How To Ensure Code Quality with SonarQube on Ubuntu 18.04

Posted January 11, 2019 1.7k DEVELOPMENT SYSTEM TOOLS NGINX MYSQL LET'S ENCRYPT UBUNTU 18.04

By: Namo

The author selected Internet Archive to receive a donation as part of the Write for DOnations program.

Introduction

Code quality is an approximation of how useful and maintainable a specific piece of code is. Quality code will make the task of maintaining and expanding your application easier. It helps ensure that fewer bugs are introduced when you make required changes in the future.

SonarQube is an open-source tool that assists in code quality analysis and reporting. It scans your source code looking for potential bugs, vulnerabilities, and maintainability issues, and then presents the results in a report which will allow you to identify potential issues in your application.

The SonarQube tool itself is made out of two parts: a scanner, which is an application that would be installed locally on the developer's machine to do the code analysis, and a centralized server for record-keeping and reporting. A single SonarQube server instance can support multiple scanners, enabling you to centralize code quality reports from many developers in a single place.

In this guide, you will deploy a SonarQube server and scanner to analyze your code and create code quality reports. Then you'll perform a test on your machine by scanning an example code with the SonarQube scanner.

Prerequisites

Before you begin this guide you'll need the following:

- One Ubuntu 18.04 server with **3GB or more** memory set up by following this Initial Server Ubuntu 18.04, including a sudo non-root user and a firewall.

- Oracle Java 8 installed on the server, configured by following the Oracle JDK section in this Oracle JDK installation tutorial.
- Nginx and MySQL, configured by following the Nginx and MySQL sections in this LEMP installation guide.
- Certbot (the Let's Encrypt client), configured by following How To Secure Nginx with Let's Encrypt on Ubuntu 18.04.
- A fully-qualified domain name and an A record pointing to the server where you'll install SonarQube. If you're using DigitalOcean's DNS service, this DNS record setup guide will help you set that up. We'll use `sonarqube.example.com` in this tutorial.

Step 1 — Preparing for the Install

You need to complete a few steps to prepare for the SonarQube installation. As SonarQube is a Java application that will run as a service, and because you don't want to run services as the **root** user, you'll create another system user specifically to run the SonarQube services. After that, you'll create the installation directory and set its permissions, and then you'll create a MySQL database and user for SonarQube.

First, create the **sonarqube** user:

```
$ sudo adduser --system --no-create-home --group --disabled-login sonarqube
```

This user will only be used to run the SonarQube service, so this creates a system user that can't log in to the server directly.

Next, create the directory to install SonarQube into:

```
$ sudo mkdir /opt/sonarqube
```

SonarQube releases are packaged in a zipped format, so install the `unzip` utility that will allow you to extract those files.

```
$ sudo apt-get install unzip
```

Next, you will create a database and credentials that SonarQube will use. Log in to the MySQL server as the **root** user:

```
$ sudo mysql -u root -p
```

Then create the SonarQube database:


```
mysql> CREATE DATABASE sonarqube;
```

Now create the credentials that SonarQube will use to access the database.

```
mysql> CREATE USER sonarqube@'localhost' IDENTIFIED BY 'some_secure_password';
```

Then grant permissions so that the newly created user can make changes to the SonarQube database:

```
mysql> GRANT ALL ON sonarqube.* to sonarqube@'localhost';
```

Then apply the permission changes and exit the MySQL console:

```
mysql> FLUSH PRIVILEGES;
mysql> EXIT;
```

Now that you have the user and directory in place, you will download and install the SonarQube server.

Step 2 — Downloading and Installing SonarQube

Start by changing the current working directory to the SonarQube installation directory:

```
$ cd /opt/sonarqube
```

Then, head over to the SonarQube downloads page and grab the download link for SonarQube 7.5 Community Edition. There are many versions and flavors of SonarQube available for download on the page, but in this specific tutorial we'll be using SonarQube 7.5 Community Edition.

After getting the link, download the file:

```
$ sudo wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-7.5.zip
```

Unzip the file:

```
$ sudo unzip sonarqube-7.5.zip
```

Once the files extract, delete the downloaded zip file, as you no longer need it:

```
$ sudo rm sonarqube-7.5.zip
```

Finally, update the permissions so that the **sonarqube** user will own these files, and be able to read and write files in this directory:

```
$ sudo chown -R sonarqube:sonarqube /opt/sonarqube
```

Now that all the files are in place, we can move on to configuring the SonarQube server.

