

ASSIGNMENT – 1

Name: B N L Pravallika Id: 2024tm93314

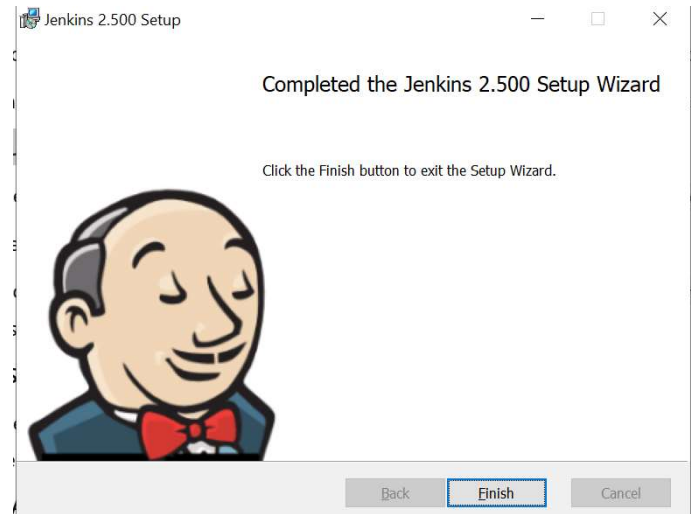
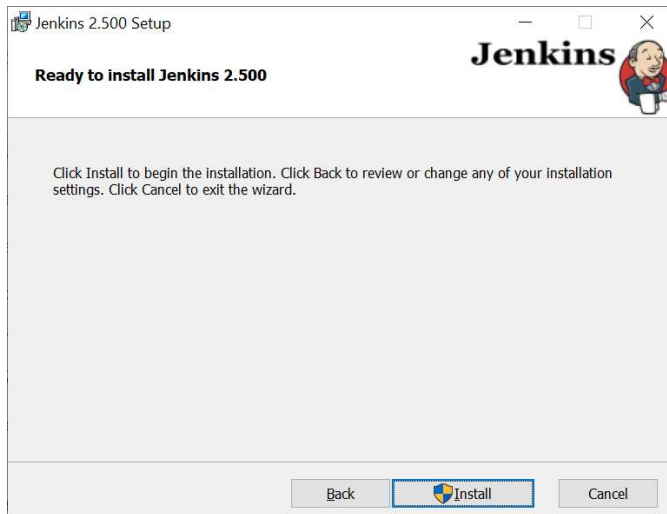
INSTALL JENKINS ON THE MASTER NODE (WINDOWS)

Step 1: Download Jenkins

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

Step 2: Install Jenkins

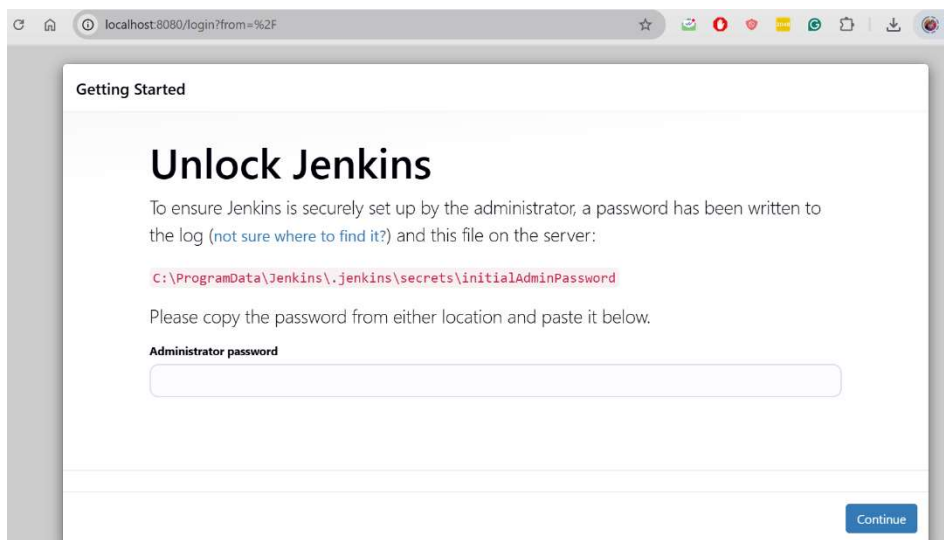
Run the .msi file and follow the installation steps. Choose the default installation option and keep the default Jenkins installation directory.



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.



