



Hands-On Jenkins Implementation for Real-World CI/CD Automation

Proud to showcase my practical journey working with Jenkins — a powerful CI/CD automation tool — where I executed and tested real-time tasks like installation, job configuration, Git integration, pipelines, and master-agent setups to streamline the software development lifecycle.

Introduction to Jenkins & CI/CD

What is Jenkins?

Jenkins is an **open-source automation server** that helps automate the building, testing, and deploying of applications. It supports Continuous Integration and Continuous Deployment (CI/CD) and is a widely adopted tool in the DevOps ecosystem.

What is CI/CD?

- **CI (Continuous Integration)**: Developers merge code into a shared repository several times a day. Jenkins builds and tests code automatically.
- **CD (Continuous Delivery/Deployment)**: After passing tests, the code is automatically deployed to production or staging environments.

Why Jenkins?

Jenkins is one of the most popular automation tools in the DevOps ecosystem due to its flexibility, extensibility, and strong community support. Here's why it stands out:

- **Open-Source and Free** – Available to everyone with continuous updates and community contributions
- **Platform-Independent** – Works seamlessly across Windows, Linux, and macOS
- **Extensive Plugin Ecosystem** – Over 1800 plugins available to integrate with tools like Git, Maven, Docker, Selenium, AWS, Kubernetes, and more
- **Large Community Support** – A vast community ensures better support, updates, and shared knowledge
- **Easy Integration** – Connects effortlessly with version control systems, build tools, testing frameworks, and deployment tools
- **Distributed Builds** – Master-agent architecture enables parallel execution of jobs across multiple nodes
- **User-Friendly & Flexible** – Provides both a simple web interface for configuration and powerful scripting via pipelines (Groovy)



How Jenkins Works (Behind the Scenes)

1. **Developers push code** to a version control system like **GitHub**
2. **Jenkins detects the change** (via Poll SCM or Webhook)
3. It **pulls the code** and sends it to a **build tool** (like Maven)
4. Once the code is built, Jenkins can:
 - o **Run automated tests** (e.g., with Selenium)
 - o **Deploy to test or production servers**
 - o **Archive artifacts** or send notifications

This whole process is defined as:

- **Freestyle Jobs** (GUI-based task automation)
- **Pipelines** (script-based, using Groovy)

Example Workflow:

GitHub (Code Commit) → Jenkins → Build → Test → Deploy → Notify

Jenkins acts like a **bridge** between code and deployment, helping teams deliver faster and more reliably.

Jenkins Installation on Linux (EC2)

1. Update packages

```
yum update -y
```

2. Add Jenkins repo

```
wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
```

3. Import key

```
rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
```

4. Upgrade system

```
yum upgrade -y
```

5. Install Java

```
yum install java-17-amazon-corretto -y
```

6. Install Jenkins

```
yum install jenkins -y
```



7. Enable & Start Jenkins

```
systemctl enable jenkins
```

```
systemctl start jenkins
```

8. Check Jenkins status

```
systemctl status jenkins
```

```
root@ip-172-31-44-192:~# systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
     Active: active (running) since Tue 2025-08-05 11:09:26 UTC; 7min ago
       Main PID: 20108 (java)
          Tasks: 38 (limit: 1124)
        Memory: 295.6M (peak: 321.0M)
         CPU: 18.850s
        CGroup: /system.slice/jenkins.service
                  └─20108 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Aug 05 11:09:18 ip-172-31-44-192 jenkins[20108]: b3ela4e24404b5d89892c6824753685
Aug 05 11:09:18 ip-172-31-44-192 jenkins[20108]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Aug 05 11:09:18 ip-172-31-44-192 jenkins[20108]: ****
Aug 05 11:09:26 ip-172-31-44-192 jenkins[20108]: 2025-08-05 11:09:26.72740000 [id=30]      INFO      jenkins.InitReactorRunner$1#onAttained: Completed initialization
Aug 05 11:09:26 ip-172-31-44-192 jenkins[20108]: 2025-08-05 11:09:26.75640000 [id=23]      INFO      hudson.lifecycle.Lifecycle$1#onReady: Jenkins is fully up and running
Aug 05 11:09:26 ip-172-31-44-192 jenkins[20108]: system@[]: Started jenkins.service Jenkins Continuous Integration Server.
Aug 05 11:09:27 ip-172-31-44-192 jenkins[20108]: 2025-08-05 11:09:27.75040000 [id=47]      INFO      hudson.util.DownloadService$Downloadable#load: Obtained the updated data file for hudson.t
Aug 05 11:09:27 ip-172-31-44-192 jenkins[20108]: 2025-08-05 11:09:27.750+0000 [id=47]      INFO      hudson.util.Retriger#start: Performed the action check updates server successfully in
lines 1-20/20 (END)
```

