

Homework – Analysis and Review Code

Javier Augustson

Buat 1 pipeline cicd yang terintegrasi dengan sonarqube dan jelaskan step by stepnya.

Jawab :

Untuk melakukan hal tersebut, akan digunakan CI/CD pada Github Action. SonarQube akan digunakan secara self-hosted yang berarti Sonarqube akan diinstal ke lokal host.

Karena SonarQube menggunakan self-hosted, maka diperlukan runner Github Action secara self-hosted. Secara default Runner Github Action berjalan di cloud, sehingga tidak bisa mengakses localhost di computer lokal.

Install SonarQube ke local host

Local host yang digunakan merupakan Mac sehingga memiliki operator perintah yang sedikit berbeda jika menggunakan Ubuntu.

Untuk memulai rangkaian instalasi, pastikan Homebrew sudah terinstal pada Mac.

Ketik `brew --version` untuk cek Homebrew. Jika belum ada dapat lakukan **instalasi Homebrew** :

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

```
javiersosial@javiers-MacBook-Pro ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

==> Checking for `sudo` access (which may request your password)...
Password:
==> This script will install:
/usr/local/bin/brew
/usr/local/share/doc/homebrew
/usr/local/share/man/man1/brew.1
/usr/local/share/zsh/site-functions/_brew
/usr/local/etc/bash_completion.d/brew
/usr/local/Homebrew
```

Cek jika Homebrew telah berhasil di install :

`brew --version`

```
javiersosial@javiers-MacBook-Pro ~ % brew --version
Homebrew 4.4.22
```

Jika homebrew sudah terinstal, dapat dilanjutkan dengan **instalasi java** :

```
brew install openjdk@17
```

```
javiersosial@javiers-MacBook-Pro ~ % brew install openjdk@17
==> Downloading https://ghcr.io/v2/homebrew/core/openjdk/17/ma
#####
```

Cek jika instalasi Java telah berhasil :

```
java -version
```

```
javiersosial@javiers-MacBook-Pro ~ % java -version
openjdk version "17.0.14" 2025-01-21
OpenJDK Runtime Environment Homebrew (build 17.0.14+0)
OpenJDK 64-Bit Server VM Homebrew (build 17.0.14+0, mixed mode, sharing)
```

Setelah java berhasil di Install, Langkah selanjutnya adalah **instalasi SonarQube** :

```
brew install sonarqube
```

```
javiersosial@javiers-MacBook-Pro ~ % brew install sonarqube
==> Downloading https://formulae.brew.sh/api/formula.jws.json
==> Downloading https://formulae.brew.sh/api/cask.jws.json
```

Setelah instalasi selesai, SonarQube dijalankan di background dan otomatis restart saat login dengan perintah :

```
brew services start sonarqube
```

```
javiersosial@javiers-MacBook-Pro ~ % brew services start sonarqube
==> Tapping homebrew/services
Cloning into '/usr/local/Homebrew/Library/Taps/homebrew/homebrew-services'...
remote: Enumerating objects: 4101, done.
remote: Counting objects: 100% (537/537), done.
remote: Compressing objects: 100% (205/205), done.
```

Untuk cek apakah SonarQube sudah berjalan, digunakan perintah :