Step 3 — Configuring the SonarQube Server

We'll need to edit a few things in the SonarQube configuration file. Namely:

- We need to specify the username and password that the SonarQube server will use for the database connection.
- We also need to tell SonarQube to use MySQL for our back-end database.
- We'll tell SonarQube to run in server mode, which will yield improved performance.
- We'll also tell SonarQube to only listen on the local network address since we will be using a reverse proxy.

Start by opening the SonarQube configuration file:

```
$ sudo nano sonarqube-7.5/conf/sonar.properties
```

First, change the username and password that SonarQube will use to access the database to the username and password you created for MySQL:

/opt/sonarqube/sonarqube-7.5/conf/sonar.properties

```
...

sonar.jdbc.username=sonarqube
sonar.jdbc.password=some_secure_password

...
```

Next, tell SonarQube to use MySQL as the database driver:

/opt/sonarqube/sonarqube-7.5/conf/sonar.properties

```
...

sonar.jdbc.url=jdbc:mysql://localhost:3306/sonarqube?useUnicode=true&characterEncoding=utf8&rewri

...
```

As this instance of SonarQube will be run as a dedicated server, we could add the `-server` option to activate SonarQube's server mode, which will help in maximizing performance.

Nginx will handle the communication between the SonarQube clients and your server, so you will tell SonarQube to only listen to the local address.

/opt/sonarqube/sonarqube-7.5/conf/sonar.properties

```
...

sonar.web.javaAdditionalOpts=-server
sonar.web.host=127.0.0.1
```

Once you have updated those values, save and close the file.

Next, you will use Systemd to configure SonarQube to run as a service so that it will start automatically upon a reboot.

Create the service file:

```
$ sudo nano /etc/systemd/system/sonarqube.service
```

Add the following content to the file which specifies how the SonarQube service will start and stop:

/etc/systemd/system/sonarqube.service

```
[Unit]
Description=SonarQube service
After=syslog.target network.target

[Service]
Type=forking

ExecStart=/opt/sonarqube/sonarqube-7.5/bin/linux-x86-64/sonar.sh start
ExecStop=/opt/sonarqube/sonarqube-7.5/bin/linux-x86-64/sonar.sh stop

User=sonarqube
Group=sonarqube
Restart=always

[Install]
WantedBy=multi-user.target
```


You can learn more about systemd unit files in Understanding Systemd Units and Unit Files.

Close and save the file, then start the SonarQube service:

```
$ sudo service sonarqube start
```

Check the status of the SonarQube service to ensure that it has started and is running as expected:

```
$ service sonarqube status
```

If the service has successfully started, you'll see a line that says "Active" similar to this:

```
● sonarqube.service - SonarQube service
   Loaded: loaded (/etc/systemd/system/sonarqube.service; enabled; vendor preset
   Active: active (running) since Sat 2019-01-05 19:00:00 UTC; 2s ago
```

Next, configure the SonarQube service to start automatically on boot:

```
$ sudo systemctl enable sonarqube
```

At this point, the SonarQube server will take a few minutes to fully initialize. You can check if the server has started by querying the HTTP port:

```
$ curl http://127.0.0.1:9000
```

Once the initialization process is complete, you can move on to the next step.

Step 4 — Configuring the Reverse Proxy

Now that we have the SonarQube server running, it's time to configure Nginx, which will be the reverse proxy and HTTPS terminator for our SonarQube instance.

Start by creating a new Nginx configuration file for the site:

```
$ sudo nano /etc/nginx/sites-enabled/sonarqube
```

Add this configuration so that Nginx will route incoming traffic to SonarQube:

/etc/nginx/sites-enabled/sonarqube

```
server {
    listen 80;
    server_name sonarqube.example.com;
```

```
    location / {
        proxy_pass http://127.0.0.1:9000;
    }
}
```

Save and close the file.

Next, make sure your configuration file has no syntax errors:

```
$ sudo nginx -t
```

If you see errors, fix them and run `sudo nginx -t` again. Once there are no errors, restart Nginx:

```
$ sudo service nginx restart
```

For a quick test, you can now visit `http://sonarqube.example.com` in your web browser. You'll be greeted with the SonarQube web interface.

