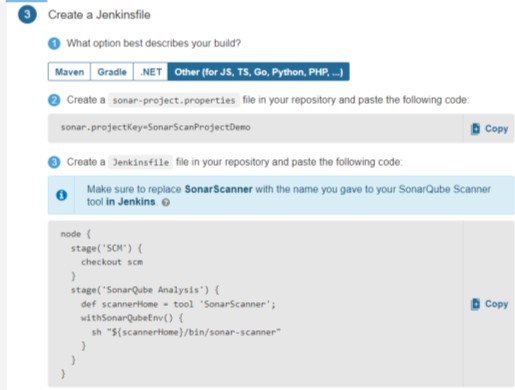
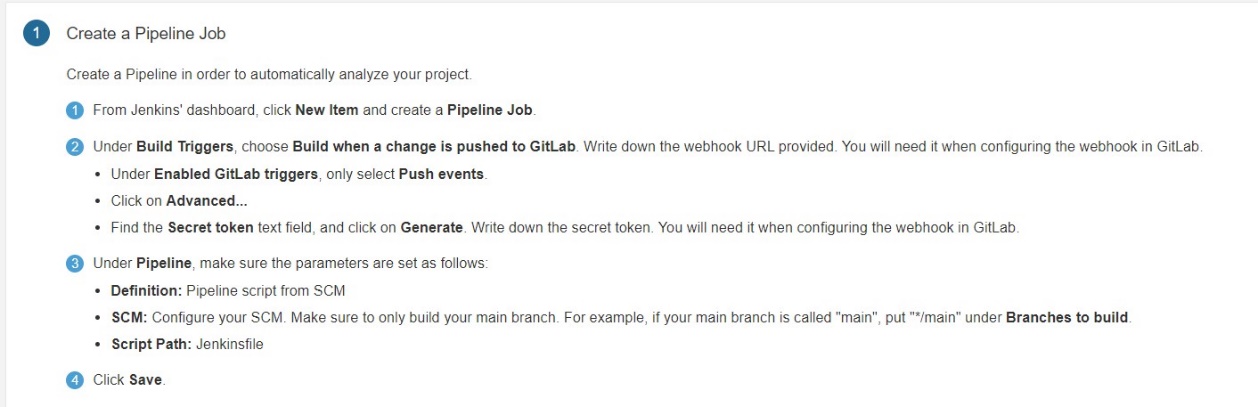
|  |  |
| --- | --- |
| A close up of a logo  Description automatically generated | *DEPARTMENT OF INFORMATION TECHNOLOGY* |

|  |  |
| --- | --- |
| Semester | T.E. Semester V – Information Technology |
| Subject | Advance DevOps Lab |
| Subject Professor In-  charge | Prof.Indu Anoop |
| Laboratory |  |

|  |  |  |
| --- | --- | --- |
| Student Name | Shreyash Kakde | |
| Roll Number | 20101A0049 | |
| Grade and Subject Teacher’s Signature |  |  |

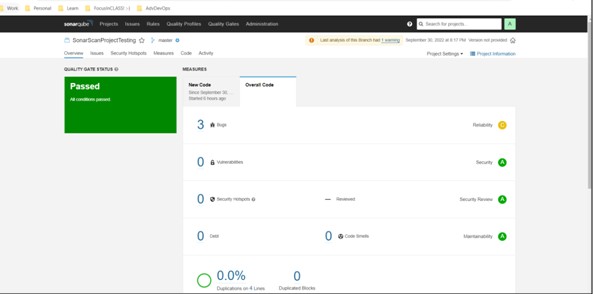
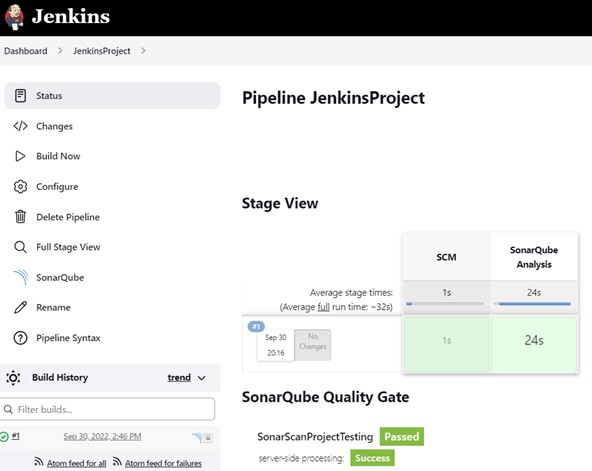
|  |  |  |
| --- | --- | --- |
| Experim ent | 8 | |
| Problem Stateme  nt | Create a Jenkins CICD pipeline with SonarQube/GitLab Integration to perform a static analysis of the code to detect bugs , code smells, and security vulnerabilities  on a sample Web/Java/Python application. | |
| Resourc es / Apparat us Require  d | Hardware:Computer System | Software:Web Browser |
| Details | Static application security testing (SAST), or static analysis, is a testing methodology that analyses source code to find security vulnerabilities that make your organization's applications susceptible to attack. SAST scans an application before the code is compiled. It's also known as white box testing.  What is SonarQube?  SonarQube is an open-source platform to check the quality of the code. We can generate reports of code with duplicate codes, dead codes, logical error, null-pointers, coding guidelines, testing, bugs, security vulnerabilities, code coverage, etc. We can set up Quality Gates, which allows setting policies like coverage by the new code, bugs count, security rating, reliability rating, etc. This ensures that no build is deployed to production or other environments without passing the quality standard. SonarQube support quality check for most of the well-known languages like Java, C/C++, Groovy, Python, Javascript, PHP, Swift, etc.  Software quality measurement is a quantitative process summing up weighted attribute values, which in part describe specific software characteristics. For each characteristic, a set of such measurable attributes is defined.  Software characteristics: | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1. Whether the coding has been done following a specific convention 2. Whether well-known/established good practices have been followed and well- known/established bad practices have been avoided 3. Are there any potential bugs and performance issues, security vulnerabilities 4. Is there any duplicate code 5. Is the code logic very complex 6. Whether the public API has good documentation and comments 7. Whether the code has unit tests 8. Whether the code follows good design and architecture principles     Important terms: | | | |
|  | **Quality Gate** | is an indicator of code quality that can be configured to give a go/no-go signal on the current release-worthiness of the code. It indicates whether your code is clean and can move forward.   * A passing (green) Quality Gate means the code meets your standard and is ready to be merged. * A failing (red) Quality Gate means there are issues to address. |  |
| **Bug** | An issue that represents something wrong in the code. If this has not broken  yet, it will, and probably at the worst possible moment. This needs to be fixed. |  |
| **Code Smell** | A maintainability-related issue in the code. Leaving it as-is means that at best maintainers will have a harder time than they should making changes to the code. At worst, they'll be so confused by the state of the code that they'll introduce additional errors as they make changes. |  |
| **Vulnerability** | A security-related issue which represents a backdoor for attackers |  |
| Code | **Prerequisites [Refer Exp7: After Integration]**   1. A SonarQube Server [Refer SonarQube Installation document] 2. A Jenkins server [Refer Jenkins Installation document] 3. GitLab account 4. Integration of all three of above | | | |



**PART D: Trigger a build by committing code to GitLab and Jenkins CICD pipeline job begins**

**PARTC: Create a Jenkins CICD pipeline job and create a webhook**



PART E: View static analysis report

|  |  |
| --- | --- |
|  |  |
| Conclusi on | Created a Jenkins CICD pipeline with SonarQube/GitLab Integration to perform a  static analysis of the code to detect bugs , code smells, and security vulnerabilities on a sample Web application. |