**EXPERIMENT NO. – 7**

**AIM:** To describe Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube / GitLab

**LO:** LO1:- To explain the fundamentals of Cloud Computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements.

LO4:- To identify and solve application vulnerabilities earlier and help integrate security in the development process using SAST Techniques.

**THEORY:**

* **SAST (Static Analysis Security Testing)**

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 Problems does SAST solve?**

SAST takes place very early in the software development life cycle ([SDLC](https://www.synopsys.com/glossary/what-is-sdlc.html)) as it does not require a working application and can take place without code being executed. It helps developers identify vulnerabilities in the initial stages of development and quickly resolve issues without breaking builds or passing on vulnerabilities to the final release of the application.

SAST tools give developers real-time feedback as they code, helping them fix issues before they pass the code to the next phase of the SDLC. This prevents security-related issues from being considered an afterthought. SAST tools also provide graphical representations of the issues found, from source to sink. These help you navigate the code easier. Some tools point out the exact location of vulnerabilities and highlight the risky code. Tools can also provide in-depth guidance on how to fix issues and the best place in the code to fix them, without requiring deep security domain expertise.

Developers can also create the customized reports they need with SAST tools; these reports can be exported offline and tracked using dashboards. Tracking all the security issues reported by the tool in an organized way can help developers remediate these issues promptly and release applications with minimal problems. This process contributes to the creation of a secure SDLC.

It’s important to note that SAST tools must be run on the application on a regular basis, such as during daily/monthly builds, every time code is checked in, or during a code release.

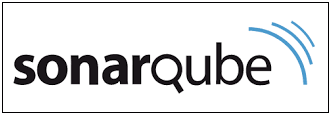
* **Why SAST is an important Security Activity?**

Developers dramatically outnumber security staff. It can be challenging for an organization to find the resources to perform code reviews on even a fraction of its applications. A key strength of SAST tools is the ability to analyze 100% of the code base. Additionally, they are much faster than manual secure code reviews performed by humans. These tools can scan millions of lines of code in a matter of minutes. SAST tools automatically identify critical vulnerabilities—such as [buffer overflows](https://www.synopsys.com/blogs/software-security/detect-prevent-and-mitigate-buffer-overflow-attacks/), [SQL injection](https://www.synopsys.com/glossary/what-is-sql-injection.html), [cross-site scripting](https://www.synopsys.com/glossary/what-is-cross-site-scripting.html), and others—with high confidence. Thus, integrating static analysis into the SDLC can yield dramatic results in the overall quality of the code developed.

* **SonarQube**

It is an open-source security tool which is established by Sonar Source. It is used to test the quality of the code and execute the automatic reviews with the help of identifying the bugs, code analysis and security exposures on various programming languages such as Java, [C#](https://www.javatpoint.com/c-sharp-tutorial), [JavaScript](https://www.javatpoint.com/javascript-tutorial), [PHP](https://www.javatpoint.com/php-tutorial), Ruby, [Cobol](https://www.javatpoint.com/cobol), [C](https://www.javatpoint.com/c-programming-language-tutorial)/[C++](https://www.javatpoint.com/cpp-tutorial) and so on of the web applications. SonarQube tool is written on the [JAVA programming language](https://www.javatpoint.com/java-tutorial).

It will generate the reports of the code coverage, complexity of code, repeated code, security weakness, and bugs. It offers complete analysis with multiple tools like [Ant](https://www.javatpoint.com/apache-ant-tutorial), [Maven](https://www.javatpoint.com/maven-tutorial), [Gradle](https://www.javatpoint.com/gradle), [Jenkins](https://www.javatpoint.com/jenkins), and so on.



**Features of SonarQube:**

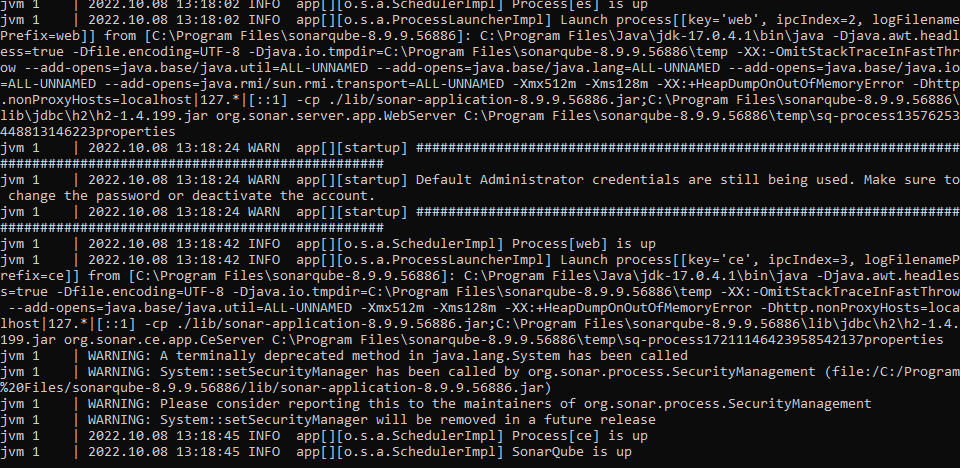
* It will integrate with multiple development environments like Visual Studio, Eclipse, and IntelliJ IDEA over the SonarLint plug-ins.
* It also supports some external tools such as GitHub, LDAP, and Active Directory.
* It can record the metric history and deliver the evolution graphs.
* It will help us to identify the complex issues.
* It will provide application security.

