

# SonarQube Setup with GCP and GitHub Actions - Detailed Instructional Guide

Created by CS2340 Instructional Team (Special Thanks to your TA - Pawan Medidi)

**YouTube Video instructions here:** <https://www.youtube.com/watch?v=QWnt-1RI7jI>

Following this video step-by-step should be sufficient to complete the setup process. **This setup needs to be done by 1 person on a team.**

This PDF is here for those who get confused in the video with certain steps. Please make sure to follow very carefully. You can follow the video OR this PDF to complete the setup. Finally, if you get confused in any step, ask in the respective Ed discussion megathread.

**IMPORTANT:** One of the **most crucial steps** is outlined at the end in **Step 7, 'Final Things to Do - IMPORTANT.'** Without following these two steps, the process will **NOT** work. It's essential to complete them, whether you use the PDF or the YouTube video. The method shown in the video might vary due to differences in GCP account configurations or GitHub permissions. If you encounter issues, refer to this PDF for additional guidance. After thorough testing, our approach has worked for all who have tested it.

## Introduction

This guide will provide a step-by-step, **click-by-click** walkthrough of setting up **SonarQube** on **Google Cloud Platform (GCP)** and integrating it with **GitHub Actions** for automated code quality analysis. Along the way, we will cover what SonarQube is, how you can utilize GCP's free credits, and how GitHub Actions functions as a powerful CI/CD tool.

## What is SonarQube?

**SonarQube** is an open-source platform that performs continuous inspection of your code, detecting bugs, vulnerabilities, and other issues like code smells. SonarQube ensures code quality by providing developers with detailed reports, metrics, and trends that help improve the health of their projects.

## GCP Free Credits

**Google Cloud Platform (GCP)** provides **\$300 in free credits** to new users, which you can use for over **90 days**. This is a perfect opportunity to run virtual machines, containers, and storage needed for deploying and hosting SonarQube. The GCP credits will be **more than sufficient** for

running this setup for the duration of this semester. **Important: Only one member per group needs to set up the GCP account. A credit card is required for verification, but Google will not charge unless you manually upgrade to a paid plan. We will send several reminders towards the end of Sprint 4 to ensure you shut down your project to avoid any charges.**

## What are GitHub Actions?

**GitHub Actions** is a CI/CD (continuous integration, continuous deployment) tool that automates and manages your software development workflow directly within GitHub. It allows you to build, test, and deploy your code automatically after pushing changes, enabling continuous integration and delivery. Integrating GitHub Actions with SonarQube allows you to run quality checks on your code every time you push changes to your repository.

## Detailed Setup Guide

### Step 1: Setting up a Service Account on GCP

#### 1. Sign in to Google Cloud Platform (GCP)

- a. Create a new Google account. This would be better if you have used free google cloud credits using your google account before.
- b. Go to [console.cloud.google.com](https://console.cloud.google.com) and log in with your new Google credentials.

#### 2. Enable Required APIs


- a. Select APIs & Services > Dashboard in the left navigation menu.
- b. Create a new project.
- c. Click **Enable APIs and Services** at the top.
- d. Search for and enable the following APIs:
  - i. **Cloud Compute Engine API**
  - ii. **IAM API**
  - iii. **Cloud Resource Manager API**
- e. It may ask you to set up a Billing account when you try and enable an API. Choose Enable Billing → Manage Billing Account → Enter payment profile as organization, and enter your details and payment method.

 Try Google Cloud for free

## Step 2 of 2 Payment Information Verification


Your payment information helps us reduce fraud and abuse. If using a credit or debit card, you won't be charged until you manually activate your full account.

Payments profile

Georgia Institute of Technology  
Organization • United States • ID: 0001 0010 0070 [Change](#) 

Your payment information is saved in a payments profile, which is associated with your Google Account and shared across Google services. [Learn more about payments profile](#)

Payment method

 Amex \*\*\*\* 2000 [Change](#)

[START FREE](#)

## Access to all Google Cloud products








Get everything you need to build and run your apps, websites and services, including Firebase and the Google Maps API.

## \$300 credit for free


Put Google Cloud to work with \$300 in credit to spend over the next 90 days.

## No autocharge after free trial ends

We ask you for your credit card to make sure you are not a robot. If you use a credit or debit card, you won't be charged unless you manually activate your full account.

Google Cloud my-sonarqube-project       

← Product details

 **Compute Engine API**  
[Google Enterprise API](#)  
Compute Engine API

[ENABLE](#) [TRY THIS API](#)

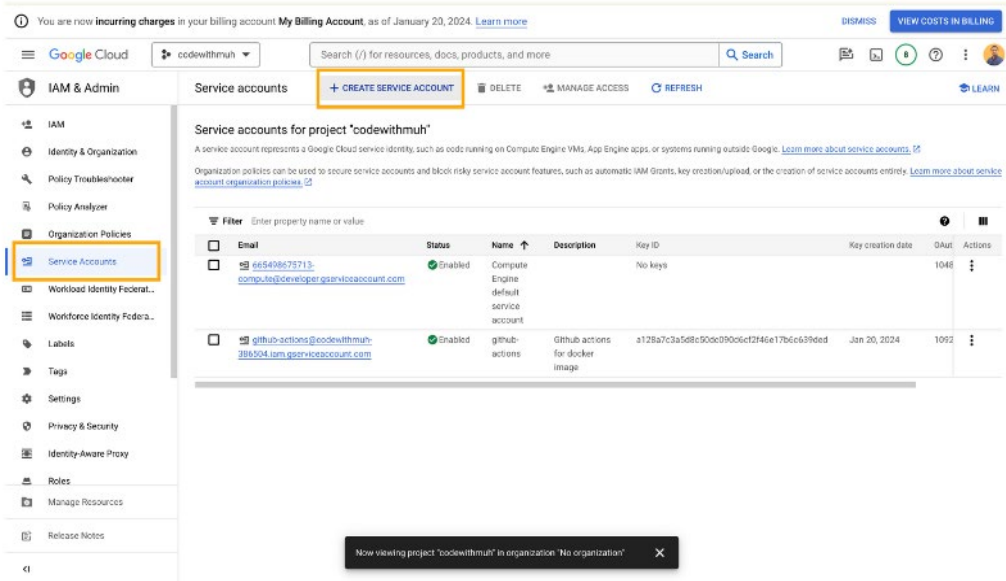
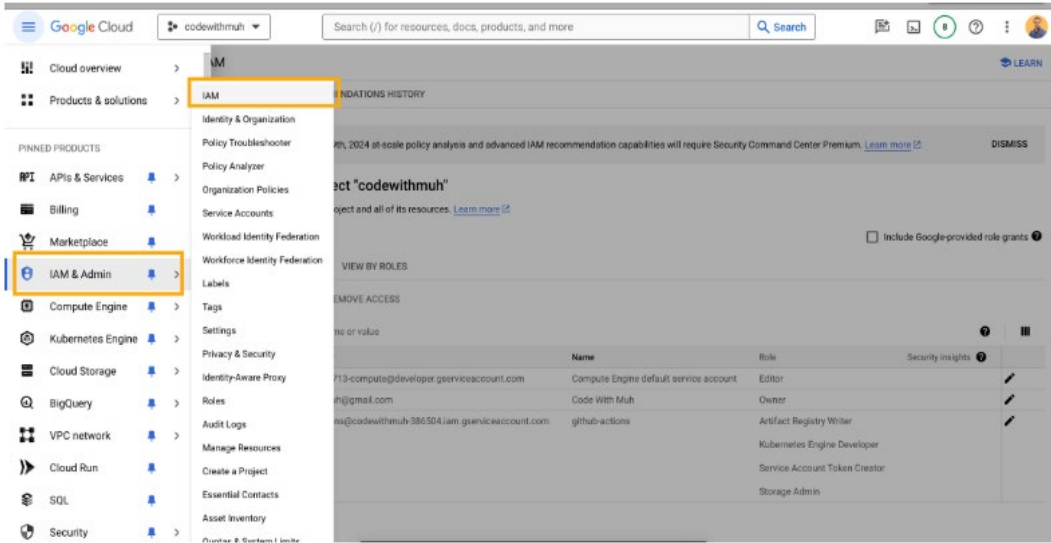
[OVERVIEW](#) [DOCUMENTATION](#) [SUPPORT](#) [RELATED PRODUCTS](#)

