Automated Website Deployment with Jenkins, SonarQube, and Docker

Automated Website Deployment with Jenkins, SonarQube, and Docker" signifies a project or initiative that involves deploying a website using an automated approach with the help of three key technologies: Jenkins, SonarQube, and Docker.

Technologies Used :

- Jenkins
- Ubuntu
- AWS (EC2 instances)
- Docker
- Github

Steps that were followed in the "Automated Website Deployment with Jenkins, SonarQube, and Docker" project:

a) Source Code Acquisition:

- Downloaded a website template from an online resource.

b) GitHub Repository:

- Uploaded the website template code to a GitHub repository.

- Created a centralized and collaborative space for code storage and version control.

c) Jenkins Pipeline:

- Configured a Jenkins pipeline for automation. The pipeline includes steps for building and deploying the website code.

- Automated the process of compiling and packaging the code.

d) SonarQube Quality Analysis:

- Set up a separate AWS instance.

- Installed SonarQube on the instance.

- Configured SonarQube to analyze the quality of the website's code.

- Conducted code quality checks to identify issues, vulnerabilities, and code quality violations.

- Ensured that the code met high-quality standards.

e) Dockerized Deployment:

- Set up another AWS instance.
- Created a Docker image of the website.
- Configured Docker to deploy the website using the Docker image.
- Dockerization ensures consistency and portability across different environments.

f) Continuous Integration:

- Jenkins enables continuous integration, automatically triggering builds and tests upon code changes.
- This accelerates the development cycle and ensures code quality throughout the project.

g) Cost Optimization:

- Carefully managed infrastructure resources on AWS to optimize costs.
- Leveraged the AWS Free Tier to minimize expenses.

Key features and benefits of the "Automated Website Deployment with Jenkins, SonarQube, and Docker" project:

1. Efficiency: Automation with Jenkins streamlines the entire deployment process, reducing manual intervention and human errors. This leads to faster and more reliable deployments.

2. Code Quality Assurance: SonarQube ensures that the website's code meets high-quality standards by identifying and reporting issues, vulnerabilities, and code smells. This helps in maintaining a clean and secure codebase.

3. Consistency: Docker containerization ensures that the website runs consistently across different environments. It eliminates "it works on my machine" issues and ensures that the website behaves consistently in development, testing, and production environments.

4. Scalability: Docker containers can be easily scaled up or down, allowing for efficient resource utilization and the ability to handle increased traffic or workloads as needed.

5. Version Control: Using Git and GitHub for code storage allows for version control, collaboration, and tracking changes over time. This enhances code management and collaboration among team members.

6. Resource Optimization: Managing infrastructure resources on AWS, including instances and storage, optimizes costs by utilizing the AWS Free Tier and selecting the appropriate instance types.

7. Java Compatibility: Ensuring that all instances have Java JDK 17 installed and aligned minimizes compatibility issues and guarantees a consistent runtime environment.

