



Automated CI/CD Pipeline Implementation Using Jenkins on RHEL

Description:

This project involved setting up and configuring a full Jenkins-based Continuous Integration/Continuous Deployment (CI/CD) pipeline in a Red Hat Enterprise Linux (RHEL) environment. Jenkins, an open-source automation server, was used to automate the software development lifecycle by enabling efficient integration and delivery of code changes.

I started by installing and configuring Jenkins on a RHEL server and then proceeded to explore the Jenkins dashboard thoroughly. I created multiple freestyle jobs for different stages of the build and deployment process. Each job was configured with appropriate build triggers, such as GitHub webhooks and periodic polling, to ensure that code changes pushed to the repository were detected automatically.

Build steps were added to pull code from a GitHub repository, compile it, and perform automated tasks. Post-build actions were configured to notify stakeholders via email and archive artifacts. I created and organized views within Jenkins to manage and monitor related jobs more effectively, allowing for better project structure and visual clarity.

I installed several essential plugins such as Git, Pipeline, and Email Extension to extend Jenkins capabilities. Jenkins' console output logs were reviewed regularly to trace job execution, debug errors, and confirm successful builds. Additionally, I explored the use of credentials management and secure parameter passing, which is critical in real-world CI/CD use cases.

Throughout the project, I became familiar with job chaining, build dependencies, Jenkins' security model, and how to manage plugin updates. This project helped me gain a deep, practical understanding of how Jenkins supports continuous integration and delivery workflows, allowing development teams to deliver software faster and more reliably.



- **Check Jenkins status**

```
thisara@localhost:~  
File Edit View Search Terminal Help  
● jenkins.service - Jenkins Continuous Integration Server  
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; disabled; vendor preset: disabled)  
  Active: active (running) since Thu 2025-07-10 10:26:21 PDT; 14s ago  
    Main PID: 8038 (java)  
      Tasks: 65 (limit: 48845)  
     Memory: 621.9M  
    CGroup: /system.slice/jenkins.service  
            └─8038 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=
```

- **Creating a project**

New Item

Enter an item name

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

- **Create build steps**

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

≡

Execute shell ?

×

Command

[See the list of available environment variables](#)

```
echo "hello world"
```



THISARA KANDAGE

UNDERGRADUATE - SLIIT

[E-mail](#) [LinkedIn](#) [GitHub](#) [Website](#)

- **Build is done**

Builds ⋮ 🔗

Today

✓ #1 10:34 AM ▼

- **Pull a GitHub repo**

Source Code Management

Connect and manage your code repository to automatically pull the latest code for your builds.

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://github.com/

❗ Please enter Git repository.

- **Again, build the project**

Builds ⋮ 🔗

/

Today

✓ #2 10:43 AM ▼

✓ #1 10:34 AM ▼



- **Console output**

```
First time build. Skipping changelog.  
[Jobtest1] $ /bin/sh -xe /tmp/jenkins18104106786377823949.sh  
+ echo 'hello world'  
hello world  
Finished: SUCCESS
```

- **If need to build happen again and again in fix interval of time.**
- **For that create another job**

New Item

Enter an item name

Jobtest2

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Source Code Management

Connect and manage your code repository to automatically pull the latest code for your builds.

☐ None

☒ Git ?

Repositories ?

Repository URL ?

Please enter Git repository.



• Add Build triggers

Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

- ☐ Trigger builds remotely (e.g., from scripts) ?
 - ☐ Build after other projects are built ?
 - ☐ Build periodically ?
 - ☐ GitHub hook trigger for GITScm polling ?
 - ☒ Poll SCM ?
- Schedule ?

* * * * *

(5 starts for build for in every single minute)

• Install required plugins

Q build pipeline

Install Name ↓



Build Pipeline 2.0.2

User Interface

Build Tools

Other Post-Build Actions

This plugin renders upstream and downstream connected jobs that typically form a build pipeline. In addition, it offers the ability to define manual triggers for jobs that require intervention prior to execution, e.g. an approval process outside of Jenkins.



• Create new view for jobs

New view

Name

Type

☐

Build Pipeline View

Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.

☐

List View

Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

☐

My View

This view automatically displays all the jobs that the current user has an access to.

Create

Upstream / downstream config

Select Initial Job ?

• Add post-build actions for job 1

Post-build Actions

Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.

≡ Build other projects ?



Projects to build

☒

Trigger only if build is stable

☐

Trigger even if the build is unstable

☐

Trigger even if the build fails



• Add build triggers for job 2

Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

- ☐ Trigger builds remotely (e.g., from scripts) ?
- ☒ Build after other projects are built ?

Projects to watch

Jobtest1,

- ☒ Trigger only if build is stable
- ☐ Trigger even if the build is unstable
- ☐ Trigger even if the build fails
- ☐ Always trigger, even if the build is aborted

• Build pipeline view

The screenshot shows the Jenkins interface for a pipeline named 'pipeline#1'. The 'Build Pipeline' header is visible. Below it, a toolbar contains icons for Run, History, Configure, Add Step, Delete, and Manage. The main area displays a list of pipeline builds. The first build, labeled '#1 job1', is highlighted in green, indicating it is the current build. The build details for this job show it was started on 'Sat 5, 2022 4:39:34 AM' and took '0:01 sec' to complete. The user 'admin' is listed as the trigger.

The screenshot shows the Jenkins interface for a pipeline named 'pipeline#1'. The 'Build Pipeline' header is visible. Below it, a toolbar contains icons for Run, History, Configure, Add Step, Delete, and Manage. The main area displays a list of pipeline builds. The third build, labeled '#3 job1', is highlighted in green, indicating it is the current build. The build details for this job show it was started on 'Sat 05, 2022 12:52:36 AM' and took '0:12 sec' to complete. The user 'admin' is listed as the trigger.



Summary– What I Learned

- Installed and configured Jenkins on a RHEL-based system.
- Created and managed multiple freestyle jobs for automation tasks.
- Integrated GitHub repositories for continuous code fetching.
- Configured build triggers (poll SCM, webhook triggers).
- Defined and executed build steps and post-build actions.
- Installed and managed Jenkins plugins for enhanced functionality.
- Reviewed Jenkins console output logs for debugging and verification.
- Managed views for better job organization and monitoring.
- Gained experience with Jenkins dashboard, job configuration, and credentials handling.
- Understood key concepts of CI/CD pipelines and how Jenkins facilitates automation.