SONARCLOUD VIA GITHUB ACTIONS

PETRO KOLOSOV

Abstract. Explains how to integrate SonarCloud with GitHub actions in a simple and quick approach.

Contents

1. Introduction

First choose which repository to analyze, it is rather obvious initial step, however. I choose to test it on my old repository `github.com/kolosovpetro/IoC-Container`.

1.1. Login to the SonarCloud.

Next we login to the `sonarcloud.io` using your GitHub account

Date: October 31, 2022.

Key words and phrases. SonarCloud, Github Actions, DevOps, CICD .

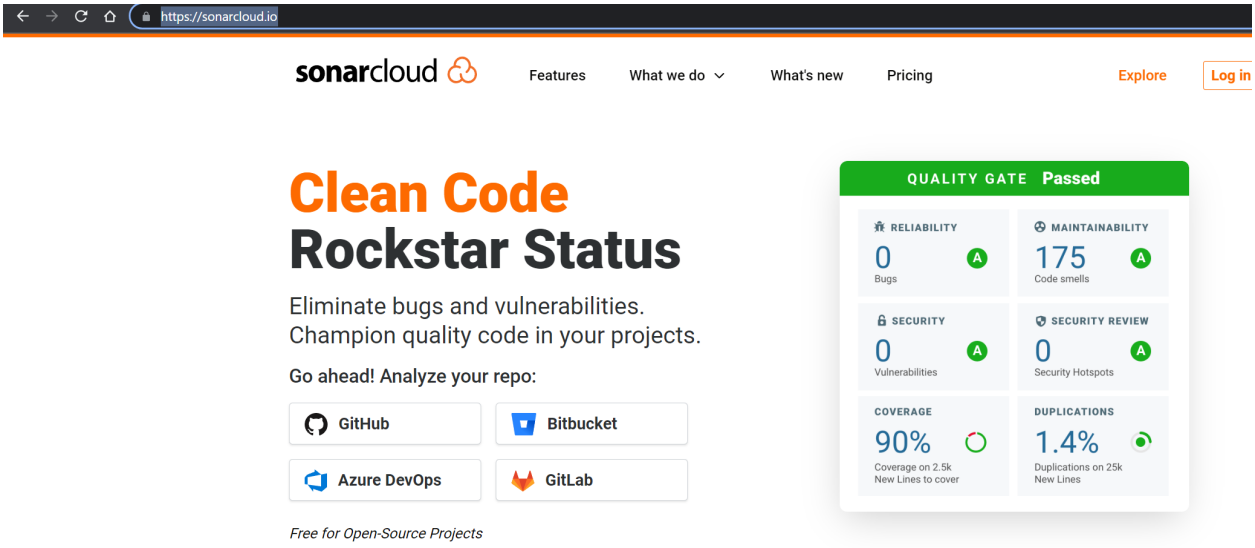

Figure 1. SonarCloud login page. Use login via GutHub.

1.2. **Create an organization at SonarCloud.** Now we create an organization we use to analyze your project, I created it as

