INDUSTRIAL TRAINING REPORT

ON

AWS AND DEVOPS

AT

SV GLOBAL SERVICES INDIA PRIVATE LIMITED

Submitted in the partial fulfillment of the

Requirement for the award of

IN

DIPLOMA IN COMPUTER SCIENCE AND ENGINEERING

Submitted by

GOGU ABHINAV

(22054-CS-054)

GOVERNMENT INSTITUTE OF ELECTRONICS

(Affiliated to SBTET-TS, HYD), EAST MARREDPALLY

SECUNDERABAD, 500026

GOVERNMENT INSTITUTE OF ELECTRONICS

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitle **"AMAZON WEB SERVICES (AWS) AND DEVOPS"** has been carried by **GOGU ABHINAV** bearing PIN **22054-CS-054**

in partial fulfilment for the award of DIPLOMA IN COMPUTER SCIENCE ENGINEERING to the STATE BOARD OF TECHNICAL EDUCATION AND TRAINING, Hyderabad at GOVERNMENT INSTITUTE OF ELECTRONICS, East Marredpally , Hyderabad during the Academic year 2024-2025

INTERNAL EXAMINER **EXTERNAL EXAMINER**

HEAD OF DEPT **PRINCIPAL**

ACKNOWLEDGEMENT

The Immense satisfaction and delight that accompanies the successful completion this task would be incomplete without the mention of the people whose constant guidance and encouragement have crowned our efforts with success.

Our sincere regards to **MRS. P. ANNAPURNA [M.E], PRINCIPAL, GOVERNMENT INSTITUTE OF ELECTRONICS** for this encouragement and support during all the stages of the project.

We express our sincere gratitude to **MR S.P VENKAT REDDY [M TECH] , HEAD OF THE DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, GOVERNMENT INSTITUTE OF ELECTRONICS** for his interminable support and encouragement and providing us with outstanding facilities for the successful completion of our project.

It is our privilege and pleasure to express my found sense of respect, gratitude and indebtedness to our guide **MRS. V. KALPANA [L/CSE]**, for her indefatigable inspiration, cogent discussion, constructive criticisms and encouragement throughout the dissertation work.

We are also thankful to other staff members for their support during the course of the project and in completing the project successfully.

We thank all those who directly and indirectly helped us in the successful completion of the project in time

DECLARATION:

We the undersigned declare that the project report entitled **"INDUSTRIAL TRAINING REPORT"** written and submitted by us in an original work done under the guidance of **Sri. T. NITESH.** The matter here in is not reproduced from any other source. I hereby declare that this project was outcome of efforts and is not been submitted to any other university for the award of any degree or diploma.

GOGU ABHINAV **22054-CS-054**

ABSTRACT

This industrial training report presents a comprehensive overview of the practical experience and technical knowledge gained during the six-month training period at SV Global Services India Pvt. Ltd., Hyderabad, focusing on the core areas of AWS and DevOps. The project emphasizes the significance of modern software development methodologies and deployment strategies through the implementation of DevOps tools such as Git, Jenkins, Maven, Docker, Kubernetes, and Selenium.

The report also explores the role of infrastructure as code (IAC), containerization, and monitoring in streamlining IT operations and improving development productivity. Through real-time application of tools and collaborative practices, the training enhanced the team's skills in project automation, deployment, and infrastructure management, aligning with current industry standards.

This project marks a significant step towards bridging academic knowledge with industrial practices, preparing students for real-world DevOps and cloud engineering roles.

INDEX

SNO	CONTENTS	PAGE .NO
1	INTRODUCTION	7-8
	1.1 DevOps	9
	1.2 DevOps Technologies	10
	1.3 CI-CD Pipeline	11-12
	1.4 GIT & GIT HUB	13-14
	1.5 JENKINS	15
	1.6 DOCKER	16-17
2	Introduction to Jenkins	18
	2.1 Key Features of Jenkins	19
	2.2 Jenkins Applications	20-21
3	Introduction to Maven	22
4	Installing Jenkins and Mavens	23-25
5	Building a Maven project in Jenkins	26-28
6	Output	29
7	Conclusion and References	30

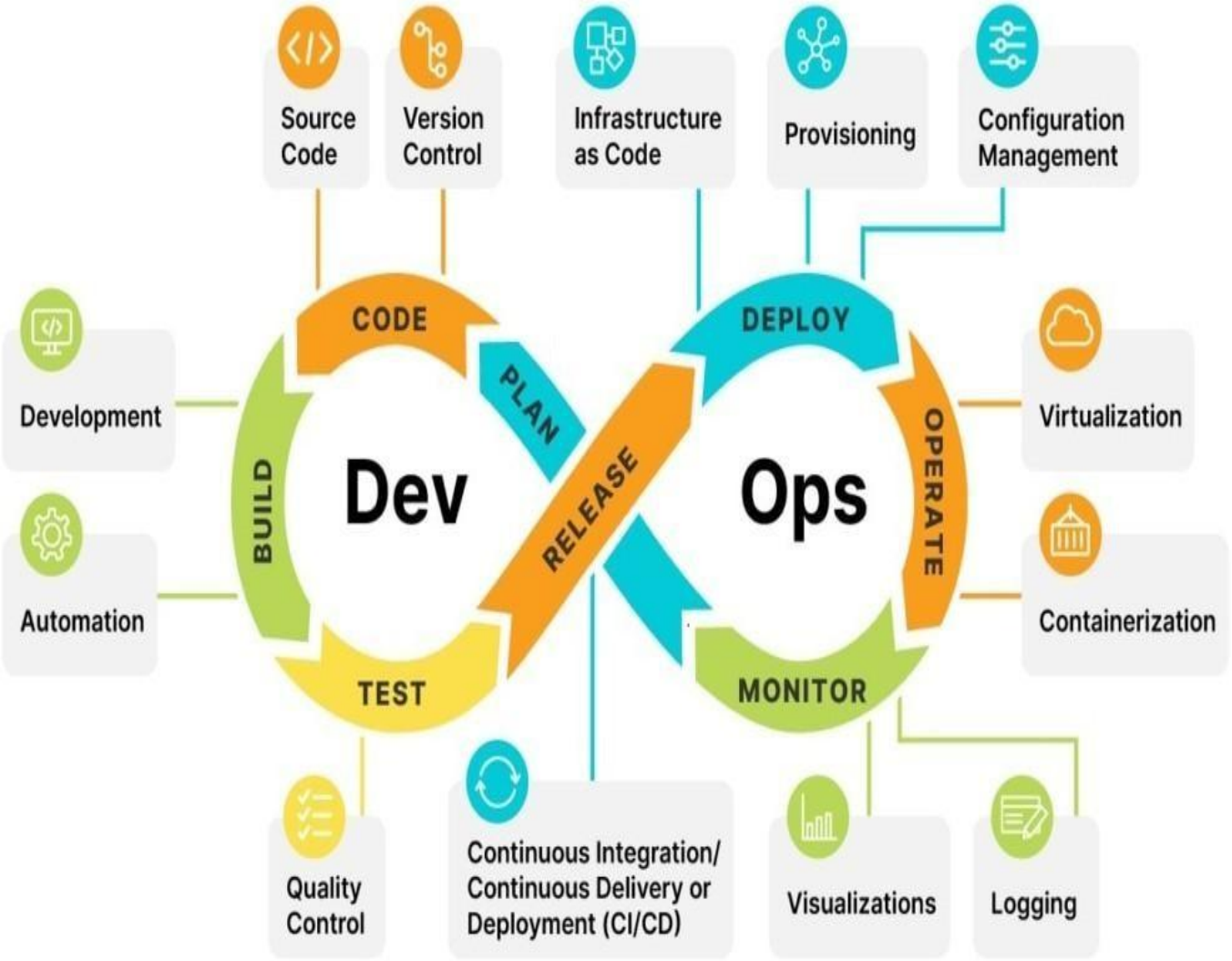
INTRODUCTION TO DEVOPS

Software is the backbone of modern technological advancement, powering everything from mobile applications to complex enterprise systems. In its essence, software refers to a set of instructions or programs that enable computers and other devices to perform specific tasks or functions. From simple utilities that organize daily tasks to sophisticated algorithms driving artificial intelligence, software encompasses a vast array of applications that shape our digital world.