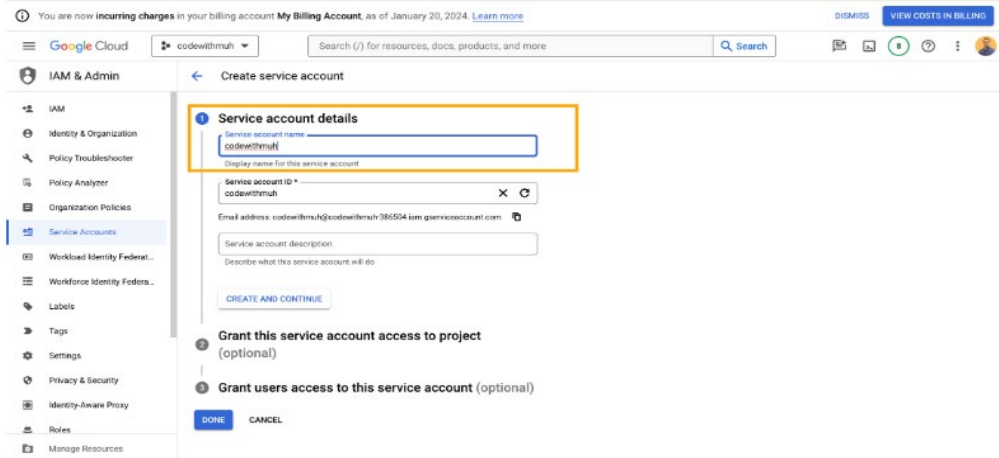
### 3. Navigate to IAM & Admin

In the left sidebar, scroll down and select **IAM & Admin** to open the IAM Dashboard.

### 4. Create a Service Account

- On the left-hand menu, select **Service Accounts**.
- Click **+ CREATE SERVICE ACCOUNT** at the top.
- Name your service account (e.g., sonarqube-service-account).
- Optionally, add a description, then click **CREATE AND CONTINUE**.





## 5. Grant Access to the Service Account

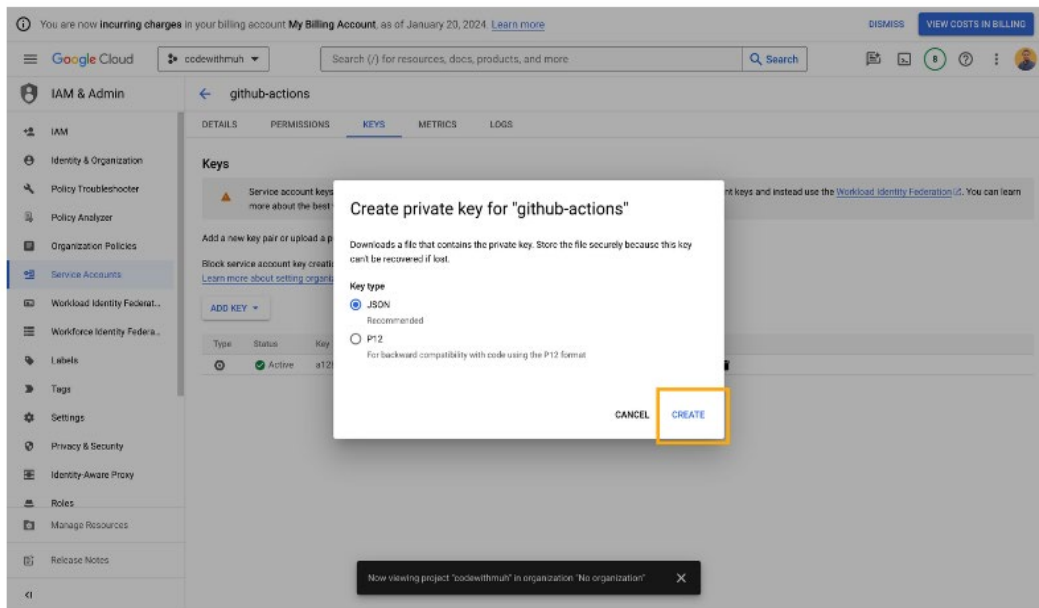
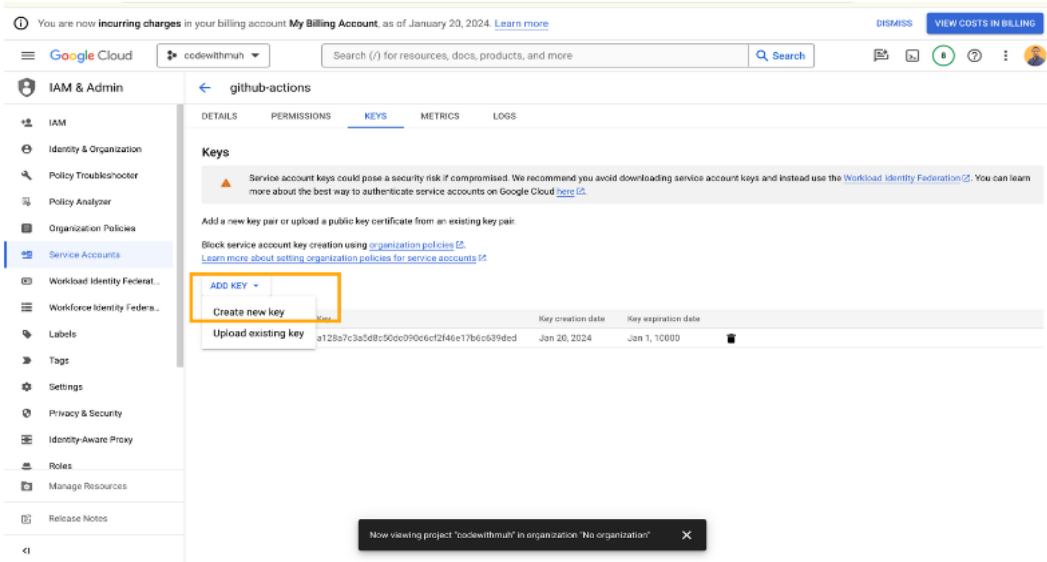
- a. On the next page, under the **Role** section:
  - i. Click **Select a role**.
  - ii. Choose **Storage > Storage Admin**.
- b. Click **Continue**.
- c. Skip the **Grant users access** step by clicking **Done**.

You now have a service account with the **Storage Admin** role.

## 6. Download the JSON Key

- a. In the list of service accounts, find the account you just created.
- b. Click the three dots (options) on the right side, then select **Manage Keys**.
- c. Click **ADD KEY > Create new key**.
- d. In the pop-up window:
  - i. Select **JSON** as the key type.
  - ii. Click **Create**.

This will download a JSON file to your computer, which you'll use later in GitHub Actions.