**SonarScanner:**

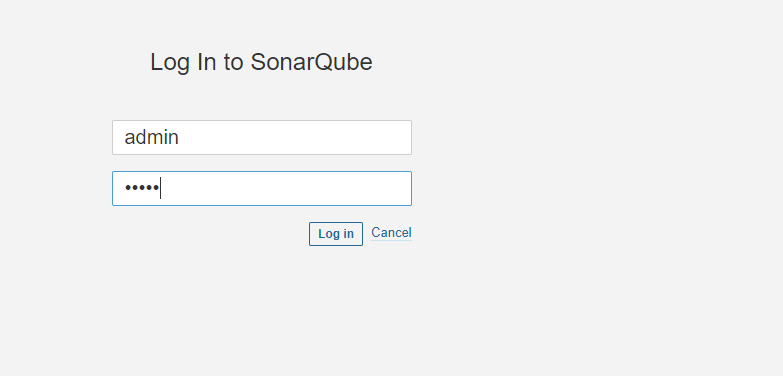
* SonarScanner is a separate client type application that in connection with the SonarQube server will run project analysis and then send the results to the SonarQube server to process it. SonarScanner can handle most programming languages supported by SonarQube except C# and VB. It is usually located on continuous integration agents (workers) or in separate docker images depending on your project flow.

**Output:-**

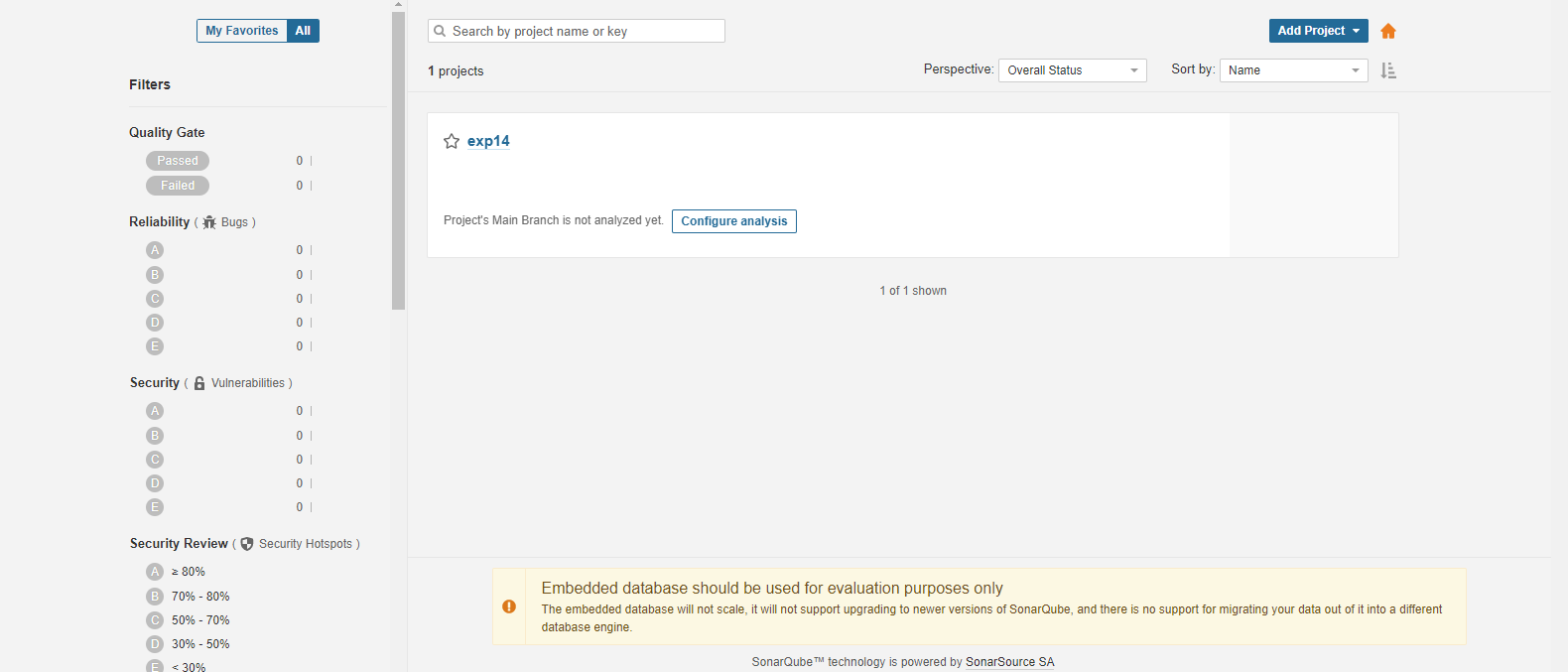
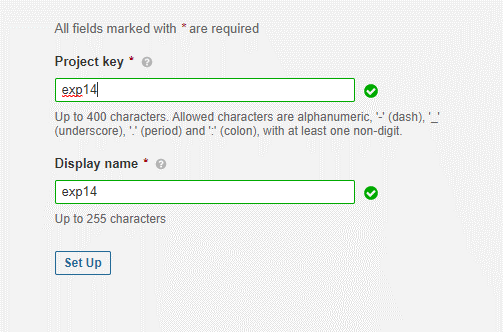
1. Open sonarcube on your sysem