Screenshot 1: Jenkins service status

Steps to Access and Login to Jenkins via Web Interface

Step 1: Open Jenkins URL in Browser

- Action:** Open your browser and enter the following URL:
- <http://<your-public-ip>:8080>

Example:

<http://13.234.67.120:8080>

- Jenkins welcome page should load.



Screenshot 2: Jenkins Welcome Page



Step 2: Unlock Jenkins

- **Action:** Copy the password from the file:
- `cat /var/lib/jenkins/secrets/initialAdminPassword`
- Paste the password into the unlock field on the Jenkins page and click **Continue**.

```
root@ip-172-31-44-192:~# id jenkins
uid=111(jenkins) gid=113(jenkins) groups=113(jenkins)
root@ip-172-31-44-192:~# cat /var/lib/jenkins/secrets/initialAdminPassword
b3e1a4ea24404b5d89892c6824753685
[...]
```

Screenshot 3: Unlock Jenkins Password Terminal

The screenshot shows the Jenkins 'Getting Started' screen with the title 'Unlock Jenkins'. It contains instructions: 'To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:' followed by the path `/var/lib/jenkins/secrets/initialAdminPassword`. Below this, it says 'Please copy the password from either location and paste it below.' There is a text input field labeled 'Administrator password' containing several dots ('.....'). At the bottom right is a blue 'Continue' button.

Screenshot 4: Jenkins Unlock Page

Step 3: Customize Jenkins

- **Action:** Choose either:
 - **Install suggested plugins** (Recommended)
 - **Select plugins to install** (Custom)
- Click on **Install suggested plugins**.



The screenshot shows the 'Getting Started' dialog box for Jenkins 2.516.1. It has a title 'Customize Jenkins' and a sub-instruction: 'Plugins extend Jenkins with additional features to support many different needs.' There are two main buttons: 'Install suggested plugins' (which says 'Install plugins the Jenkins community finds most useful.') and 'Select plugins to install' (which says 'Select and install plugins most suitable for your needs.'). The background is light grey.

Screenshot 5: Customize Jenkins Plugins

Step 4: Plugin Installation in Progress

- **Action:** Wait while Jenkins installs the necessary plugins.

The screenshot shows the 'Getting Started' dialog box for Jenkins 2.516.1. It has a title 'Getting Started'. Below it is a table of available Jenkins plugins. The table has columns for 'Folders', 'Available Plugins', and 'Installed Plugins'. The 'Available Plugins' column includes: Timestamper, Pipeline, Git, Email Extension, OWASP Markup Formatter, GitHub Branch Source, SSH Build Agents, Mailer, Ant, Pipeline: GitHub Groovy Libraries, Matrix Authorization Strategy, Dark Theme, Build Timeout, Gradle, Pipeline Graph View, LDAP, and Credentials Binding. The 'Installed Plugins' column shows checkmarks next to Folders, OWASP Markup Formatter, Build Timeout, Gradle, Pipeline Graph View, LDAP, and Credentials Binding. To the right of the table, there is a detailed list of dependencies for the 'Credentials Binding' plugin, including commons-lang3 v3.x Jenkins API, Ionicons API, commons-text API, commons-variant API, and various sub-dependencies like ASH API, JPath API, Structs, Pipeline: Step API, bouncycastle API, Credentials, Pipeline: Credentials, Variant, SSH Credentials, SOH API, and Pipeline: API. At the bottom, it says '** - required dependency'.

Screenshot 6: Plugin Installation Progress

Step 5: Create First Admin User

- **Action:** Fill out the admin user form:

- Username
- Password
- Full name
- Email address



- Click **Save and Continue**.

Getting Started

Create First Admin User

Username
Jayshri

Password
.....

Confirm password
.....