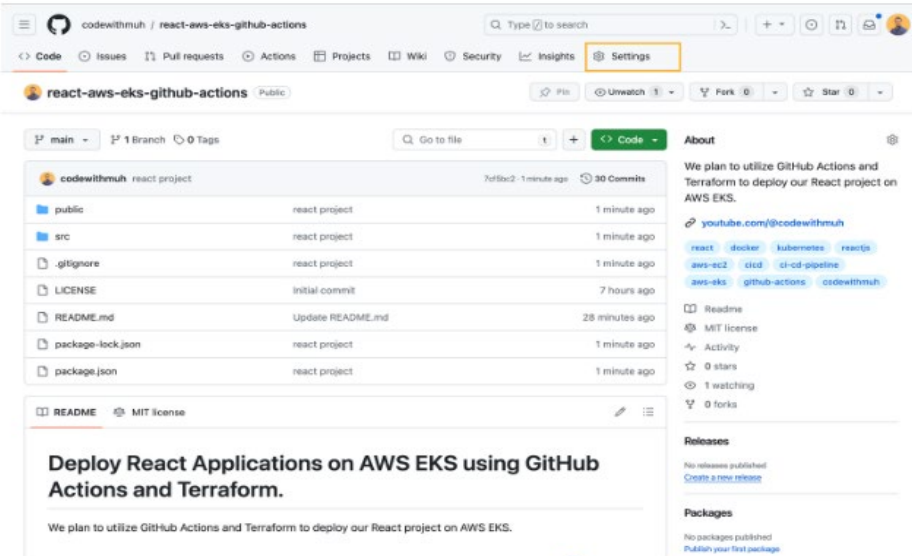
## Step 2: Configuring GitHub Actions with Secrets

### 1. Push Your Code to GitHub:

You should be using your group's GitHub repository. The root folder should have 3 folders: Project, SOLID\_GRASP, and Code Smells. SonarQube will be analyzing all 3 folders to create a leaderboard for the project, SOLID\_GRASP, and Code smell assignments.

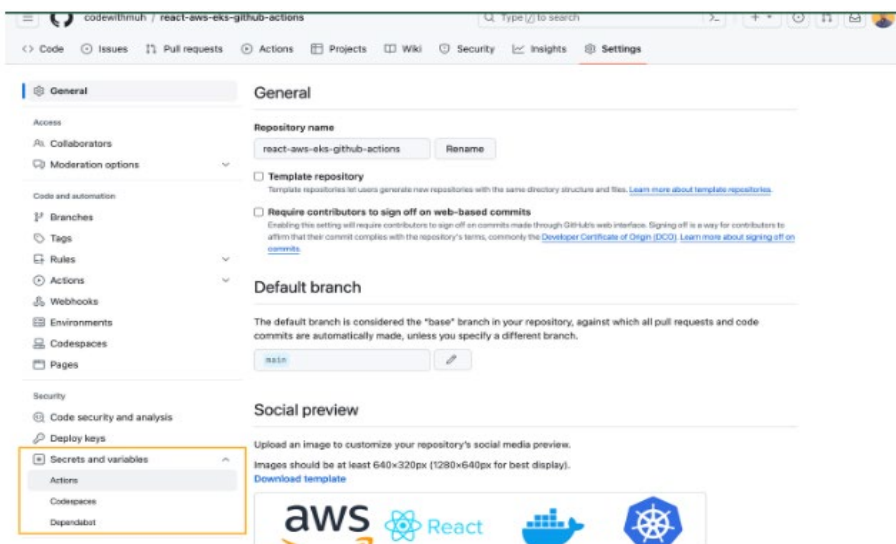
## 2. Add Secrets to GitHub Actions:

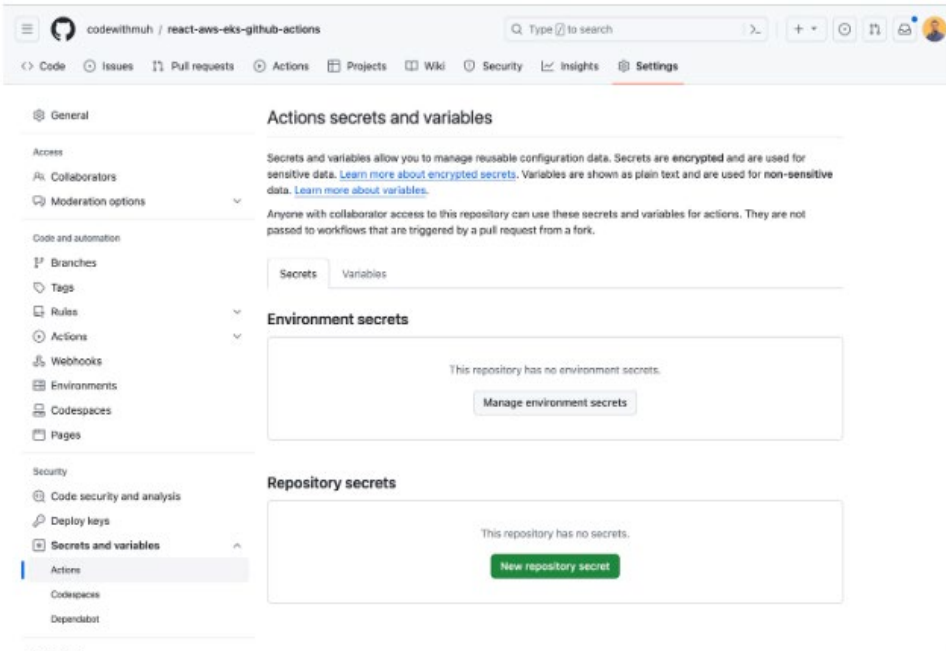
- a. Go to your repository page on GitHub.
- b. Click the **Settings** tab at the top.



Click on Secrets and Variables and then click on Actions.

- c. In the left sidebar, click **Secrets and variables**, then click **Actions**.
- d. Click **New repository secret**. You'll add the following secrets:
  - i. **Google Application Credentials**:
    1. For **Name**, enter GOOGLE\_APPLICATION\_CREDENTIALS
    2. For **Value**, open the JSON key file you downloaded earlier, copy its entire content, and paste it into the value field. Click **Add secret**.





ii. **Google Project ID:**

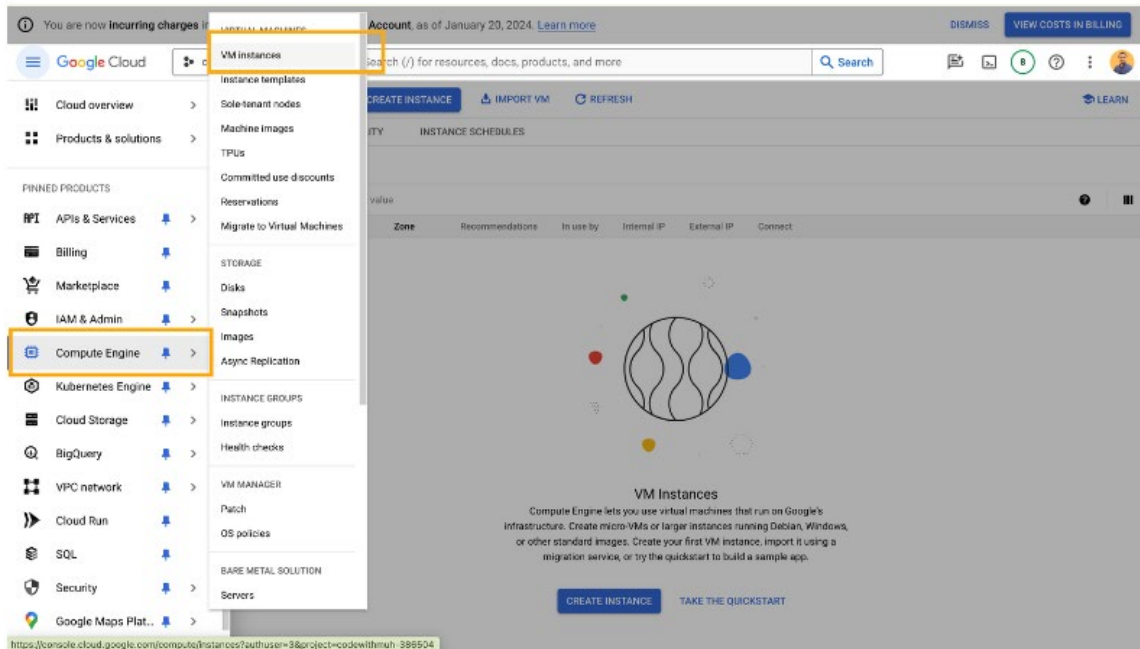
1. Create a new repository secret on the same page as the picture above.
2. For **Name**, enter `GOOGLE_PROJECT_ID`
3. For **Value**, go to the GCP Console, click the project name in the top navigation bar to view your **Project ID**, and copy/paste it here. Click **Add secret**.