Now we'll use Let's Encrypt to create HTTPS certificates for our installation so that data will be securely transferred between the server and your local machine. Use `certbot` to create the certificate for Nginx:

```
$ sudo certbot --nginx -d sonarqube.example.com
```

If this is your first time requesting a Let's Encrypt certificate, Certbot will prompt for your email address and EULA agreement. Enter your email and accept the EULA.

Certbot will then ask how you'd like to configure your security settings. Select the option to redirect all requests to HTTPS. This will ensure that all communication between clients and the SonarQube server gets encrypted.

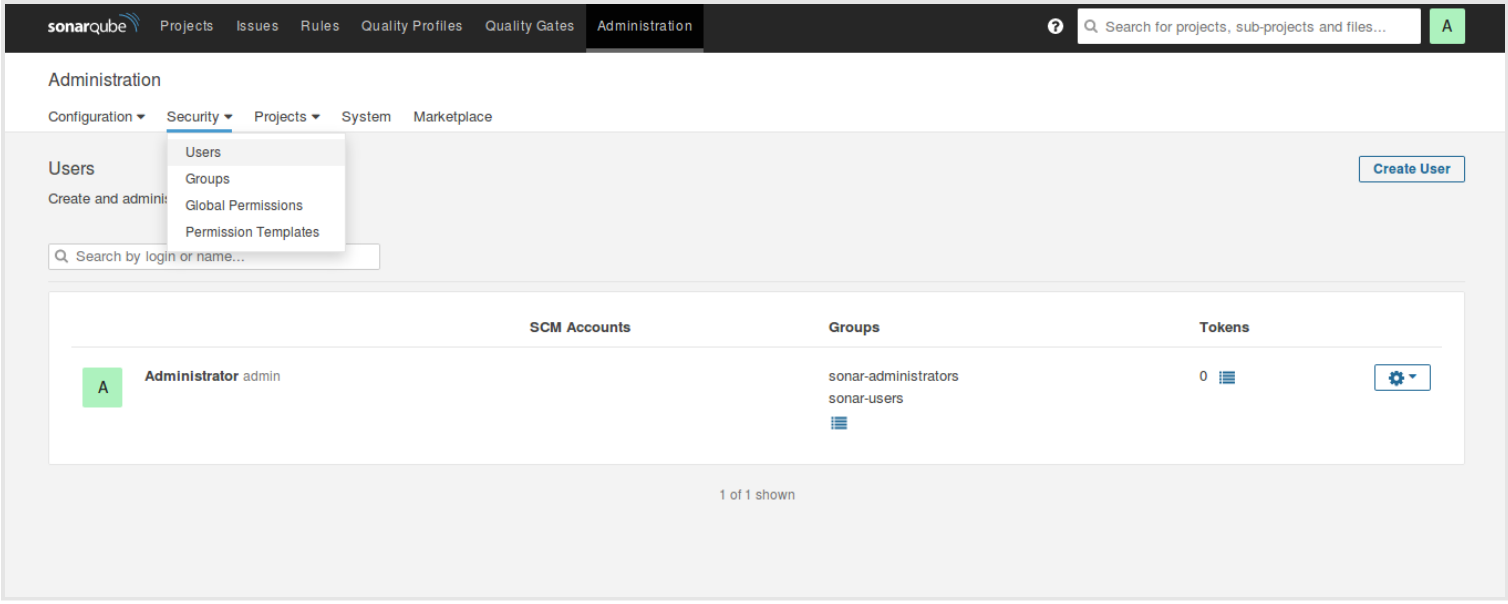
Now that we're done setting up the reverse proxy, we can move on to securing our SonarQube server.

Step 5 — Securing SonarQube

SonarQube ships with a default administrator username and password of **admin**. This default password is not secure, so you'll want to update it to something more secure as a good security practice.