As technology continues to evolve, so does the demand for innovative software solutions that streamline processes, enhance productivity, and enrich user experiences.

Understanding the principles of software development, including design, coding, testing, and maintenance, is essential for navigating today's digital landscape and driving meaningful change through technology.

Continuous growth in software is not merely a trend but a fundamental aspect of the industry's evolution. With technology advancing at an unprecedented pace, software development constantly expands its horizons to meet ever-changing demands and challenges. For this continuous growth of software DevOps also plays a crucial play. We use DevOps in almost every sector for better utilization and better productive


Source Code
Version Control
Infrastructure as Code
Provisioning
Configuration Management
CODE
PLAN
DEPLOY
Development
Virtualization
BUILD
Dev
RELEASE
Ops
OPERATE
Automation
Containerization
TEST
MONITOR
Quality Control
Continuous Integration/ Continuous Delivery or Deployment (CI/CD)
Visualizations
Logging

DEVOPS

DevOps an amalgamation of "development" and "operations," represents a transformative approach to software development and IT operations. It emphasizes collaboration, communication, and integration between traditionally siloed teams, aiming to streamline the software delivery process and improve overall organizational performance.

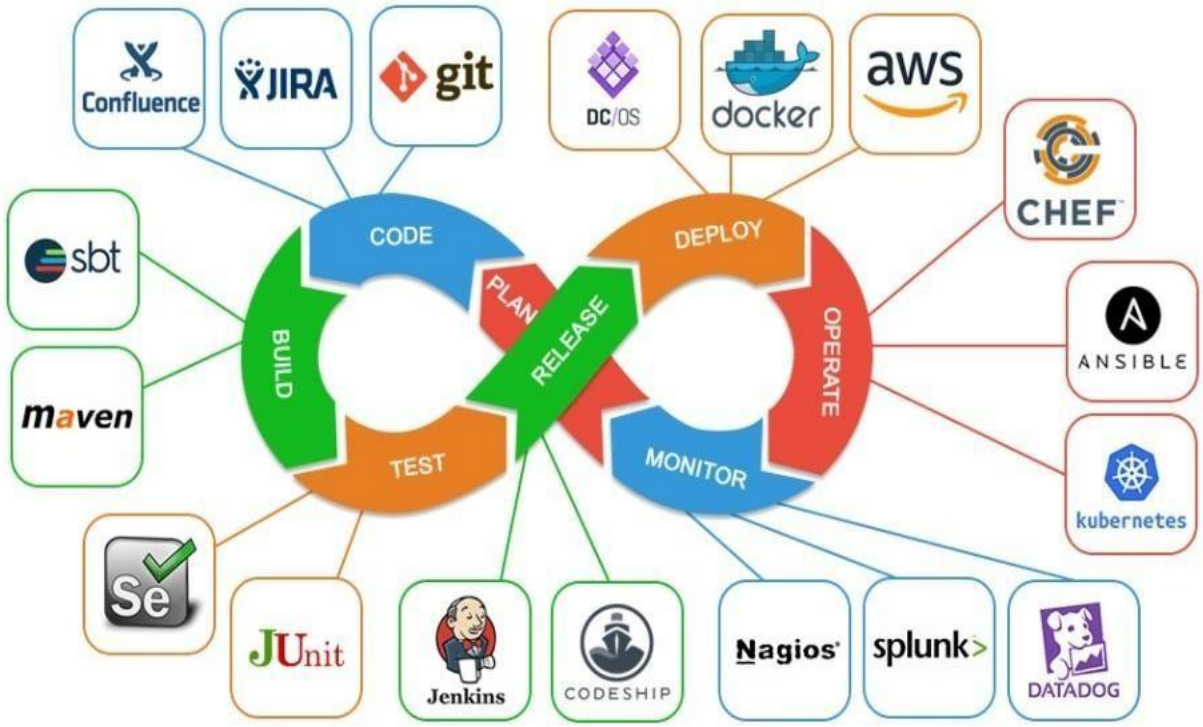
In today's fast-paced digital landscape, DevOps has emerged as a cornerstone of modern software development practices, empowering teams to innovate faster, respond to market changes more effectively, and drive business success.

Devops goes beyond continuous integration and continuous delivery to enable near-instantaneous deployment of products and services in the cloud.

DevOps Technologies

There are many technologies used in DevOps. Each technology is used for specific function.

Some of the technologies with their usage are:



- **GIT** – Used for writing code in DevOps.
- **MAVEN**- Used for build.
- **SELENIUM** – Used for test the project.
- **JENKINS** – Used for release of project.
- **DOCKER** – Used for deploy project.
- **KUBERNETES** – Used to operate project.
- **SPLUNK** – Used for monitoring project

CI–CD Pipeline

Continuous Integration (CI) and Continuous Delivery (CD) are key practices in DevOps aimed at automating and streamlining the software development and delivery process. A CI/CD pipeline is a set of automated processes that enable teams to build, test, and deploy software rapidly and reliably.

Continuous Integration (CI):

- Automated Builds: Developers commit code changes to a shared repository multiple times a day. Each commit triggers an automated build process, where the code is compiled, integrated, and tested in a controlled environment.
- Automated Testing: Automated tests, including unit tests integration tests and other forms of testing are executed as part of the build process. These tests at the code changes haven't introduced any regressions or defects.
- Code Quality Checks: Static code analysis tools are used to analyze the code for potential issues such as code smells, style violations, and security vulnerabilities. This helps maintain code quality and consistency across the codebase.
- Immediate Feedback: The CI process provides immediate feedback to developers, highlighting any build or test failures. Developers can quickly identify and fix issues before they escalate reducing the risk of integration problems and ensuring a stable codebase.

Continuous Delivery (CD):

- Automated Deployment: Once code changes have passed the CI process, they are automatically deployed to a staging or production-like environment. This allows teams to quickly validate changes in a real-world environment and gather feedback from stakeholders.

- Deployment Pipelines: CD pipelines orchestrate the deployment process, including provisioning infrastructure, configuring environments, deploying applications, and running smoke tests. Deployment pipelines are typically defined as code and version- controlled allowing for repeatability and consistency.

- Incremental Releases: CD enables teams to release software incrementally and frequently delivering value to customers faster and reducing time-to-market. Releases are small, incremental changes that can be deployed with minimal risk and disruption.

- Rollback Mechanisms: CD pipelines include rollback mechanisms that allow teams to revert to a previous known-good state in case of deployment failures or issues. This provides confidence in the deployment process and minimizes downtime or impact on users.

CI/CD Pipeline:

End-to-End Automation: The CI/CD pipeline automates the entire software delivery process from code commit to production deployment. It encompasses all stages of the development lifecycle, including build, test, and deploy.

Integration with Tools: CI/CD pipe lines integrate with various tools and services, such as version control systems (e.g.,Git), build servers (e.g.,Jenkins) ,testing frameworks, artifact repositories, and deployment platforms (e.g., Kubernetes, AWS, Azure).

