ASSIGNMENT – 1 Name: B N L Pravallika Id: 2024tm93314

Git Repo: https://github.com/Lakshmi-Pravallika/MyJenkinsApplication2

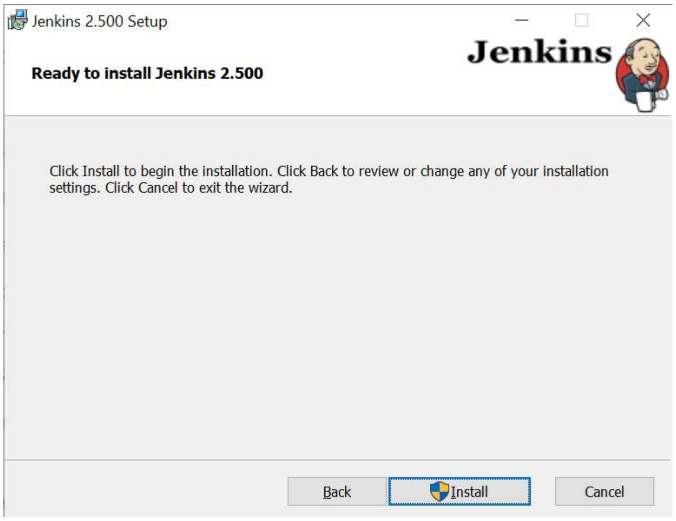
INSTALL JENKINS ON THE MASTER NODE (WINDOWS)

Step 1: Download Jenkins

Go to the Jenkins download page and download the Windows installer (.msi).

Step 2: Install Jenkins

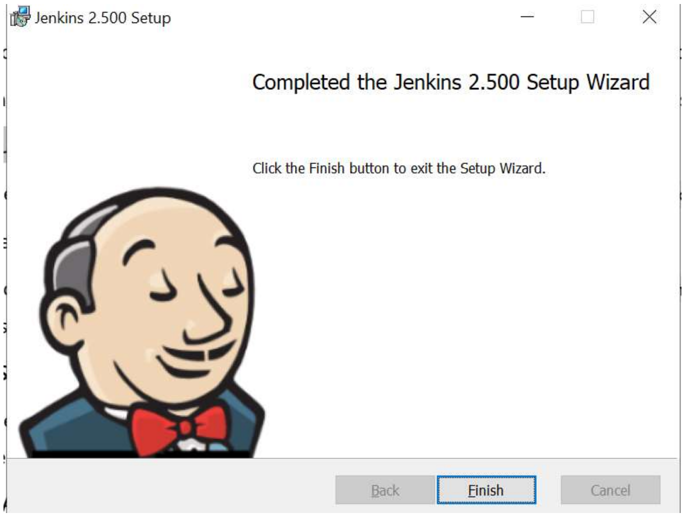
Run the .msi file and follow the installation steps. Choose the default installation option and keep the default Jenkins installation directory.



Jenkins 2.500 Setup

Ready to install Jenkins 2.500

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.



Jenkins 2.500 Setup

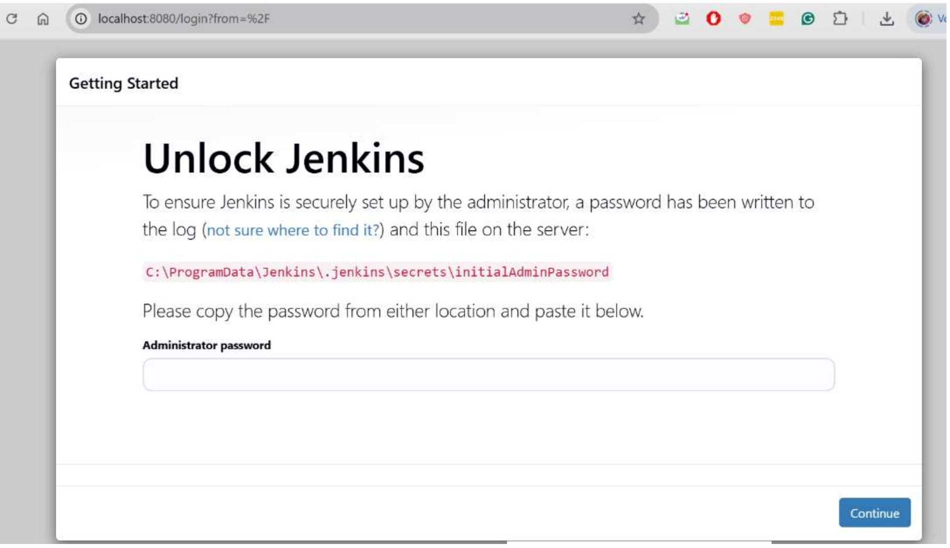
Completed the Jenkins 2.500 Setup Wizard

Click the Finish button to exit the Setup Wizard.

Step 3: Access Jenkins

After installation, Jenkins will be accessible in your web browser at http://localhost:8080. The first time you open Jenkins, it will ask for an unlock key.

Find the key in the file C:\Program Files (x86)\Jenkins\secrets\initialAdminPassword and paste it in the browser to unlock Jenkins.



localhost:8080/login?from=%2F

Getting Started

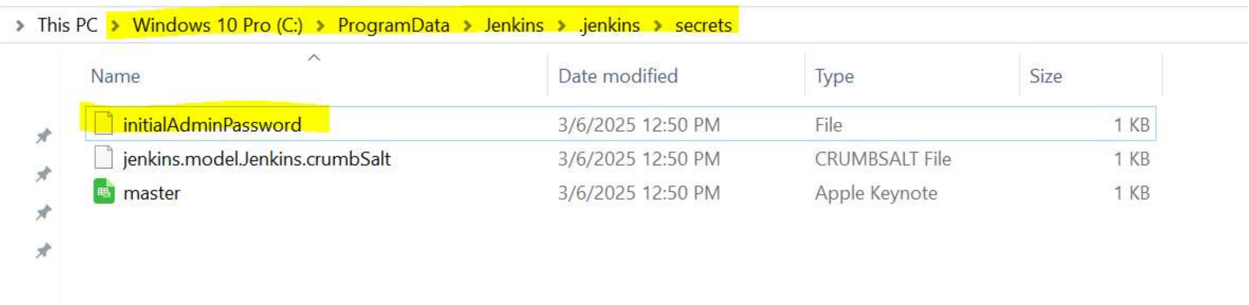
Unlock Jenkins

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:

```
C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword
```

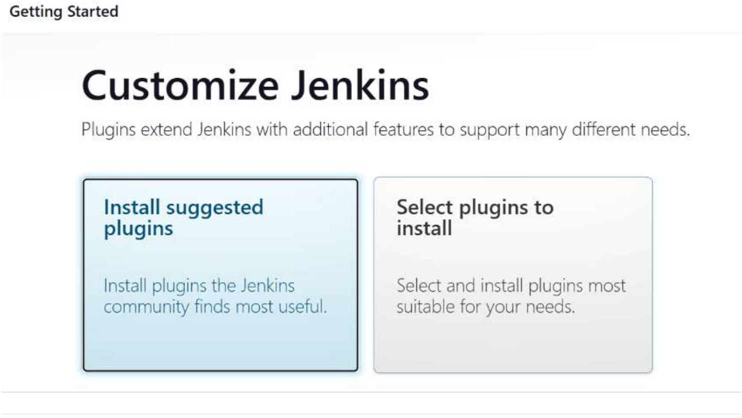
Please copy the password from either location and paste it below.

Administrator password



This PC > Windows 10 Pro (C:) > ProgramData > Jenkins > .jenkins > secrets

Name	Date modified	Type	Size
initialAdminPassword	3/6/2025 12:50 PM	File	1 KB
jenkins.model.Jenkins.crumbSalt	3/6/2025 12:50 PM	CRUMBSALT File	1 KB
master	3/6/2025 12:50 PM	Apple Keynote	1 KB



Getting Started

Customize Jenkins

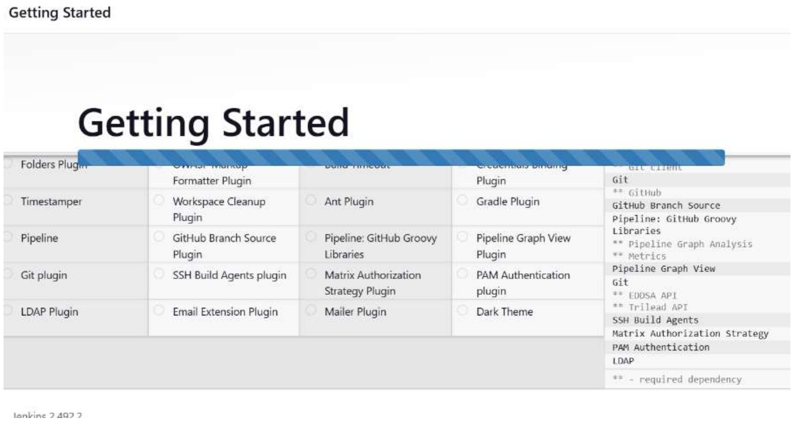
Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.



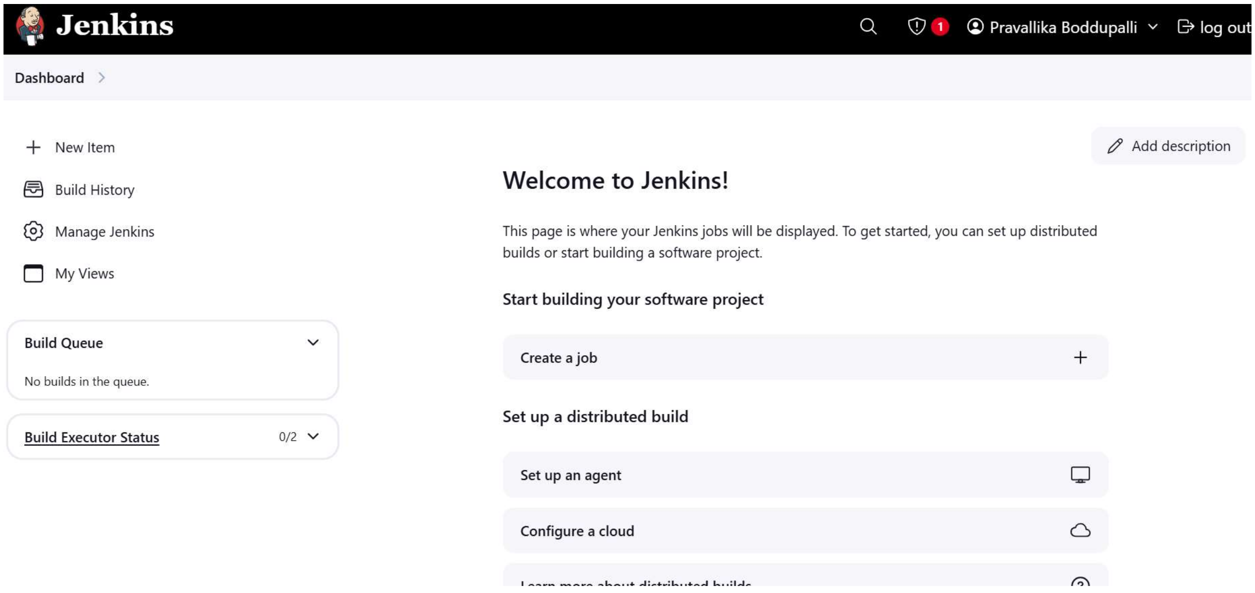
Getting Started

Getting Started

○ Folders Plugin	○ OWASP Markup Formatter Plugin	○ Build Timeout	○ Credentials Binding Plugin
○ Timestamper	○ Workspace Cleanup Plugin	○ Ant Plugin	○ Gradle Plugin
○ Pipeline	○ GitHub Branch Source Plugin	○ Pipeline: GitHub Groovy Libraries	○ Pipeline Graph View Plugin
○ Git plugin	○ SSH Build Agents plugin	○ Matrix Authorization Strategy Plugin	○ PAM Authentication plugin
○ LDAP Plugin	○ Email Extension Plugin	○ Mailer Plugin	○ Dark Theme

```
** Git Client
Git
** GitHub
GitHub Branch Source
Pipeline: GitHub Groovy
Libraries
** Pipeline Graph Analysis
** Metrics
Pipeline Graph View
Git
** EDDSA API
** Trilead API
SSH Build Agents
Matrix Authorization Strategy
PAM Authentication
LDAP
** - required dependency
```

Jenkins 2.492.2



Build Queue

No builds in the queue.