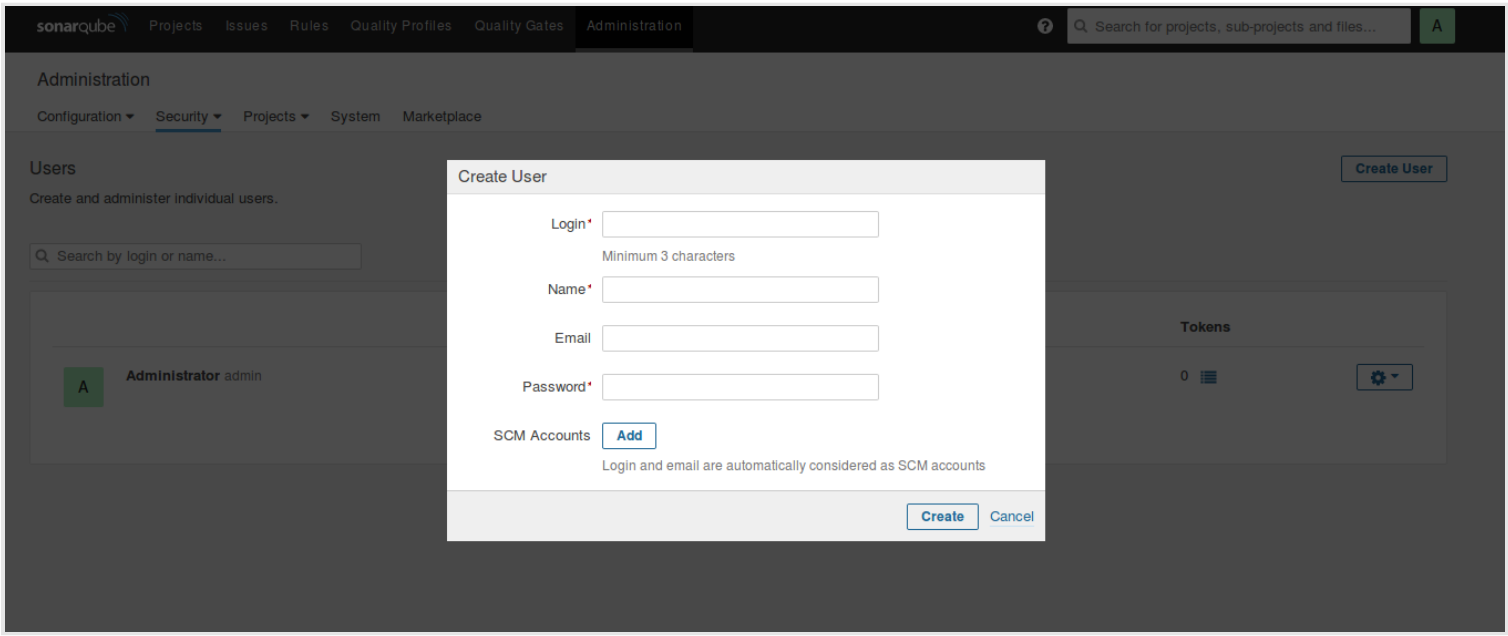
Start by visiting the URL of your installation, and log in using the default credentials. If prompted to start a tutorial, simply click **Skip this tutorial** to get to the dashboard.

Once logged in, click the **Administration** tab, select **Security** from the drop-down list, and then select **Users**:



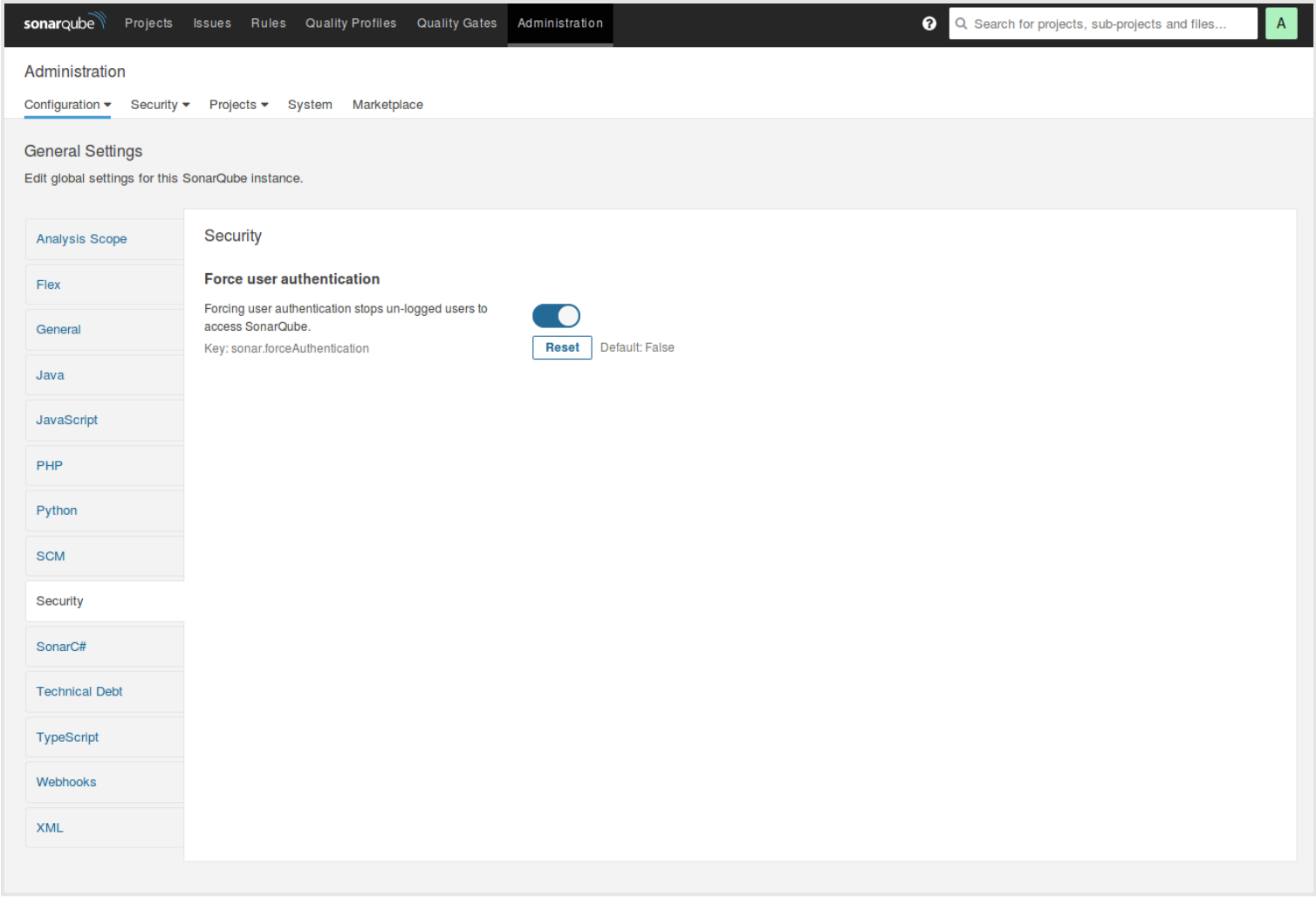
From here, click on the small cog on the right of the "Administrator" account row, then click on "Change password". Be sure to change the password to something that's easy to remember but hard to guess.

Now create a normal user that you can use to create projects and submit analysis results to your server from the same page. Click on the **Create User** button on the top-right of the page:



Then create a token for a specific user by clicking on the button in the "Tokens" column and giving this token a name. You'll need this token later when you invoke the code scanner, so be sure to write it down in a safe place.

Finally, you may notice that the SonarQube instance is wide-open to the world, and anyone could view analysis results and your source code. This setting is highly insecure, so we'll configure SonarQube to only allow logged-in users access to the dashboard. From the same **Administration** tab, click on **Configuration**, then **General Settings**, and then **Security** on the left pane. Flip the switch that says **Force user authentication** to enable authentication, then click on the **Save** button below the switch.



Now that you're done setting up the server, let's set up the SonarQube scanner.

Step 6 — Setting Up the Code Scanner

SonarQube's code scanner is a separate package that you can install on a different machine than the one running the SonarQube server, such as your local development workstation or a continuous delivery server. There are packages available for Windows, MacOS, and Linux which you can find at the SonarQube web site

In this tutorial, you'll install the code scanner on the same server that hosts our SonarQube server.