Visibility and Traceability: CI/CD pipelines provide visibility and trace ability into the entire software delivery process

GIT

- GIT is a distributed version control system (DVCS) designed to track changes in source code during software development. Developed by Linus Torvaldsin 2005, Git has become one of the most popular version control systems in the world due to its speed, efficiency, and flexibility.

- Unlike centralized version control systems, Git is distributed, meaning every developer has a complete copy of the repository, including the entire history of changes. This allows developers to work offline commit changes locally and synchronize with remote repositories when online.

- Git provides powerful branching and merging capabilities, allowing developers to create isolated branches to work on features or fixes independently. Branches can be merged back into the main branch (e.g., ‘master’ or ‘main’) when the changes are complete, enabling parallel development and collaboration.

- Git supports remote repositories, allowing developers to collaborate with others and share code across different locations. Remote repositories serve as centralized hubs for sharing and synchronizing changes between team members. Common remote hosting services for Git repositories include GitHub, GitLab, and Bitbucket.

GIT HUB

- GitHub is a web-based platform built on top of Git, providing additional features and services for hosting Git repositories managing projects and collaborating with others.

- GitHub allows developers to host Git repositories online providing a central location for storing ,sharing, and collaborating on code. Repositories can be public, allowing anyone to view and contribute, or private, restricted to specific collaborators.

- GitHub integrates with a widerange of third-party tools and services, such as CI/CD pipelines, code quality analyzers, and project management tools.

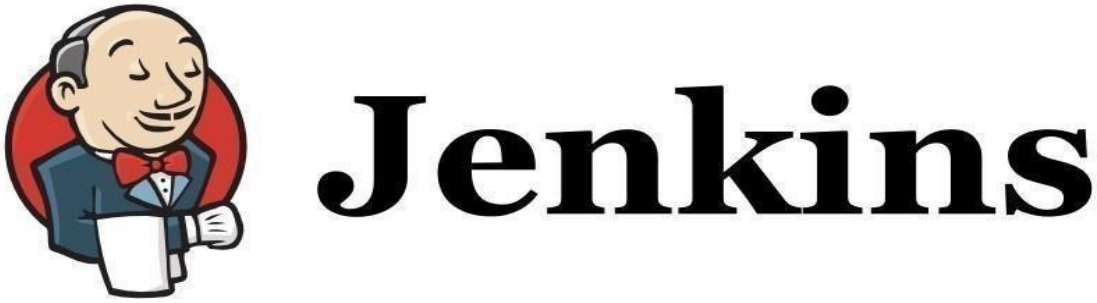
- GitHub offers various collaboration tools including wikis, project boards, and discussions, to facilitate communication and coordination among team members.

- GitHub facilitates code review and collaboration through pull requests (PRs). Developers can create pull requests to propose changes, discuss modifications, and request feedback from collaborators PRs provide a structured workflow for reviewing, commenting, and merging code changes into the main repository.

Git and GitHub form a powerful combination for version control, collaboration, and project management in software development. Together, they enable teams to work more efficiently collaborate effectively and deliver high-quality software products.

Jenkins

Jenkins is an open-source automation server widely used for continuous integration (CI) and continuous delivery (CD) pipelines in software development. It enable steams to automate various aspects of the software delivery process, including building, testing, and deploying applications, thereby increasing efficiency, reducing errors, and accelerating time-to-market.



- Jenkins automates the process of integrating code changes from multiple developers into a shared repository. It monitors version control systems (e.g., Git, Subversion) for changes and triggers build jobs automatically whenever new code is committed .This ensures that changes are tested and integrated early and often, reducing integration issues and improving code quality.

- Jenkins facilitates continuous delivery by automating the process of deploying applications to various environments, such as development, testing, staging, and production It orchestrates deployment pipelines, including building artifacts, running tests, and deploying to target environments, allowing teams to release software rapidly, reliably, and frequently.

- Jenkins is a versatile and powerful automation server that enables teams to implement CI/CD practices, automate repetitive tasks and stream line the software delivery process. Its flexibility, extensibility, and scalability make it a popular choice for organizations of all sizes and industries looking to adopt DevOps practices and accelerate their software delivery lifecycle.

Docker

Docker is a popular platform used for developing, shipping, and running applications in containers. Containers are lightweight, portable, and self-sufficient environments that encapsulate an application and its dependencies, allowing it to run reliably across different computing environments.

1) Containerization:

- Docker enables containerization, which involves packaging an application and its dependencies into a container image. Containers isolate the application from its host environment ensuring consistency and portability across different platforms and environments. This allows developers to build, ship, and run applications reliably, regardless of the underlying infrastructure.

2) Docker Engine:

- At the core of Docker is the Docker Engine a light weight runtime environment that runs containers on a host system. The Docker Engine includes the Docker daemon (docker) and the Docker command-line interface (CLI), which allows users to interact with containers and manage containerized applications.

3) Docker file:

- Docker uses Docker files, simple text files with a set of instructions, to define the configuration and components of a container image. Docker files specify the base image, dependencies, environment variables, and commands needed to build the container image. Developers can version control Docker files and use them to automate the container build process.

4) Orchestration and Management:

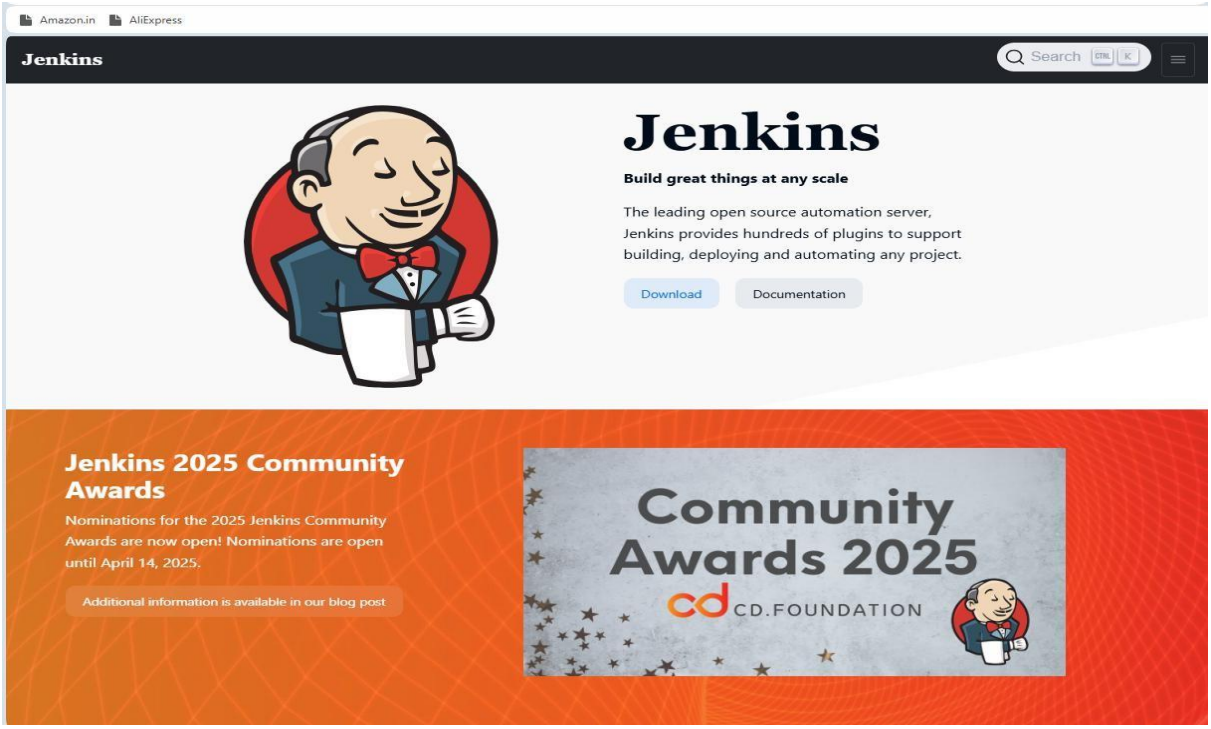
- Docker offers tools and services for orchestrating and managing containers at scale. Docker Swarm ,built into Docker Engine, allows users to deploy and manage container clusters across multiple hosts providing high availability and scalability for containerized applications. Additionally, Kubernetes, an open-source container orchestration platform, can be integrated with Docker for advanced container management and automation.