`petro-kolosov-own-repos`

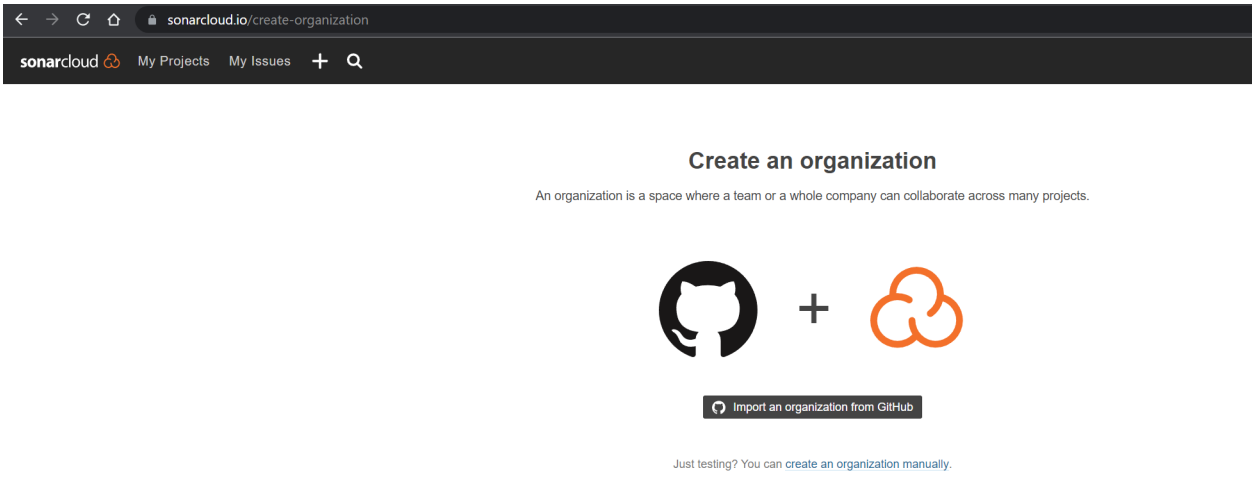

Figure 2. Create organization form, use `create an organization manually` at the bottom.

Then enter the key of the organization

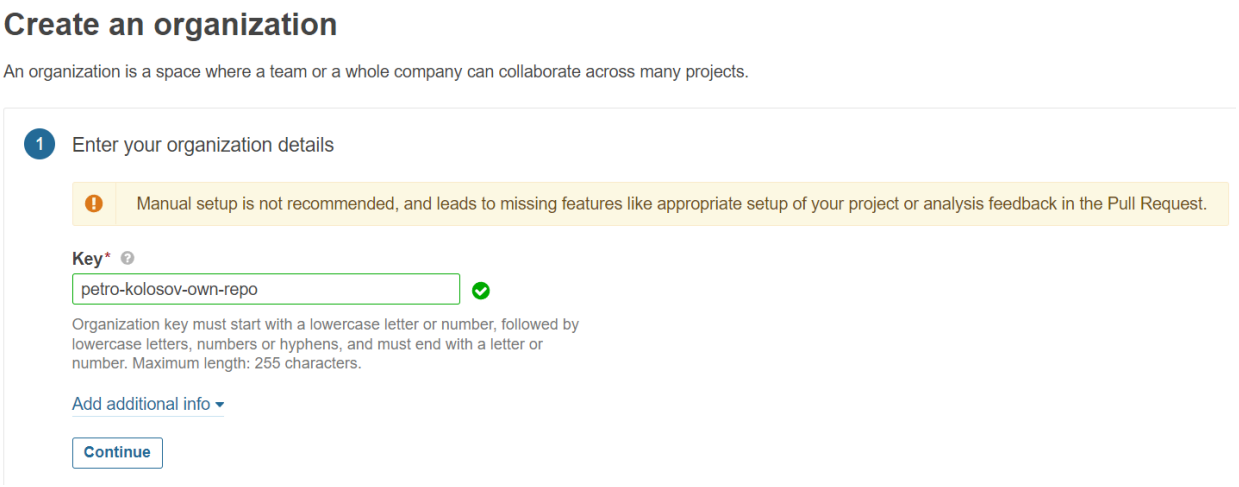

Figure 3. Enter the key of the organization.