Start by creating a directory for the scanner:

```
$ sudo mkdir /opt/sonarscanner
```

Then change into that directory:

```
$ cd /opt/sonarscanner
```

Download the SonarQube scanner for Linux using `wget`:

```
$ sudo wget https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar
```

Next, extract the scanner:

```
$ sudo unzip sonar-scanner-cli-3.2.0.1227-linux.zip
```

Then delete the zip archive file:

```
$ sudo rm sonar-scanner-cli-3.2.0.1227-linux.zip
```

After that, you'll need to modify a few settings to get the scanner working with your server install. Open the configuration file for editing:

```
$ sudo nano sonar-scanner-3.2.0.1227-linux/conf/sonar-scanner.properties
```

First, tell the scanner where it should submit the code analysis results. Un-comment the line starting with `sonar.host.url` and set it to the URL of your SonarQube server:

/opt/sonarscanner/sonar-scanner-3.2.0.1227-linux/conf/sonar.properties

```
sonar.host.url=https://sonarqube.example.com
```

Save and close the file. Now make the scanner binary executable:

```
$ sudo chmod +x sonar-scanner-3.2.0.1227-linux/bin/sonar-scanner
```

Then create a symbolic link so that you can call the scanner without specifying the path:

```
$ sudo ln -s /opt/sonarscanner/sonar-scanner-3.2.0.1227-linux/bin/sonar-scanner /usr/local/bin/sonar
```

Now that the scanner is set up, we're ready to run our first code scan.

Step 7 — Running a Test Scan on SonarQube Example Projects

If you'd like to just poke around with SonarQube to see what it can do, you might consider running a test scan on the SonarQube example projects. These are example projects created by the SonarQube team that contains many issues that SonarQube will then detect and report.

Create a new working directory in your home directory, then change to the directory:

```
$ cd ~
$ mkdir sonar-test && cd sonar-test
```

Download the example project:

```
$ wget https://github.com/SonarSource/sonar-scanning-examples/archive/master.zip
```

Unzip the project and delete the archive file:

```
$ unzip master.zip
$ rm master.zip
```

Next, switch to the example project directory:

```
$ cd sonar-scanning-examples-master/sonarqube-scanner
```

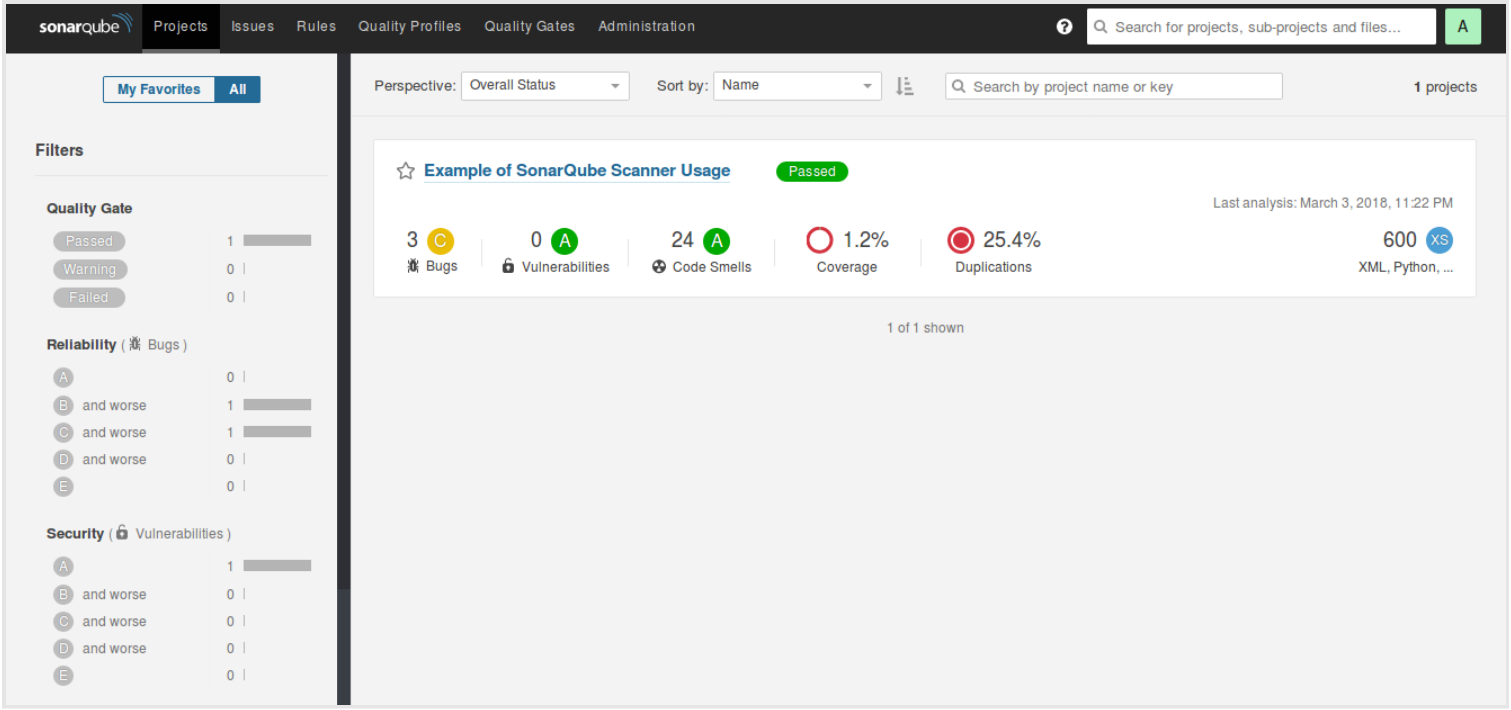
Run the scanner, passing it the token you created earlier:

```
$ sonar-scanner -D sonar.login=your_token_here
```