5) DevOps Integration:

- Docker plays a central role in DevOps practices, enabling teams to implement continuous integration (CI) and continuous delivery (CD) workflows. Containers provide a standardized packaging format for applications, facilitating automation, reproducibility, and collaboration across development, testing, and deployment pipelines.

Docker simplifies the process of building, shipping, and running applications in containers, providing developers with a powerful tool for building modern, cloud-native applications and adopting DevOps practices effectively. Its simplicity, portability, and ecosystem of tools make it a popular choice for organizations looking to modernize their application development and deployment workflows.

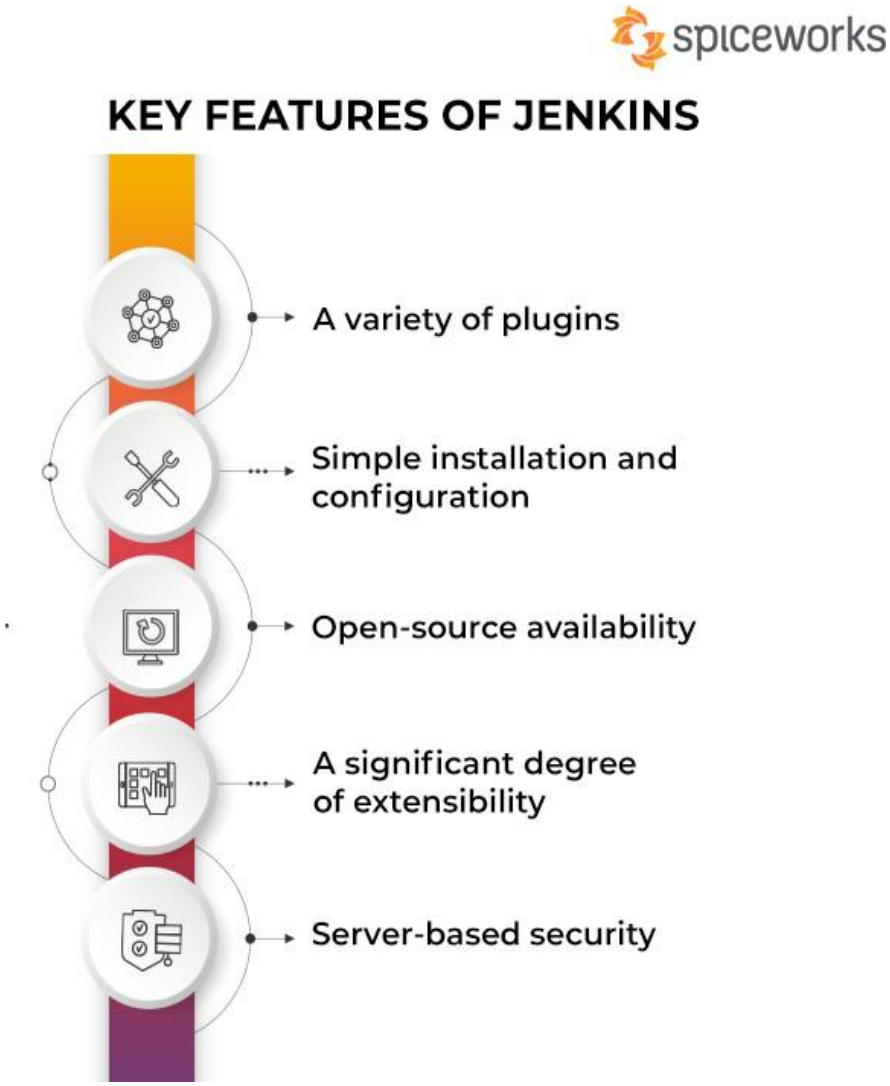
JENKINS

- Jenkins is an opensource **continuous integration /continuous delivery** and deployment (CI/CD) automation software
- **DevOps** tool written in the **Java** programming language. It is used to implement CI/CD workflows called pipelines.
- CI/CD pipelines automate testing and reporting on isolated changes in a larger codebase in real time. They also facilitate the integration of disparate branches of the code into a main branch.
- Pipelines rapidly detect defects in a codebase, build the software, automate testing of builds, prepare the codebase for deployment and delivery, and ultimately deploy code to containers and virtual machines
- 

Key Features of Jenkins

Jenkins is simple to set up and customize. Jenkins has many plugins that give it a lot of versatility. It delivers code instantaneously, generates a report after deployment, highlights errors in code or tests, and detects and resolves various issues in near real-time. It's also ideal for integration because it's all done automatically.

There is also a fantastic support community. These features in more detail:



Jenkins Applications

Jenkins is a popular open-source automation server that provides a robust platform for implementing continuous integration and continuous delivery (CI/CD) pipelines. Its flexibility and extensive plugin ecosystem allow it to fit into nearly any development workflow, supporting a wide range of use cases across software development and deployment. Here are some detailed applications of Jenkins:

1. **Continuous Integration (CI)**

Jenkins is primarily used to automate the process of continuous integration, where it builds and tests code every time a change is committed to a version control system. This helps developers detect issues early in the development cycle, improving code quality and reducing the time needed to validate and release new software updates.

- **Automated Builds:** Jenkins can compile and build code from various environments and languages.
- **Automated Testing:** It can run a suite of tests (unit, integration, system) on new code to ensure it doesn't break anything.

2. **Continuous Delivery (CD)**

Beyond continuous integration, Jenkins can automate steps in software delivery, making it easier to deploy and release new versions. It allows for the automation of the deployment process, making sure that you can release reliably at any time.

- **Automated Deployment**: Jenkins can automate the deployment of applications to various environments, including testing, staging, and production.
- **Rollbacks**: It can also automate the rollback of a deployment if the deployment fails, ensuring quick recovery from errors.

3. **Infrastructure as Code (IAC)**

Jenkins is used to implement Infrastructure as Code practices, which involve managing and provisioning infrastructure through code instead of through manual processes.

- Configuration Management: Jenkins can integrate with tools like Ansible, Chef, and Puppet to automate the configuration of servers.
- Server Provisioning: It can also use scripts or templates to create or update servers.

4. **Monitoring and Reporting**

Jenkins can be configured to monitor its own performance and to generate reports on various aspects of the development process.

- Build Monitoring: Jenkins can keep track of build success rates and notify developers of failures.
- Performance Trends: It can generate reports that track performance metrics over time, helping teams understand trends.

5. **DevOps and Multibranch Pipeline**

Jenkins supports DevOps practices by enabling teams to implement multibranch pipelines, where each branch of the version control system can have its own tailored CI/CD pipeline.

- Pipeline as Code: Jenkins Pipelines allow defining build, test, and deploy stages that are stored in a Jenkins file and versioned along with the code.
- Parallel Execution: Jenkins can execute jobs in parallel, reducing the time required for builds and tests.

6. **Containerization Support**

Jenkins has robust support for Docker and Kubernetes, allowing teams to use containers for builds, tests, and deployments.

- **Docker Integration**: Jenkins can build Docker images and push them to Docker registries.
- **Kubernetes Integration**: It can manage Kubernetes pods, enabling dynamic provisioning of agents for builds and tests.