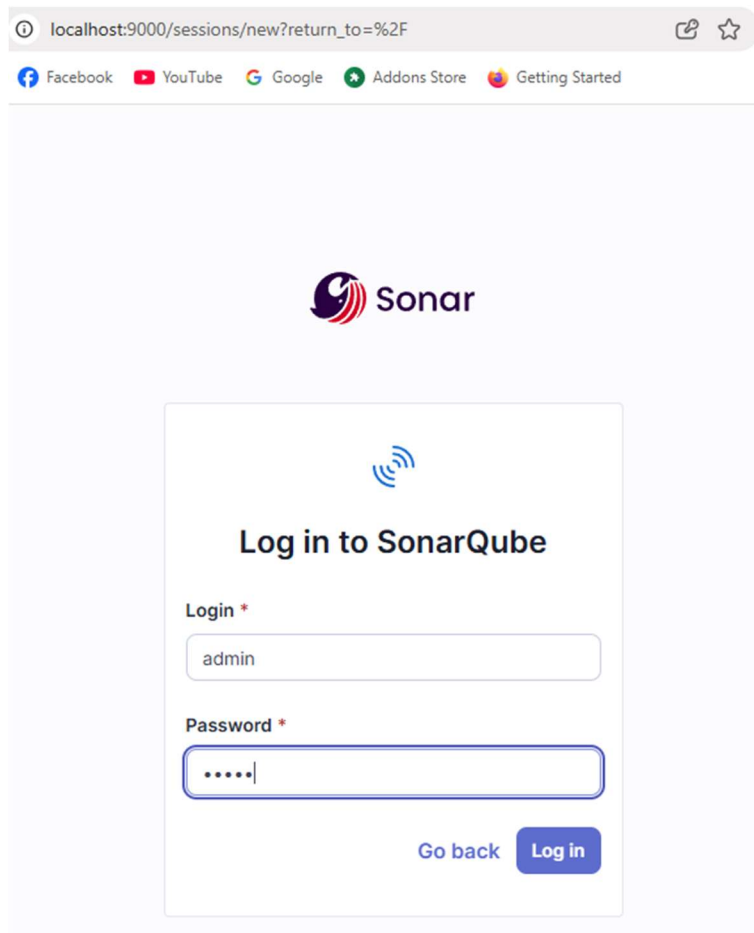
```
brew services list
```

```
javiersosial@javiers-MacBook-Pro ~ % brew services list

Name      Status  User           File
sonarqube started javiersosial ~/Library/LaunchAgents/homebrew.mxcl.sonarqube.plist
```


Jika sudah berjalan, SonarQube dapat diakses di browser melalui :


<http://localhost:9000>



localhost:9000/sessions/new?return_to=%2F

Facebook YouTube Google Addons Store Getting Started

 Sonar



Log in to SonarQube

Login *

admin

Password *

.....

Go back Log in

Gunakan login : admin, password : admin. Setelah masuk akan diminta membuat password baru.

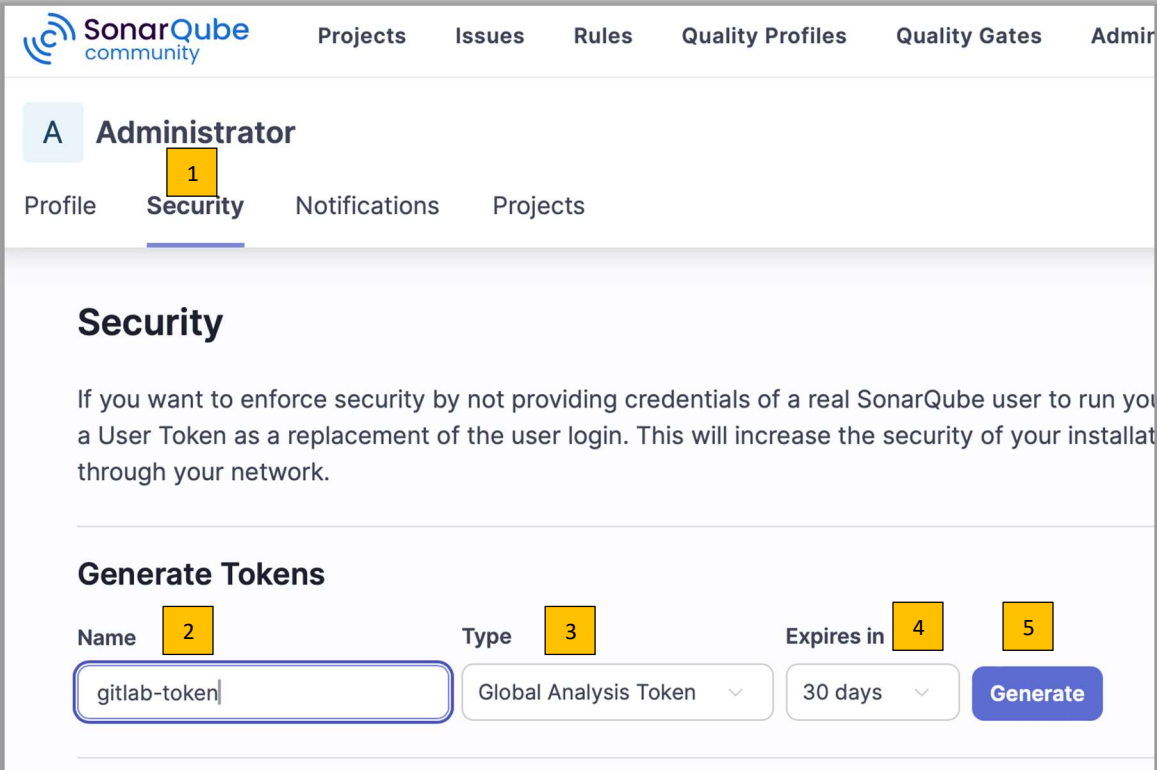
Setelah berhasil masuk ke halaman SonarQube, **buat token baru di SonarQube** yang akan digunakan untuk koneksi antara Github Action dengan SonarQube nantinya.

