Automatic Deployment of Code Using Jenkins Pipeline

Introduction

Continuous Integration and Continuous Deployment (CI/CD) are essential practices in modern software development, enabling teams to deliver high-quality software efficiently. Jenkins, an open-source automation tool, plays a pivotal role in implementing CI/CD pipelines. This document provides a comprehensive overview of automating code deployment using Jenkins.

What is Jenkins?

Jenkins is a popular automation server that enables developers to build, test, and deploy applications. Its robust plugin ecosystem and support for pipeline scripting make it an ideal choice for automating deployment workflows.

Features of Jenkins

- Extensible through plugins.
- Support for pipelines as code.
- Integration with version control systems like Git, SVN, and more.
- Scalable and supports distributed builds.

Jenkins Pipeline Overview

A Jenkins pipeline is a suite of plugins supporting the implementation of CD pipelines in Jenkins. Pipelines are defined using a domain-specific language (DSL) called Jenkinsfile.

Steps to Automate Code Deployment Using Jenkins Pipeline:

1. Prerequisites

- Install Jenkins on your server. (could be same as target server or a different server).
- Install necessary plugins (e.g., Git Plugin, Pipeline Plugin, SSH Plugin, etc.).
- Set up version control (e.g., GitHub and Git).
- Configure a deployment environment (e.g., a staging/production).

2. Define the Jenkins Pipeline

The Pipeline can be seen inside the Jenkins portal on the server.

The Developer will just get the PAT token to push the code on GitHub. Everything else will be handled inside the Jenkins pipeline.

There are three individual pipelines – Front-end, Back-end and Admin., for a smooth execution of each part of the project.

3. Pipeline Stages Explanation:

Stage 1: Checkout Code and Pull

- Pull the latest code from the version control system.
- Ensures the pipeline uses the most recent changes.

Stage 2: Install

- Install the project dependencies and packages.

Stage 3: Build

- Compile and build the application to ensure it’s free of syntax errors.

Stage 4: Run/ Restart and Test the Application

- Execute unit, integration, or functional tests.
- Ensure the application is stable before deployment.

Stage 5: Deploy to the server

- Securely transfer built artifacts to the deployment server.
- Restart the application to apply changes.

4. Configure Jenkins

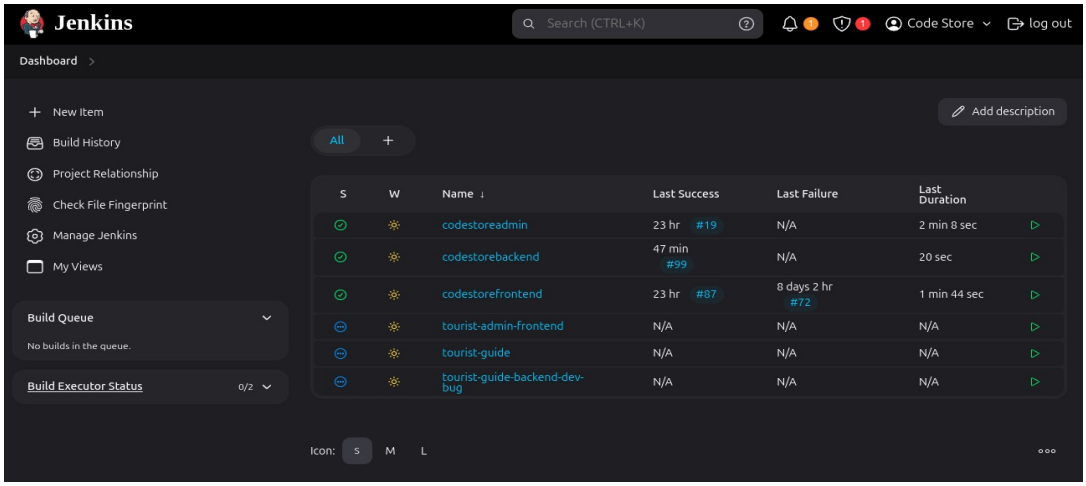
1. **Set up Credentials**:
    - Add an SSH key in Jenkins for secure access to the deployment server.
    - Add GitHub credentials in case of private repositories.
2. **Install Plugins**:
    - Install the necessary plugins for Git, SSH, and build tools.
3. **Create a Jenkins Job**:
    - Use the Pipeline project type.
    - Define the repository and branch for the pipeline.