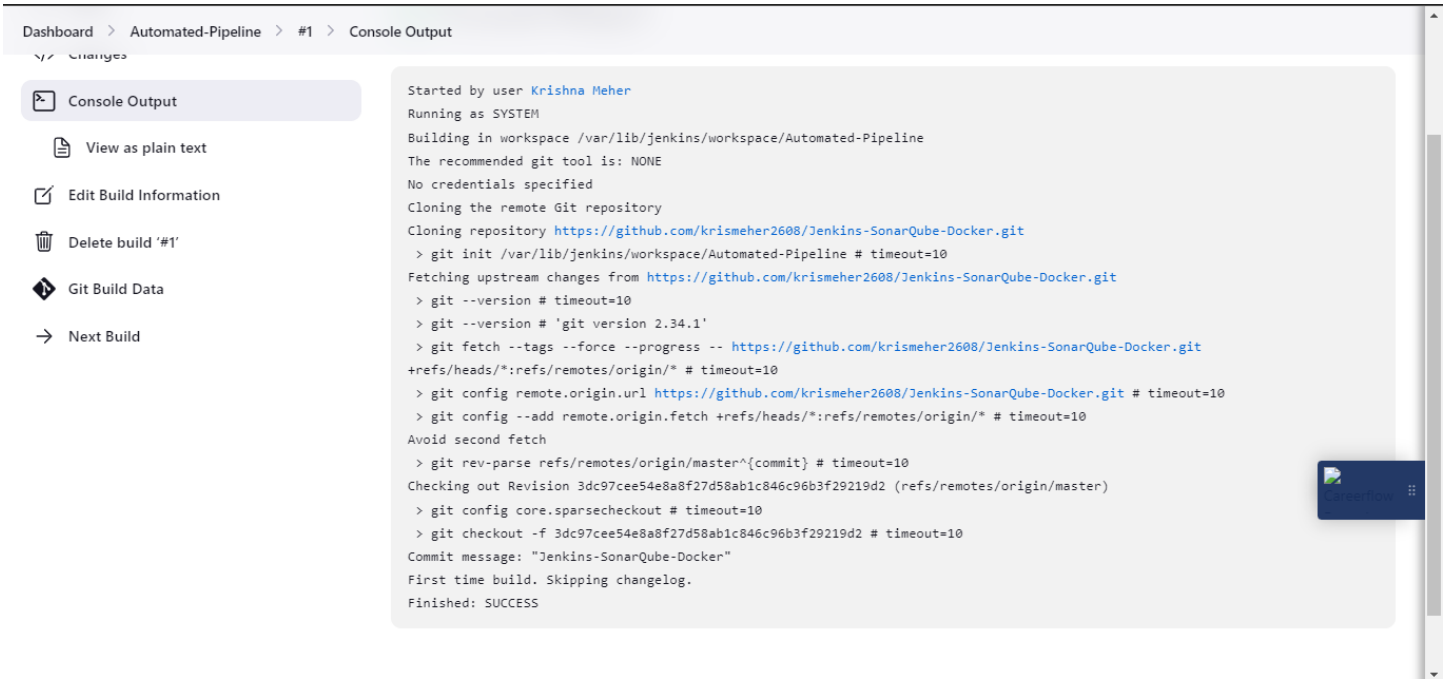
8. Continuous Integration: Jenkins enables continuous integration, automatically building and testing code changes as they are pushed to the repository. This accelerates the development cycle and ensures code quality from the beginning.

9. Security: SonarQube's code analysis helps identify and address security vulnerabilities early in the development process, reducing the risk of security breaches.

11. Cost Optimization: Careful resource management on AWS, including using the AWS Free Tier wherever possible, helps keep infrastructure costs in check.

Results of the Deployment in sequence :

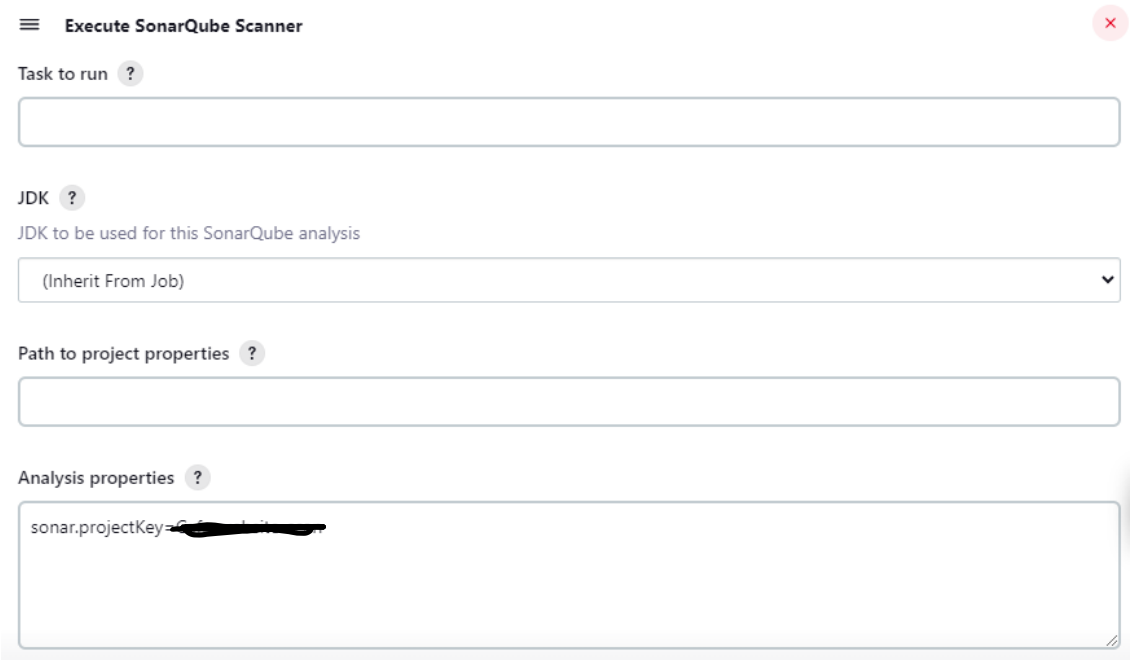
Jenkins : (Building the code)