Buat Token SonarQube Baru

1. Klik ikon profil (pojok kanan atas)
2. Pilih "My Account"



3. Masuk ke tab **"Security"**
4. Pada bagian **Generate Tokens**, lakukan:
 - Masukkan nama token (misalnya: gitlab-token)
 - Pilih **scope: Global analysis token**
 - Klik **"Generate"**



The screenshot shows the SonarQube community interface. At the top, there's a navigation bar with links: Projects, Issues, Rules, Quality Profiles, Quality Gates, and Admin. Below this, the user is logged in as 'Administrator' (indicated by a blue 'A' icon). The 'Security' tab is selected, highlighted with a yellow box labeled '1'. The 'Security' section has a heading and a paragraph explaining the purpose of User Tokens. Below this is the 'Generate Tokens' section. It contains four fields: 'Name' (labeled with a yellow box '2'), 'Type' (labeled with a yellow box '3'), 'Expires in' (labeled with a yellow box '4'), and a 'Generate' button (labeled with a yellow box '5'). The 'Name' field contains the text 'gitlab-token'. The 'Type' dropdown is set to 'Global Analysis Token'. The 'Expires in' dropdown is set to '30 days'.

5. **Salin token yang muncul** (Token hanya muncul sekali!).

Token tersebut akan digunakan sebagai secrets di Github repositori supaya CI/CD dalam repo tersebut dapat mengakses SonarQube untuk dilakukan scanning.

Install Github runner self-hosted

Karena SonarQube berjalan di localhost, ada beberapa hal yang perlu diperhatikan saat menjalankan GitHub Actions:

✗ **Masalah:** GitHub Actions tidak bisa mengakses localhost

Secara default, runner GitHub Actions berjalan di cloud, sehingga tidak bisa mengakses localhost di komputer.

✓ **Solusi:** Jalankan Runner GitHub Actions di Local Machine

Agar GitHub Actions bisa berkomunikasi dengan **SonarQube di localhost**, kita harus menjalankan **self-hosted runner** di komputer.

Setup Self-Hosted GitHub Actions Runner

Jalankan runner GitHub Actions di komputer tempat SonarQube berjalan.