Build Executor Status 0/2

Welcome to Jenkins!

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.

Start building your software project

Create a job +

Set up a distributed build

Set up an agent

Configure a cloud

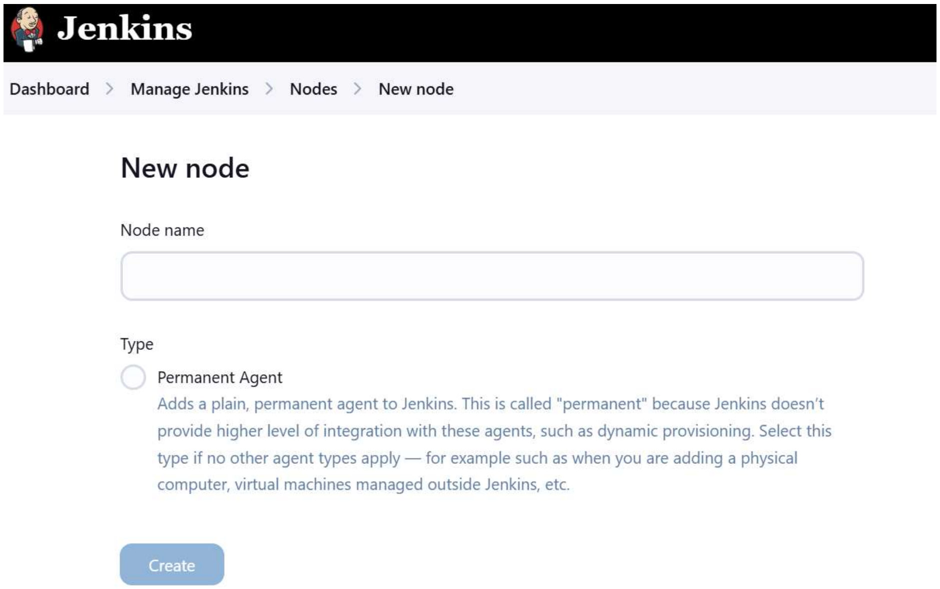
Learn more about distributed builds

SETUP SLAVE NODE (WINDOWS)

Step :1 Enable Remote Access on Jenkins:

a) Creating a new Node

Go to Jenkins dashboard → Manage Jenkins → Manage Nodes and Clouds → New Node.



New node

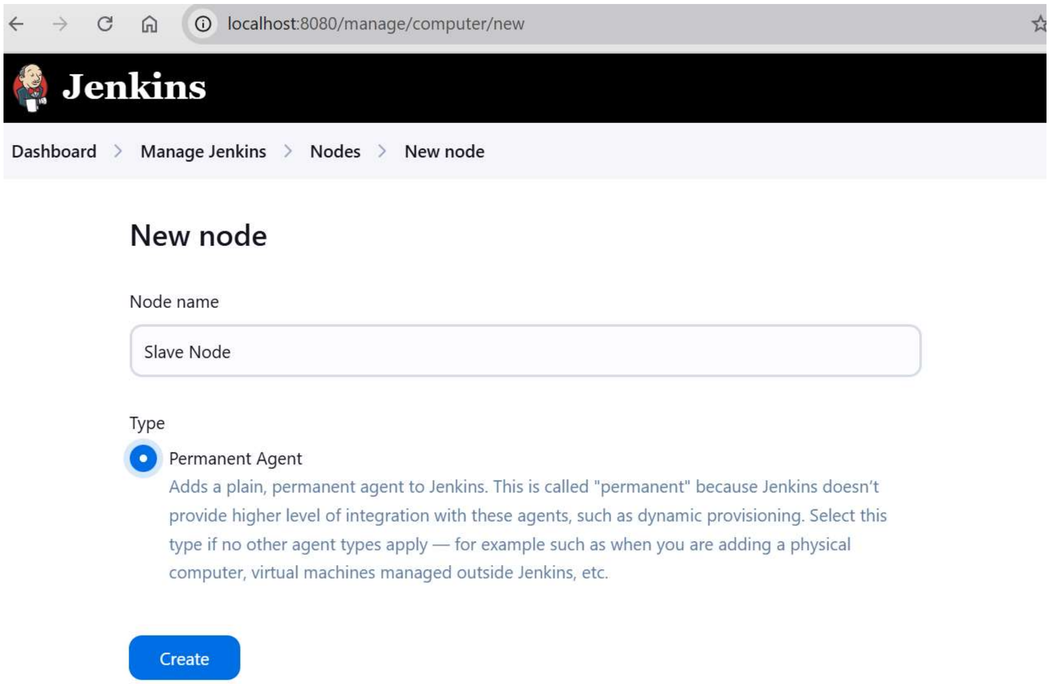
Node name

Type

○ Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Enter a name for the slave node and select "Permanent Agent". Click "OK".


localhost:8080/manage/computer/new
Jenkins
Dashboard > Manage Jenkins > Nodes > New node

New node

Node name: Slave Node

Type

● Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

b) **Configure Slave Node**:

Set the following details:

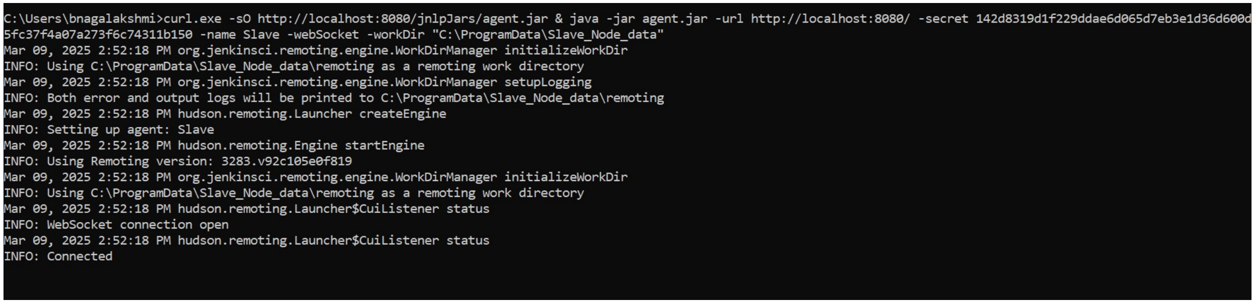
- **Remote root directory**: A folder where Jenkins will store the files on the slave machine.
- **Labels**: This is used to assign specific jobs to the slave node.
- **Launch method**: Choose "Launch agent via Java Web Start" or "Launch agent via SSH" (depending on your setup).

After filling out the node configuration, click **Save**.

c) **Connect to Slave Node:**

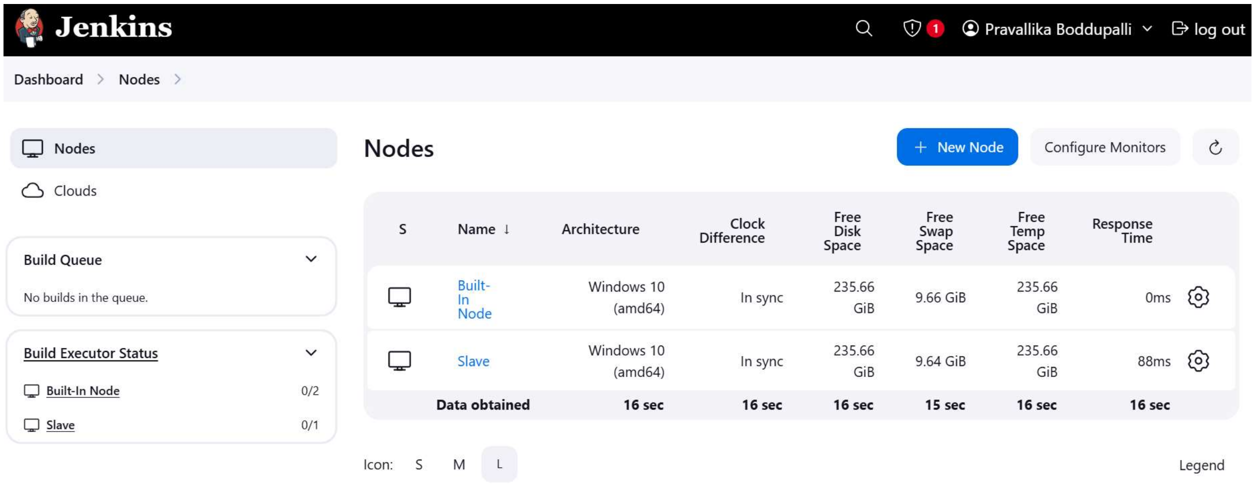
Execute this in the cmd

curl.exe -sO http://localhost:8080/jnlpJars/agent.jar & java -jar agent.jar -url http://localhost:8080/ -secret 142d8319d1f229ddae6d065d7eb3e1d36d600d5fc37f4a07a273f6c74311b150 -name Slave -webSocket -workDir "C:\ProgramData\Slave_Node_data"



```
C:\Users\bnagalakshmi>curl.exe -sO http://localhost:8080/jnlpJars/agent.jar & java -jar agent.jar -url http://localhost:8080/ -secret 142d8319d1f229ddae6d065d7eb3e1d36d600d
5fc37f4a07a273f6c74311b150 -name Slave -webSocket -workDir "C:\ProgramData\Slave_Node_data"
Mar 09, 2025 2:52:18 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using C:\ProgramData\Slave_Node_data\remoting as a remoting work directory
Mar 09, 2025 2:52:18 PM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to C:\ProgramData\Slave_Node_data\remoting
Mar 09, 2025 2:52:18 PM hudson.remoting.Launcher createEngine
INFO: Setting up agent: Slave
Mar 09, 2025 2:52:18 PM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 3283.v92c105e0f819
Mar 09, 2025 2:52:18 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using C:\ProgramData\Slave_Node_data\remoting as a remoting work directory
Mar 09, 2025 2:52:18 PM hudson.remoting.Launcher$CuiListener status
INFO: WebSocket connection open
Mar 09, 2025 2:52:18 PM hudson.remoting.Launcher$CuiListener status
INFO: Connected
```

The slave is up and running now.



No builds in the queue.