Full name
Jayshri Ashok Landge

E-mail address
jayshrilandge123@gmail.com

Jenkins 2.516.1 Skip and continue as admin **Save and Continue**

Screenshot 7: Create Admin User

Step 6: Instance Configuration

- **Action:** Jenkins will prompt to confirm the Jenkins URL. Leave it as default or update if needed.
- Click **Save and Finish**.

Getting Started

Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.
The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.516.1 Not now **Save and Finish**

Screenshot 8: Configure Jenkins URL



Step 7: Jenkins is Ready!

- **Action:** Click **Start using Jenkins**.
- **Action:** You are now logged into the Jenkins Dashboard.

The screenshot shows the Jenkins dashboard at the URL 15.188.119.96:8080. The title bar says "Jenkins". The main content area is titled "Welcome to Jenkins!". It includes a message: "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project." Below this is a section titled "Start building your software project" with a "Create a job" button and a "+" icon. Another section titled "Set up a distributed build" contains links for "Set up an agent" (with a monitor icon) and "Configure a cloud" (with a cloud icon). At the bottom right, there are links for "REST API" and "Jenkins 2.516.1".

Screenshot 9: Jenkins Dashboard

Installing & Removing HTTPD via Jenkins

Install HTTPD:

```
sudo yum install -y httpd  
sudo systemctl start httpd  
sudo chkconfig httpd on  
echo "Welcome to Jenkins" | sudo tee /var/www/html/index.html
```

The screenshot shows the Jenkins build history for the "install_httpd_job" project. The title bar says "Jenkins / install_httpd_job". The left sidebar has links for "Status", "Changes", "Workspace", "Build Now", "Configure", "Delete Project", and "Rename". The main content area shows a green checkmark icon next to "install_httpd_job". Below it is a "Permalinks" section with a bulleted list of four build links. Underneath is a "Builds" section with a table showing one build entry: "#2 11:55 AM".

Screenshot 10: Jenkins Build Success Status Of Install httpd



Remove HTTPD:

```
sudo yum remove -y httpd
```

```
sudo rm -rf /var/www/html/index.html
```

The screenshot shows the Jenkins interface with a job named "remove_httpd_job". The "Console Output" tab is selected, displaying the command output. The output shows the user running "sudo apt remove -y apache2" and the process of removing the package, including dependency resolution and file removal.

```
Started by user Jayshri Ashok Landge
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/remove_httpd_job
[remove_httpd_job] $ /bin/sh -xe /tmp/jenkins137800804075569797666.sh
+ sudo apt remove -y apache2

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists...
Building dependency tree...
Reading state information...
The following packages were automatically installed and are no longer required:
  apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1t64 libdbd-sqlite3
  libaprutil1-ldap libaprutil1t64 liblua5.4-0 ssl-cert
Use 'sudo apt autoremove' to remove them.
The following packages will be REMOVED:
  apache2
0 upgraded, 0 newly installed, 1 to remove and 5 not upgraded.
After this operation, 465 kB disk space will be freed.
(Reading database ...
(Reading database ... 5%
(Reading database ... 10%
(Reading database ... 15%
(Reading database ... 20%
(Reading database ... 25%
```

Screenshot 11: Console output of HTTPD remove job in Jenkins

Cloning GitHub Repositories

1 Without Plugin:

Use shell in Jenkins job:

```
sudo yum install -y git
```

```
sudo git clone https://github.com/username/repo.git
```

The screenshot shows the Jenkins interface with a job named "clone_repo_job". The "Console Output" tab is selected, displaying the command output. The output shows the user running "git clone https://github.com/username/repo.git" and the process of cloning the repository, including package updates and dependency resolution.

```
Started by user Jayshri Ashok Landge
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/clone_repo_job
[clone_repo_job] $ /bin/sh -xe /tmp/jenkins2025066108219347213.sh
+ sudo apt update

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Hit:1 http://eu-west-3.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://eu-west-3.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://eu-west-3.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu mobile-security InRelease
Ign:5 https://pkgs.jenkins.io/debian-stable binary/ InRelease
Hit:6 https://pkgs.jenkins.io/debian-stable binary/ Release
Reading package lists...
Building dependency tree...
Reading state information...
5 packages can be upgraded. Run 'apt list --upgradable' to see them.
+ sudo apt install -y git

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists...
Building dependency tree...
Reading state information...
git is already the newest version (1:2.43.0-1ubuntu7.3).
The following packages were automatically installed and are no longer required:
  apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1t64 libdbd-sqlite3
  libaprutil1-ldap libaprutil1t64 liblua5.4-0 ssl-cert
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
+ git clone https://github.com/jayshriLandge10/Devops_auto_deploy_website.git
Cloning into 'Devops_auto_deploy_website'...
Finished: SUCCESS
```