Next, choose a plan for your organization, I use free one

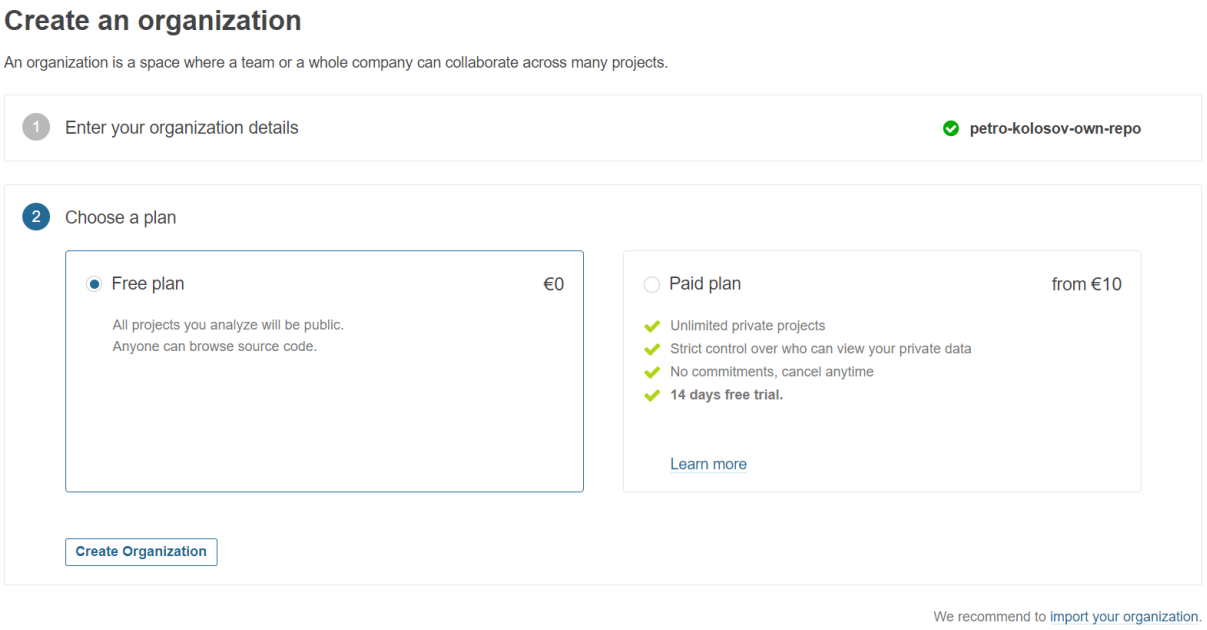

Figure 4. Choose a plan for your organization.

Click create then, so we have finished second step, creating of the organization.

1.3. **Create a project on behalf of organization.** At your recently created organization, click the `Analyze new project` button