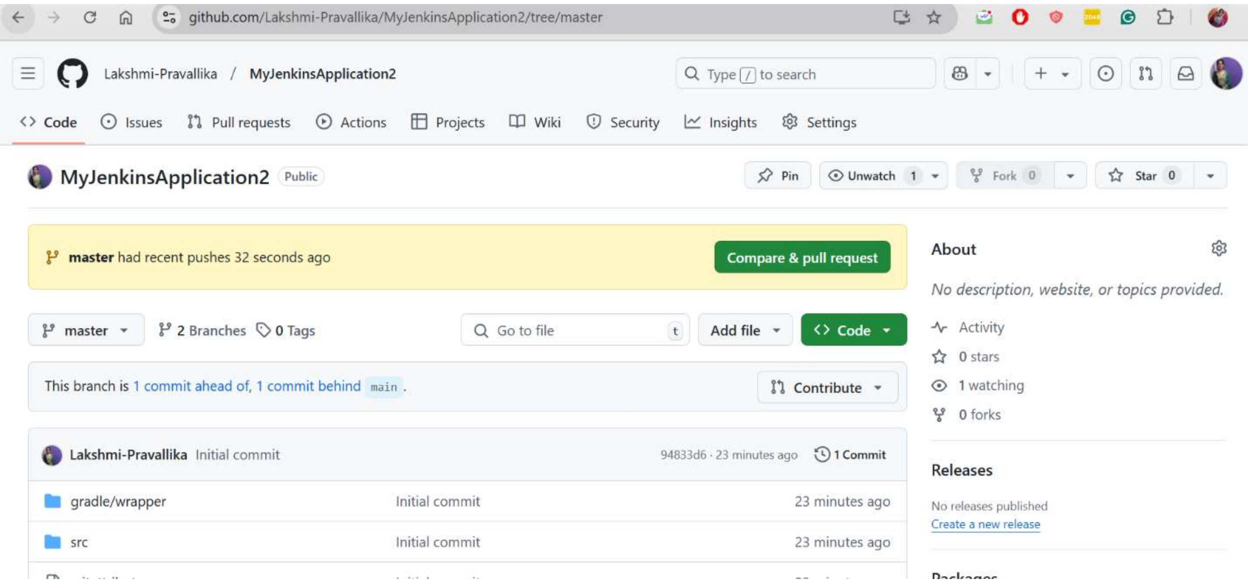
Built-In Node 0/2

Slave 0/1

Nodes

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Windows 10 (amd64)	In sync	235.66 GiB	9.66 GiB	235.66 GiB	0ms
	Slave	Windows 10 (amd64)	In sync	235.66 GiB	9.64 GiB	235.66 GiB	88ms
	Data obtained	16 sec	16 sec	16 sec	15 sec	16 sec	16 sec

Create a Git Repository



github.com/Lakshmi-Pravallika/MyJenkinsApplication2/tree/master

master had recent pushes 32 seconds ago

This branch is 1 commit ahead of, 1 commit behind main .

Lakshmi-Pravallika Initial commit 94833d6 · 23 minutes ago 1 Commit

gradle/wrapper	Initial commit	23 minutes ago
src	Initial commit	23 minutes ago

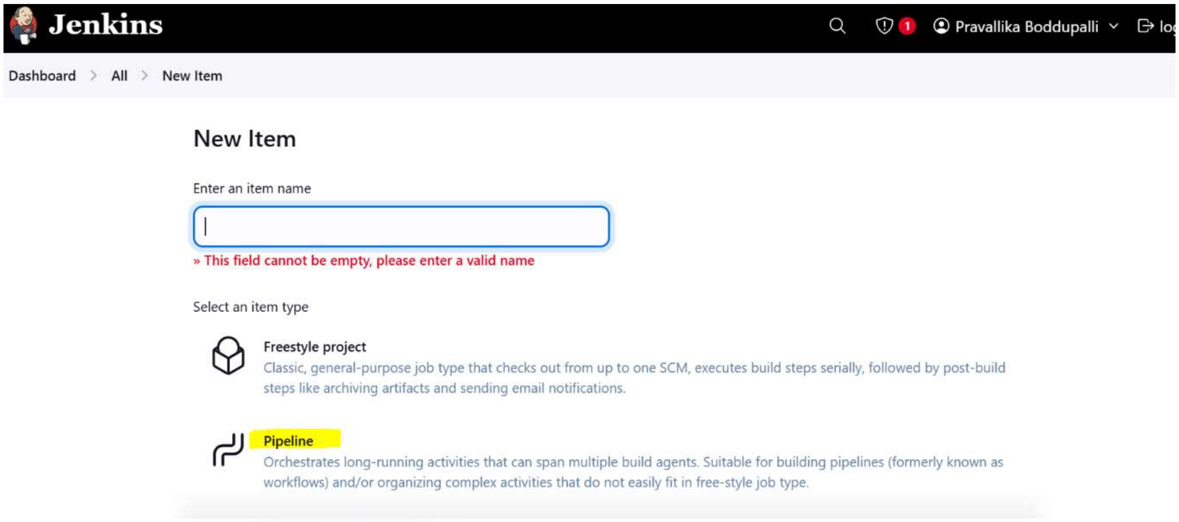
No description, website, or topics provided.

No releases published

CREATING JENKINS PIPELINE

Create a New Pipeline Project

- Go to Jenkins → **New Item**.
- Select "Pipeline" and give it a name (e.g., MyPipeline).
- Choose "Pipeline" under "Project" and click "OK."



New Item

Enter an item name

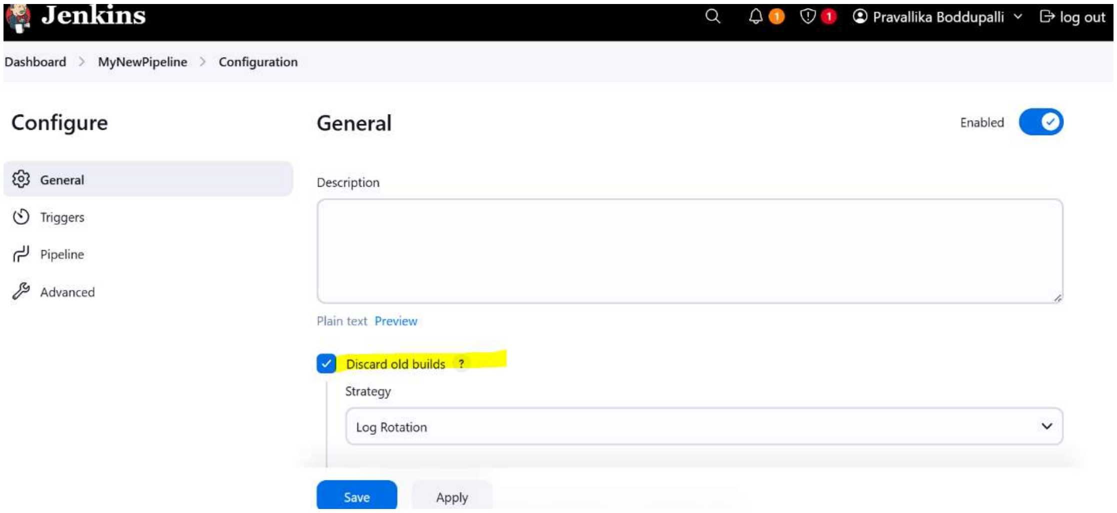
» This field cannot be empty, please enter a valid name

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.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

2) Select discard old builds.



Dashboard > MyNewPipeline > Configuration

General

Enabled

Description

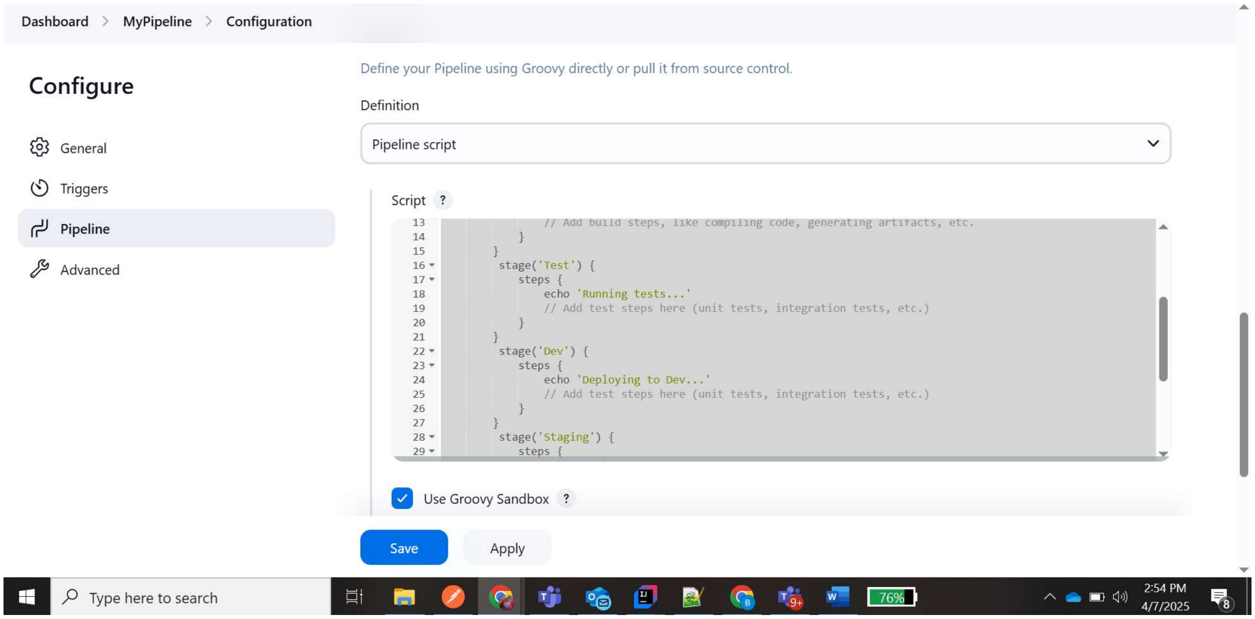
Plain text Preview

☑ Discard old builds ?

Strategy

Log Rotation

3) Define Pipeline script



Dashboard > MyPipeline > Configuration

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script

Script ?

```
                // Add build steps, like compiling code, generating artifacts, etc.
            }
        }
         stage('Test') {
            steps {
                echo 'Running tests...'
                // Add test steps here (unit tests, integration tests, etc.)
            }
        }
         stage('Dev') {
            steps {
                echo 'Deploying to Dev...'
                // Add test steps here (unit tests, integration tests, etc.)
            }
        }
         stage('Staging') {
            steps {
```

☑ Use Groovy Sandbox ?

