Configuring GitHub With SonarQube

SonarQube

First we need to download SonarQube in our local system and we need to up the SonarQube.

Step 1: Download the SonarQube. It will be in Zip file extract that file to any drive.

Step 2: Open the folder which you have extracted.

Step 3: Go to required folder sonarqube-9.4.0.54424\sonarqube-9.4.0.54424\bin\windows-x86-64\

From the above mentioned path open command promp and you run the command as

StartSonar.bat

Step 4: Then in Browser Open the port number <http://localhost:9000/>.

This is how we up the SonarQube locally.

With this integration, you'll be able to:

* **Import your GitHub repositories** - Import your GitHub repositories into SonarQube to easily set up SonarQube projects.
* **Analyse projects with GitHub Actions** - Integrate analysis into your build pipeline. Starting in [Developer Edition](https://redirect.sonarsource.com/editions/developer.html), Sonar Scanners running in GitHub Actions jobs can automatically detect branches or pull requests being built so you don't need to specifically pass them as parameters to the scanner.
* **Report your Quality Gate status to your branches and pull requests** - (starting in [Developer Edition](https://redirect.sonarsource.com/editions/developer.html)) See your Quality Gate and code metric results right in GitHub so you know if it's safe to merge your changes.
* **Authenticate with GitHub** - Sign in to SonarQube with your GitHub credentials.

**Import your GitHub repositories** :

1. Create your GitHub App.
2. Install your GitHub App in your organization.
3. Update your SonarQube global settings with your GitHub App information.

Creating an GitHub App:

Step 1: Open your GitHub Account.

Step 2: Go to Settings>Developer Settings> GitHub App.

Step 3: Click on New GitHub App

Step 4: In next page we need give details about our App and then we need to click on Create an app.

Specify the following settings in your app:

* GitHub App Name
* Homepage URL
* User authorization callback URL
* Webhook URL
* Grant access for the Repository permissions.

**Installing GitHub App in your organization:**

Step 1:Go to GitHub Account.

Step 2: On the right side plane Search for setting and click on that.

Step 3: Click on Developer settings and choose GitHub Apps.

Step 4: Select your App.

Step 5: In that App page click on Install App.

While installing it will ask for all repositories or for only one repository we need to choose that.

Updating SonarQube global settings with GitHub App information

After you've created and installed GitHub App, update the global SonarQube settings to finish integration and allow for the import of GitHub projects.

* **Configuration Name** – The name used to identify your GitHub configuration at the project level.
* **GitHub URL** – Provide Your GitHub URL.
* **GitHub App ID** – The App ID is found on your GitHub App's page on GitHub at **Settings > Developer Settings > GitHub Apps**.
* **Client ID** – The Client ID is found on your GitHub App's page.
* **Client secret** – The Client secret is found on your GitHub App's page.
* **Private Key** – Your GitHub App's private key. You can generate a .pem file from your GitHub App's page under **Private keys**. Copy and paste the whole contents of the file here.

**Analysing projects with GitHub Actions**

Sonar Scanners running in GitHub Actions can automatically detect branches and pull requests being built so you don't need to specifically pass them as parameters to the scanner.

To analyse projects with GitHub Actions, you need to:

* Create your GitHub Secrets.
* Configure your workflow YAML file.
* Commit and push your code to start the analysis.

### Creating your GitHub Secrets

* SONAR\_TOKEN – Generate a SonarQube [token](https://docs.sonarqube.org/latest/user-guide/user-token/) and, in GitHub, create a new repository secret in GitHub with SONAR\_TOKEN as the **Name** and the token you generated as the **Value**.
* SONAR\_HOST\_URL – In GitHub, create a new repository secret with SONAR\_HOST\_URL as the **Name** and your SonarQube server URL as the **Value**.

**Configuring .github/workflows/build.yml file**

This section shows how to configure our .github/workflows/build.yml file.

set up your build according to your SonarQube edition:

* **Community Edition** – Community Edition doesn't support multiple branches, so you should only analyse your main branch. You can restrict analysis to your main branch by setting it as the only branch in your on.push.branches configuration in your workflow YAML file, and not using on.pull\_request.
* **Developer Edition and above** – GitHub Actions can build specific branches and pull requests if you use on.push.branches and on.pull-requests configurations as shown in the examples below.

### Commit and push your code

Commit and push your code to start the analysis. Each new push you make on your branches or pull requests will trigger a new analysis in SonarQube.

## **Reporting your Quality Gate status in GitHub**

After creating and installing GitHub App, SonarQube can report your Quality Gate status and analysis metrics directly to your GitHub branches and pull requests.

To do this, add a project from GitHub by clicking the **Add project** button in the upper-right corner of the **Projects** homepage and select **GitHub** from the drop-down menu.

Then, follow the steps in SonarQube to analyse your project. SonarQube automatically sets the project settings required to show your Quality Gate in your branches and pull requests.

If you're creating your projects manually or adding Quality Gate reporting to an existing project, see the following section.

### Reporting your Quality Gate status in manually created or existing projects

SonarQube can also report your Quality Gate status to GitHub pull requests and branches for existing and manually-created projects.

After you've created and installed your GitHub App and updated your global DevOps Platform Integration settings as shown in the **Importing your GitHub repositories into SonarQube** section above, set the following project settings at **Project Settings > General Settings > DevOps Platform Integration**:

* **Configuration name** – The configuration name that corresponds to your GitHub instance.
* **Repository identifier** – The path of your repository URL.

**Authenticating with GitHub**

To allow users to log in with GitHub credentials, use the GitHub App that you created above (see the **Importing your GitHub repositories using a GitHub App** section for more information) and update your global SonarQube settings.

To update your global SonarQube settings:

1. **Enabled** – set the switch to true.
2. **Client ID** – the Client ID is found below the GitHub App ID on your GitHub App's page.
3. **Client Secret** – the Client secret is found below the Client ID on your GitHub App's page.

Now, from the login page, users can connect their GitHub accounts with the new "Log in with GitHub" button.

### Creating a dedicated app for authentication

If we want to use a dedicated app for GitHub authentication, we can create a GitHub OAuth app. You'll find general instructions for creating a GitHub OAuth App [here](https://docs.github.com/en/free-pro-team@latest/developers/apps/creating-an-oauth-app). Specify the following settings in your OAuth App:

* **Homepage URL** – the public URL of your SonarQube server. For example, https://sonarqube.mycompany.com. For security reasons, HTTP is not supported, and you must use HTTPS. The public URL is configured in SonarQube at **Administration > General > Server base URL**.
* **Authorization callback URL** – your instance's base URL. For example, https://yourinstance.sonarqube.com.

**After creating your app, update your global SonarQube settings:**

1. **Enabled** – set the switch to true.
2. **Client ID** – the Client ID is found below the GitHub App ID on your GitHub App's page.
3. **Client Secret** – the Client secret is found below the Client ID on your GitHub App's page.

Now, from the login page, users can connect their GitHub accounts with the new "Log in with GitHub" button.