Screenshot 12: Console Output showing successful git clone



2 With Plugin (Public Repo):

Configure Git in Source Code Management → Git URL → Save → Build

A screenshot of the Jenkins configuration interface for a job named "Git_Pugin_job". The left sidebar shows options like General, Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions. The "Source Code Management" section is selected. It shows a "Repository URL" input field containing "https://github.com/jayshrilandge30/Netflix-clone.git". Below it is a "Credentials" dropdown menu which is currently empty. An "Advanced" button is at the bottom right of the configuration panel.

Screenshot 13: Source Code Management configuration

3 With Plugin (Private Repo):

Add Git credentials (username and PAT) → Select from dropdown → Save → Build

A screenshot of the Jenkins configuration interface for the same job. The "Source Code Management" section is selected. The "Credentials" dropdown menu now contains a single item: "880a1a0c-f7c0-40cb-b4aa-21c1ecabc3c9". The rest of the interface is identical to Screenshot 13, showing the repository URL and the "Advanced" button.

Screenshot 14: Credential configuration for Git in Jenkins

Automating Builds

Poll SCM

Cron schedule like * * * * * for polling → Jenkins triggers build on change



The screenshot shows the Jenkins job configuration interface for a job named 'Git_Pugin_job'. The 'Triggers' section is active. Under 'Poll SCM', a cron schedule is set to '* * * * *'. A note below states: 'No schedules so will only run due to SCM changes if triggered by a post-commit hook'. There is also an unchecked checkbox for 'Ignore post-commit hooks'.

Screenshot 15: Build Triggers - Poll SCM configuration

Git Webhook

- GitHub → Webhooks → Add Jenkins URL /github-webhook/
- Jenkins → Manage Jenkins → GitHub server → Secret Token

The screenshot shows the GitHub settings page for a repository named 'Devops_auto_deploy_website'. The 'Webhooks' section is active. It shows a single webhook configuration for Jenkins with the URL 'http://35.180.34.30:8080/github-webhook/push'. The status message indicates 'Last delivery was successful.' with an 'Edit' and 'Delete' button.

Screenshot 16: Webhook setup in GitHub

User & Access Management

Create User:

Manage Jenkins → Manage Users → Create User

The screenshot shows the Jenkins 'Manage Users' page. It lists two users: 'Jayshri' and 'Rohan'. Each user has a profile icon, a 'Name' column, and edit/delete icons. A note at the top states: 'These users can log into Jenkins. This is a sub set of [this list](#), which also contains auto-created users who really just made some commits on some projects and have no direct Jenkins access.'

User ID	Name	Action
Jayshri	Jayshri Ashok Landge	
Rohan	Rohan Ashok Landge	

Screenshot 17: User creation

Jayshri Ashok Landge



Configure Permissions:

Matrix-based security → Add user → Assign permissions → Save

The screenshot shows the Jenkins 'Manage Jenkins' interface under the 'Security' section. It displays a matrix-based permission configuration. The columns represent actions (View, ThreadDump, HealthCheck, Tag) and the rows represent users (User/group, Admin, Anonymous). The 'Anonymous' row has checkboxes for View, ThreadDump, and HealthCheck. The 'Authenticated Users' row has checkboxes for View, ThreadDump, and HealthCheck. The 'Rohan Ashok Landge' row has checkboxes for View, ThreadDump, and HealthCheck, with the 'HealthCheck' checkbox being checked.