SonarQube:

Setting the configuration in the Jenkins :

1. Adding the sonar.projectKey for the integration of SonarQube with Jenkins for the particular project



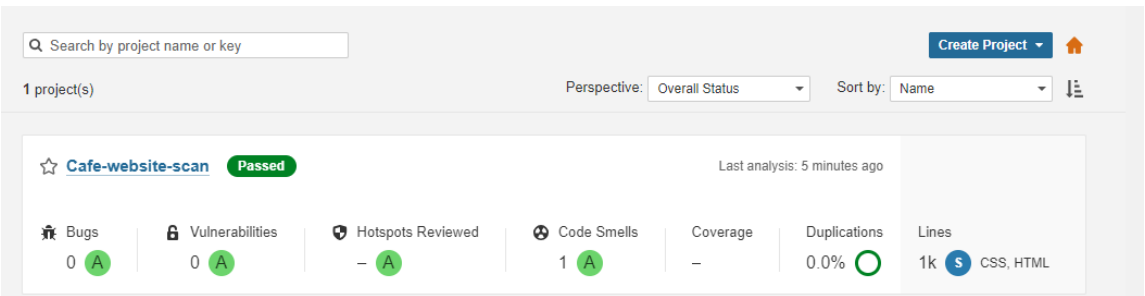
2. Building the code again after integration to check for the quality issues :
   Jenkins console :
   Error : jdk wasn't same un both Jenkins and SonarQube instances

```
 > git checkout -f 72dc82b9de37c2dd23f5fa135dd80dab1b9f3bf4 # timeout=10
Commit message: "Testing Webhooks"
 > git rev-list --no-walk 72dc82b9de37c2dd23f5fa135dd80dab1b9f3bf4 # timeout=10
[Automated-Pipeline] $ /var/lib/jenkins/tools/hudson.plugins.sonar.SonarRunnerInstallation/SonarScanner/bin/sonar-scanner -
Dsonar.host.url=http://54.163.38.77:9000 ******** -Dsonar.projectKey=Cafe-website-scan -
Dsonar.projectBaseDir=/var/lib/jenkins/workspace/Automated-Pipeline
Error: LinkageError occurred while loading main class org.sonarsource.scanner.cli.Main
        java.lang.UnsupportedClassVersionError: org/sonarsource/scanner/cli/Main has been compiled by a more recent version
of the Java Runtime (class file version 61.0), this version of the Java Runtime only recognizes class file versions up to
55.0
WARN: Unable to locate 'report-task.txt' in the workspace. Did the SonarScanner succeed?
ERROR: SonarQube scanner exited with non-zero code: 1
Finished: FAILURE
```

After aligning with the same version of jdk got the expected output:

