Report of static source code analysis through SonarQube plugin

I am pleased to report that we have successfully completed a static source code analysis using the SonarQube plugin in our Jenkins pipeline for our Python-based project.

1. **Setup and Configuration**

   The pipeline starts with a checkout from the remote Git repository 'https://github.com/HassaanKhurram/COMP421-A2-251684003.git', specifically from the 'python' branch.

2. **SonarQube Scan**

   The next stage was the SonarQube scan. This was set up to use the SonarQube environment "SonarServer". The scan was initiated using a shell script command running the Python script 'hazrat.py'. This allowed the SonarQube scanner to analyze our source code and provide insights about its quality.

3. **Build**

   After the SonarQube scan, the pipeline proceeded to the build stage, where the same Python script was executed. This helped verify the functionality of the code along with the static analysis.

4. **Test**

   While the pipeline as provided didn't perform any specific tests, in general, this stage would involve using tools like pytest for unit testing or Snyk for security testing.

5. **Deploy**

   The last stage in the pipeline was the deployment stage, marked as successful after the preceding stages were completed without errors.

The integration of the SonarQube plugin within our Jenkins pipeline provided several benefits:

- We were able to perform automatic reviews with static code analysis to detect bugs, code smells, and security vulnerabilities.
- This allowed us to ensure that the code complies with coding standards and is maintainable and secure.

- The setup was also helpful in reducing technical debt by making sure that all issues are identified early on in the development process.

However, it's worth noting that the provided Jenkins pipeline script did not include specific steps to perform SonarQube analysis on the source code. To do this, you'd typically use a command similar to `sh 'sonar-scanner -Dsonar.projectKey=hazrat -Dsonar.sources=hazrat.py'` as suggested in the commented code.

The pipeline as provided executes the Python script 'hazrat.py' within the SonarQube environment, but without an explicit call to sonar-scanner, SonarQube may not perform an analysis. The details of how this was set up may be dependent on your specific SonarQube server configuration.