Script:

```
pipeline {
   agent any
   stages {
        stage('Git') {
            steps {
                git branch: 'main', url: 'https://github.com/Lakshmi-Pravallika/MyJenkinsApplication2.git'
            }
        }
        stage('Build') {
            steps {
                echo 'Building the project...'
            }
        }
         stage('Test') {
            steps {
                echo 'Running tests...'
                   }
        }
         stage('Dev') {
            steps {
                echo 'Deploying to Dev...'
            }
        }
         stage('Staging') {
            steps {
                echo 'Deploying to Staging'
                     }
        }
        stage('Production') {
            steps {
                echo 'Deploying to Prod'
            }
         }
      }
   }
```

Click Save and Apply

Pipeline Stage View Plugin for Jenkins:

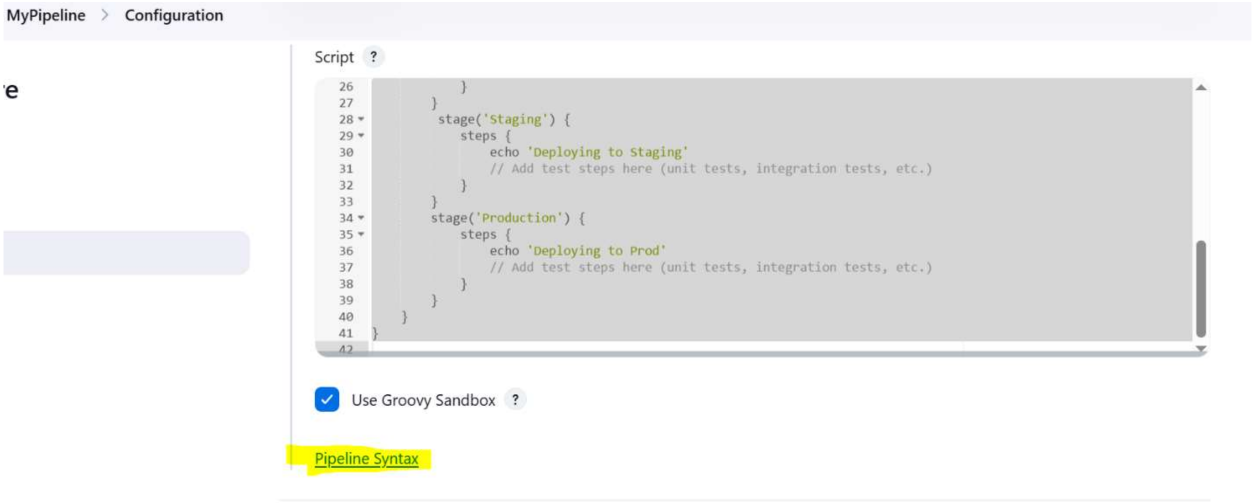
The **Pipeline Stage View Plugin** is a Jenkins plugin that provides a **graphical view of the pipeline's execution stages**. It shows the progress of the pipeline by visualizing the stages, whether they're **running**, **successful**, or **failed**. This plugin helps you monitor the status of individual stages in your Jenkins pipeline with a simple, easy-to-read display.

Steps:

- Go to Manage Jenkins
- Go to Manage Plugins
- Go to Available
- Use the search bar to look for **Pipeline Stage View Plugin**.
- Select the **Pipeline Stage View Plugin** from the list.
- Click the **Install without Restart** button to install the plugin.

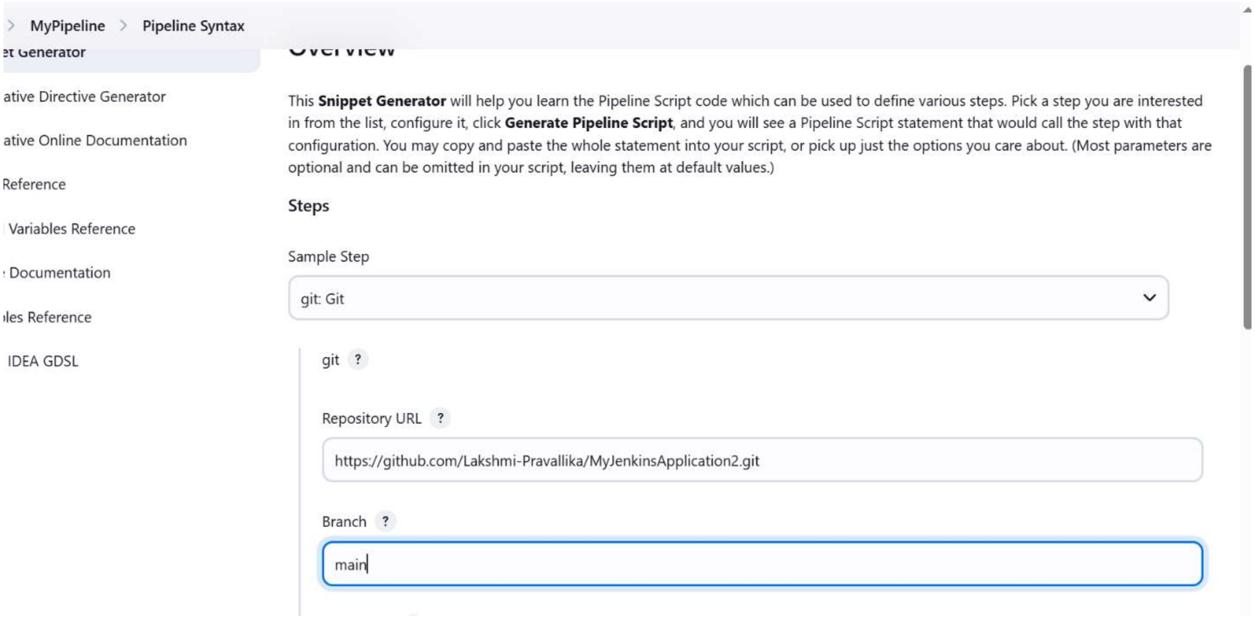
Integrating Git with Jenkins

1) Click on Pipeline Syntax



```
            }
        }
         stage('Staging') {
            steps {
                echo 'Deploying to Staging'
                // Add test steps here (unit tests, integration tests, etc.)
            }
        }
        stage('Production') {
            steps {
                echo 'Deploying to Prod'
                // Add test steps here (unit tests, integration tests, etc.)
            }
        }
    }
}
```

2) Select GIT and enter repository URL



This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

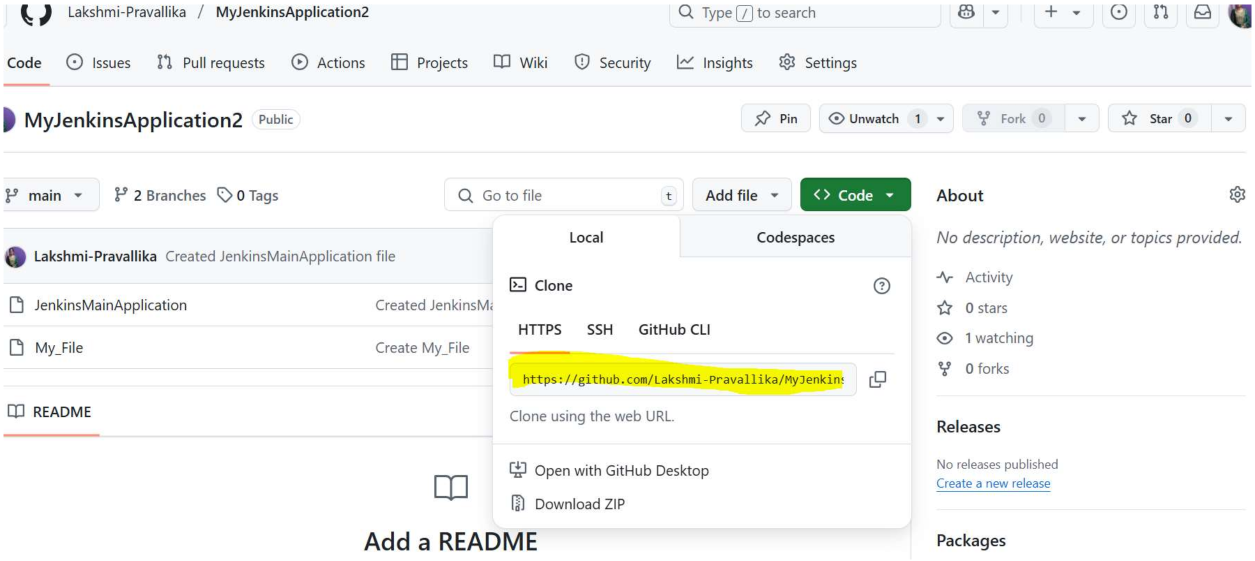
Sample Step: git: Git

git ?

Repository URL ?: https://github.com/Lakshmi-Pravallika/MyJenkinsApplication2.git

Branch ?: main

No credentials are required for public repository. If the repository is private, enter the GIT credentials.



Lakshmi-Pravallika Created JenkinsMainApplication file

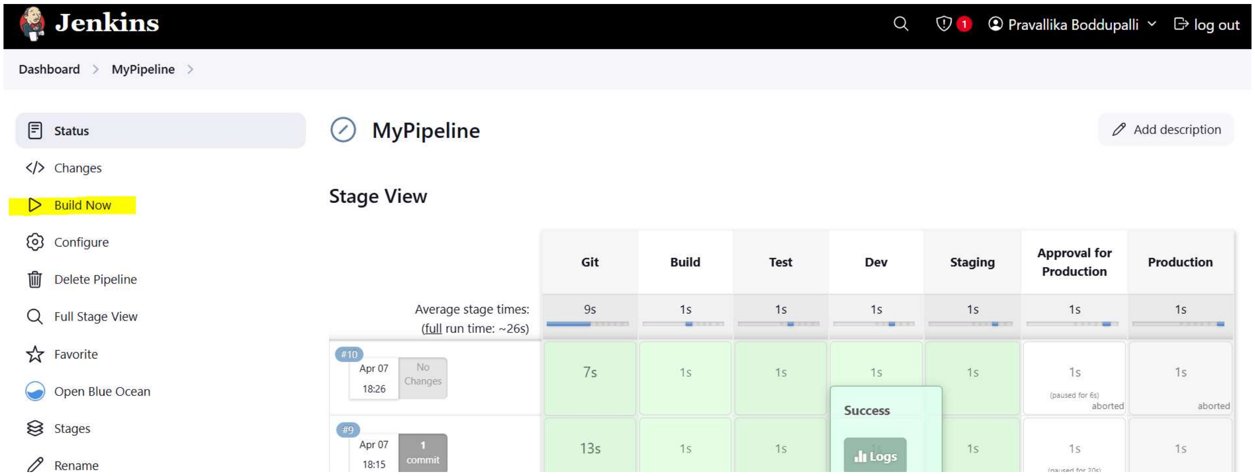
JenkinsMainApplication	Created JenkinsMa
My_File	Create My_File

https://github.com/Lakshmi-Pravallika/MyJenkins

Clone using the web URL.

No description, website, or topics provided.

3) Click on Save.
4) Click on Build Now



	Git	Build	Test	Dev	Staging	Approval for Production	Production
Average stage times: (full run time: ~26s)	9s	1s	1s	1s	1s	1s	1s
#10 Apr 07 18:26 No Changes	7s	1s	1s	1s	1s	1s (paused for 6s) aborted	1s aborted
#9 Apr 07 18:15 1 commit	13s	1s	1s	Success Logs	1s	1s (paused for 20s)	1s

The pipeline script has the above shown stages i.e.,

1) **GIT** Stage where the repo is cloned.
2) **BUILD** Stage where the application is built.
3) **TEST** Stage where the tests in the application are ran.
4) **DEV** Stage where the application is deployed to Dev environment.
5) **STAGING** Stage where the application is deployed to Stage environment.
6) **APPROVAL FOR PROD** Stage where the pipeline awaits user approval for prod deployment.
7) **PRODUCTION** Stage where the application goes live to prod.