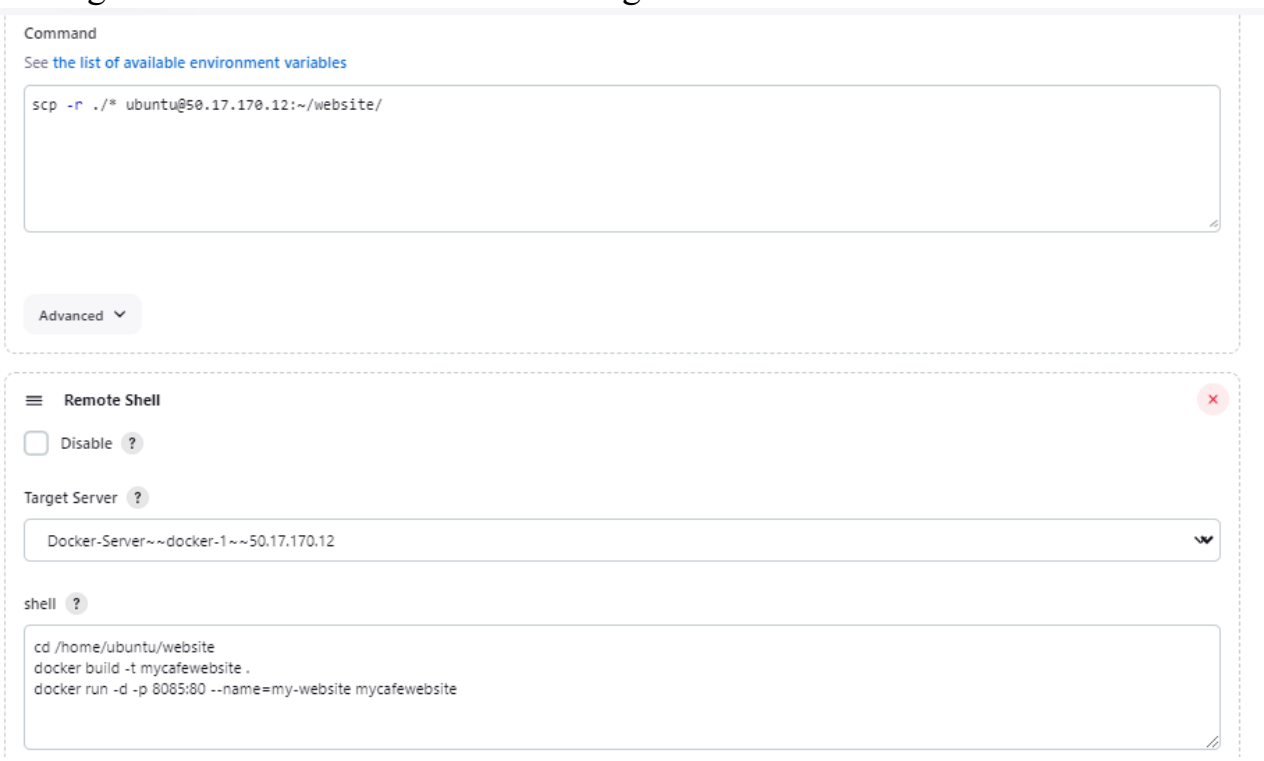
```
INFO: SCM Publisher 3/3 source files have been analyzed (done) | time=177ms
INFO: CPD Executor Calculating CPD for 1 file
INFO: CPD Executor CPD calculation finished (done) | time=8ms
INFO: Analysis report generated in 90ms, dir size=184.6 kB
INFO: Analysis report compressed in 16ms, zip size=35.8 kB
INFO: Analysis report uploaded in 40ms
INFO: ANALYSIS SUCCESSFUL, you can find the results at: http://54.163.38.77:9000/dashboard?id=Cafe-website-scan
INFO: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
INFO: More about the report processing at http://54.163.38.77:9000/api/ce/task?id=AYphvBbupU7GPyeZXqBf
INFO: Analysis total time: 10.170 s
INFO: ------------------------------------------------------------------------
INFO: EXECUTION SUCCESS
INFO: ------------------------------------------------------------------------
INFO: Total time: 14.292s
INFO: Final Memory: 16M/60M
INFO: ------------------------------------------------------------------------
Finished: SUCCESS
```

Results in the SonarQube Scanner:



Docker :

1. Setting the environment in Jenkins to integrate with Docker :



The website on the successful completion of Docker Image building and running it :

http://50.17.170.12/8085