID th	
e saved...						<a href="https://avd.aquasec.com/nvd/cve-2019-18276">https://avd.aquasec.com/nvd/cve-2019-18276</a>	
binutils	CVE-2017-13716			2.31.1-16		binutils: Memory leak with the C++ symbol demangle	
routine						in libiberty	
						<a href="https://avd.aquasec.com/nvd/cve-2017-13716">https://avd.aquasec.com/nvd/cve-2017-13716</a>	
dw	CVE-2018-1000876					integer overflow leads to heap-based buffer overfl	
in						objdump	
						<a href="https://avd.aquasec.com/nvd/cve-2018-1000876">https://avd.aquasec.com/nvd/cve-2018-1000876</a>	

```

deploy
succeeded 2 minutes ago in 1m 51s

> ✓ 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 529004a51c3519f99d6017a007df1b9938b77b1d3e26e22a51f69aaee3a2a0b

> ✓ 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

```

The screenshot shows a GitHub Actions pipeline status. At the top, it says "Triggered via push 12 minutes ago" and "Status Success". Below that, it shows "Total duration 7m 13s" and "Artifacts -". The main part of the screenshot displays a workflow named "deployed.yml" triggered by "on: push". It consists of two steps: "Build" and "deploy". The "Build" step took 4m 33s and was successful. The "deploy" step took 1m 59s and was also successful. Both steps are represented by green checkmarks.

Job completed.

Let's go to the Ec2 ssh connection

Provide this command



kubectl get all



```
YfMUW9R00A6qn-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
service/kubernetes   ClusterIP   10.100.0.1   <none>
service/tetris-service   LoadBalancer  10.100.212.20  a0725d5c76f3e47a8af3c7391c061307-244742789.ap-south-1.elb.amazonaws.com  80:31849/TCP   123m
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



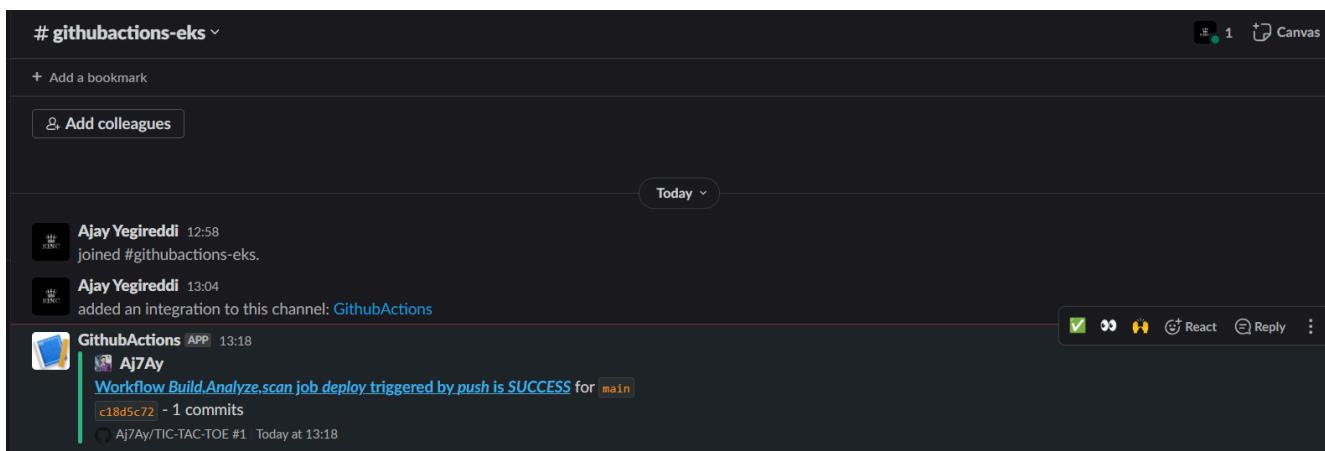
```
name: Build,Analyze,scan
on:
  push:
    branches:
      - main
jobs:
  build-analyze-scan:
    name: Build
    runs-on: [self-hosted]
    steps:
      - name: Checkout code
        uses: actions/checkout@v2
        with:
          fetch-depth: 0 # Shallow clones should be disabled for a beta
  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



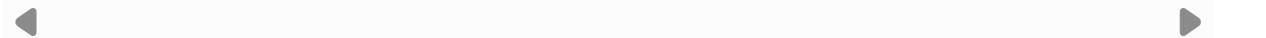
To delete the Eks cluster



```

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.

THANKS.

Aws

ci/cd

devops

DevSecOps

Docker

GitHubActions

Jenkins

kuberenetes

Terraform

Trivy



**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

### 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](#)

© 2024 · Powered by [Mr Cloud Book](#)

[Follow Us on YouTube](#)