1 **Masuk ke repo GitHub kamu**

2 **Buka:**

- **Settings → Actions → Runners**
- Klik **"New self-hosted runner"**

The screenshot shows the GitHub Actions Runners page for a repository named 'my-project' by user 'javersosial'. The page is divided into a left sidebar with navigation links and a main content area. The 'Runners' section is active, showing a table of runners. Annotations are placed as follows:


- 1**: Points to the repository name 'my-project' in the top header.
- 2**: Points to the 'Settings' link in the top navigation bar.
- 3**: Points to the 'Runners' link in the left sidebar.
- 4**: Points to the 'New self-hosted runner' button in the top right of the Runners section.


Runners	Status
javers-MacBook-Pro self-hosted macOS X64	Offline ...


3 Pilih OS sesuai sistem kamu (MacOS/Linux/Windows)

4 Ikuti perintah instalasi di terminal, contoh untuk Mac/Linux:

Runner image

☒  macOS

☐  Linux

☐  Windows

Architecture

x64

▼

Download

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

# Download the latest runner package
$ curl -o actions-runner-osx-x64-2.322.0.tar.gz -L
https://github.com/actions/runner/releases/download/v2.322.0/actions-runner-osx-x64-2.322.0.tar.gz

# Optional: Validate the hash
$ echo "aa0fc262363912167dcdbc746ffcdf7b8996bc587f51cf1bab38ad86cf70b6ea actions-
runner-osx-x64-2.322.0.tar.gz" | shasum -a 256 -c

# Extract the installer
$ tar xzf ./actions-runner-osx-x64-2.322.0.tar.gz
```

Configure

```
# Create the runner and start the configuration experience
$ ./config.sh --url https://github.com/javiersosial/my-project --token
BMXIST4XS7JKM6EY6WR4VE3HX2F6E

# 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
```

```
tar xzf ./actions-runner-osx-x64.tar.gz
```

```
javiersosial@javiers-MacBook-Pro ~ % mkdir actions-runner && cd actions-runner
javiersosial@javiers-MacBook-Pro actions-runner % curl -o actions-runner-osx-x64-2.322.0.tar.gz -L https://github.com/actions/runner/releases/download/v2.322.0/actions-runner-osx-x64-2.322.0.tar.gz
```

% Total	% Received	% Xferd	Average Dload	Speed Upload	Time Total	Time Spent	Time Left	Current Speed
0	0	0	0	0	--:--:--	--:--:--	--:--:--	0
100 74.6M	100 74.6M	0	0	9.8M	0	0:00:07	0:00:07	--:--:-- 12.1M

```
javiersosial@javiers-MacBook-Pro actions-runner % echo "aa0fc262363912167dcdbc74|6ffcdf7b8996bc587f51cf1bab38ad86cf70b6ea actions-runner-osx-x64-2.322.0.tar.gz"
| shasum -a 256 -c
actions-runner-osx-x64-2.322.0.tar.gz: OK
javiersosial@javiers-MacBook-Pro actions-runner % tar xzf ./actions-runner-osx-x64-2.322.0.tar.gz
javiersosial@javiers-MacBook-Pro actions-runner % ./config.sh --url https://github.com/javiersosial/my-project --token BMXISTYUUFQYSQ56SYJDX3HXX2YK
```

Self-hosted runner registration

```
✓ Runner successfully added
✓ Runner connection is good

# Runner settings

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

✓ Settings Saved.
```

Jalankan runner :