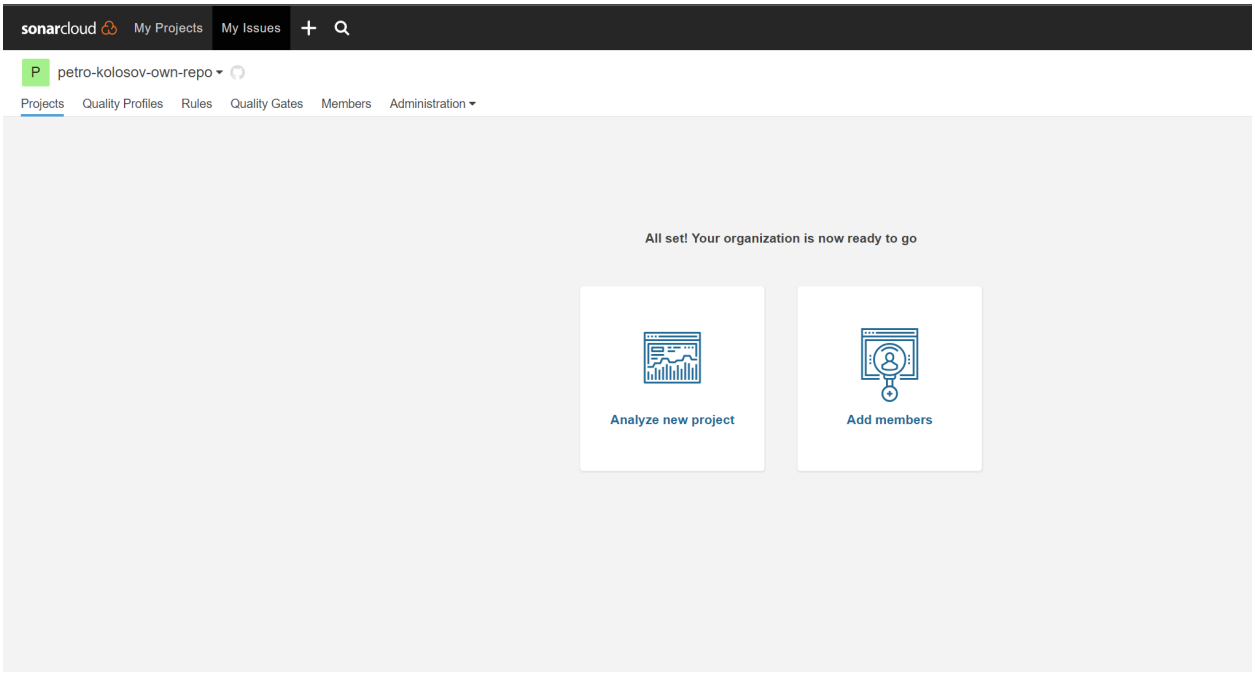

Figure 5. Click the `Analyze new project` button

Configure the name of your project

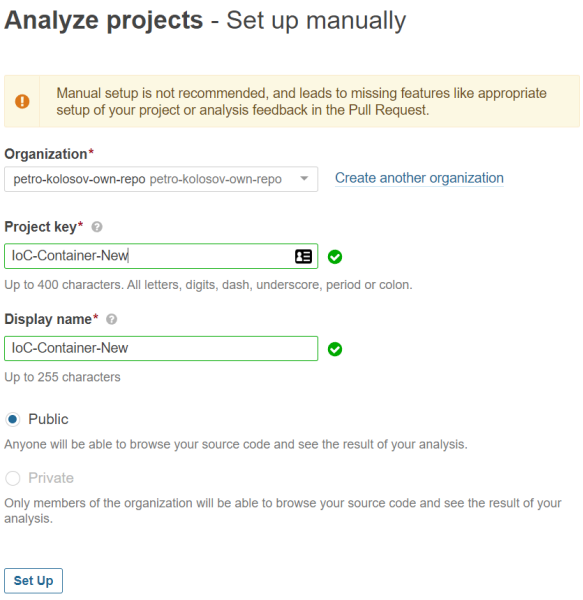

Figure 6. Configure the name of your project

Click `Set Up` and choose `Configure With Github Actions`, so that it looks like

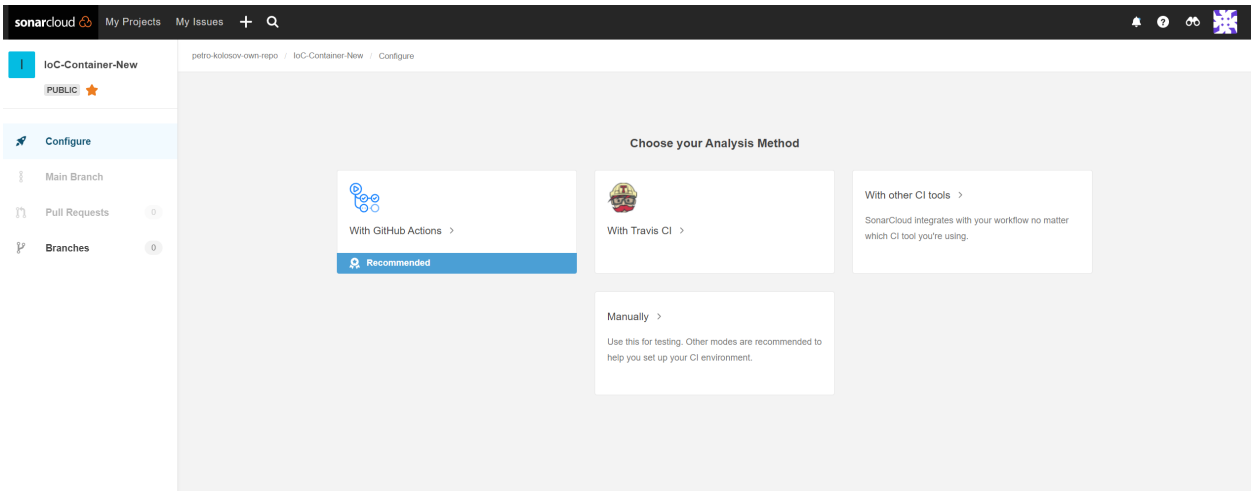

Figure 7. `Configure With Github Actions`.