Getting Started

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


Install plugins the Jenkins community finds most useful.

Select and install plugins most suitable for your needs.

Getting Started

Folders Plugin	Swagger Markup Formatter Plugin	Build-Invoker	SecureCodeBringing Plugin	Git Client
Timestampers	<input type="radio"/> Workspace Cleanup Plugin	<input type="radio"/> Ant Plugin	<input type="radio"/> Grade Plugin	Git ** Github Github Branch Source Pipeline: Github Groovy Libraries ** Pipeline Graph Analysis ** Metrics Pipeline Graph View Git ** ODS4 API ** Trilead API SSH Build Agents Matrix Authorization Strategy PAM Authentication LDAP
Pipeline	<input type="radio"/> Github Branch Source Plugin	<input type="radio"/> Pipeline: Github Groovy Libraries	<input type="radio"/> Pipeline Graph View Plugin	
Git plugin	<input type="radio"/> SSH Build Agents plugin	<input type="radio"/> Matrix Authorization Strategy Plugin	<input type="radio"/> PAM Authentication plugin	
LDAP Plugin	<input type="radio"/> Email Extension Plugin	<input type="radio"/> Mailer Plugin	<input type="radio"/> Dark Theme	

lanxing 2 402 2

**Jenkins**

🔍

🛡️ 1

👤 Pravallika Boddupalli ▾

🔗 log out

Dashboard ▸

+ New Item

📁 Build History

⚙️ Manage Jenkins

📌 My Views

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 ☁️

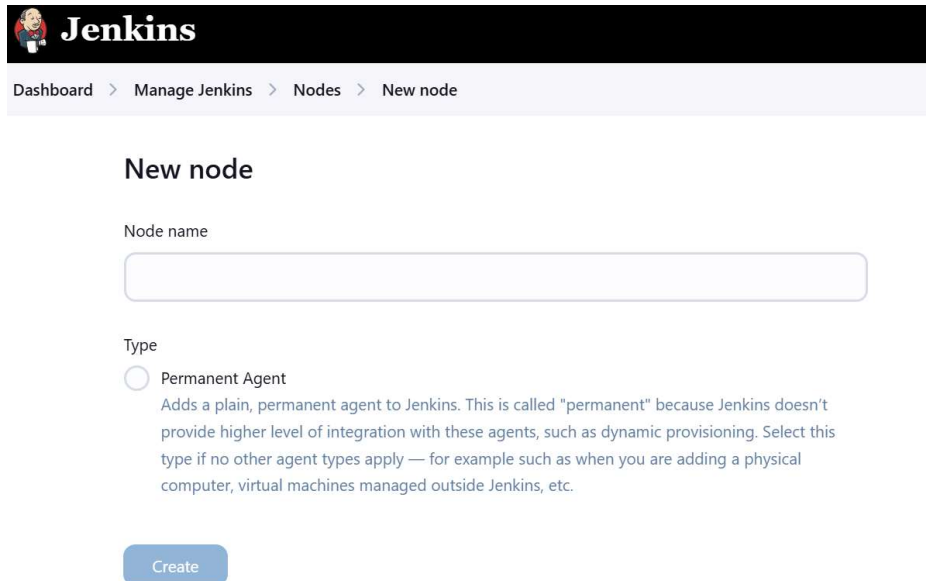
Learn more about distributed builds 📖

SETUP SLAVE NODE (WINDOWS)

Step :1 Enable Remote Access on Jenkins:

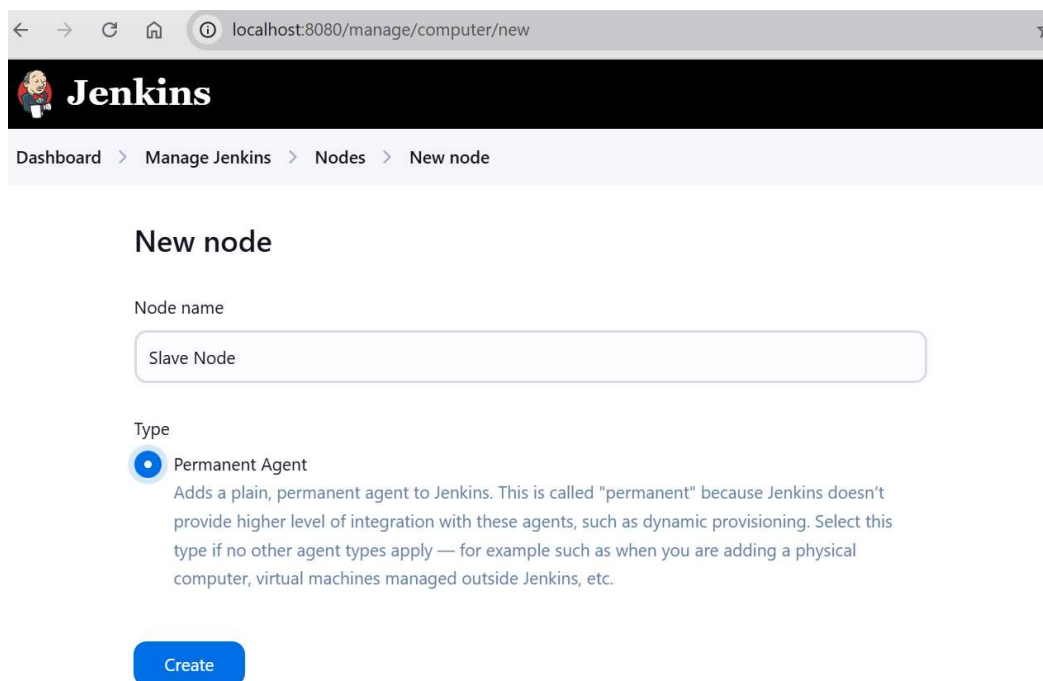
a) Creating a new Node

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



The image shows the Jenkins 'New node' form. At the top is the Jenkins logo and a breadcrumb trail: Dashboard > Manage Jenkins > Nodes > New node. The main heading is 'New node'. Below it is a 'Node name' label followed by an empty text input field. Underneath is a 'Type' label with a radio button selected for 'Permanent Agent'. A descriptive text follows: '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.' At the bottom is a blue 'Create' button.

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



This image shows the same Jenkins 'New node' form as above, but with the 'Node name' field filled with 'Slave Node'. The 'Permanent Agent' radio button remains selected. The 'Create' button is still at the bottom.

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**.

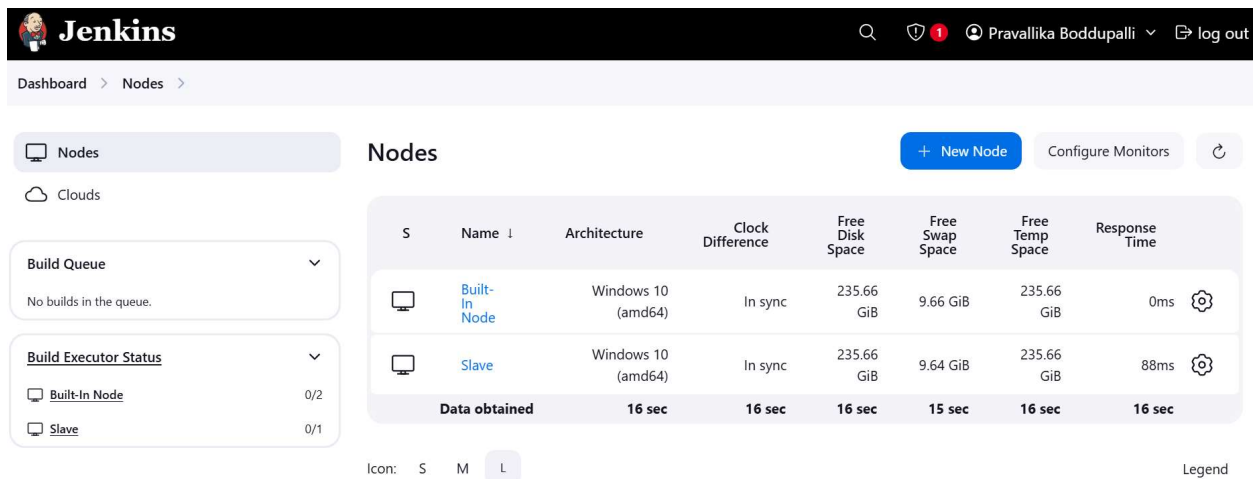
c) Connect to Slave Node:

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 142d8319d1f229ddae6d065d7eb3e1d36d600d5fc37f4a07a273f6c74311b150 -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.