```
./run.sh
```

```
javiersosial@javiers-MacBook-Pro actions-runner % ./run.sh
✓ Connected to GitHub
Current runner version: '2.322.0'
2025-02-25 15:56:12Z: Listening for Jobs
```

Integrasikan SonarQube dengan CI/CD Github Action

Tambahkan Secrets ke GitHub

- Pergi ke repo GitHub.
- Masuk ke **Settings > Secrets and variables > Actions > New repository secret**.
- Tambahkan:
 - SONAR_TOKEN: Token yang baru dibuat.
 - SONAR_HOST_URL: URL SonarQube (misalnya, http://localhost:9000 atau URL server SonarQube-mu).

The screenshot shows the GitHub repository settings page for 'my-project' by 'javiersosial'. The page is titled 'Actions secrets and variables'. The left sidebar has a yellow box '1' next to the repository name and a yellow box '2' next to the 'Settings' tab. The main content area has a yellow box '3' next to the 'Actions' tab in the left sidebar and a yellow box '4' next to the 'Manage environment secrets' button. The 'Repository secrets' table shows two secrets: SONAR_HOST_URL and SONAR_TOKEN, both updated 10 hours ago.

Name	Last updated
SONAR_HOST_URL	10 hours ago
SONAR_TOKEN	10 hours ago

Actions secrets / New secret

Name *

SONAR_TOKEN

Secret *

sqa_fbd923bb33e08da8047164ea1fdaebdea4aa

Add secret

Actions secrets / New secret

Name *

SONAR_HOST_URL

Secret *

http://localhost:9000

Setelah menambahkan secrets, dapat dilakukan **instalasi SonarScanner pada lokal host**. Instalasi Sonar scanner juga dengan mengunduh dan unzip melalui job CI/CD tetapi kurang efisien karena harus hal tersebut setiap kali commit menjalankan workflow.

SonarScanner adalah **CLI (Command Line Interface) tool** yang digunakan untuk mengirim kode sumber ke **SonarQube** agar bisa dianalisis. Tool ini membaca kode, mencari bug, kerentanan keamanan, code smells, dan masalah lainnya, lalu mengirim hasilnya ke SonarQube untuk ditampilkan dalam bentuk laporan.

Instal SonarScanner

```
brew install sonar-scanner
```

```
javiersosial@javiers-MacBook-Pro ~ % brew install sonar-scanner
==> Downloading https://formulae.brew.sh/api/formula.jws.json
==> Downloading https://formulae.brew.sh/api/cask.jws.json
==> Downloading https://ghcr.io/v2/homebrew/core/sonar-scanner/ma
.....
```

Cek apakah Sonar Scanner telah berhasil terinstal :

```
sonar-scanner -v
```

```
javiersosial@javiers-MacBook-Pro ~ % sonar-scanner -v
22:44:36.608 INFO Scanner configuration file: /usr/local/Cel
.0.2.4839/libexec/conf/sonar-scanner.properties
22:44:36.611 INFO Project root configuration file: NONE
```

Setelah memasukkan secrets di Github dan menginstal SonarScanner, **buat workflow pipeline CI/CD di Github Action.**

Buat atau edit file `.github/workflows/sonarqube.yml`

Tambahkan konfigurasi pipeline CI/CD seperti ini:

```
name: SonarQube Analysis
on:
  push:
    branches:
      - main
jobs:
  sonarqube:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Repository
        uses: actions/checkout@v4

      - name: Setup Java
        uses: actions/setup-java@v3
        with:
          distribution: 'temurin'
          java-version: '17'

      - name: Cache SonarQube packages
        uses: actions/cache@v3
        with:
          path: ~/.sonar/cache
          key: ${{ runner.os }}-sonar
          restore-keys: ${{ runner.os }}-sonar

      - name: SonarQube Scan
        run: |
          sonar-scanner \
            -Dsonar.projectKey=nama_project \
            -Dsonar.sources=. \
            -Dsonar.host.url=${{ secrets.SONAR_HOST_URL }} \
            -Dsonar.login=${{ secrets.SONAR_TOKEN }}
```

🔍 Penjelasan Workflow CI/CD SonarQube di GitHub Actions

Workflow ini digunakan untuk menjalankan **analisis kode dengan SonarQube** setiap kali ada push ke branch main. Berikut adalah langkah-langkahnya:

📌 Rincian Workflow

1 Nama Workflow

```
name: SonarQube Analysis
```

- 🔗 Workflow ini diberi nama **"SonarQube Analysis"**, agar mudah dikenali di GitHub Actions.

2 Kapan Workflow Berjalan?