Screenshot 18: Matrix permission panel

Connecting Jenkins Master and Agent

A. Using JNLP:

- Launch EC2 (Slave)
- Install Java
- Create workspace: mkdir /dir1
- Jenkins → New Node → Permanent Agent
- Launch using JNLP
- Run agent using nohup command

The screenshot shows the Jenkins 'Manage Jenkins' interface under the 'Nodes' section. It lists two nodes: 'Built-In Node' and 'node1'. The 'node1' node has a status of 'Data obtained' and a response time of '58 sec'. Below the table is a terminal window showing the command: 'curl -s0 http://35.181.153.100:8080/jnlpJars/agent.jar & nohup java -jar agent.jar -url http://35.181.153.100:8080/ -secret b23d88a45490bb08c6cf78b5d00612b67105c4ae138169b9746349e41bb89915 -name model -webSocket -workDir "/dir1" & [1] 16515'.

Screenshot 19: Node creation and JNLP command on Jenkins



B. Using SSH:

- Launch EC2 instance (Agent)
- Install Java
- Create directory /home/ec2-user/dir2
- Jenkins → New Node → Launch via SSH
- Add credentials (username + PEM key)
- Trust SSH key → Save

The screenshot shows a browser window with the URL `35.181.153.100:8080/computer/ubuntu-agent/log`. The Jenkins logo is at the top left. The page title is "Log". The log output shows the following:

```
SHELL=/bin/bash
SHELLLOPTS=braceexpand:hashall:interactive-comments
SHLVL=1
SSH_CLIENT="172.31.39.65 44046 22"
SSH_CONNECTION="172.31.39.65 44046 172.31.43.77 22"
TERM=dumb
UID=1000
USER=ubuntu
XDG_RUNTIME_DIR=/run/user/1000
XDG_SESSION_CLASS=user
XDG_SESSION_ID=18
XDG_SESSION_TYPE=pty
"-"
[08/05/25 17:38:14] [SSH] Starting sftp client.
[08/05/25 17:38:14] [SSH] Copying latest remoting.jar...
[08/05/25 17:38:15] [SSH] Copied 1,397,044 bytes.
Expanded the channel window size to 4MB
[08/05/25 17:38:15] [SSH] Starting agent process: cd "/home/ubuntu/jenkins-agent" && java -jar remoting.jar -workDir /home/ubuntu/jenkins-agent -jar-cache /home/ubuntu/jenkins-agent/remoting/jarCache
Aug 05, 2025 5:38:15 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/jenkins-agent as a remoting work directory
Aug 05, 2025 5:38:15 PM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/ubuntu/jenkins-agent/remoting
<-->[JENKINS REMOTING CAPACITY]=>channel started
Remoting version: 3380_v27b_9314fd1a_4
Launcher: SSHLauncher
Communication Protocol: Standard in/out
This is a Unix agent
Agent successfully connected and online
```

Screenshot 20: SSH Node configuration and status: Connected

Multi-configuration (Matrix) Jobs

- Jenkins → New Item → Multi-configuration project
- Add Axis: Label Expression → Values: built-in, instance1
- Configure build script → Run on both nodes in parallel

The screenshot shows the Jenkins configuration interface for a multi-configuration job named "multi_config_job". The left sidebar shows "Configure" sections: General, Advanced Project Options, Source Code Management, Triggers (selected), Configuration Matrix, Environment, Build Steps, and Post-build Actions.

The "Triggers" section is expanded, showing options like Trigger builds remotely, Build after other projects are built, Build periodically, GitHub hook trigger for GITScm polling, and Poll SCM.

The "Configuration Matrix" section is expanded, showing a "Label expression" input field with "label_exp" entered, and a "Label Expressions" dropdown containing "built-in" and "prod". A note below states: "Label built-in matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list. Label prod matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list."

Screenshot 21: Axis configuration



The screenshot shows the Jenkins interface for a 'multi_configi_job'. The left sidebar has links for Status, Changes, Workspace, Build Now, Configure, Delete Multi-configuration project, Rename, and a Builds section with a filter. The main area is titled 'Project multi_configi_job' and contains sections for 'Configurations' (built-in, prod) and 'Permalinks'. It lists four recent builds: Last build (#2), Last stable build (#2), Last successful build (#2), and Last completed build (#2), all from 10 seconds ago. A 'Builds' table shows one entry for build #2 at 5:50 PM.