This will take a while. Once the scan is complete, you'll see something like this on the console:

```
INFO: Task total time: 14.128 s
INFO: ------------------------------------------------------------------------
INFO: EXECUTION SUCCESS
INFO: ------------------------------------------------------------------------
INFO: Total time: 21.776s
INFO: Final Memory: 17M/130M
INFO: ------------------------------------------------------------------------
```

The example project's report will now be on the SonarQube dashboard like so:



Now that you've confirmed that the SonarQube server and scanner works with the test code, you can use SonarQube to analyze your own code.

Step 8 — Running a Scan on Your Own Code

To have SonarQube analyze your own code, start by transferring your project to the server, or follow Step 6 to install and configure the SonarQube scanner on your workstation and configure it to point to your SonarQube server.

Then, in your project's root directory, create a SonarQube configuration file:

```
$ nano sonar-project.properties
```

You'll use this file to tell SonarQube a few things about your project.

First, define a *project key*, which is a unique ID for the project. You can use anything you'd like, but this ID must be unique for your SonarQube instance:

sonar-project.properties

```
# Unique ID for this project
sonar.projectKey=foobar:hello-world

...
```

Then, specify the project name and version so that SonarQube will display this information in the dashboard:

sonar-project.properties

```
...

sonar.projectName=Hello World Project
sonar.projectVersion=1.0

...
```

Finally, tell SonarQube where to look for the code files. Note that this is relative to the directory in which the configuration file resides. Set it to the current directory:

sonar-project.properties

```
# Path is relative to the sonar-project.properties file. Replace "\" by "/" on Windows.
sonar.sources=.
```

Close and save the file.

You're ready to run a code quality analysis on your own code. Run `sonar-scanner` again, passing it your token:

```
$ sonar-scanner -D sonar.login=your_token_here
```

Once the scan is complete, you'll see a summary screen similar to this:

```
INFO: Task total time: 5.417 s
INFO: ------------------------------------------------------------------------
INFO: EXECUTION SUCCESS
INFO: ------------------------------------------------------------------------
INFO: Total time: 9.659s
INFO: Final Memory: 39M/112M
INFO: ------------------------------------------------------------------------
```

The project's code quality report will now be on the SonarQube dashboard.

Conclusion

In this tutorial, you've set up a SonarQube server and scanner for code quality analysis. Now you can make sure that your code is easily maintainable by simply running a scan — SonarQube will tell you where the potential problems might be!

From here, you might want to read the SonarQube Scanner documentation to learn how to run analysis on your local development machine or as part of your build process.

By: Namo

Upvote (3) Subscribe Share

Editor:
Haley Mills

Related Tutorials

How to Manually Set Up a Prisma Server on Ubuntu 18.04

How To Display Data from the DigitalOcean API with React

How To Set Up Jupyter Notebook with Python 3 on Ubuntu 18.04

How to Install Node.js and Create a Local Development Environment on macOS

How To Use Git: A Reference Guide

2 Comments

Leave a comment...

SCROLL TO TOP

Log In to Comment

Gilasiha *January 13, 2019*

0 Great!
check new articles at gilasiha.ir

teleworm1337 *January 18, 2019*

0 Quality code will make the task of maintaining and expanding your application easier. https://get-shareit.com