

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:

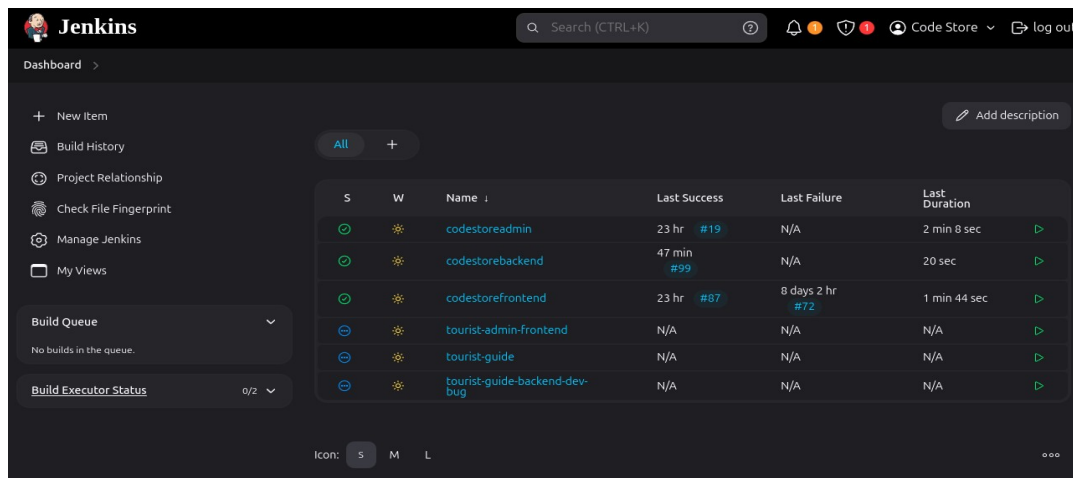
- Use the Pipeline project type.
- Define the repository and branch for the pipeline.

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:

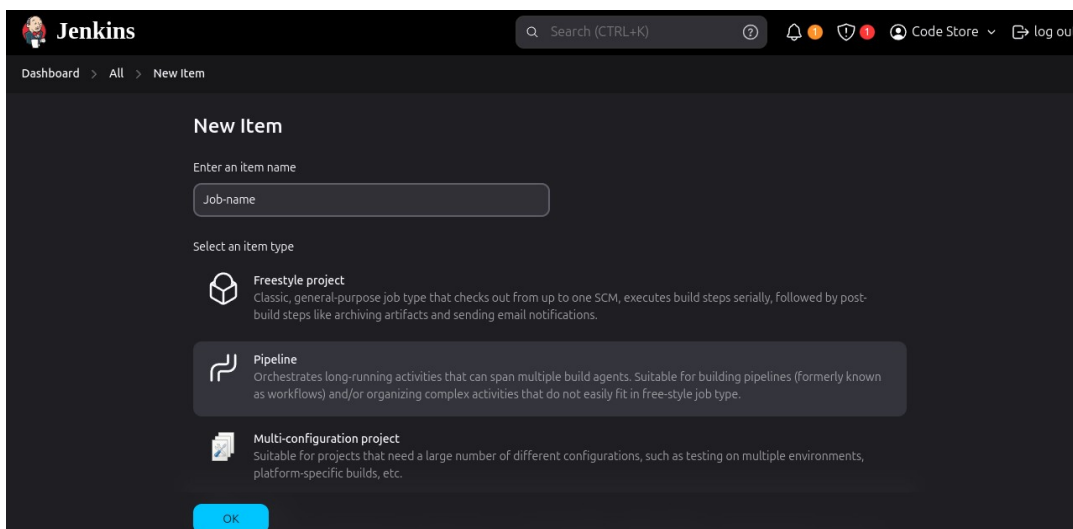
1. Install and run jenkins on server (Virtual Machine).
2. Run jenkins. On the jenkins dashboard click on **New Item** on left.



The screenshot shows the Jenkins Dashboard. On the left sidebar, there are links for 'New Item', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', and 'My Views'. Below these are 'Build Queue' (showing 'No builds in the queue') and 'Build Executor Status' (showing '0/2'). The main area displays a table of jobs. The table has columns for 'S' (status), 'W' (webhook), 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. The jobs listed are 'codestoreadmin', 'codestorebackend', 'codestorefrontend', 'tourist-admin-frontend', 'tourist-guide', and 'tourist-guide-backend-dev-bug'. At the bottom, there is a filter for 'Icon' with options 'S', 'M', and 'L'.