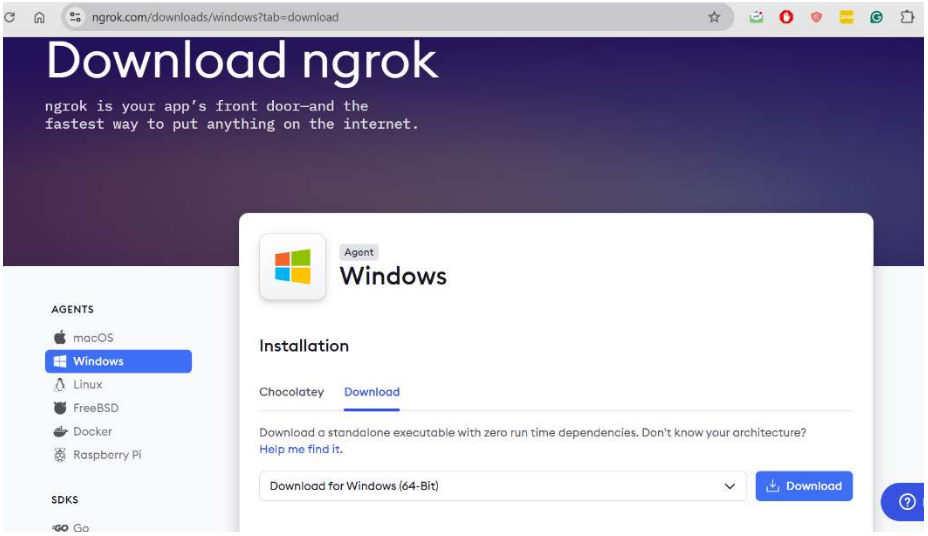
INTEGRATING GIT WITH JENKINS FOR AUTO BUILD TRIGGER

- In your GitHub repository, go to **Settings** > **Webhooks** > **Add webhook**.
- In the **Payload URL**, enter your Jenkins server's URL with the /github-webhook/ endpoint (e.g., http://localhost:8080/github-webhook/).
- Set **Content type** to application/json.
- Choose **Just the push event** or other triggers as per your requirement.
- Click **Add webhook**.

GETTING JENKINS SERVER'S URL VIA NGROK:

- Go to ngrok.com and signup with email id.

- Download the setup for windows.



ngrok.com/downloads/windows?tab=download

Download ngrok

ngrok is your app's front door—and the fastest way to put anything on the internet.

AGENTS
- macOS
- Windows
- Linux
- FreeBSD
- Docker
- Raspberry Pi

SDKS
- Go

Agent

Windows

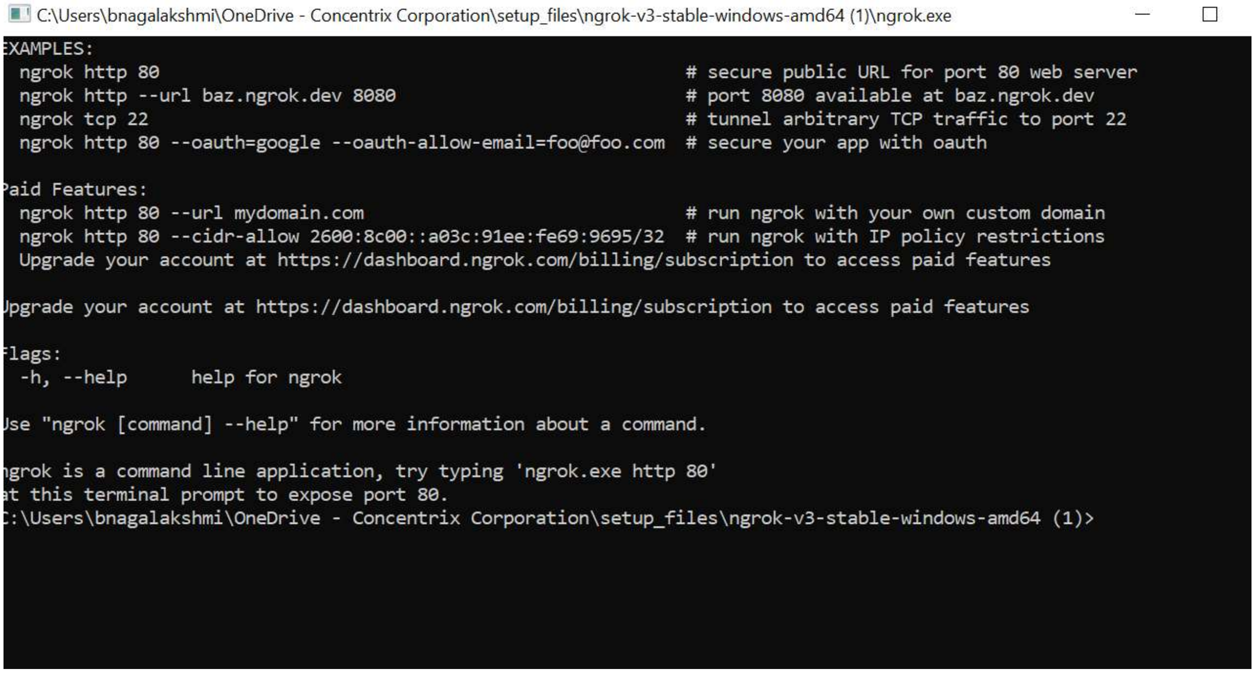
Installation

Chocolatey | Download

Download a standalone executable with zero run time dependencies. Don't know your architecture? Help me find it.

Download for Windows (64-Bit) | Download

- Run the ngrok exe file.

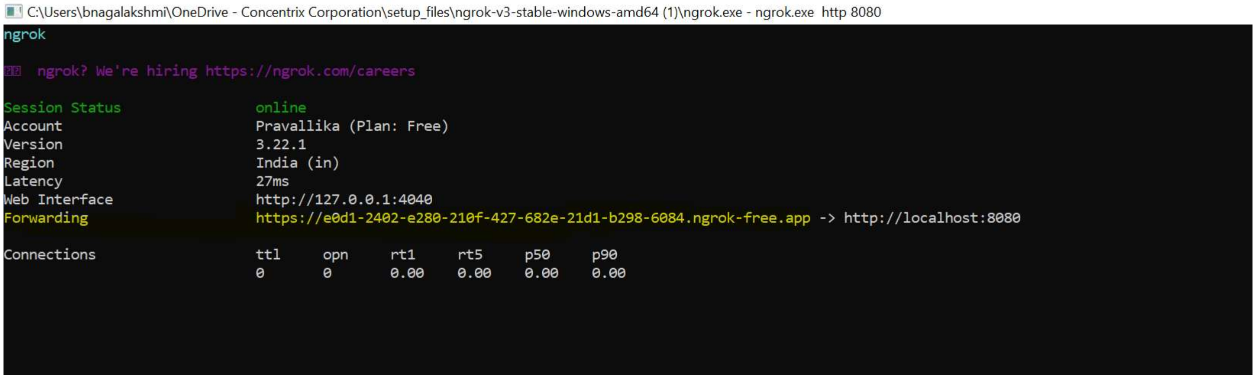


C:\Users\bnagalakshmi\OneDrive - Concentrix Corporation\setup_files\ngrok-v3-stable-windows-amd64 (1)\ngrok.exe

```
EXAMPLES:
  ngrok http 80                                                 # secure public URL for port 80 web server
  ngrok http --url baz.ngrok.dev 8080                           # port 8080 available at baz.ngrok.dev
  ngrok tcp 22                                                  # tunnel arbitrary TCP traffic to port 22
  ngrok http 80 --oauth=google --oauth-allow-email=foo@foo.com  # secure your app with oauth

Paid Features:
  ngrok http 80 --url mydomain.com                              # run ngrok with your own custom domain
  ngrok http 80 --cidr-allow 2600:8c00::a03c:91ee:fe69:9695/32  # run ngrok with IP policy restrictions
  Upgrade your account at https://dashboard.ngrok.com/billing/subscription to access paid features

Upgrade your account at https://dashboard.ngrok.com/billing/subscription to access paid features

Flags:
  -h, --help      help for ngrok

Use "ngrok [command] --help" for more information about a command.

ngrok is a command line application, try typing 'ngrok.exe http 80'
at this terminal prompt to expose port 80.
C:\Users\bnagalakshmi\OneDrive - Concentrix Corporation\setup_files\ngrok-v3-stable-windows-amd64 (1)>
```

- Add the token using **ngrok config add-authtoken <token>**

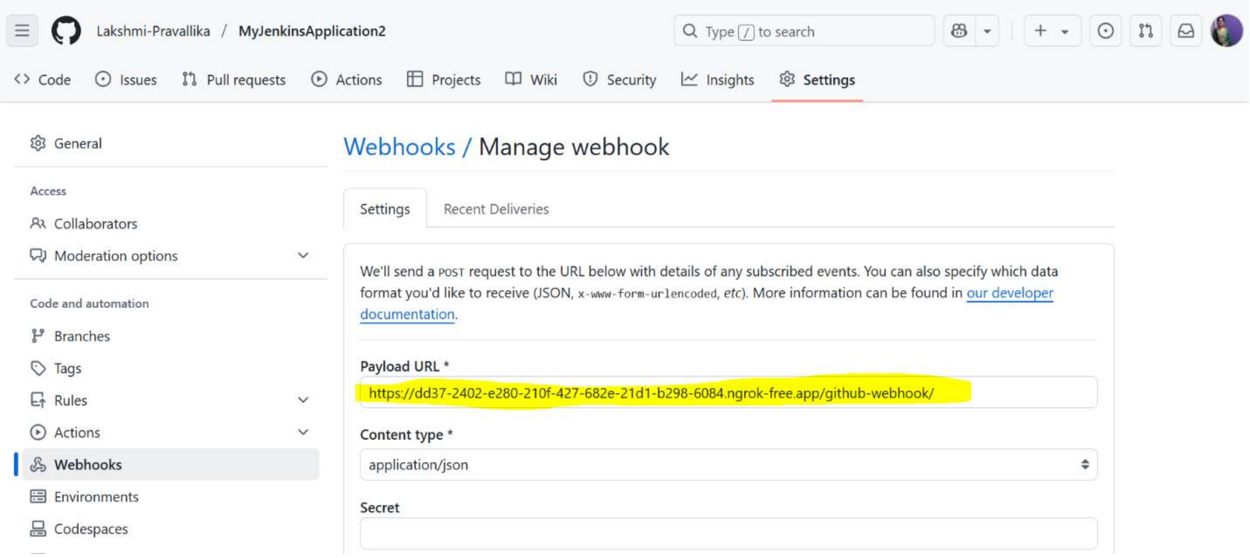
- Start the endpoint on the port on which Jenkins is running.