7. **Third-Party Integration**

Jenkins' extensive plugin ecosystem allows it to integrate with virtually any tool used in software development, from version control systems like Git to issue tracking systems like JIRA, and artifact repositories like Artifactory.

8. **Security and Compliance**

Jenkins can help enforce security policies and compliance standards by automating security scans and compliance checks.

- **Static Code Analysis**: Plugins can scan source code for security vulnerabilities.

MAVEN

- Maven is a build tool that uses a **POM (project object model)** to help build processes through plugins. It's not unlike how MS Build helps with C#, or MAKE with C/C++, or even npm /Grunt and JavaScript.
- A POM is the core of a project's configuration in Maven. It's an XML file containing info about the project, configuration details, and default values for most projects.
- Maven helps developers maintain Java-based applications through projects that organize code files and build scripts to run compiler tools, version numbers for compiled code, and dependency management that lets one project reference a version of another project.
- Jenkins allows you to run Maven, and decide when to call a POM file, what condition to call, and what to do with the outcome. And since Jenkins can listen to different events, e.g svn commit, the current time is 12:00 AM, etc. Jenkins and Maven can become quite a powerful duo. For example, you can ask Jenkins to trigger a build or run through all JUnit tests whenever a new code is committed and then, if the unit tests are passed, deploy on a target machine.

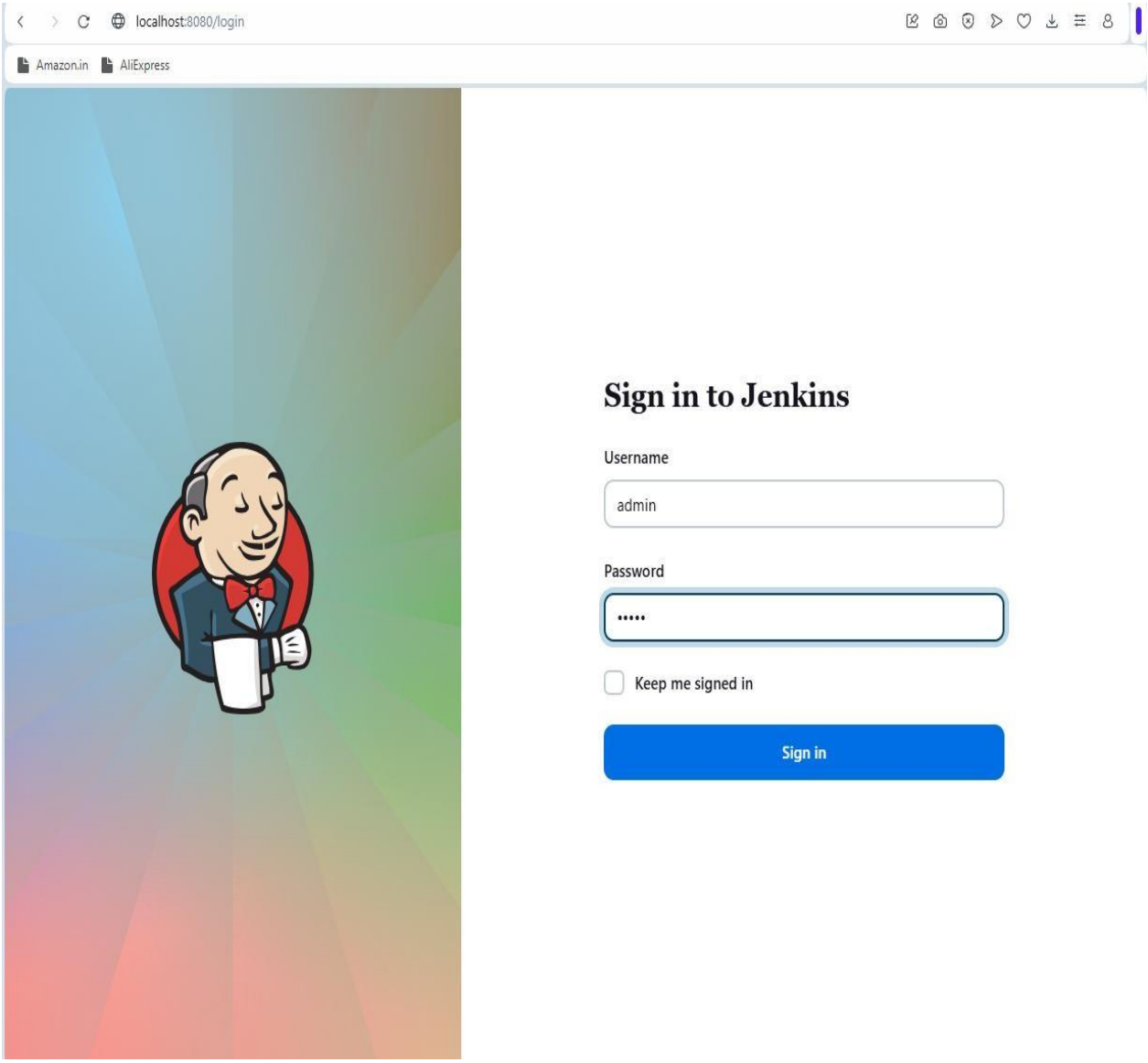
- Maven projects are then published to a **Maven Repository,** which is essentially like a web-based file share. There are index files in Maven repositories that list what projects and versions are stored in the repository, as well as metadata files that describe what each project is.

- Adding Maven to your Jenkins is just as simple as adding any other plugin. Simply install the Maven plugin in Jenkins. This will add a “Build” section to your projects where you can specify exactly what to execute.