Create a specified GitHub actions secret at your repository on GitHub

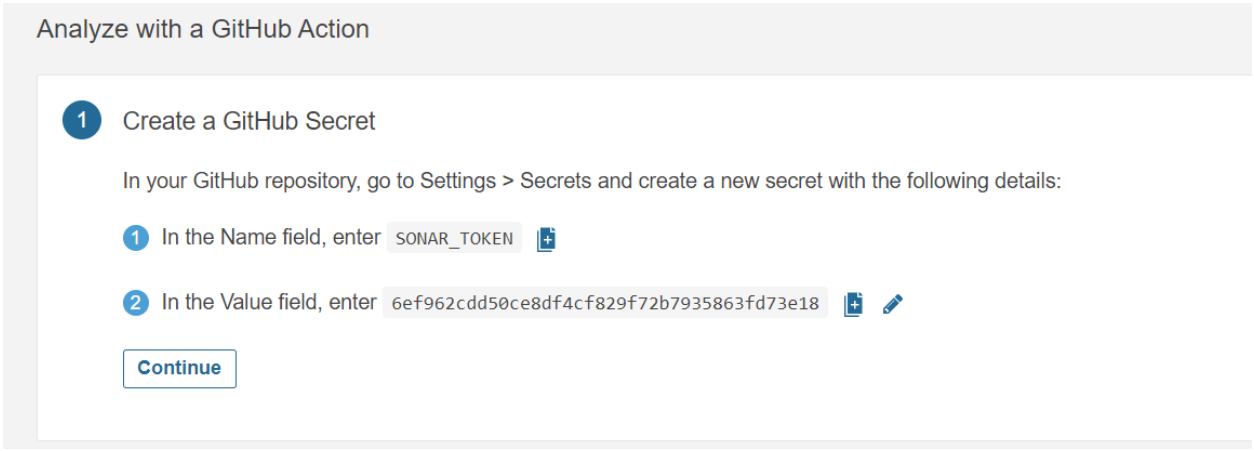

Figure 8. Create a specified GitHub actions secret.

Do not worry, this key won't work and here just for example. Clicking `Continue` and then `.NET` explores an example of the pipeline we are able to use on the GitHub as an action of our repository.