```
on:
```

```
  push:  
    branches:  
      - main
```

- 🔗 Workflow akan berjalan **setiap kali ada push ke branch main**.
- 🔗 Bisa diubah untuk berjalan di branch lain atau saat pull request.

3 Definisi Job

```
jobs:  
  sonarqube:  
    runs-on: ubuntu-latest
```

- 🔗 **Job bernama sonarqube** dijalankan di runner **Ubuntu terbaru (ubuntu-latest)**.

4 Langkah-Langkah dalam Job

🔗 4.1 Checkout Repository

```
- name: Checkout Repository  
  uses: actions/checkout@v4
```

- 🔗 Mengambil kode terbaru dari repository agar bisa dianalisis.

🔗 4.2 Setup Java 17

```
- name: Setup Java  
  uses: actions/setup-java@v3  
  with:  
    distribution: 'temurin'  
    java-version: '17'
```

- 🔗 **Menginstal Java 17** (diperlukan oleh SonarScanner).
- 🔗 Menggunakan **Temurin JDK** (Adoptium).

🔗 4.3 Cache SonarQube Packages

```
- name: Cache SonarQube packages
```

```
uses: actions/cache@v3
with:
  path: ~/.sonar/cache
  key: ${ runner.os }-sonar
  restore-keys: ${ runner.os }-sonar
```

- 🔗 **Caching paket SonarQube** untuk mempercepat eksekusi di build berikutnya.
- 🔗 Cache akan disimpan di `~/.sonar/cache` pada runner.

🔗 4.4 Jalankan SonarScanner untuk Analisis Kode