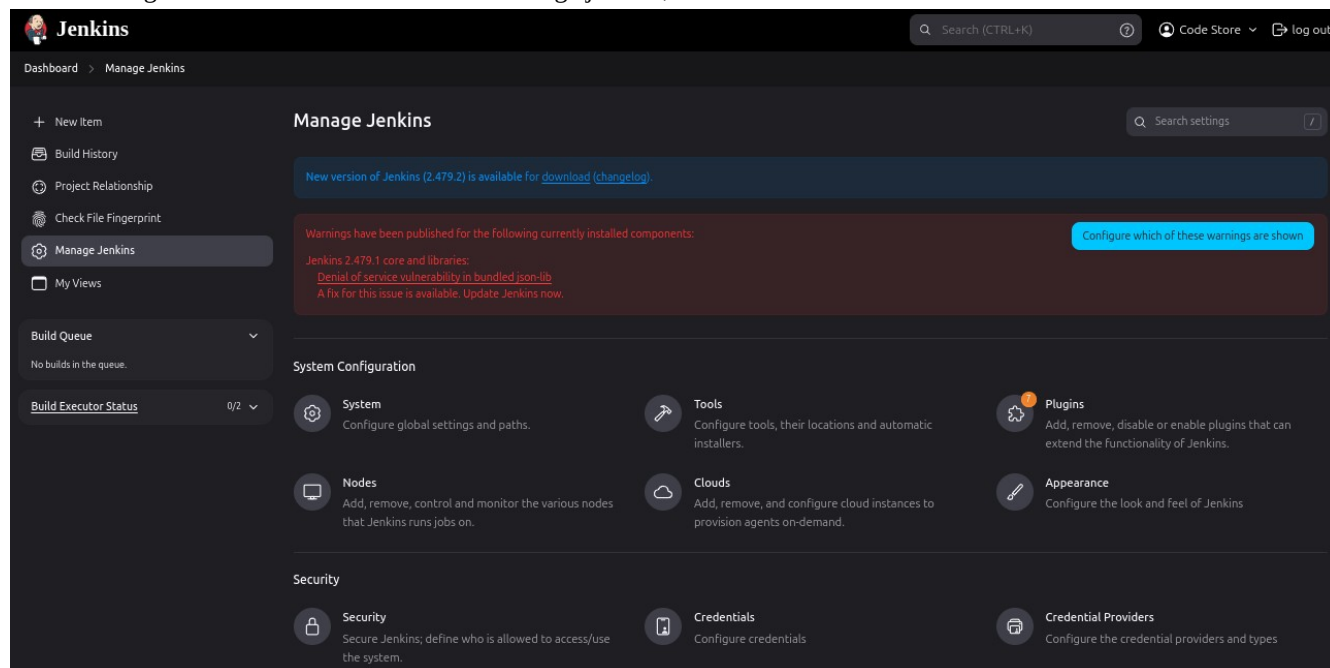
S	W	Name	Last Success	Last Failure	Last Duration
🟢	🔔	codestoreadmin	23 hr #19	N/A	2 min 8 sec
🟢	🔔	codestorebackend	47 min #99	N/A	20 sec
🟢	🔔	codestorefrontend	23 hr #87	8 days 2 hr #72	1 min 44 sec
🔴	🔔	tourist-admin-frontend	N/A	N/A	N/A
🔴	🔔	tourist-guide	N/A	N/A	N/A
🔴	🔔	tourist-guide-backend-dev-bug	N/A	N/A	N/A

3. Create a new pipeline job.



The screenshot shows the 'New Item' form in Jenkins. It has a header 'New Item' and a sub-header 'Enter an item name'. Below this is a text input field labeled 'Job-name'. Underneath is a section 'Select an item type' with three options: 'Freestyle project' (described as a classic, general-purpose job type), 'Pipeline' (described as orchestrating long-running activities), and 'Multi-configuration project' (described as suitable for projects needing different configurations). At the bottom is an 'OK' button.