Installing Jenkins and Maven

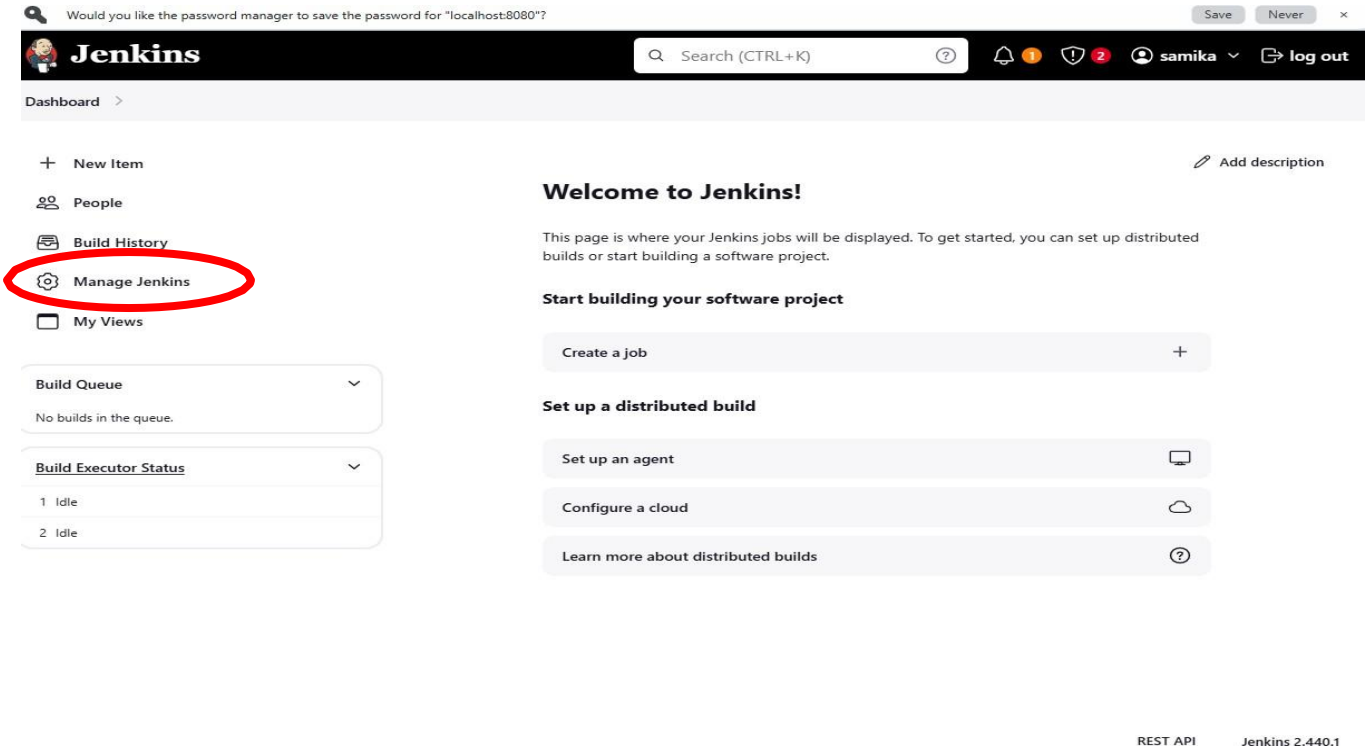
Step 1: Installing Jenkins

- Install Jenkins in our local desktop
- Sign into the Jenkins

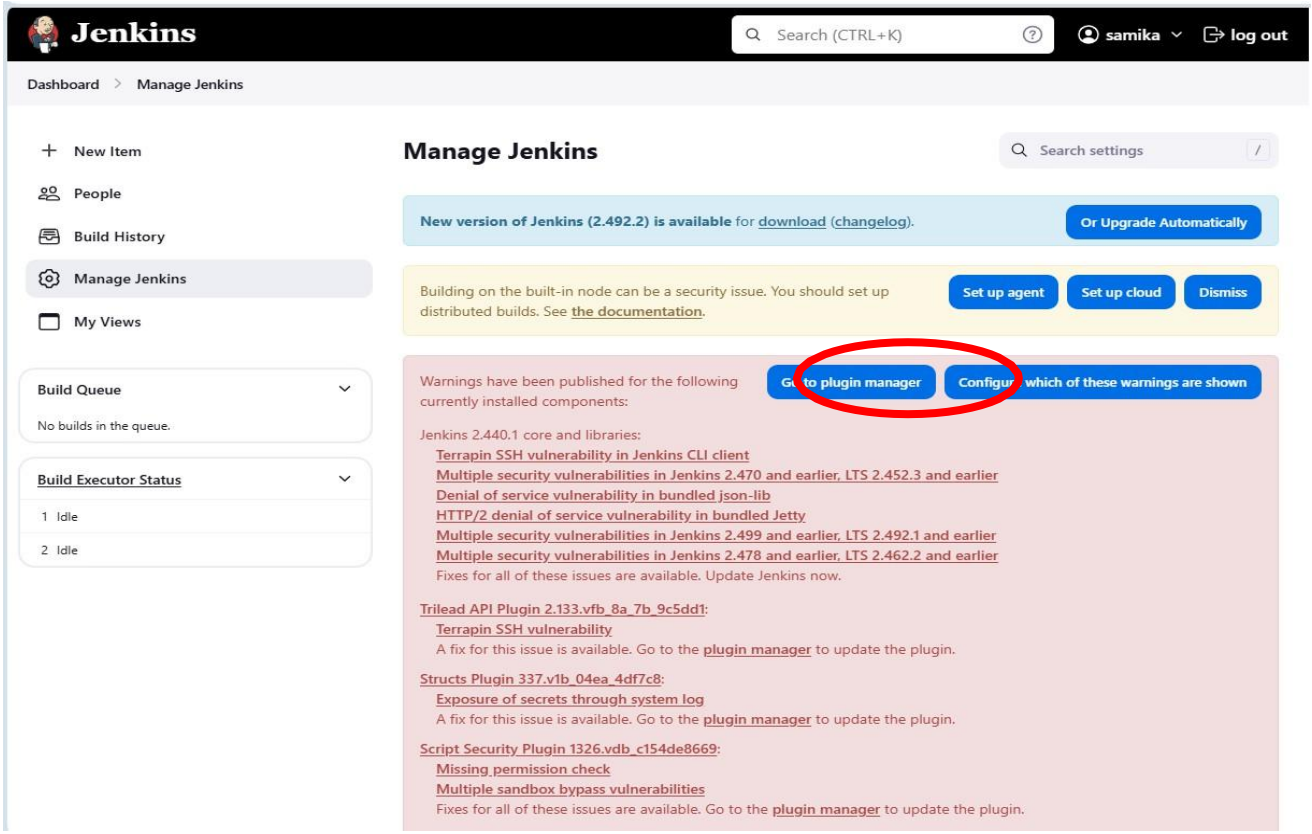


Step 2: Installing Maven

- Go to Manage Jenkins



- Go to the Plugin manager

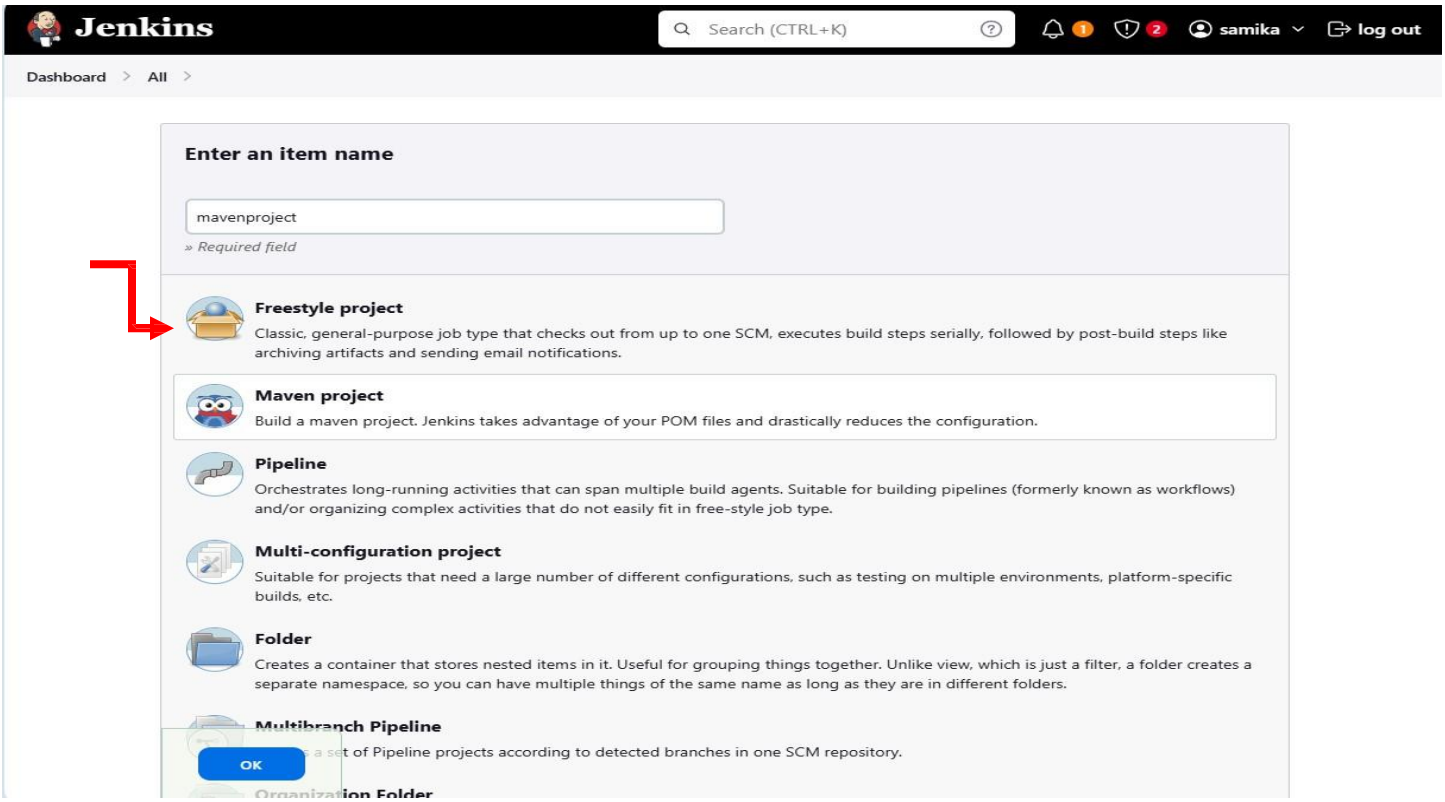


- Go to the Available Plugins