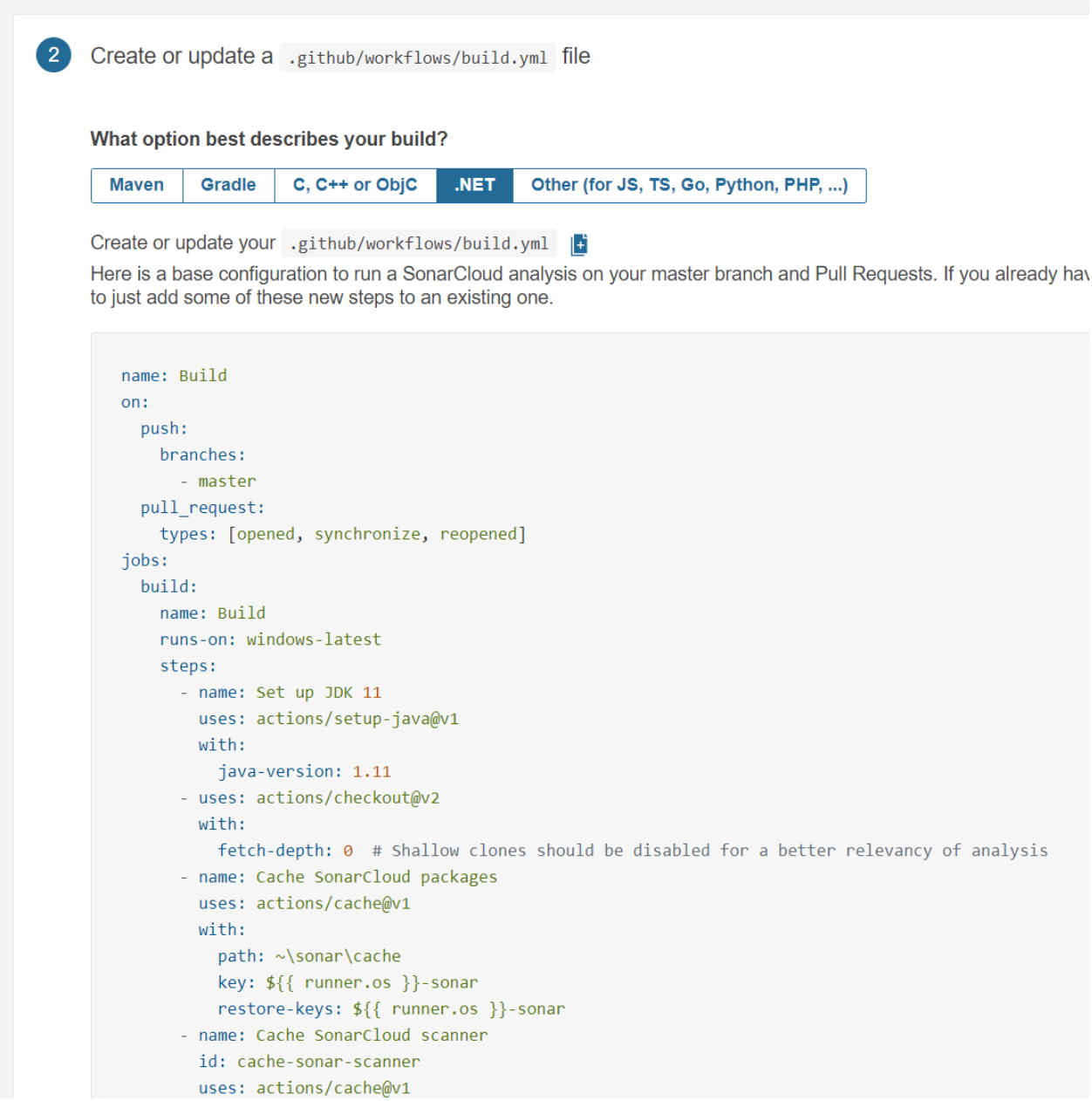

Figure 9. SonarCloud analyzer pipeline example.

Above pipeline runs on the `Windows` runner, but I prefer ubuntu runner for the reasons of performance so that I change it following way

```
name: Run SonarCloud analysis
on:
  push:
    branches: [ develop ]
  pull_request:
    types: [ opened, synchronize, reopened ]
```

```
  workflow_dispatch:
jobs:
  run-sonarcloud-analysis:
    name: Run SonarCloud Analysis
    runs-on: ubuntu-latest
    steps:
      - name: Set up JDK 11
        uses: actions/setup-java@v1
        with:
          java-version: 1.11
      - uses: actions/checkout@v2
        with:
          fetch-depth: 0

      - name: Cache SonarCloud packages
        uses: actions/cache@v1
        with:
          path: ~/sonar/cache
          key: ${{ runner.os }}-sonar
          restore-keys: ${{ runner.os }}-sonar

      - name: Cache SonarCloud scanner
        id: cache-sonar-scanner
        uses: actions/cache@v1
        with:
          path: ./.sonar/scanner
          key: ${{ runner.os }}-sonar-scanner
          restore-keys: ${{ runner.os }}-sonar-scanner

      - name: Setup .NET 6 SDK
        uses: actions/setup-dotnet@v1
        with:
          dotnet-version: 6.0.x

      - name: Install SonarCloud scanner
        if: steps.cache-sonar-scanner.outputs.cache-hit != 'true'
        run: |
```

```
          mkdir -p ./.sonar/scanner
          chmod a+rwx ./.sonar/scanner
          dotnet tool update dotnet-sonarscanner --tool-path ./.sonar/scanner

      - name: Analyze project
        run: |
          ./.sonar/scanner/dotnet-sonarscanner begin /k:"IoC-Container" /o:"petro-koloso
/d:sonar.login="${{ secrets.SONAR_TOKEN }}" /d:sonar.host.url="https://sonarcloud.io"
          dotnet build -c Release
          ./.sonar/scanner/dotnet-sonarscanner end /d:sonar.login="${{ secrets.SONAR_TOK
}}"
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
```