Conclusion

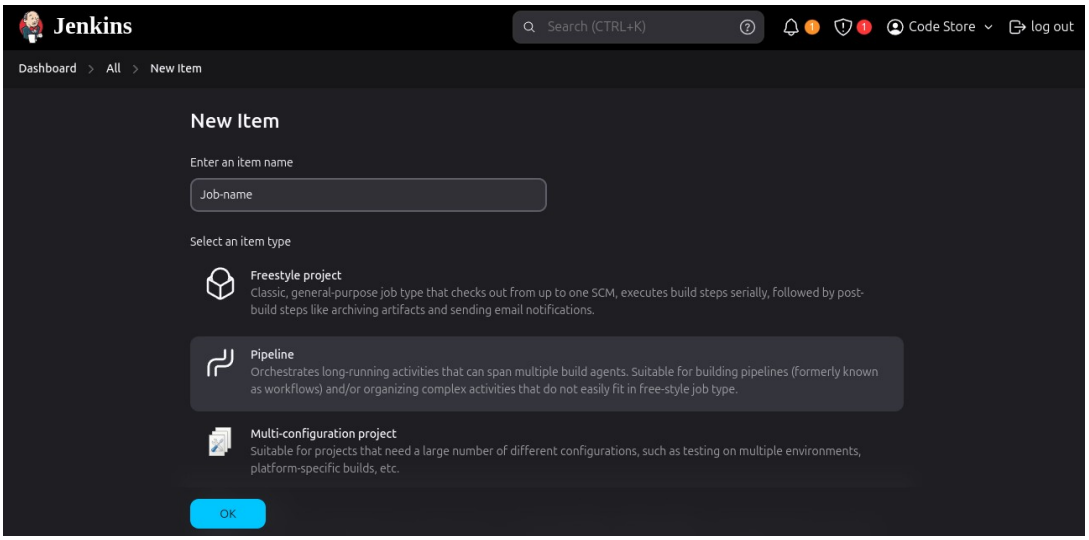
Automating code deployment using Jenkins pipelines streamlines the CI/CD process, reducing manual interventions and enhancing the reliability of releases. By defining robust pipelines and adhering to best practices, teams can achieve seamless deployments with minimal downtime.

How to setup the pipeline:

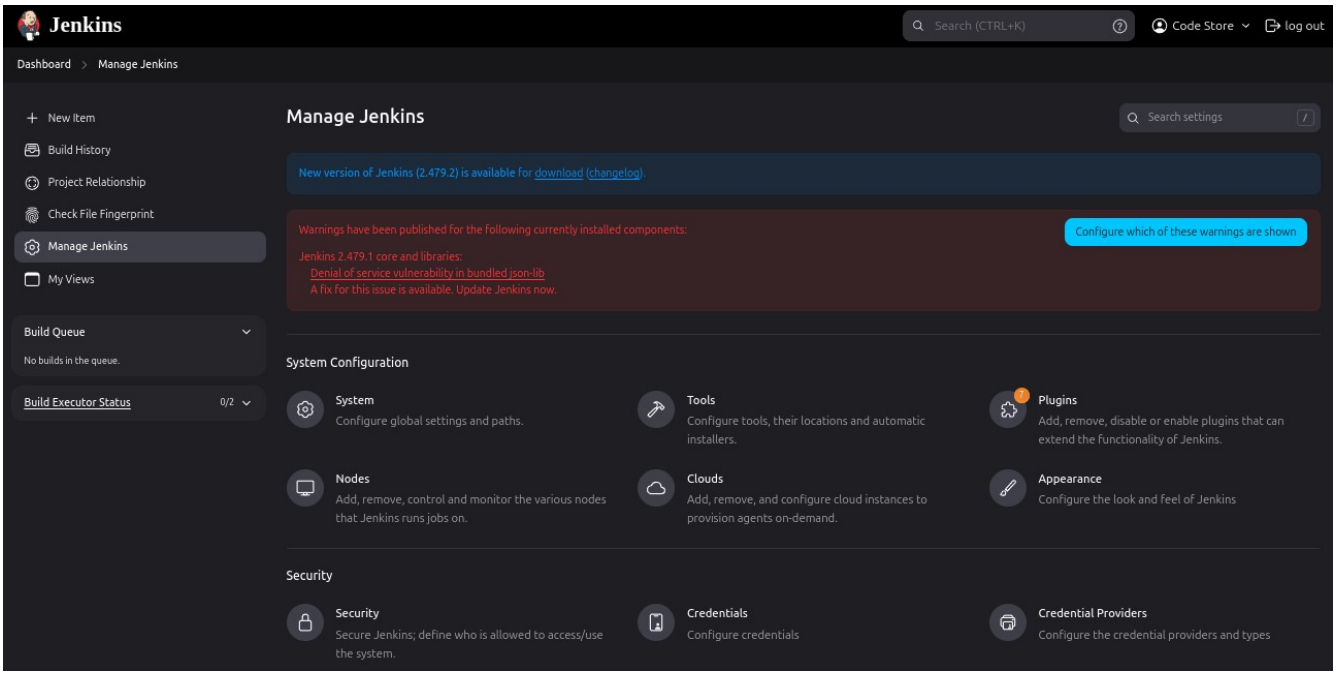
1. Install and run jenkins on server (Virtual Machine).
2. Run jenkins. On the jenkins dashboard click on **New Item** on left.