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



Jenkins

Dashboard > Manage Jenkins

Search (CTRL+K)

Code Store log out

Search settings

New version of Jenkins (2.479.2) is available for [download](#) ([changelog](#)).

Warnings have been published for the following currently installed components:

Configure which of these warnings are shown

Jenkins 2.479.1 core and libraries:
Denial of service vulnerability in bundled json-lib
A fix for this issue is available. Update Jenkins now.

System Configuration

System: Configure global settings and paths.

Tools: Configure tools, their locations and automatic installers.

Plugins: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Nodes: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Clouds: Add, remove, and configure cloud instances to provision agents on-demand.

Appearance: Configure the look and feel of Jenkins.

Security

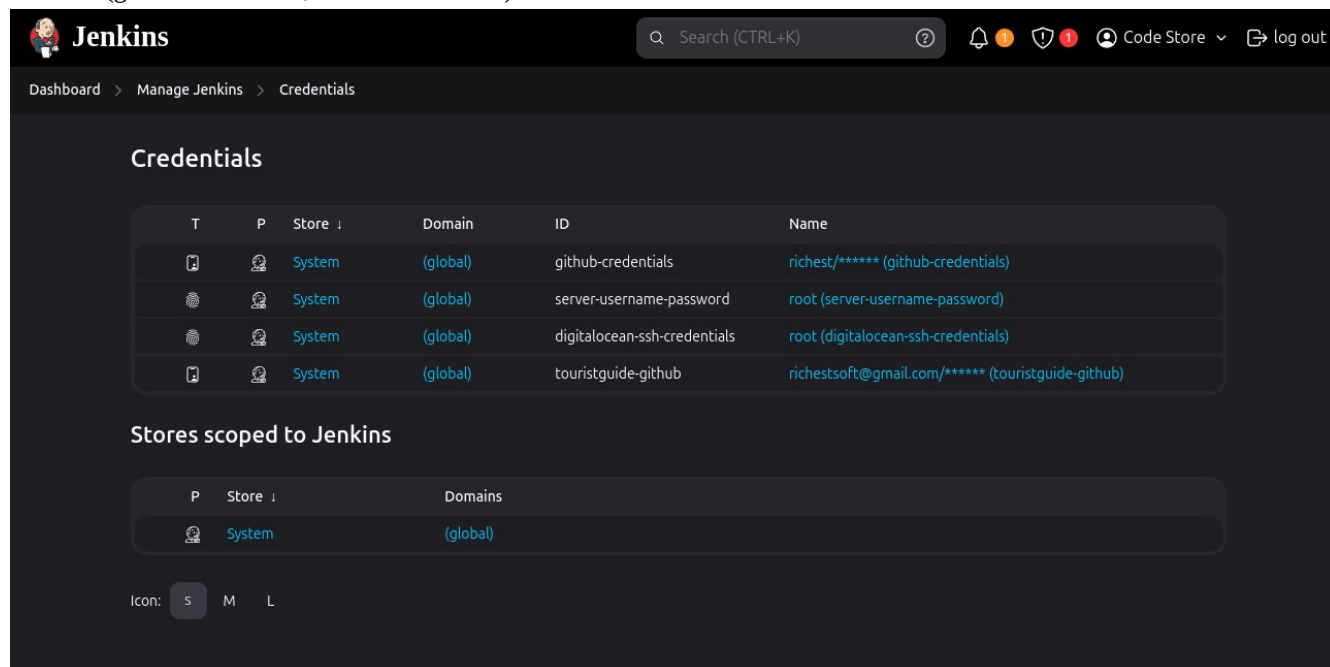
Security: Secure Jenkins; define who is allowed to access/use the system.

Credentials: Configure credentials.

Credential Providers: Configure the credential providers and types.

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)



Jenkins

Search (CTRL+K)

Code Store log out

Dashboard > Manage Jenkins > Credentials

Credentials

T	P	Store	Domain	ID	Name
		System	(global)	github-credentials	richest/***** (github-credentials)
		System	(global)	server-username-password	root (server-username-password)
		System	(global)	digitalocean-ssh-credentials	root (digitalocean-ssh-credentials)
		System	(global)	touristguide-github	richestsoft@gmail.com/***** (touristguide-github)

Stores scoped to Jenkins

P	Store	Domains
	System	(global)

Icon: **S** M L

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

master	1 Branch	0 Tags	Go to file	Add file	Code
Nikhil chnages 7109508 · yesterday 44 Commits					
public	iniitalcommit	3 weeks ago			
src	chnages	yesterday			
.gitignore	iniitalcommit	3 weeks ago			
Jenkinsfile	Update Jenkinsfile	last week			
README.md	iniitalcommit	3 weeks ago			

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.