- Install the Maven by searching in search engine

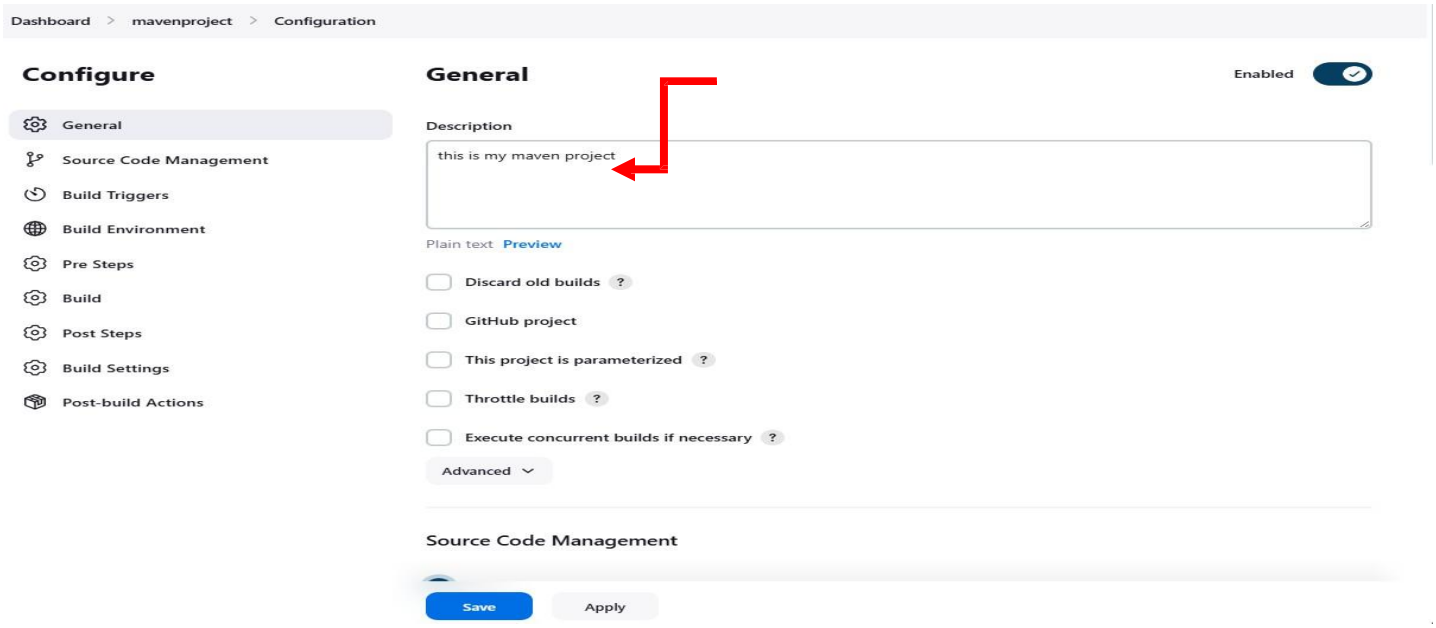
To check whether the Maven is installed, go to installed plugins and search Maven.

Building a Maven Project in Jenkins

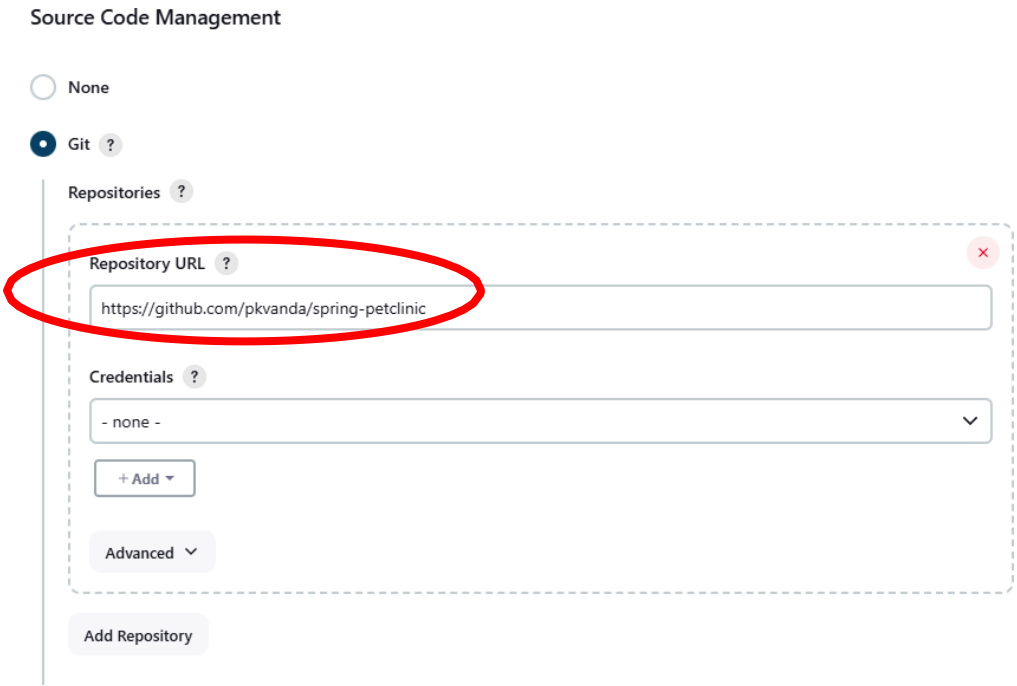
- Go to the Jenkins dashboard and click on new item.
- Now enter the item name and select freestyle project