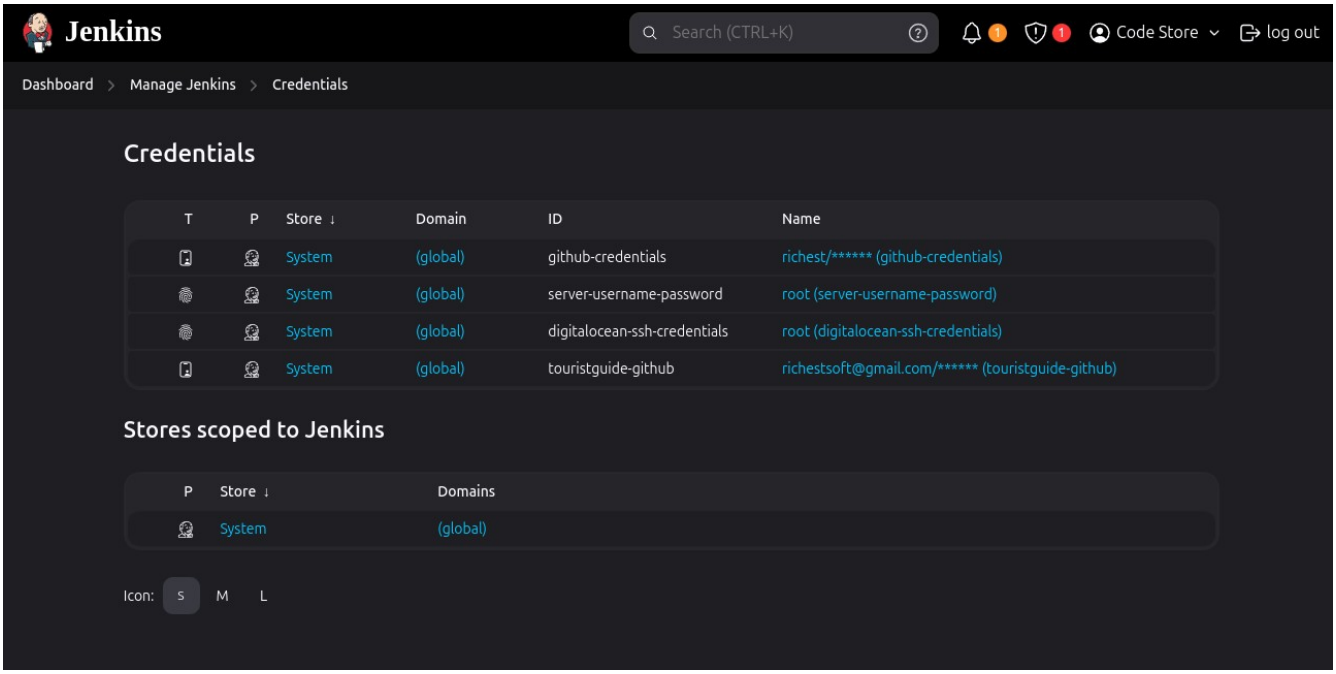
3. Create a new pipeline job.