Screenshot 22: Build results on different nodes

Jenkins Pipeline Scripts (Groovy)

Basic Pipeline:

```
pipeline {  
    agent any  
    stages {  
        stage('Build') { steps { echo 'Building...' } }  
        stage('Test') { steps { echo 'Testing...' } }  
        stage('Deploy') { steps { echo 'Deploying...' } }  
    }  
}
```

Real-Time Tasks:

- User creation: sudo useradd jayshri

The screenshot shows the Jenkins console output for a 'useradd' pipeline job. The left sidebar includes links for Status, Changes, Console Output (which is selected), Edit Build Information, Delete build #1, Timings, Pipeline Overview, Restart from Stage, Replay, Pipeline Steps, and Workspaces. The main area is titled 'Console Output' and shows the command 'sudo useradd jayshri' being run. The output log shows it was started by user Jayshri Ashok Landge, running on Jenkins, and finished successfully.

Screenshot 23: Console outputs of pipeline jobs (user creation)



- Directory operations: mkdir, touch, yum install, build other jobs

```
Started by user Jayshri Ashok Landge
[Pipeline] Start of Pipeline
[Pipeline] node
Running on ubuntu-agent in /home/ubuntu/jenkins-agent/workspace/job2
[Pipeline] {
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (stage-1)
[Pipeline] sh
+ sudo mkdir /tmp/dir11
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (pkg-installation)
[Pipeline] sh
+ sudo apt-get install -y git
Reading package lists...
Building dependency tree...
Reading state information...
git is already the newest version (1:2.43.0-1ubuntu7.3).
0 upgraded, 0 newly installed, 0 to remove and 7 not upgraded.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Screenshot 24: Console outputs of pipeline jobs (mkdir, httpd install)

Pipeline with Agent-Specific Workspaces

Run on Master:

```
agent { label 'built-in' }
```

Run on Custom Workspace:

```
agent {

node {

label 'prod'

customWorkspace '/home/ec2-user/dir3'

}

}
```



Hybrid Pipeline (Stage-1 on Agent, Stage-2 on Master):

```
pipeline {  
    agent {  
        node {  
            label "prod"  
            customWorkspace "/home/ec2-user/dir3"  
        }  
    }  
    stages {  
        stage ("stage-1") {  
            steps { sh "sudo touch file6" }  
        }  
        stage ("stage-2") {  
            agent {  
                node {  
                    label "built-in"  
                    customWorkspace "jayshri"  
                }  
            }  
            steps { sh "sudo touch file6" }  
        }  
    }  
}
```



Jenkins / job3 / #1

Console Output

Started by user Jayshri Ashok Landge
[Pipeline] Start of Pipeline
[Pipeline] node
Running on ubuntu-agent in /home/ubuntu/jenkins-agent/workspace/job3
[Pipeline] {
[Pipeline] ws
Running In /home/ubuntu/dir3
[Pipeline]
[Pipeline] stage
[Pipeline] { (stage-1)
[Pipeline] sh
+ mkdir -p /home/ubuntu/dir3
+ touch file3
+ echo File created on prod node in /home/ubuntu/dir3
File created on prod node in /home/ubuntu/dir3
[Pipeline]
[Pipeline] } // stage
[Pipeline] stage
[Pipeline] { (stage-2)
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/job3
[Pipeline]
[Pipeline] ws
Running in /var/lib/jenkins/workspace/jayshri
[Pipeline]
[Pipeline] sh
+ mkdir -p /var/lib/jenkins/workspace/jayshri
+ touch file4
+ echo File created on built-in node in /var/lib/jenkins/workspace/jayshri
File created on built-in node in /var/lib/jenkins/workspace/jayshri
[Pipeline]
[Pipeline] } // ws
[Pipeline]
[Pipeline] } // node
[Pipeline] } // stage
[Pipeline]
[Pipeline] } // ws
[Pipeline]
[Pipeline] } // node
[Pipeline] End of Pipeline
Finished: SUCCESS

Screenshot 25: Pipeline build output for hybrid workspace setup

Conclusion

Through this practical exploration of Jenkins, I've:

- Installed and configured Jenkins from scratch
- Cloned repositories (public & private)
- Created freestyle and matrix jobs
- Integrated Jenkins with GitHub using both polling and webhooks
- Built pipelines that span across master and agent nodes with custom workspaces

This journey reinforced my knowledge in **DevOps automation, pipeline scripting, CI/CD practices, and agent node management**, providing a strong foundation for tackling real-world DevOps projects.