C:\Users\bnagalakshmi\OneDrive - Concentrix Corporation\setup_files\ngrok-v3-stable-windows-amd64 (1)\ngrok.exe - ngrok.exe http 8080

```
ngrok

   ngrok? We're hiring https://ngrok.com/careers

Session Status                online
Account                       Pravallika (Plan: Free)
Version                       3.22.1
Region                        India (in)
Latency                       27ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://e0d1-2402-e280-210f-427-682e-21d1-b298-6084.ngrok-free.app -> http://localhost:8080

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

- Forwarding is the Jenkin’s URL which is to be added to the Webhook.



Webhooks / Manage webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, *etc*). More information can be found in our developer documentation.

Payload URL *

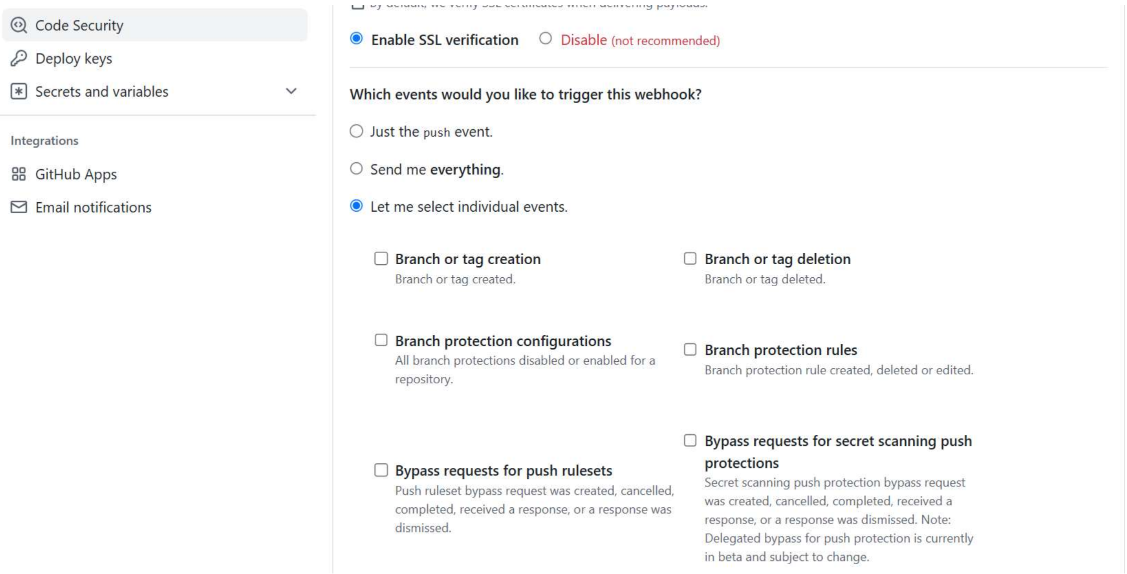
https://dd37-2402-e280-210f-427-682e-21d1-b298-6084.ngrok-free.app/github-webhook/

Content type *

application/json

Secret

- Select the events on which webhook has to be triggered, i.e., Pull Request, Push, Branch creation etc.



● Enable SSL verification ○ Disable (not recommended)

Which events would you like to trigger this webhook?

○ Just the push event.

○ Send me **everything**.

● Let me select individual events.

□ **Branch or tag creation**
Branch or tag created.

□ **Branch or tag deletion**
Branch or tag deleted.

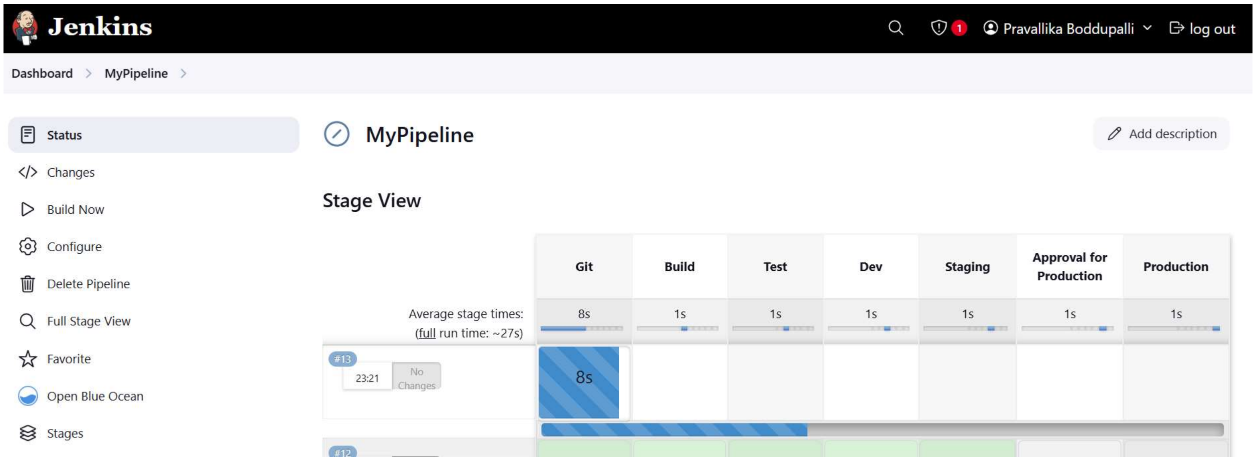
□ **Branch protection configurations**
All branch protections disabled or enabled for a repository.

□ **Branch protection rules**
Branch protection rule created, deleted or edited.

□ **Bypass requests for push rulesets**
Push ruleset bypass request was created, cancelled, completed, received a response, or a response was dismissed.

□ **Bypass requests for secret scanning push protections**
Secret scanning push protection bypass request was created, cancelled, completed, received a response, or a response was dismissed. Note: Delegated bypass for push protection is currently in beta and subject to change.

RUNNING THE PIPELINE BY CLICKING ON BUILD NOW



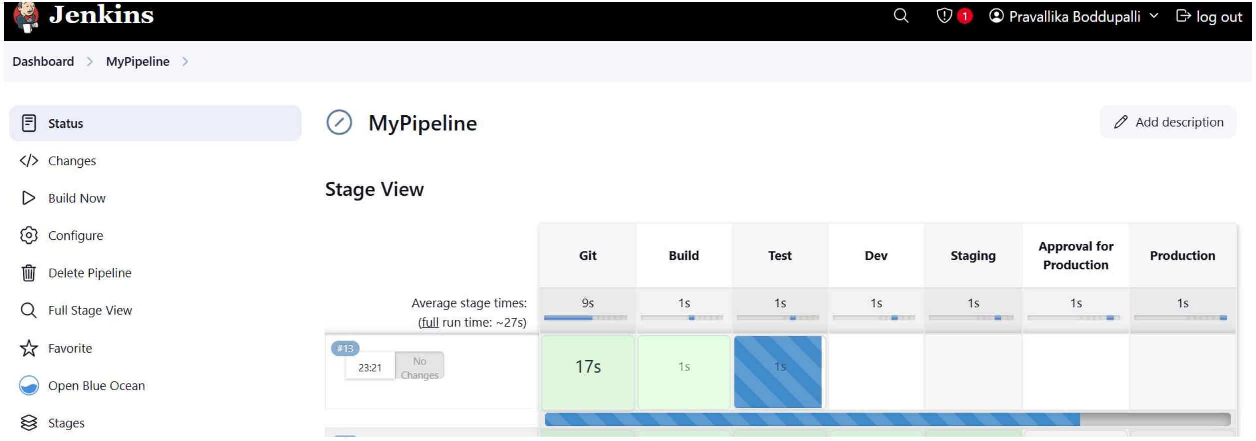
Jenkins — Pravallika Boddupalli — log out

Dashboard > MyPipeline >

MyPipeline

Stage View

	Git	Build	Test	Dev	Staging	Approval for Production	Production
Average stage times: (full run time: ~27s)	8s	1s	1s	1s	1s	1s	1s
#13 23:21 No Changes	8s						



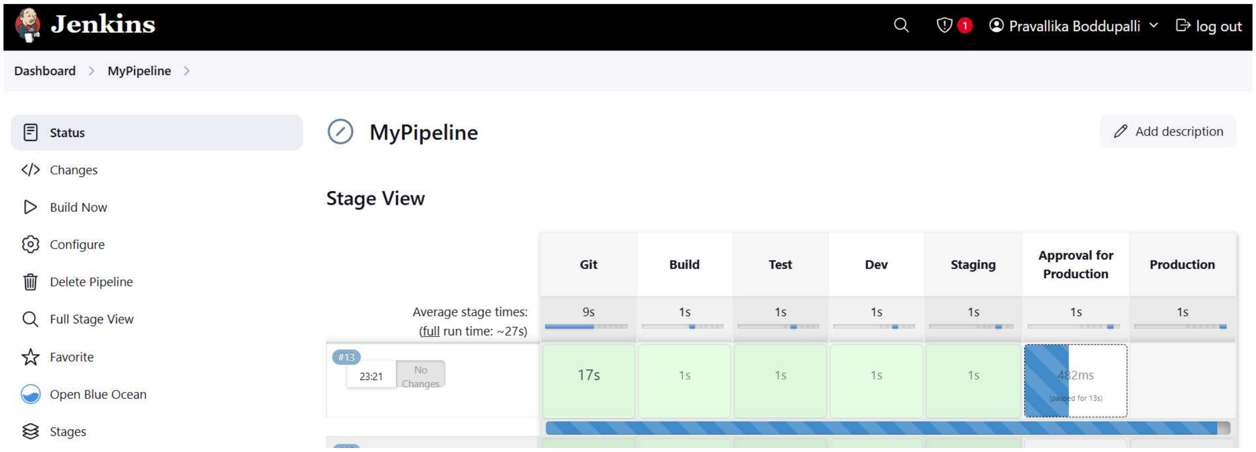
Jenkins — Pravallika Boddupalli — log out

Dashboard > MyPipeline >

MyPipeline

Stage View

	Git	Build	Test	Dev	Staging	Approval for Production	Production
Average stage times: (full run time: ~27s)	9s	1s	1s	1s	1s	1s	1s
#13 23:21 No Changes	17s	1s	1s				



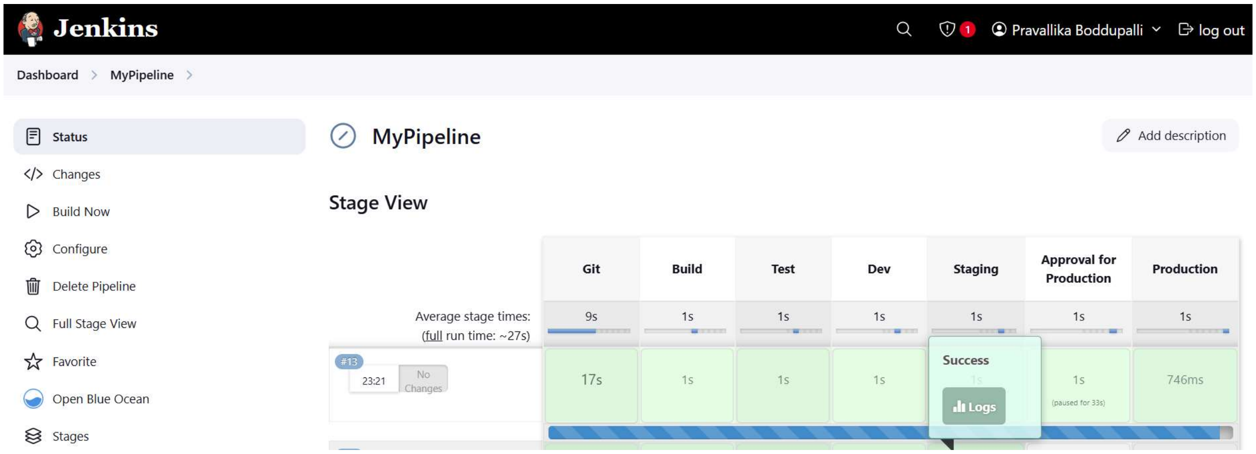
Jenkins — Pravallika Boddupalli — log out

Dashboard > MyPipeline >

MyPipeline

Stage View

	Git	Build	Test	Dev	Staging	Approval for Production	Production
Average stage times: (full run time: ~27s)	9s	1s	1s	1s	1s	1s	1s
#13 23:21 No Changes	17s	1s	1s	1s	1s	482ms (paused for 13s)	



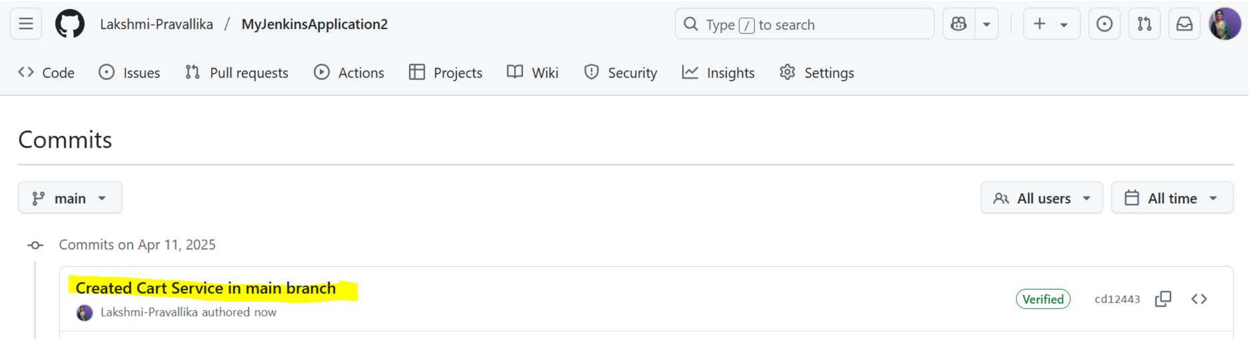
MyPipeline

Stage View

	Git	Build	Test	Dev	Staging	Approval for Production	Production
Average stage times: (full run time: ~27s)	9s	1s	1s	1s	1s	1s	1s
#13 23:21 No Changes	17s	1s	1s	1s	Success 1s Logs	1s (paused for 33s)	746ms

PUSHING CODE TO MAIN BRANCH TO TRIGGER AUTO BUILD THROUGH WEBHOOK

1) Make a commit into the main branch.
2) Created Cart Service file.

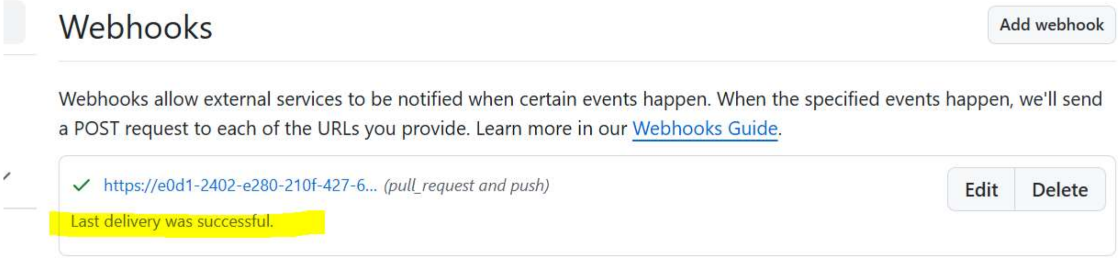


Commits

Commits on Apr 11, 2025

Created Cart Service in main branch
Lakshmi-Pravallika authored now
Verified cd12443

The webhook has triggered successfully.



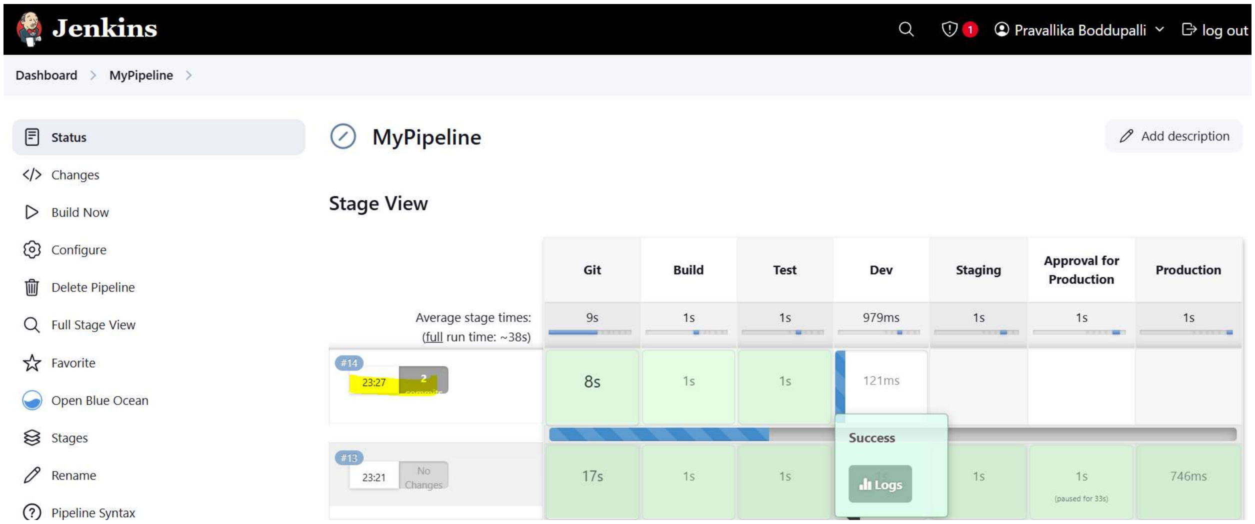
Webhooks

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.

✓ https://e0d1-2402-e280-210f-427-6... *(pull_request and push)*

Last delivery was successful.

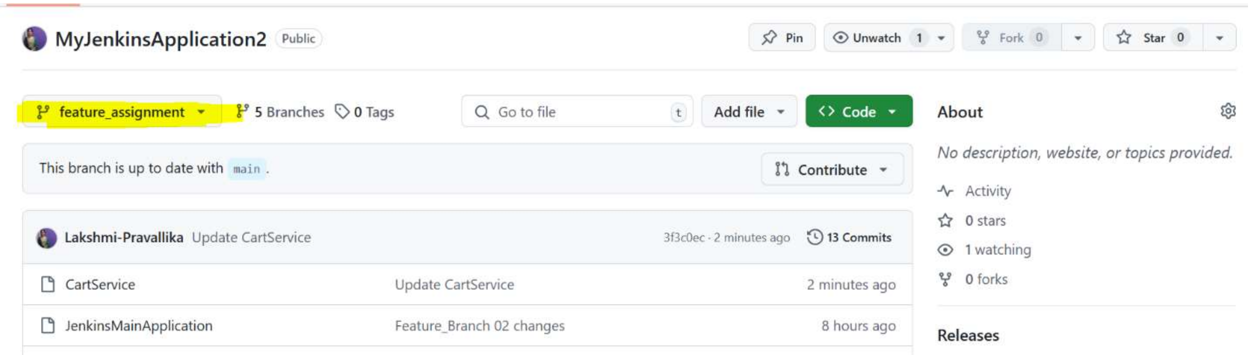