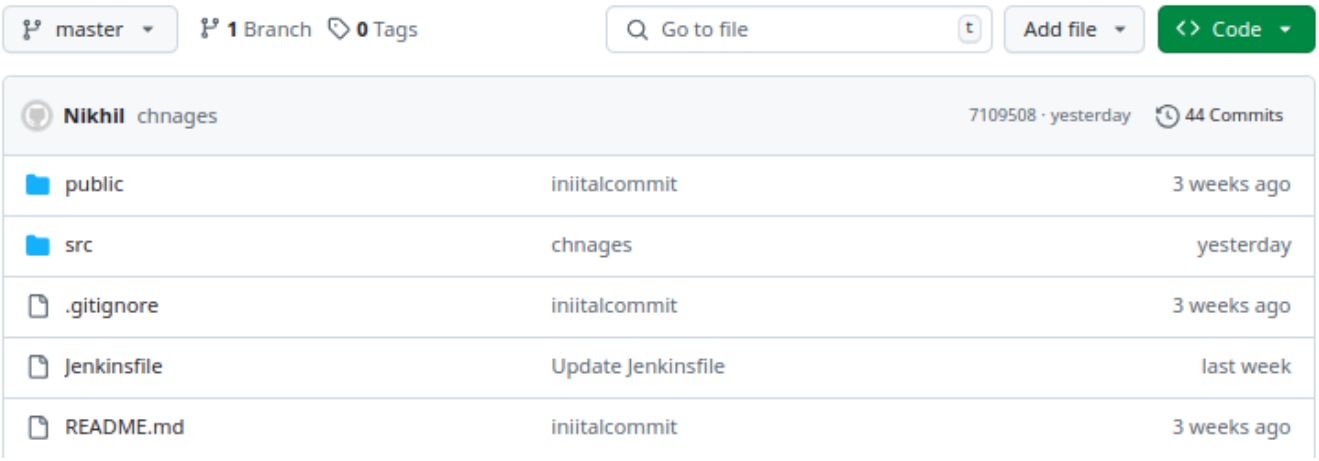
4. After this go to the dashboard and click on manage jenkins, then scroll down and search **credentials**.



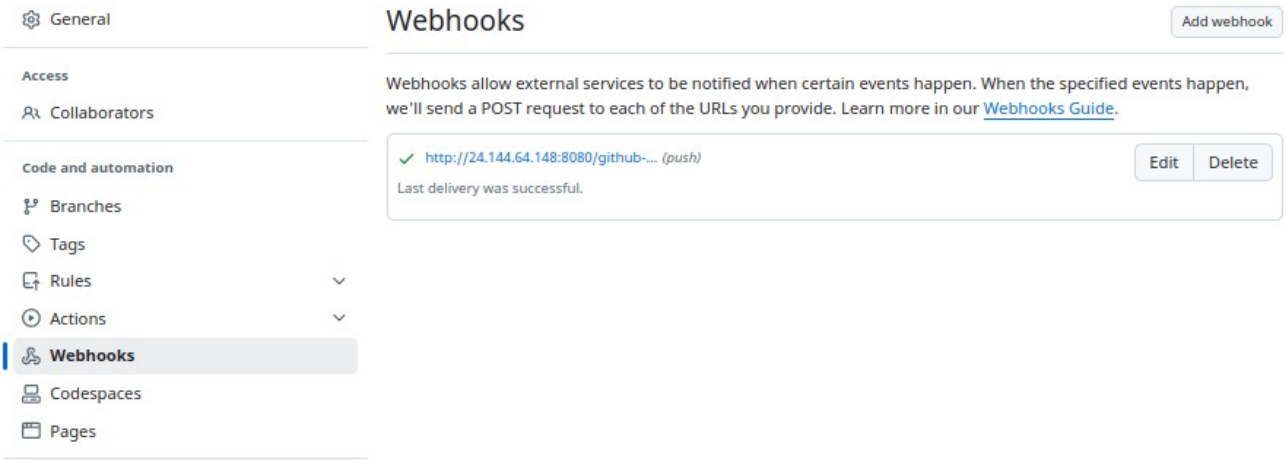
6. Click on it. Then search for **global** and click on it.
7. Click on **Add Credentials.** After that add the credentials that are required in the pipeline
   (github credentials, server credentials)