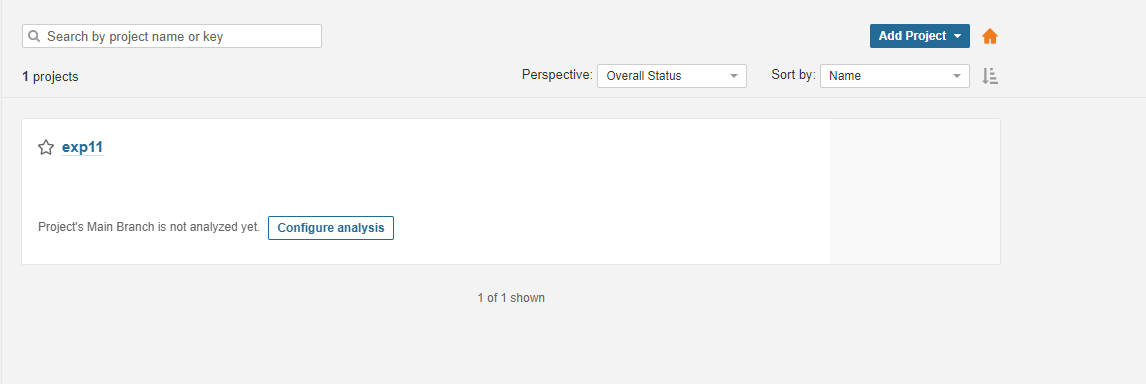
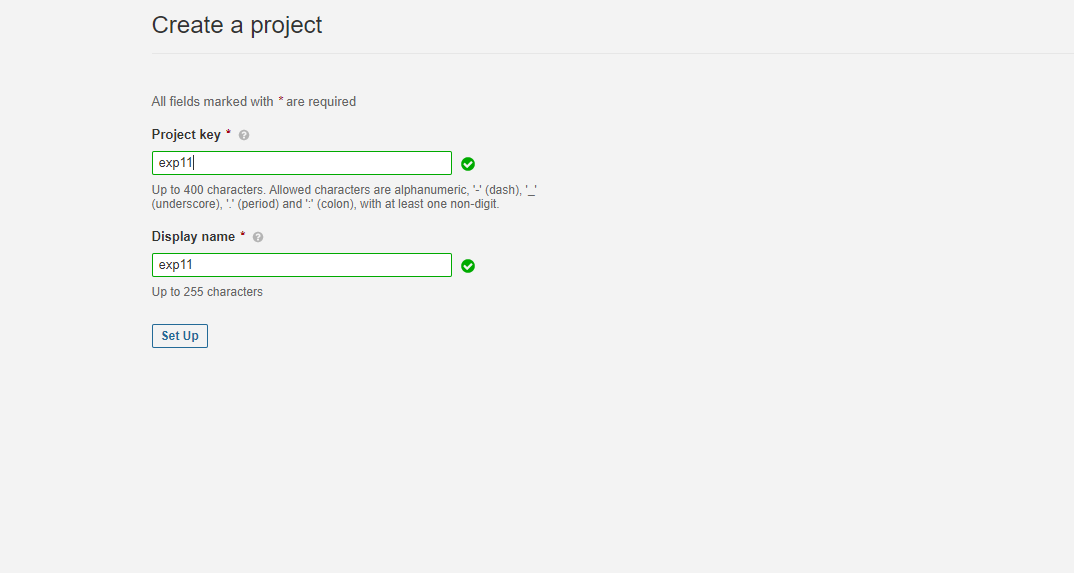
1. Login in SonarCube



3) Create a Organization



4) Creating a Project



**Conclusion:-**