General

Access

Collaborators

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Codespaces

Pages

Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

✓ http://24.144.64.148:8080/github-... (push)

Last delivery was successful.

EditDelete

How the pipeline runs:

1. The developer push the code to GitHub repository.

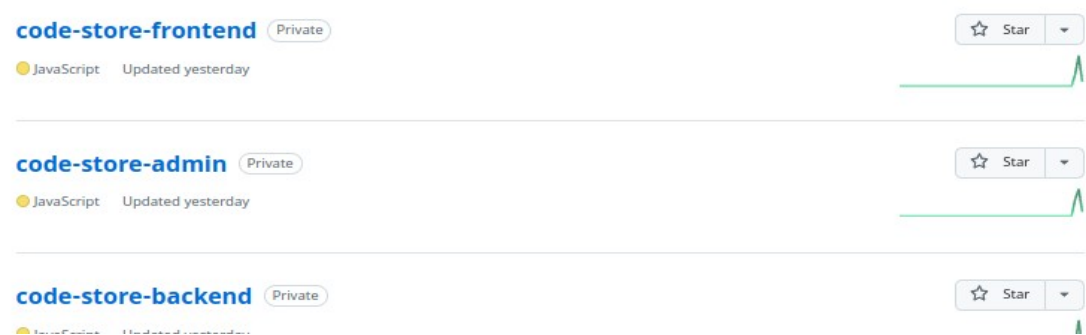
```
common > emailTemplate.js > forgotPasswordMail > description
35
36 padding: 0 20px;
37 border-radius: 15px 15px 0 0;
38
39 <h2 align="center">Reset Password</h2>
40
41 </tr>
42 </thead>
43 <tbody>
44 style="

PUT /admin/store-fcm 401 3.719 ms - 187
GET /admin/total-notification-count 401 3.634 ms - 187
GET /admin/all-notifications 401 3.764 ms - 187

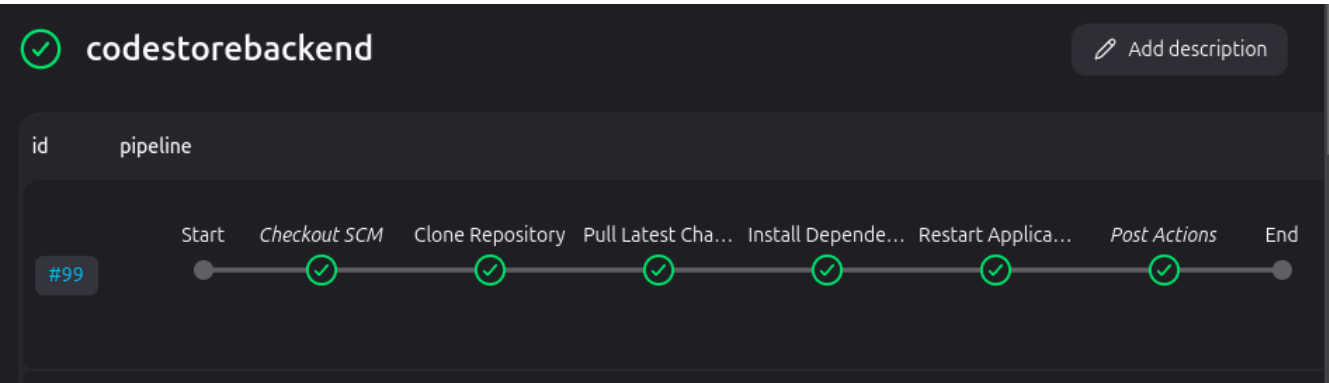
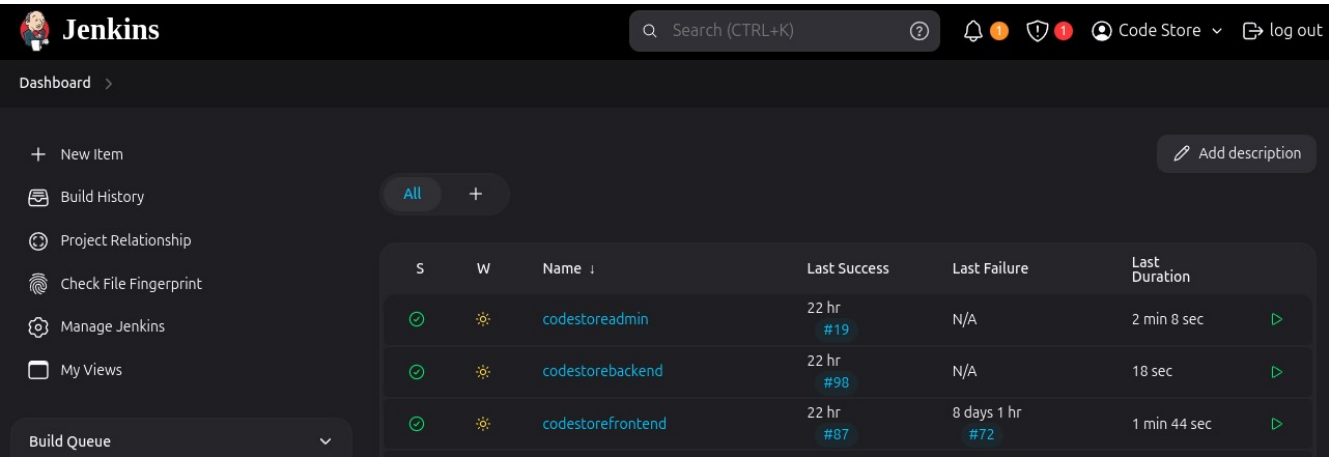
History restored

Now using node v20.10.0 (npm v10.2.3)
richestsoft@richestsoft0:~/Satyam Projects/code-store-backend$ git add .
richestsoft@richestsoft0:~/Satyam Projects/code-store-backend$ git commit -m "forgot-password-fixed"
[main 143ac3e] forgot-password-fixed
8 files changed, 41 insertions(+), 32 deletions(-)
richestsoft@richestsoft0:~/Satyam Projects/code-store-backend$ git push origin main
Enumerating objects: 34, done.
Counting objects: 100% (34/34), done.
Delta compression using up to 6 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (18/18), 4.60 KiB | 4.60 MiB/s, done.
Total 18 (delta 12), reused 0 (delta 0)
remote: Resolving deltas: 100% (12/12), completed with 12 local objects.
To https://github.com/richest/code-store-backend.git
al72d26..143ac3e main -> main
richestsoft@richestsoft0:~/Satyam Projects/code-store-backend$
```