8. Then add a file named Jenkinsfile in the github repository in which the user will write the pipeline script.

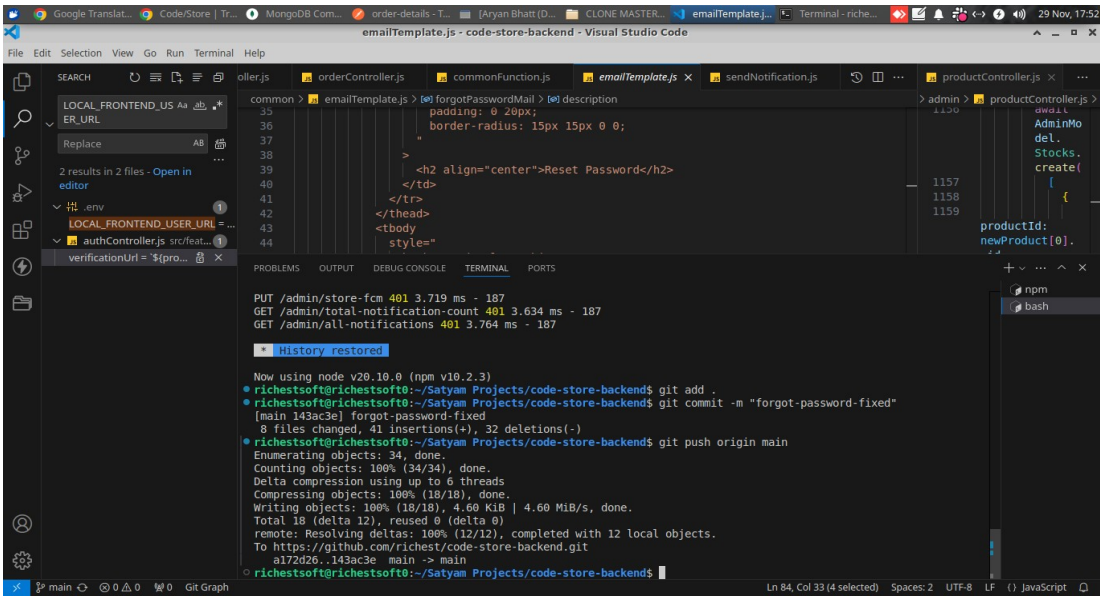


9. In the github repository settings, go to webhooks.
10. Setup webhooks that will send a push event to jenkins after that jenkins will trigger the pipeline and do the task.

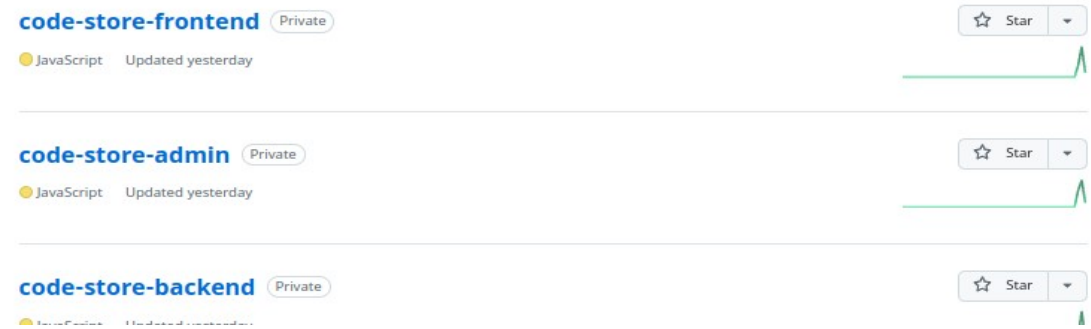


How the pipeline runs:

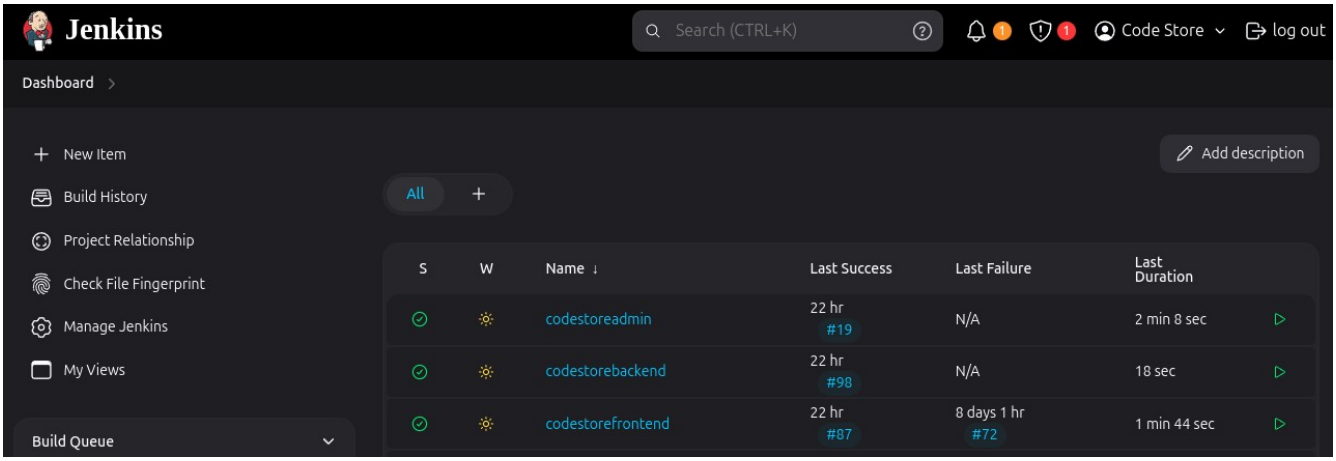
1. The developer push the code to GitHub repository.

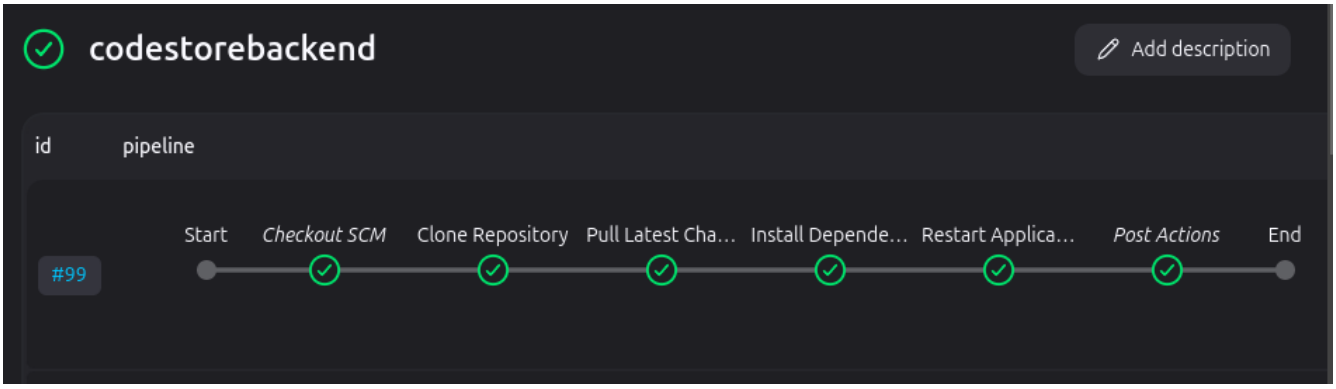


GitHub Repositories.



2. GitHub sends a push event webhook that triggers the jenkins pipeline.
3. The triggered pipeline will check for the pipeline script that will be in the pipeline configuration or inside the GitHub repository named as jenkinsfile. Inside the file pipeline is defined, under which job stages are defined.
4. The pipeline will execute all the stages one by one and deploy the code.





5. After getting success in the pipeline, the user can check if the project is deployed successfully or not.
6. If any errors, then the user can check the errors in the console output of the pipeline.