Build automatically started with commits



	Git	Build	Test	Dev	Staging	Approval for Production	Production
Average stage times: (full run time: ~38s)	9s	1s	1s	979ms	1s	1s	1s
#14 23:27 2 commits	8s	1s	1s	121ms			
#13 23:21 No Changes	17s	1s	1s	1s	1s	1s (paused for 33s)	746ms

CREATING A NEW FEATURE BRANCH AND MERGING IT TO MAIN TO TRIGGER AUTO BUILD THROUGH WEBHOOK

1) Created a new feature branch named “feature_assignment”



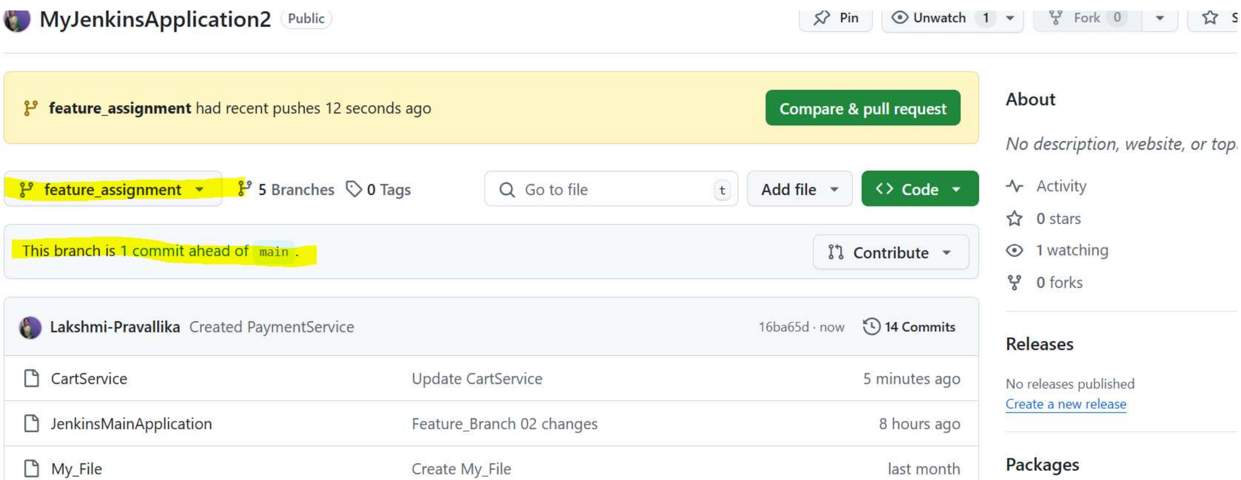
No description, website, or topics provided.

This branch is up to date with `main` .

Lakshmi-Pravallika Update CartService 3f3c0ec · 2 minutes ago 13 Commits

CartService	Update CartService	2 minutes ago
JenkinsMainApplication	Feature_Branch 02 changes	8 hours ago

2) Commit something to the feature branch. Added Payment Service



feature_assignment had recent pushes 12 seconds ago

No description, website, or top

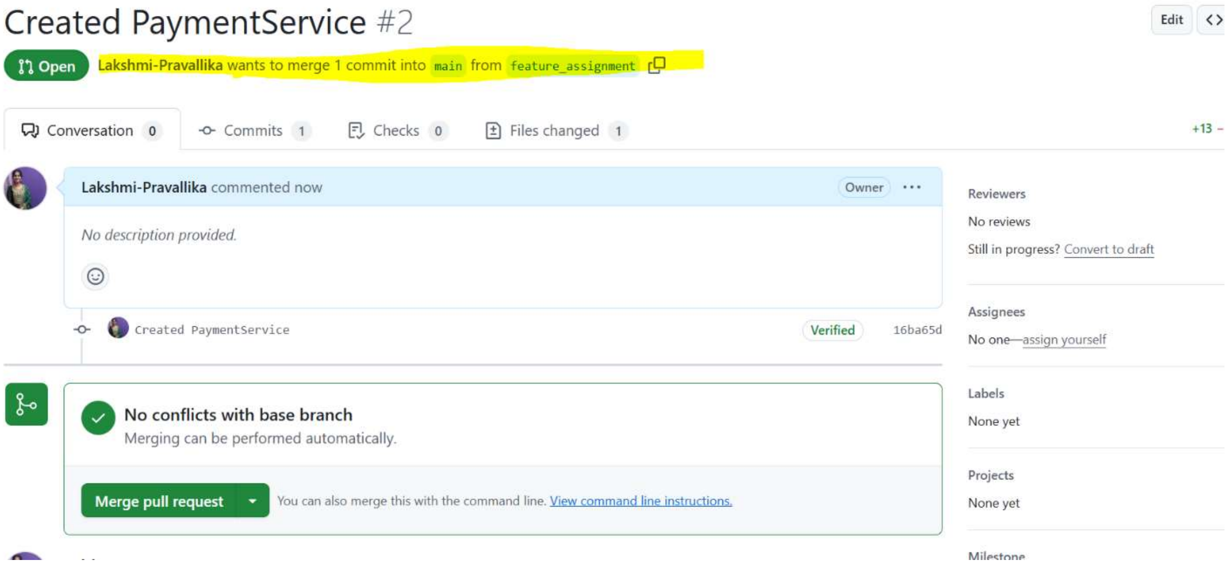
This branch is 1 commit ahead of `main` .

No releases published

Lakshmi-Pravallika Created PaymentService 16ba65d · now 14 Commits

CartService	Update CartService	5 minutes ago
JenkinsMainApplication	Feature_Branch 02 changes	8 hours ago
My_File	Create My_File	last month

3) Create a Pull Request



Created PaymentService #2

Open Lakshmi-Pravallika wants to merge 1 commit into main from feature_assignment

Conversation 0 · Commits 1 · Checks 0 · Files changed 1 · +13

Lakshmi-Pravallika commented now · Owner

No description provided.

Created PaymentService · Verified · 16ba65d

No conflicts with base branch

Merging can be performed automatically.

Merge pull request · You can also merge this with the command line. View command line instructions.

Reviewers: No reviews

Still in progress? Convert to draft

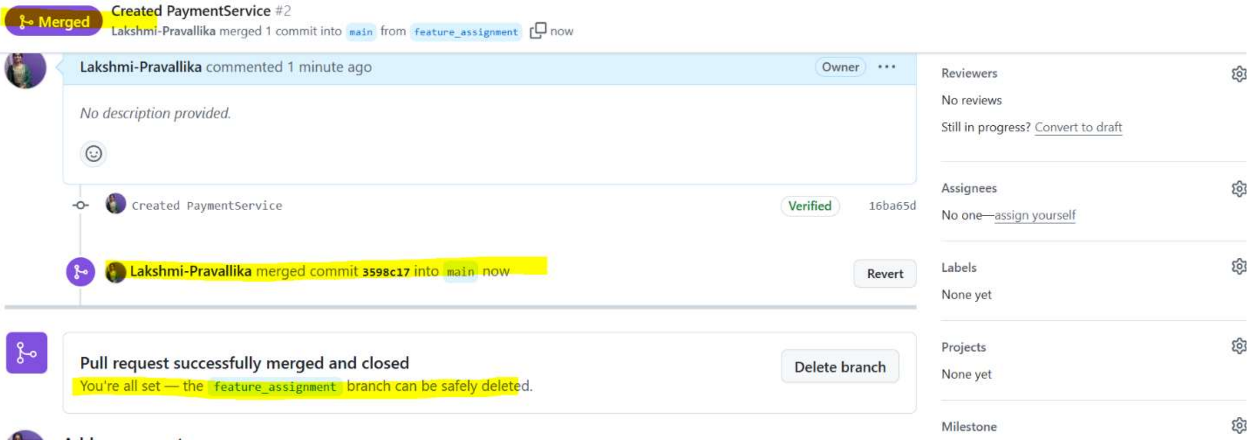
Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone

4) Merge Pull request to Main branch



Merged · Created PaymentService #2

Lakshmi-Pravallika merged 1 commit into main from feature_assignment now

Lakshmi-Pravallika commented 1 minute ago · Owner

No description provided.

Created PaymentService · Verified · 16ba65d

Lakshmi-Pravallika merged commit 3598c17 into main now

Pull request successfully merged and closed

You're all set — the feature_assignment branch can be safely deleted.

Reviewers: No reviews

Still in progress? Convert to draft

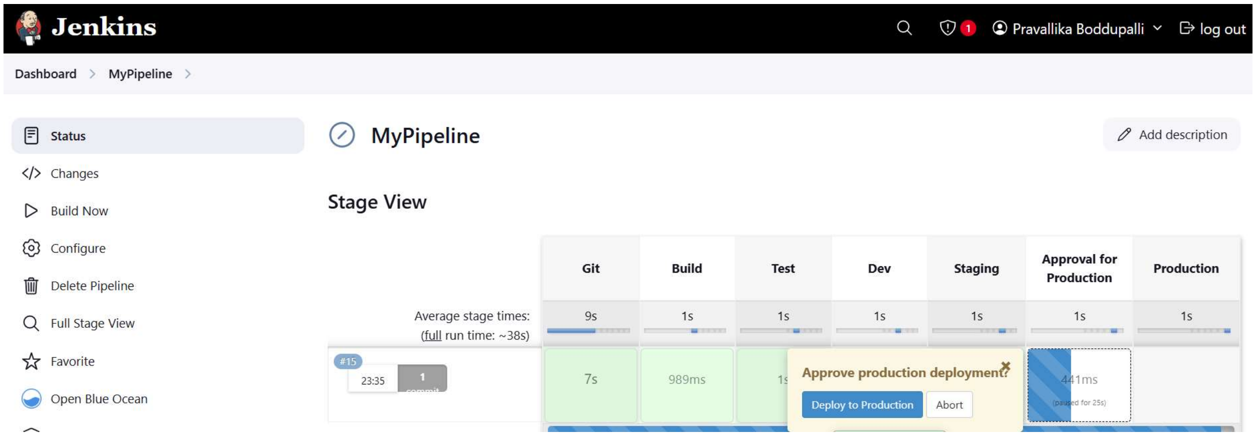
Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone

5) Automatic build started



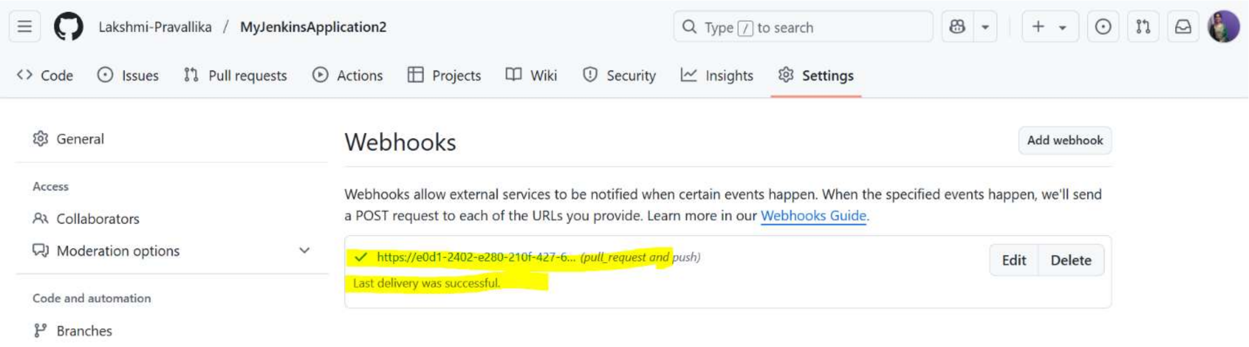
MyPipeline

Stage View

	Git	Build	Test	Dev	Staging	Approval for Production	Production
Average stage times: (full run time: ~38s)	9s	1s	1s	1s	1s	1s	1s
#15 23:35 1	7s	989ms	1s			441ms (paused for 25s)	

Approve production deployment?

6) Webhook triggered successfully



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.

✓ https://e0d1-2402-e280-210f-427-6... (pull_request and push)

Last delivery was successful.

ASSIGNMENT – 2

Case Study A – Scaling a Healthcare Analytics SaaS Product with DevOps

Overview

The healthcare analytics product is designed to perform data loading, cleaning, pre-processing, AI-driven report generation, and exporting reports in formats like PDF and Word. It also includes mailing system integration to automatically distribute reports. With the recent investment, the demand has scaled significantly—over 100 stakeholders, each generating around 1 lakh reports per month. This demands high availability, error-free processing, and performance under heavy load.

DevOps for Faster Delivery and Scaling

To meet this level of scale and reliability, DevOps plays a key role in streamlining the product delivery and maintenance processes. The first step involves containerizing the application using Docker, then deploying and scaling it through Kubernetes. This ensures better fault tolerance, ease of deployment, and resource optimization across different environments.