## Step 3: Setting Up Google Compute Engine VM

### 1. Navigate to Compute Engine:

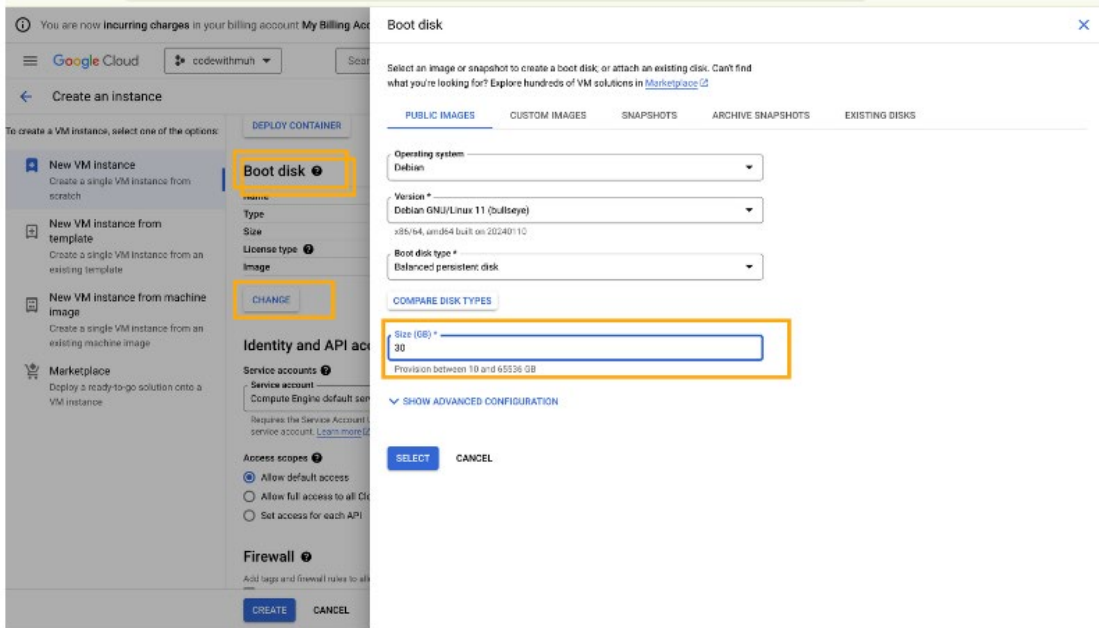
- a. In the GCP Console, click the **Navigation Menu** (three horizontal lines in the top left corner).
- b. Scroll down to **Compute Engine > VM Instances**.
- c. Click **Create Instance** at the top of the page.



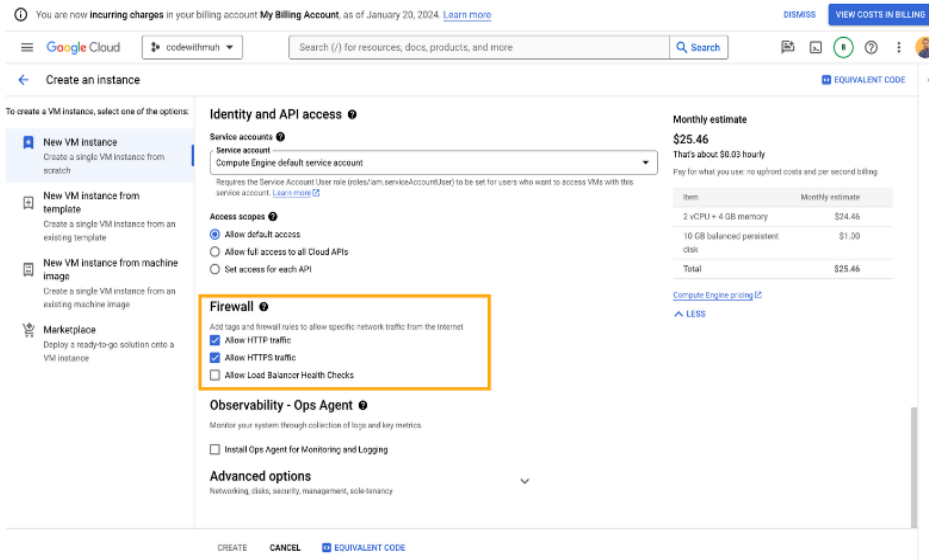


## 2. Configure the VM Instance:

- a. **Name:** Enter a name, e.g., sonarqube-vm
- b. **Region:** Choose a region close to your location for better performance (should be us-east1 for Atlanta. You can choose zone b, c, or d).
- c. **Machine Type:** Set to **e2-medium (2 vCPU, 4 GB memory)** for adequate resources.
- d. **Boot Disk:** Click **Change** and:
  - i. Select **Debian GNU/Linux** (This should be chosen regardless of your OS)
  - ii. Set **Size** to **30 GB**.
  - iii. Click **Select**.



e. **Firewall:** Check both **Allow HTTP traffic** and **Allow HTTPS traffic**.



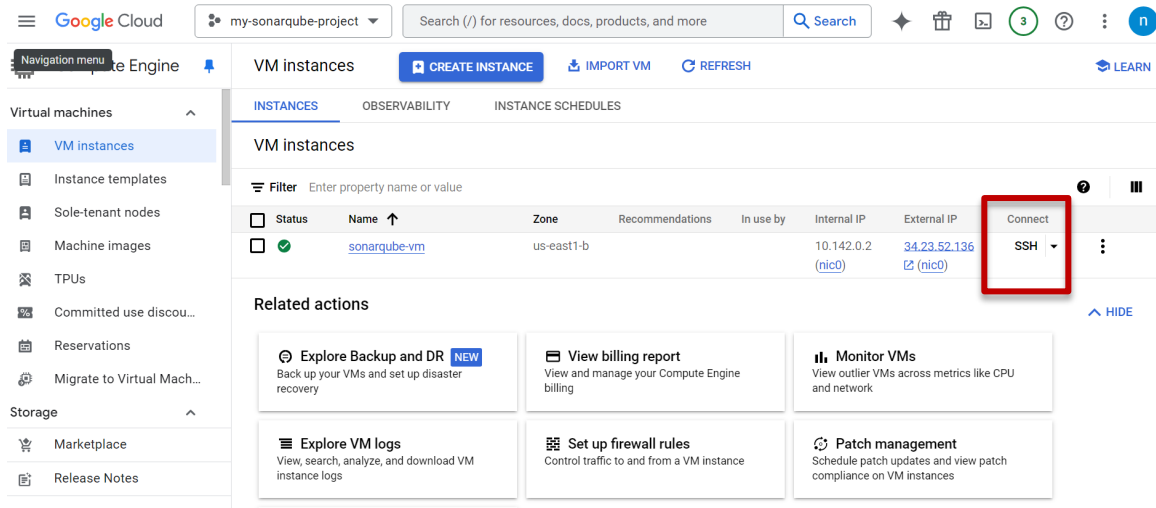
3. Create the VM:

Click **Create** at the bottom of the page.