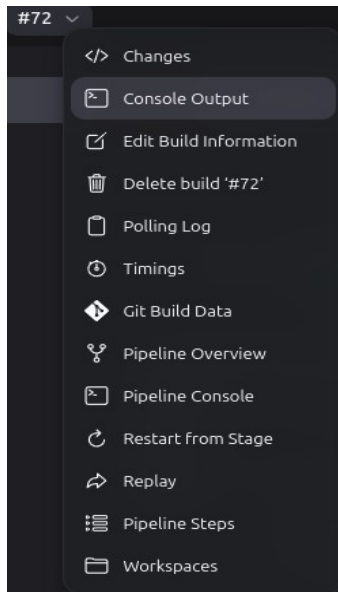
GitHub Repositories.



- 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.

A screenshot of the Jenkins console output for build #72. The title bar shows a red 'x' icon and the text 'Console Output'. On the right side of the title bar are three buttons: 'Download', 'Copy', and 'View as plain text'. The main area contains the following text:

```
Started by GitHub push by richest
Obtained Jenkinsfile from git https://github.com/richest/code-store-frontend.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/codestorefrontend
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
The recommended git tool is: git
using credential github-credentials
Cloning the remote Git repository
Cloning repository https://github.com/richest/code-store-frontend.git
> git init /var/lib/jenkins/workspace/codestorefrontend # timeout=10
Fetching upstream changes from https://github.com/richest/code-store-frontend.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
```