After the push to the `develop` branch the action is triggered and succeeded

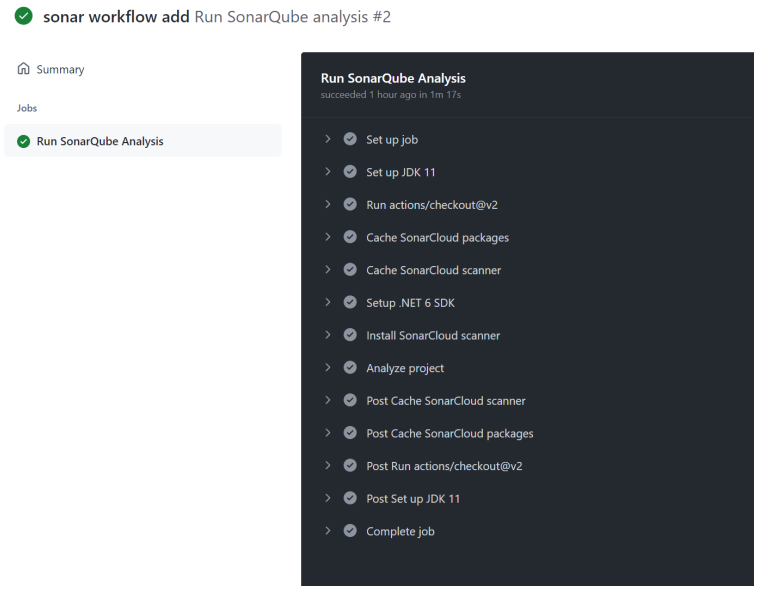

Figure 10. Action is triggered and succeeded.