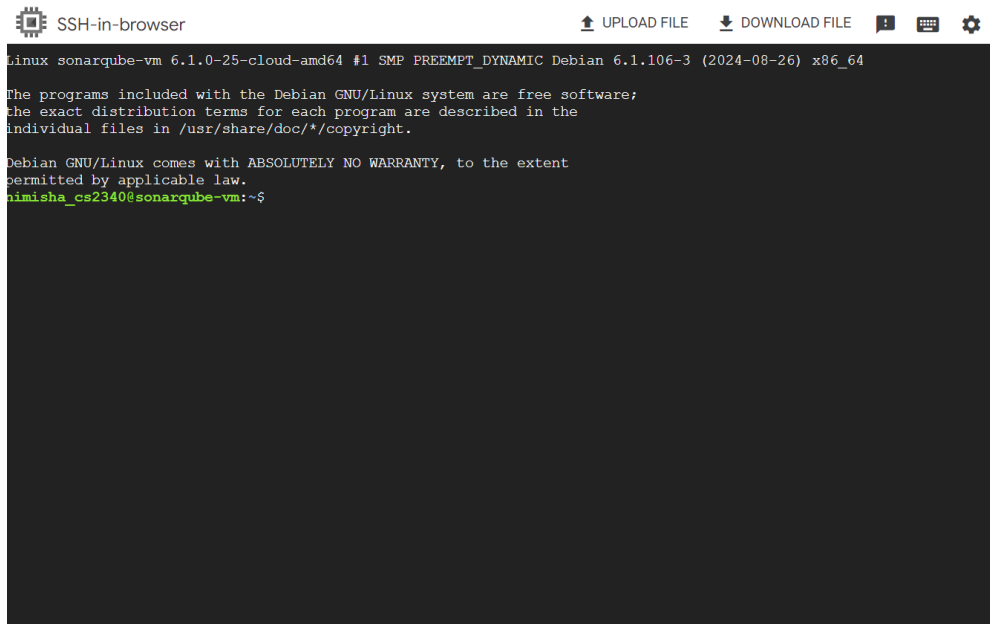
4. SSH into the VM:

a. Once your VM is created, click **SSH** on the VM instance page. To access the SSH, go to VM instances page and click the **SSH button** underneath **Connect**

b. A terminal will open where you can run commands on your VM.



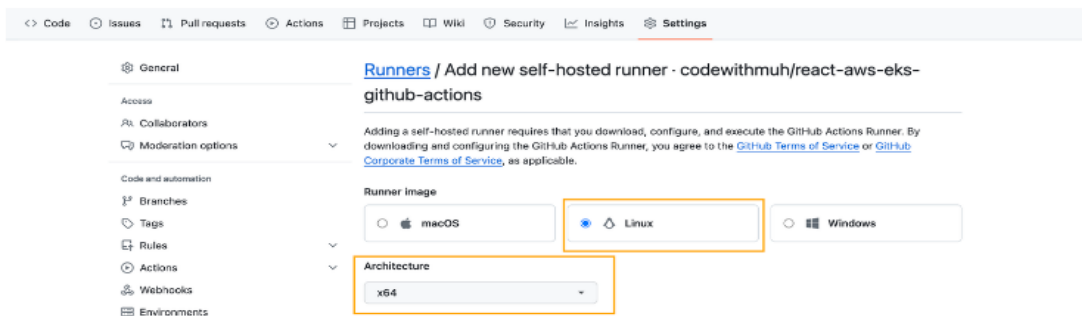
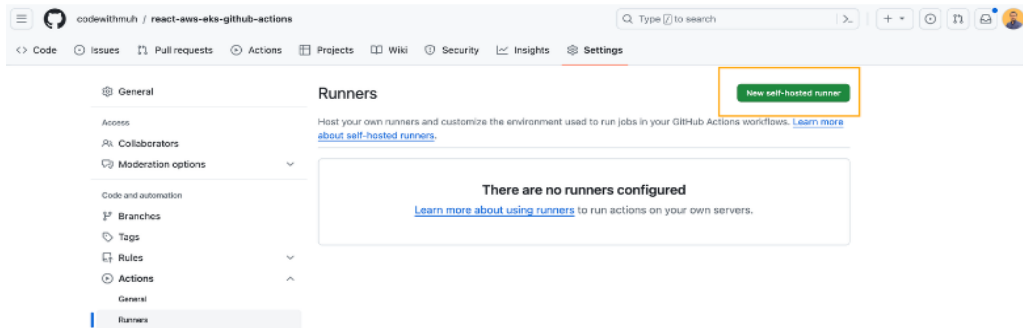
The screenshot shows the Google Cloud console interface for VM instances. The left sidebar contains navigation options like 'Virtual machines', 'Instance templates', and 'Storage'. The main content area displays a table of VM instances. One instance, 'sonarqube-vm', is highlighted. A red box highlights the 'Connect' button, which has a dropdown menu with 'SSH' selected. Below the table, there are several 'Related actions' cards such as 'Explore Backup and DR', 'View billing report', 'Monitor VMs', 'Explore VM logs', 'Set up firewall rules', and 'Patch management'.



The screenshot shows an 'SSH-in-browser' terminal window. The terminal output displays the Linux prompt: 'Linux sonarqube-vm 6.1.0-25-cloud-amd64 #1 SMP PREEMPT\_DYNAMIC Debian 6.1.106-3 (2024-08-26) x86\_64'. Below this, there is a copyright notice for Debian GNU/Linux. The terminal prompt is 'nimiasha\_cs2340@sonarqube-vm:~\$'.

## 5. Add a Self-hosted Runner to Google Compute Engine Instance:

- Now go to the GitHub repository and click on Settings → Actions → Runners. Click on Self-Hosted Runners
- Now select Linux and architecture X-64 (Again, this should be chosen regardless of the OS of your laptop).



## Download

```
# Create a folder
$ mkdir actions-runner && cd actions-runner

# Download the latest runner package
$ curl -o actions-runner-linux-x64-2.319.1.tar.gz -L
https://github.com/actions/runner/releases/download/v2.319.1/actions-runner-linux-x64-2.319.1.tar.gz

# Optional: Validate the hash
$ echo "3f6efb7488a183e291fc2c62876e14c9ee732864173734facc85a1bfb1744464 actions-runner-linux-x64-
2.319.1.tar.gz" | shasum -a 256 -c

# Extract the installer
$ tar xzf ./actions-runner-linux-x64-2.319.1.tar.gz
```

## Configure

```
# Create the runner and start the configuration experience
$ ./config.sh --url https://github.com/Nimisha-Roy/sonarqube_tetsing --token A0TJED6U7PTVQCCVJONG2Q3G74B4G

# 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
```

Sign in now to u

- c. Copy the create folder command: **`mkdir actions-runner && cd actions-runner`** from the above picture for my repository.
- d. Paste this into your SSH-in-browser terminal window

SSH-in-browser

Linux sonarqube-vm 6.1.0-25-cloud-amd64 #1 SMP PREEMPT\_DYNAMIC Debian 6.1.106-3 (2024-08-26) x86\_64

The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/\*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

```
nimisha_cs2340@sonarqube-vm:~$ mkdir actions-runner && cd actions-runner
nimisha_cs2340@sonarqube-vm:~/actions-runner$
```

- e. Download the latest runner package and extract the installer using commands on your github repo: **`curl -o actions-runner-linux-x64-2.319.1.tar.gz -L https://github.com/actions/runner/releases/download/v2.319.1/actions-runner-linux-x64-2.319.1.tar.gz`** and **`tar xzf ./actions-runner-linux-x64-2.319.1.tar.gz`** for me.

```
nimisha_cs2340@sonarqube-vm:~/actions-runner$ curl -o actions-runner-linux-x64-2.319.1.tar.gz -L https://github.com/actions/runner/releases/download/v2.319.1/actions-runner-linux-x64-2.319.1.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 0:00:00 0:00:00 0:00:00 0
100 208M 100 208M 0 0 197M 0 0:00:01 0:00:01 --:--:-- 251M
nimisha_cs2340@sonarqube-vm:~/actions-runner$ tar xzf ./actions-runner-linux-x64-2.319.1.tar.gz
nimisha_cs2340@sonarqube-vm:~/actions-runner$
```

- f. Create the runner and start the configuration experience according to commands on your GitHub repository.

```
nimisha_cs2340@sonarqube-vm:~/actions-runner$ ./config.sh --url https://github.com/Nimisha-Roy/sonarqube_tetsing --token AOTJED6U7PTVQCCVJONG2Q3G74B4G

-----
GitHub Actions
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]
Enter the name of runner: [press Enter for sonarqube-vm]

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

√ Runner successfully added
√ Runner connection is good

# Runner settings

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

√ Settings Saved.
```

g. The last step is to run it: **./run.sh**

```
nimisha_cs2340@sonarqube-vm:~/actions-runner$ ./run.sh

√ Connected to GitHub

Current runner version: '2.319.1'
2024-10-03 20:14:11Z: Listening for Jobs
█
```