```
- name: SonarQube Scan
  run: |
    sonar-scanner \
      -Dsonar.projectKey=nama_project \
      -Dsonar.sources=. \
      -Dsonar.host.url=${ secrets.SONAR_HOST_URL } \
      -Dsonar.login=${ secrets.SONAR_TOKEN }
```

🔗 Menjalankan **SonarScanner** untuk menganalisis kode dan mengirim hasil ke **SonarQube**.

🔗 Parameter yang digunakan:

- `-Dsonar.projectKey=nama_project` → **Nama unik proyek** di SonarQube.
- `-Dsonar.sources=.` → **Menganalisis seluruh repo** (. artinya semua file dalam repo).
- `-Dsonar.host.url=${ secrets.SONAR_HOST_URL }` → **URL SonarQube**, diambil dari **GitHub Secrets**.
- `-Dsonar.login=${ secrets.SONAR_TOKEN }` → **Token autentikasi**, juga dari **GitHub Secrets**.

🔗 Kesimpulan

🔗 Tujuan Workflow:

- ✓ Secara otomatis menjalankan **analisis kode dengan SonarQube** setiap kali ada push ke branch main.
- ✓ **Mendeteksi bug, vulnerabilities, dan code smells** dalam kode proyek.
- ✓ **Menggunakan caching** untuk mempercepat eksekusi di masa mendatang.

🔗 Bagaimana Workflow Ini Bekerja?

- 1 **Mengecek kode dari repo** (checkout).
- 2 **Menyiapkan environment** (Java 17).
- 3 **Menggunakan cache** untuk SonarQube agar lebih cepat.
- 4 **Menjalankan SonarScanner** dan mengirim hasil ke **SonarQube Server**.

Commit, push ke Github dan jalankan workflow tersebut. Workflow akan berjalan otomatis dan mengirim laporan ke SonarQube.

The screenshot shows the GitHub Actions interface for a workflow named 'Update sonarqube.yml #2'. The workflow is in a 'Completed' state, indicated by a green checkmark. The main panel displays a log of the workflow steps, all of which were successful:

- > ✓ Set up job
- > ✓ Checkout Repository
- > ✓ Setup Java
- > ✓ Cache SonarQube packages
- > ✓ Run SonarQube Scanner
- > ✓ Post Cache SonarQube packages
- > ✓ Post Setup Java
- > ✓ Post Checkout Repository
- > ✓ Complete job

The left sidebar shows the 'Summary' tab selected, with a 'Jobs' section listing the 'sonarqube' job as completed. Other tabs like 'Run details', 'Usage', and 'Workflow file' are also visible.

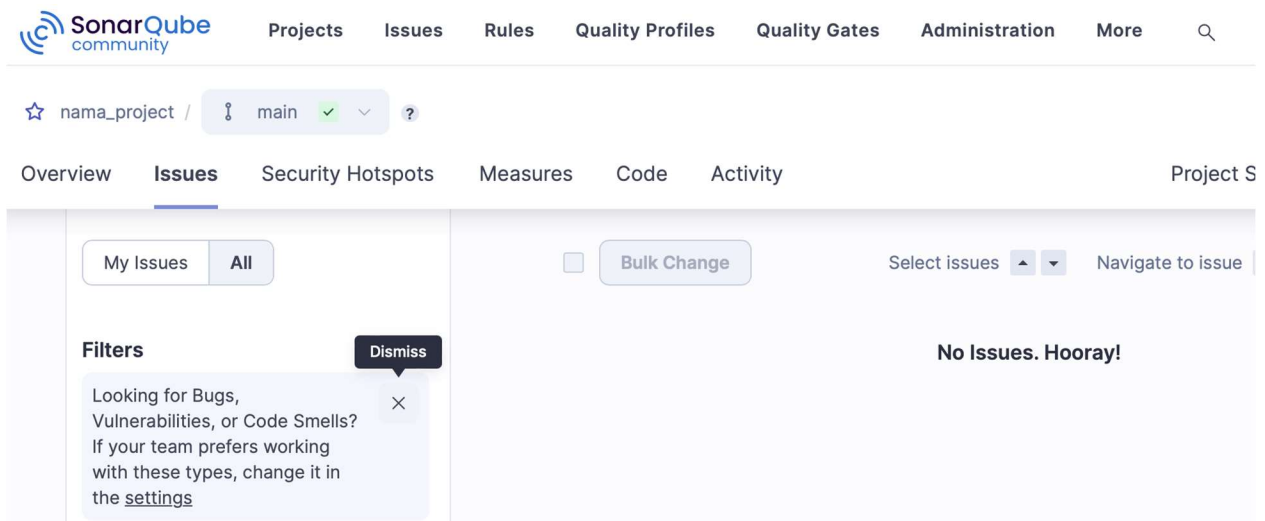
Jika workflow berhasil, berarti sudah menjalankan semua job termasuk melakukan scan pada kode di repository tersebut dan mengirimkan hasilnya ke SonarQube.

Buka SonarQube pada browser <http://localhost:9000/projects>. Terlihat proyek baru tadi sudah tertampil beserta hasil analisisnya.

The screenshot shows the SonarQube web interface. The top navigation bar includes links for 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', 'Administration', and 'More'. A notification banner at the top states: 'The way we calculate ratings has changed and it might affect your security, reliability, maintainability and security review ratings. [Learn More](#)'.

The main content area displays a list of projects. The first project, 'nama_project', is shown with a 'Passed' status (indicated by a green checkmark). The project details show 'Last analysis: 22 minutes ago' and a message: 'The main branch of this project is empty.'.

On the left sidebar, the 'Filters' section shows a 'Quality Gate' summary with 'Passed' (1) and 'Failed' (0) counts.



Hasil analisis dapat dilihat detail pada halaman tersebut, apakah passed atau failed, bagaimana tingkat security dan reliabilitasnya. Hasil ini akan disampaikan ke developer untuk ditindaklanjuti. Pada tugas ini, hasil scan repo javiersosial/my-project mendapat status passed dan tidak terdapat issue.

Rangkaian scan repository menggunakan SonarScanner melalui workflow di Github Action kemudian menampilkan hasil scan tersebut ke SonarQube telah berhasil. Ini menandakan pembuatan 1 pipeline cicd yang terintegrasi dengan sonarqube dan jelaskan step by stepnya telah berhasil dibuat.