Sonarqube-Day-2

28 November 2023 23:20

- **Console overview of Sonarqube**

- **Projects:**
  - Once you logged into the console, when you click on the projects it will be empty for the first time and here we can create projects based on our requirement.

  - This project console will started getting filled up when you try to push the analysis reports of your source code projects.

  - In sonarqube we can create projects in two ways,
    - Manually
    - Another one is by integrating sonarqube with other source code management tools like
      - □ GithHub
      - □ Gitlab
      - □ Azure devops
      - □ Bit bucket

  - Now let's start create a project manually,
    To create a project manually we have to enter owner project key & now I gave
    project key as **calculator-dev**
    sonar.login as calculator-dev

  - sonar.login is used to authenticate either from Jenkins/cmd prompt to the sonarqube.

  - Now we have to find out the command that need to be executed to publish analysis reports of your projects into sonarqube.
    Click on continue, it will give the command to execute.

  - As of now I am just saving this command into the notepad after sometime I will show you how to execute this command from the command prompt.

- The next tab **issues** it will show you the list of the issues that are present in your source code related project as of now it will be empty sincewe don't have any projects.

- **Next rules**
  - As I said earlier rules are best practices that each developer keep in mind while writing a code for his project.

  - Sonarqube supports almost 27 programming language.

  - Each programming language having some set of rules
    - Java is having 670 rules &
    - .NET having 400
      Like that each programming language having some number of rules if there is any deviation from this rules that will be reportas an bug or duplication or vulnerability.

- Next **quality profiles**
  - Quality Profile is nothing but **collection of the rules & these collection of rules(quality profile) will be applied on code during analysis of the project.**

  - By default each programming language associated with **sonar-way** quality profile.
    If you see the Java related sonar-way quality profile it's having 483 rules are active & 143 rules are inactive by default applicable to any Java related projects.

  - We can create a quality profiles on our own instead of default one and we can enable/disable some more extra rules in that quality profiles depend on the need.

  - Show this practically.
    Create java-custom-profile with 626 active rules.

- **UC: Now we will see practically how to generate static code analysis report & publish it to sonarqube.**

  - **Pre-requisites**
    - EC2 server
    - Git
    - Maven
    - Sonar-scanner
    - Sonarqube server
    - GitHub(java-repo):

  - **Clone repo:**

git clone https://github.com/chaitanyaredd/onlinebookstore.git

- **Build:**

  ```
  mvn clean install
  ```

- **Run static code analysis:**

  ```
  mvn  sonar:sonar \
      -Dsonar.projectKey=calculator
      -Dsonar.projectName=calculator
      -Dsonar.login=XXXXXXXXXXX
      -Dsonar.host.url=http://<ip-address>:9000
  ```

- Now in during code analysis sonar-way quality profile applied on code which was containing 400+ rules for java code & generated static code analysis report. This report is available in sonarqube with project name calculator.

- Here we can see all details like
  - Bugs
  - Vulnerabilities
  - Code coverage
  - Code duplications
  - Code smells

- **UC: Change quality profile for the sonar-project & new custom profile must have 600+ rules.**

  - Modify the quality profile of project to custom-java-profile.

  - Re-run static code analysis on code,

    ```
    mvn  sonar:sonar \
        -Dsonar.projectKey=calculator
        -Dsonar.projectName=calculator
        -Dsonar.login=XXXXXXXXXXX
        -Dsonar.host.url=http://<ip-address>:9000
    ```

  - Now quality profile of the project modified from **sonar-way** to **custom-java-profile**

- **Next one is quality gates**
  - Quality gates are nothing but setting threshold levels for the each measurements of project & if any measurement is below that threshold level that Sonar project status will be should failed.

  - Like if project is containing
    - bugs more than one that project status should be filled.
    - If project would coverage is less than 80% that project status will be failed.

  - Let's setup quality gate(custom-java-gate_ instead default quality gate.

  - **UC: Modify the quality gate of calculator application to custom-java-gate & make project status as failed.**
    - **Clone repo:**

      git clone https://github.com/chaitanyaredd/onlinebookstore.git
    - **Build:**

      ```
      mvn clean install
      ```
    - **Run static code analysis:**

      ```
      mvn  sonar:sonar \
          -Dsonar.projectKey=calculator
          -Dsonar.projectName=calculator
          -Dsonar.login=XXXXXXXXXXX
          -Dsonar.host.url=http://<ip-address>:9000
      ```

    - Now the calculator project status will be in failed state.

- **Homework:**
  - **Repo:** https://github.com/chaitanyaredd/petsclinic.git
  - Do the static code analysis on java project & publish report to sonarqube
  - Create custom quality & update it sonar project
  - Create custom quality gate & make the project as failed
  - Generate custom sonar.login token & use it this token during latest scan.