Let's close Runner for now.

```
ctrl + c # To close Run this Command.
```

## Step 4: Installing Docker and SonarQube on the VM

### 1. Install Docker:

a. In the SSH terminal, run the following commands to install Docker:

```
sudo apt-get update
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
843C48A565F8F04B
sudo apt-get update
```

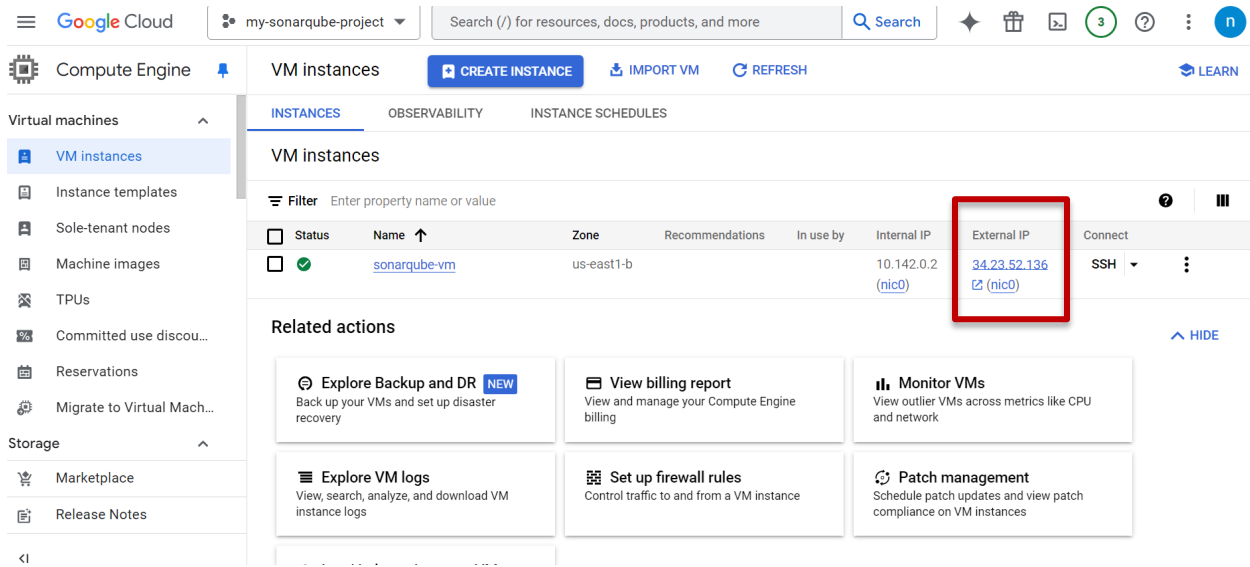
```
sudo apt-get install -y jq
sudo apt-get install docker.io -y
sudo usermod -aG docker ubuntu newgrp docker
sudo chmod 777 /var/run/docker.sock
```

b. Next, pull the SonarQube image from Docker Hub:

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

Note that we are using port 9000 in the firewall of our instance.

c. Now copy the external IP address of your Google engine instance as shown the image below.



The screenshot shows the Google Cloud Platform console for a project named 'my-sonarqube-project'. The 'VM instances' page is active, displaying a table of instances. The 'sonarqube-vm' instance is listed with an internal IP of 10.142.0.2 and an external IP of 34.23.52.136. The external IP is highlighted with a red box. Below the table, there are several 'Related actions' such as 'Explore Backup and DR', 'View billing report', 'Monitor VMs', 'Explore VM logs', 'Set up firewall rules', and 'Patch management'.

For example, mine would be **<34.23.52.136:9000>** (append **:9000** at the end)

## 2. Update Firewall Rules:

a. To access SonarQube on port 9000, you need to update the firewall rules:

i. In the GCP Console, click **VPC network > Firewall**

ii. Click **Create firewall rule**.

iii. Set the following:

1. **Name:** allow-sonarqube

2. **Targets:** All instances in the network.

3. **Source IP ranges:** 0.0.0.0/0

4. **Protocols and Ports:** Allow **TCP** on port 9000.

iv. Click **Create**.

my-sonarqube-project Search (/) for resources, docs, products, and more Search

### Create a firewall rule

**Targets**  
All instances in the network

**Source filter**  
IPv4 ranges

**Source IPv4 ranges \***  
0.0.0.0/0

**Second source filter**  
None

**Destination filter**  
None

**Protocols and ports**

Allow all

Specified protocols and ports

TCP

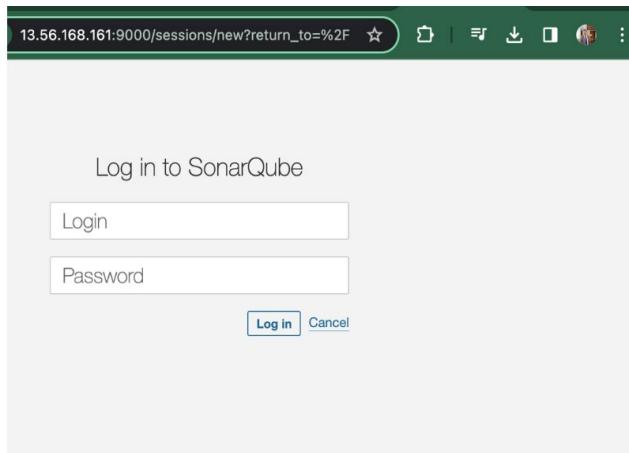
Ports  
9000

## Step 5: Accessing SonarQube Dashboard

### 1. Access SonarQube:

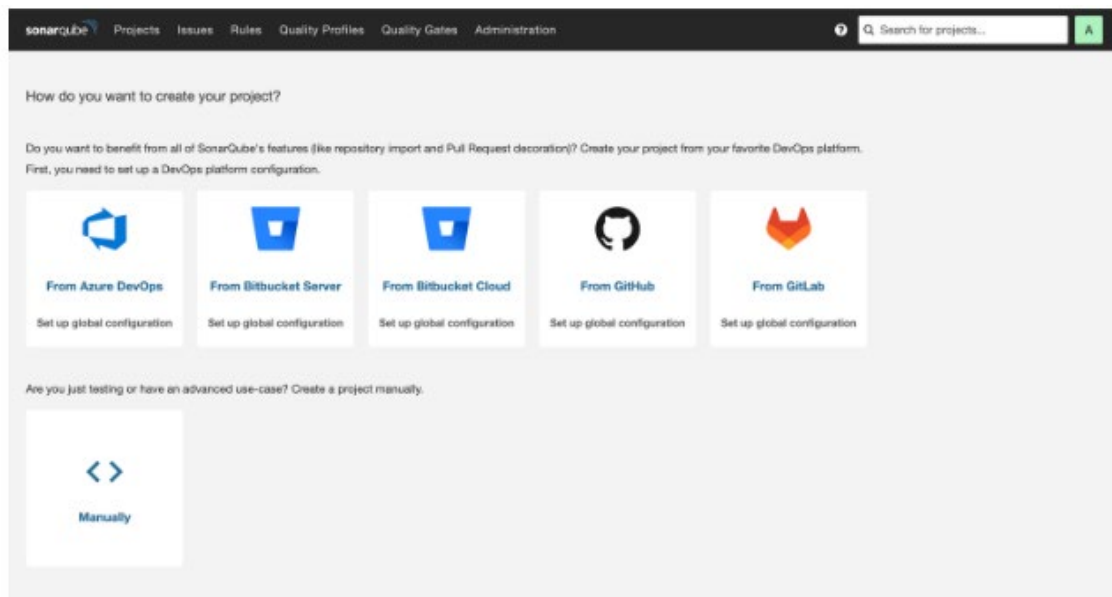
- a. In your browser, go to `http://<your-vm-external-ip>:9000`
- b. In my case, it is: `http://34.23.52.136:9000`
- c. Use the default credentials:
  - a. Username: `admin`
  - b. Password: `admin`
- d. You'll be prompted to change the password after your first login.
  - a. In case you are having issues with login, clear browser cookies.
- e. Change your password to `cs2340`**





## 2. Integrate SonarQube with GitHub Actions

- a. Go to SonarQube Dashboard
- b. Under How to create your project, Click on Manually.