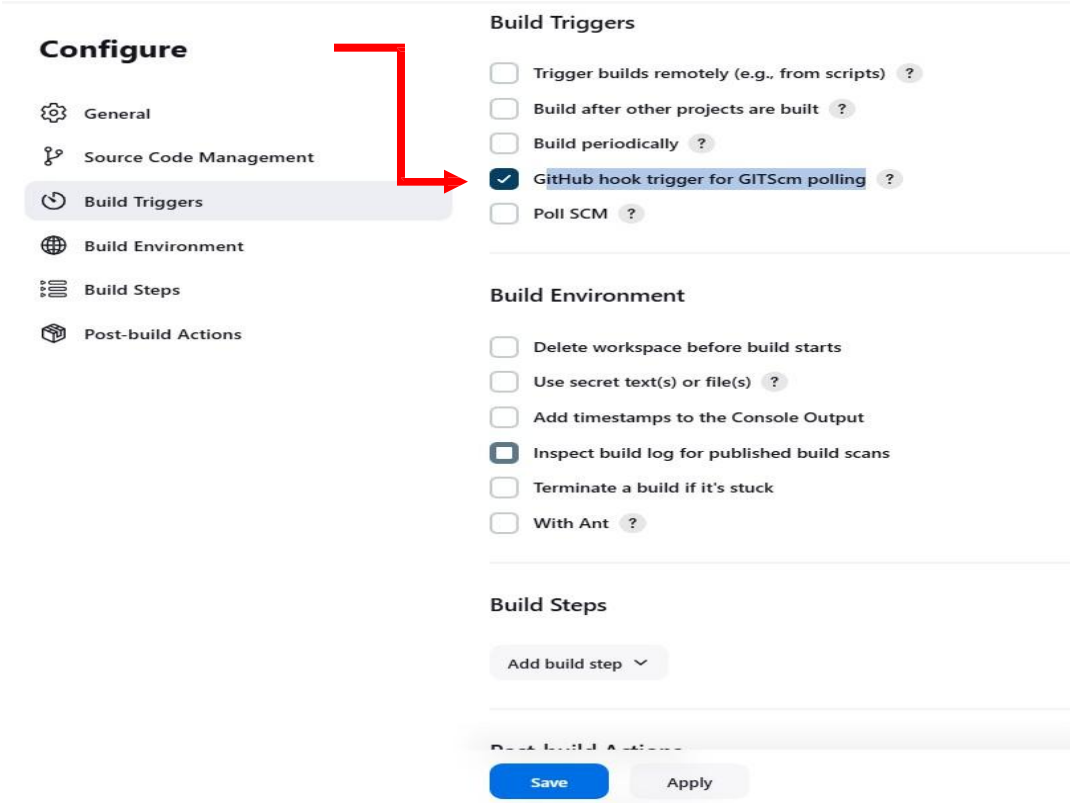
- Now write the description of project (optional)



- Now select the source code management as GIT, now paste the GIT repository URL

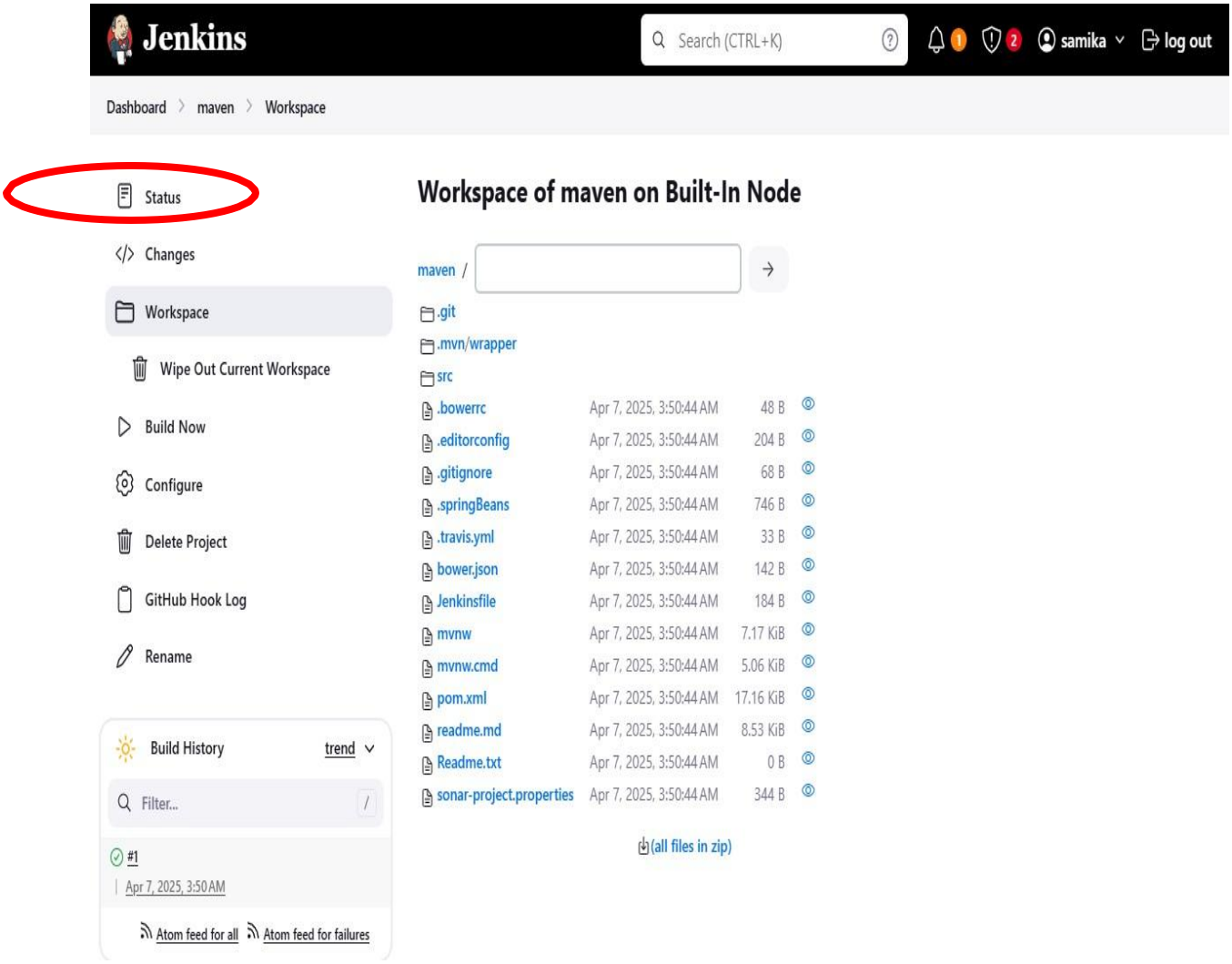


- Now select the build trigger (i.e GitHub hook trigger for GIT Scm polling)



- Now click on the Build now
- To check whether the project is build, go to the workspace then you will find the project output

Output


Jenkins
Search (CTRL+K)
samika
log out
Dashboard
maven
Workspace
Status
Changes
Workspace
Wipe Out Current Workspace
Build Now
Configure
Delete Project
GitHub Hook Log
Rename
Build History
trend
Filter...
#1
Apr 7, 2025, 3:50 AM
Atom feed for all
Atom feed for failures
Workspace of maven on Built-In Node
maven /
.git
.mvn/wrapper
src
.bowerrc Apr 7, 2025, 3:50:44 AM 48 B
.editorconfig Apr 7, 2025, 3:50:44 AM 204 B
.gitignore Apr 7, 2025, 3:50:44 AM 68 B
.springBeans Apr 7, 2025, 3:50:44 AM 746 B
.travis.yml Apr 7, 2025, 3:50:44 AM 33 B
bower.json Apr 7, 2025, 3:50:44 AM 142 B
Jenkinsfile Apr 7, 2025, 3:50:44 AM 184 B
mvnw Apr 7, 2025, 3:50:44 AM 7.17 KiB
mvnw.cmd Apr 7, 2025, 3:50:44 AM 5.06 KiB
pom.xml Apr 7, 2025, 3:50:44 AM 17.16 KiB
readme.md Apr 7, 2025, 3:50:44 AM 8.53 KiB
Readme.txt Apr 7, 2025, 3:50:44 AM 0 B
sonar-project.properties Apr 7, 2025, 3:50:44 AM 344 B
(all files in zip)

CONCLUSION

Hence we build a Maven Project in Jenkins & Using build triggers.

References

- https://www.simplilearn.com/tutorials/jenkins-tutorial/what-is-jenkins
- https://maven.apache.org/what-is-maven.html
- https://www.techtarget.com/searchitoperations/definition/GitHub