So that pull request is analyzed and reported to the SonarCloud web application

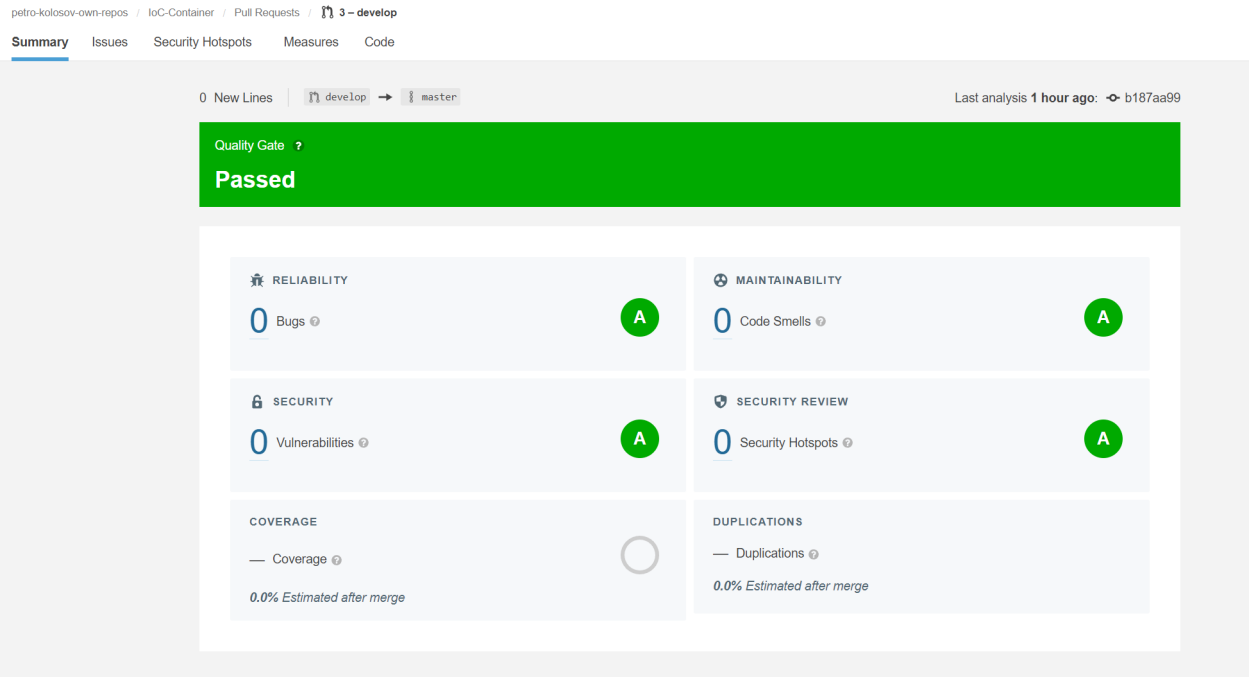

Figure 11. Pull request analysis.

After merge workflow runs again in order to analyze the main branch `master` so that we got a report as follows

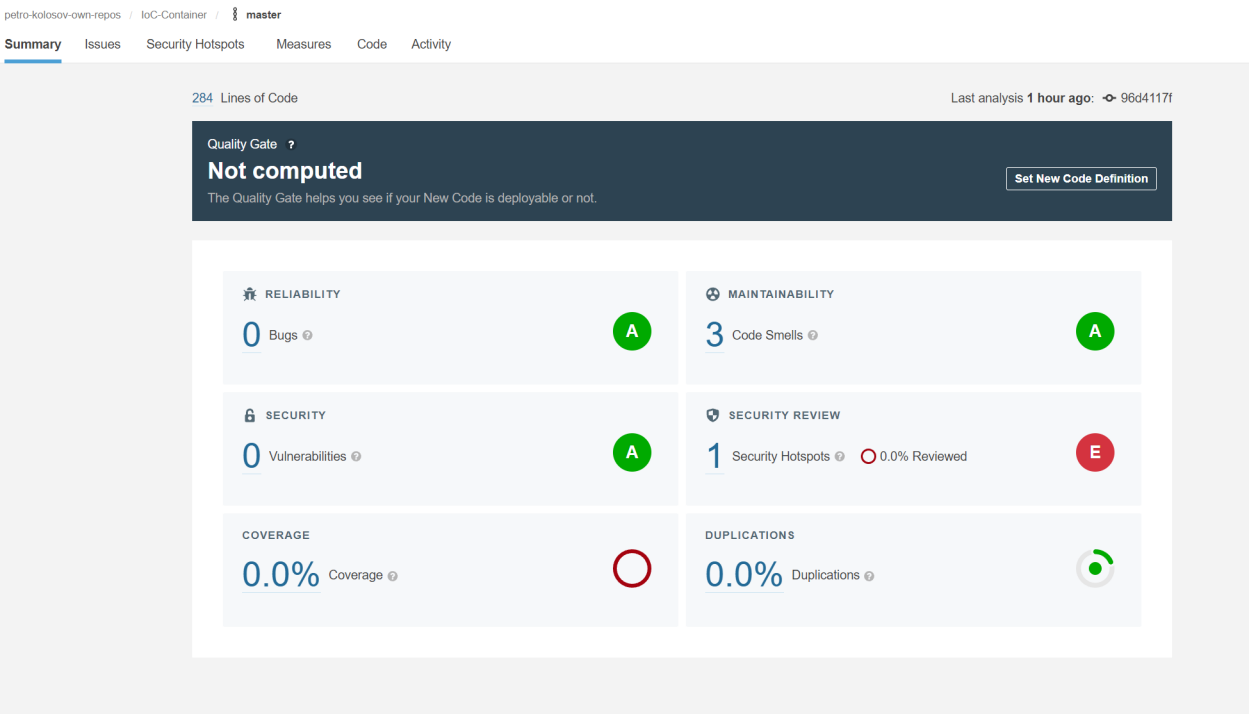

Figure 12. Main branch analysis.

Therefore, we have successfully established an integration between SonarCloud static code analyzer and GitHub repository.

- Workflow source
- SonarCloud dashboard

Email address: kolosovp94@gmail.com
URL: https://kolosovpetro.github.io