Continuous Integration and Delivery (CI/CD)

A CI/CD pipeline is essential to automate the build, test, and deployment processes. Tools like Jenkins or GitHub Actions can be used to integrate these pipelines, ensuring that every code change is thoroughly tested and deployed without manual intervention. This also enables rapid feature delivery and consistent release cycles.

Load Balancing and Distributed Processing

Given the huge volume of reports, it's critical to implement load balancing and distributed data processing. Using Kubernetes' Horizontal Pod Autoscaler in combination with tools like Apache Spark allows for parallel execution of heavy data tasks. Load balancers like NGINX ensure that the application can handle incoming traffic efficiently.

Monitoring, Logging, and Security

To ensure performance and uptime, monitoring solutions like Prometheus and Grafana are used to track metrics in real time. The ELK Stack can be implemented for comprehensive logging and issue diagnosis. Security is handled through role-based access control (RBAC), encrypted secrets, and vulnerability scanning tools like SonarQube.

Report Distribution and Automation

The mailing system can be automated using services like Amazon SES or SendGrid. These integrate well with pipelines to ensure generated reports are dispatched without delays. Batch processing and message queues (like RabbitMQ or Kafka) further help manage high volumes of outgoing reports.

Case Study B – Resolving Product Delivery Challenges in a Global Team

Overview

The company is facing severe product delivery delays, and there's a high risk of losing 20% of its customer base. The issues are rooted in outdated nightly build practices and a geographically distributed team with members in the Netherlands, America, and India. These challenges have resulted in poor collaboration, delayed feedback loops, and missed delivery deadlines.

Transition from Nightly Builds to CI/CD

One of the major pain points is the use of nightly builds, which delay the detection of issues and create bottlenecks in the delivery pipeline. The solution is to move to continuous integration, where every code change is built and tested immediately. Combined with continuous delivery or deployment, this allows the team to push tested changes to production frequently and reliably.

Improving Collaboration Across Time Zones

With the team spread across continents, establishing effective communication and workflow is essential. Adopting agile methodologies along with collaboration tools like Jira, Confluence, and Slack can greatly enhance transparency and coordination. Regular stand-ups, sprint planning, and retrospective meetings—held asynchronously or in rotating shifts—ensure that all members stay aligned regardless of time zone.

Automation and Infrastructure Management

To reduce human error and increase consistency, infrastructure and testing must be automated. Infrastructure-as-code tools like Terraform or Ansible can be used to manage deployments across environments. Automated test cases (unit, integration, and regression) integrated into the CI/CD pipeline further ensure stability and speed.

Monitoring and Feedback Loops

Introducing real-time monitoring and logging helps the team stay informed about system performance and issues. Tools like Grafana, Prometheus, and centralized logging systems ensure that problems are identified and resolved quickly, minimizing downtime and user complaints.

Results and Business Impact

By adopting these DevOps strategies, the company can overcome its delivery backlog, improve team efficiency, and accelerate product releases. This not only enhances product quality and stability but also rebuilds customer confidence, reducing churn and supporting long-term business growth.