- c. Enter a project display name and key. Use **cs2340**

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

## Create a project

All fields marked with \* are required

**Project display name \***

 ✓

Up to 255 characters. Some scanners might override the value you provide.

**Project key \***

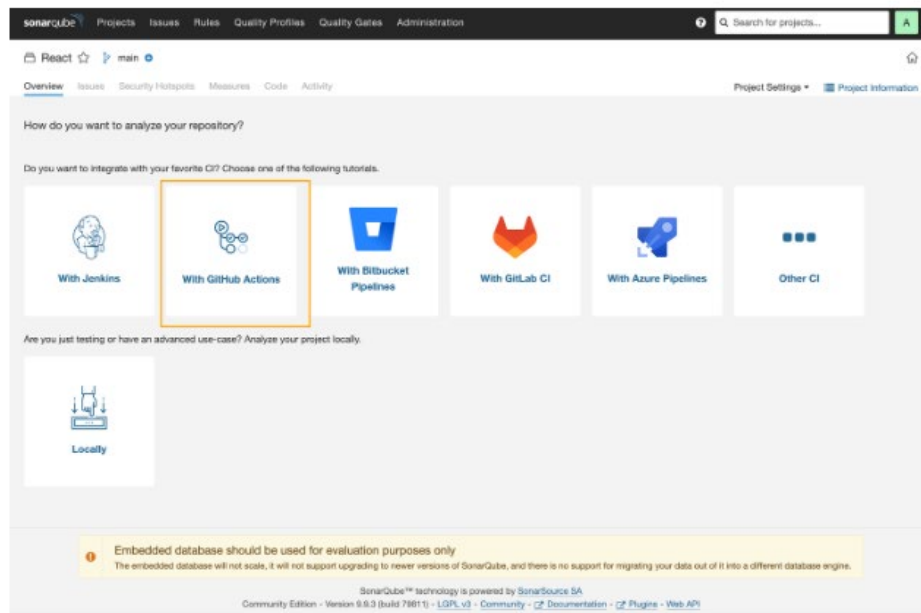
 ✓

The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '\_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

**Main branch name \***

The name of your project's default branch [Learn More](#)

d. On the next page, Click “With GitHub Actions”




### 3. Generate a Token for GitHub Actions:

- a. Click on Generate Token

## 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:

[Generate a token](#)

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://34.23.52.136:9000` 

4. Click on **Add secret**.

[Continue](#)

- b. Name the token (e.g., `github-actions`) ,expires in 90 days and click **Generate**. If asked, select token type as global analysis token.
- c. Copy the generated token.


## 4. Add GitHub secrets:

- a. Go to GitHub repository → Settings → Secrets and Variables → Actions → Add Secret
- b. Write name as `SONAR_TOKEN` and paste your generated token as the secret.
- c. Add another secret. Write name as `SONAR_HOST_URL` and paste the url shown in the value field as the secret.

## 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: [Generate a token](#)

4. Click on **Add secret**.

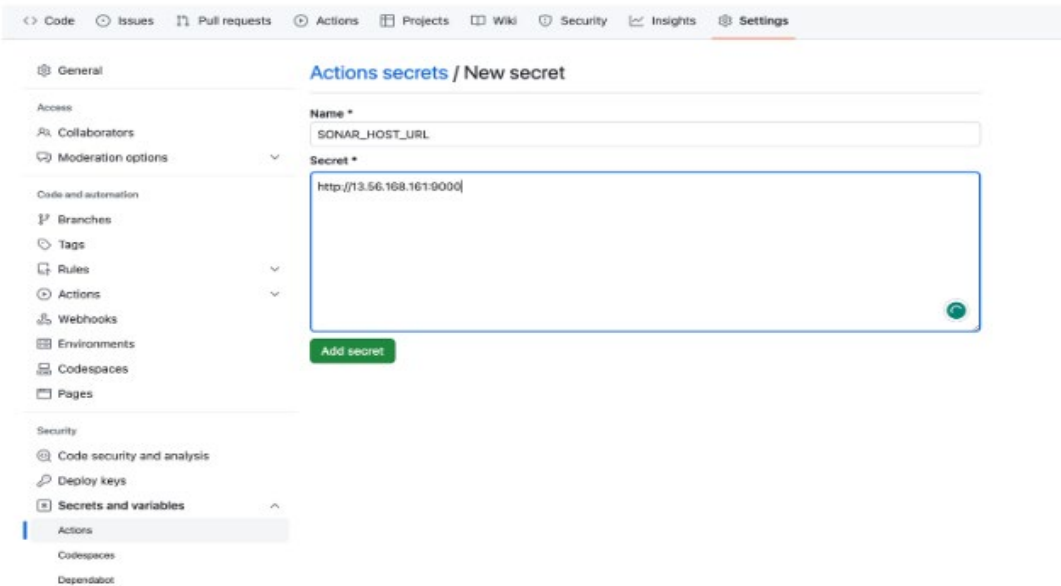
1. Click on **New repository secret**.

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

3. In the **Value** field, enter `http://34.23.52.136:9000` 

4. Click on **Add secret**.

[Continue](#)



The screenshot shows the GitHub repository settings page, specifically the 'Secrets and variables' section under 'Actions'. The 'New secret' form is visible, with the following details:

- Name:** SONAR\_HOST\_URL
- Secret:** http://13.56.168.161:9000
- Buttons:** 'Add secret' (green) and a 'Cancel' button (grey).

The left sidebar shows the navigation menu with 'Secrets and variables' expanded under the 'Security' section.

Repository secrets		New repository secret	
Name		Last updated	
GOOGLE_APPLICATION_CREDENTIALS		4 hours ago	
GOOGLE_PROJECT_ID		4 hours ago	
SONAR_HOST_URL		4 minutes ago	
SONAR_TOKEN		4 minutes ago	

## 5. Go to the Sonarqube dashboard again

Click on Other and keep this tab open for Step 6.

### 2 Create Workflow YAML File

1. What option best describes your build?

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

```
sonar.projectKey=cs2340
```

Copy

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
```

## Step 6: Setting Up GitHub Actions Workflow

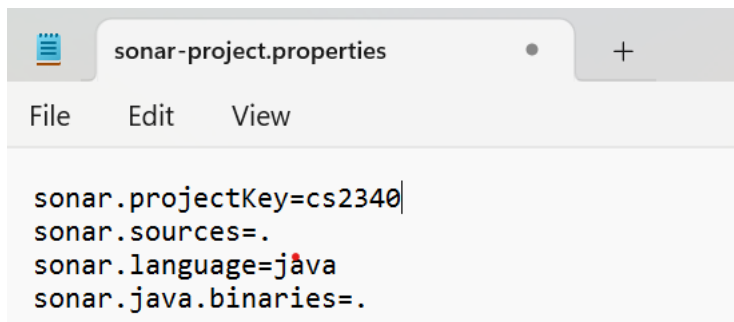
### 1. Create GitHub Actions Workflow:

- In your github's root directory, create a folder `.github/workflows/build.yml`. So there will be a folder at the root of your project called `.github`, then another folder called `workflows`, and inside is the file called `build.yml`. **.github → workflows → build.yml**
- Copy the contents from SonarQube into `build.yml` file (point 3 in the above image).

## 2. Create sonar-project.properties files

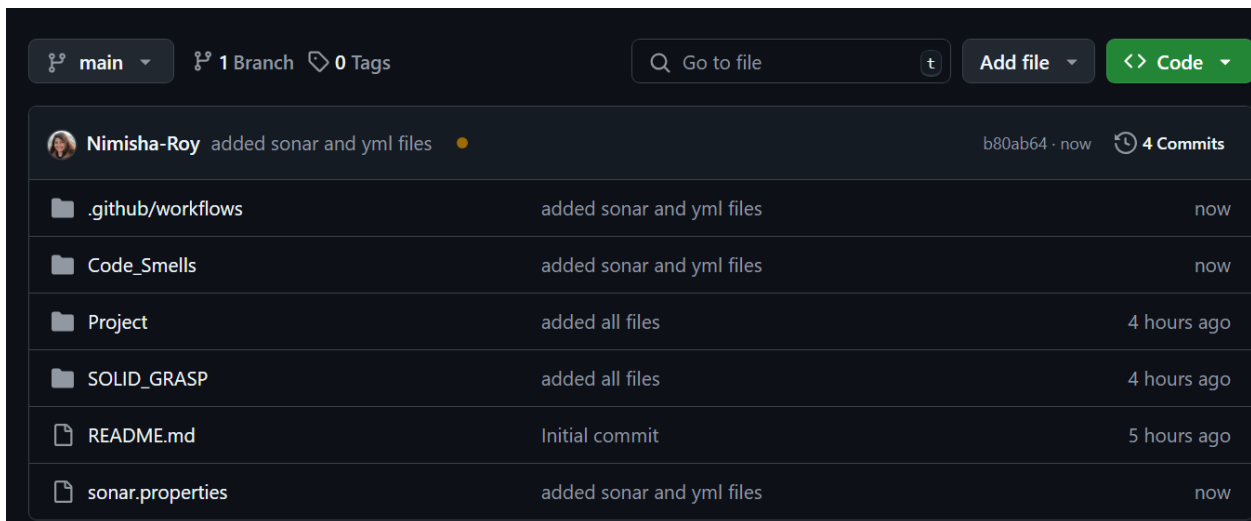
- a. In the same root directory, add another file called sonar.properties.
- b. Use the correct sonar-project.properties file **here**, **not the one in the YouTube video**:

```
sonar.projectKey=YOUR_PROJECT_KEY
sonar.sources=.
sonar.language=java
sonar.java.binaries=.
```

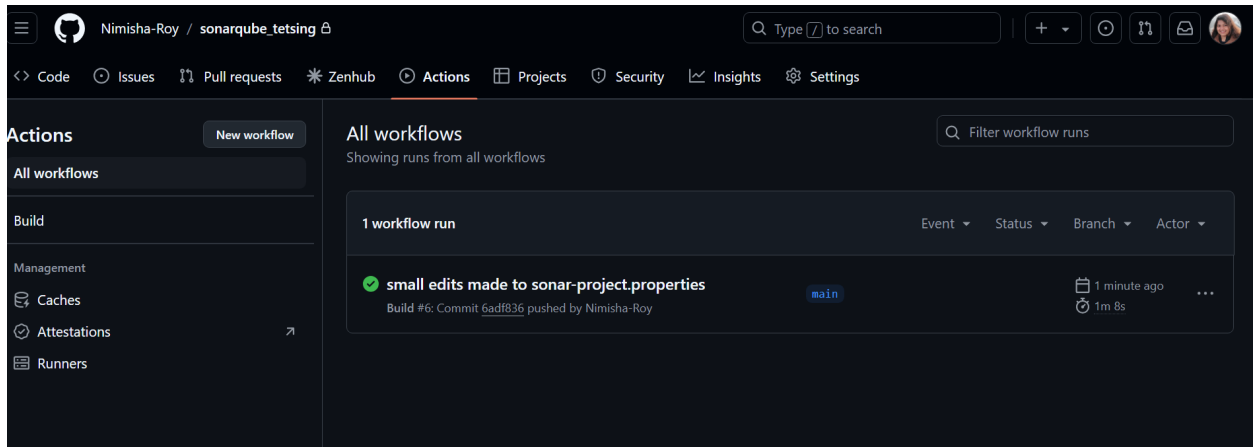


## 3. Push Changes to GitHub:

- a. Push your changes to the GitHub repository.



- b. You can go to Actions and see build success. This means that trigger GitHub Actions was triggered successfully on pushing.



- c. SonarQube scan will start automatically. You will see the results on your SonarQube browser after you click Finish Tutorial.

## Step 7: Final Things To Do - **\*\*IMPORTANT\*\***

### 1. Ensure Project is Compiled:

Make sure your project/assignment is properly compiled and that the **.class files** are present before running the SonarQube analysis. Otherwise, your GitHub Actions build may fail.

### 2. Correct SonarQube Properties File:

Use the correct `sonar-project.properties` file **here**, **not the one in the YouTube video**. **Check Step 6 point 2b.**

## **FINALLY! Look at the analysis!!**

The analysis you see is for your entire GitHub repository, including your project, SOLID\_GRASP, and code smell assignments. There will be three assignments on GradeScope corresponding to these three submissions, and the output from autograder will be the specific output from the three separate files.

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

cs23401 main Last analysis of this Branch had 2 warnings October 3, 2024 at 8:01 PM Version not provided

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

QUALITY GATE STATUS **Passed**  
All conditions passed.

MEASURES

New Code	Overall Code
3 Bugs	Reliability <span>C</span>
7 Vulnerabilities	Security <span>E</span>
10 Security Hotspots	0.0% Reviewed Security Review <span>E</span>
2d 5h Debt	164 Code Smells Maintainability <span>A</span>
0.0% Coverage on 1.6k Lines to cover	- Unit Tests
	0.3% Duplications on 39k Lines Duplicated Blocks 6

You can click on each type of error and understand and fix it. You can also go to issues to look at the errors in detail.

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

cs23401 main Last analysis of this Branch had 2 warnings October 3, 2024 at 8:01 PM Version not provided

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

My Issues All 1 / 174 issues 3d effort

Filters

- Issues in new code
- Type
  - Bug 3
  - Vulnerability 7
  - Code Smell 164
- Severity
  - Blocker 2
  - Critical 24
  - Major 83
  - Minor 62
  - Info 3
- Scope
- Resolution
- Status
- Security Category
- Creation Date
- Language
- Rule
- Tag

Project/app/google-services.json

- Make sure this Google API Key is not disclosed. 5 hours ago - L18 🔗 🔽  
 Vulnerability 🔒 Blocker 🔴 Open 🔵 Not assigned 🕒 30min effort 🗨️ Comment 🔗 cwe, owasp-a3, sans-top25-porous

Project/app/src/main/AndroidManifest.xml

- Implement permissions on this exported component. 5 hours ago - L20 🔗 🔽  
 Vulnerability 🔒 Major 🔴 Open 🔵 Not assigned 🕒 10min effort 🗨️ Comment 🔗 android, cwe, sans-top25-porous
- Implement permissions on this exported component. 5 hours ago - L40 🔗 🔽  
 Vulnerability 🔒 Major 🔴 Open 🔵 Not assigned 🕒 10min effort 🗨️ Comment 🔗 android, cwe, sans-top25-porous
- Implement permissions on this exported component. 5 hours ago - L44 🔗 🔽  
 Vulnerability 🔒 Major 🔴 Open 🔵 Not assigned 🕒 10min effort 🗨️ Comment 🔗 android, cwe, sans-top25-porous
- Implement permissions on this exported component. 5 hours ago - L48 🔗 🔽  
 Vulnerability 🔒 Major 🔴 Open 🔵 Not assigned 🕒 10min effort 🗨️ Comment 🔗 android, cwe, sans-top25-porous
- Implement permissions on this exported component. 5 hours ago - L52 🔗 🔽  
 Vulnerability 🔒 Major 🔴 Open 🔵 Not assigned 🕒 10min effort 🗨️ Comment 🔗 android, cwe, sans-top25-porous

Project/...com/example/greenplate/models/CalorieCountDecorator.java

- Replace this use of System.out or System.err by a logger. 5 hours ago - L47 🔗 🔽  
 Code Smell 🔒 Major 🔴 Open 🔵 Not assigned 🕒 10min effort 🗨️ Comment 🔗 bad-practice, cert, owasp-a3
- Replace this use of System.out or System.err by a logger. 5 hours ago - L52 🔗 🔽  
 Code Smell 🔒 Major 🔴 Open 🔵 Not assigned 🕒 10min effort 🗨️ Comment 🔗 bad-practice, cert, owasp-a3