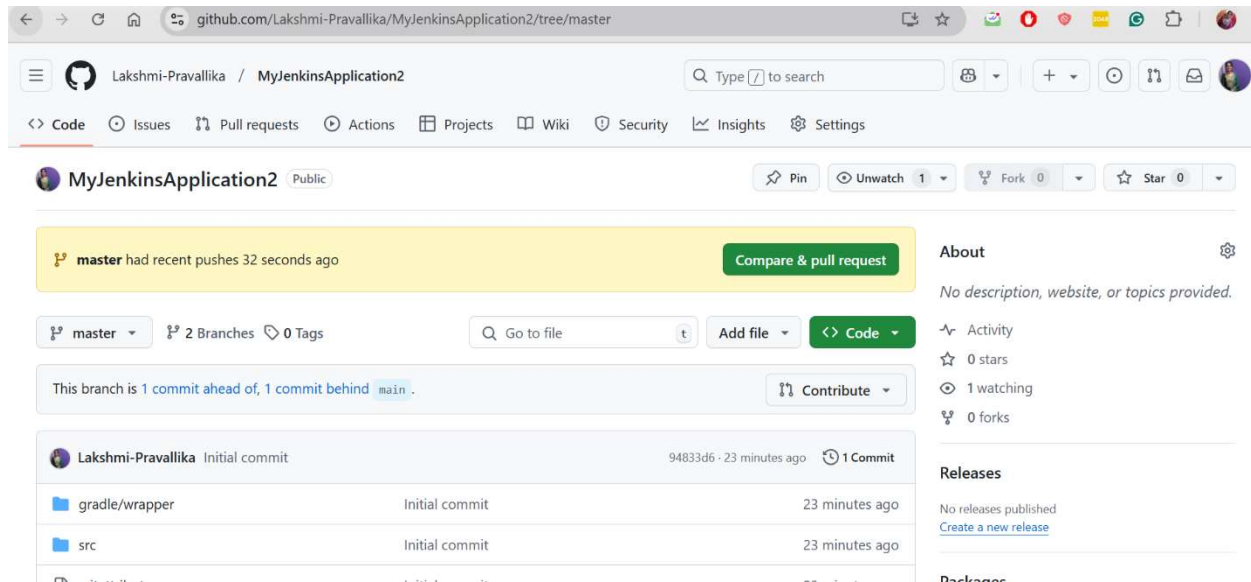
Jenkins Dashboard > Nodes >

Nodes + New Node Configure Monitors

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

Icon: S M L Legend

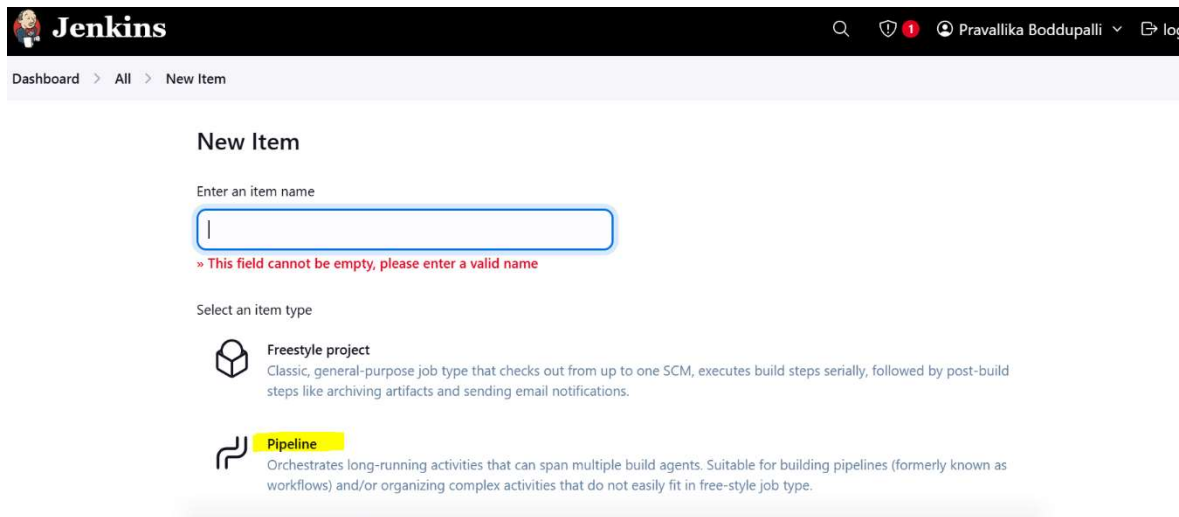
Create a Git Repository



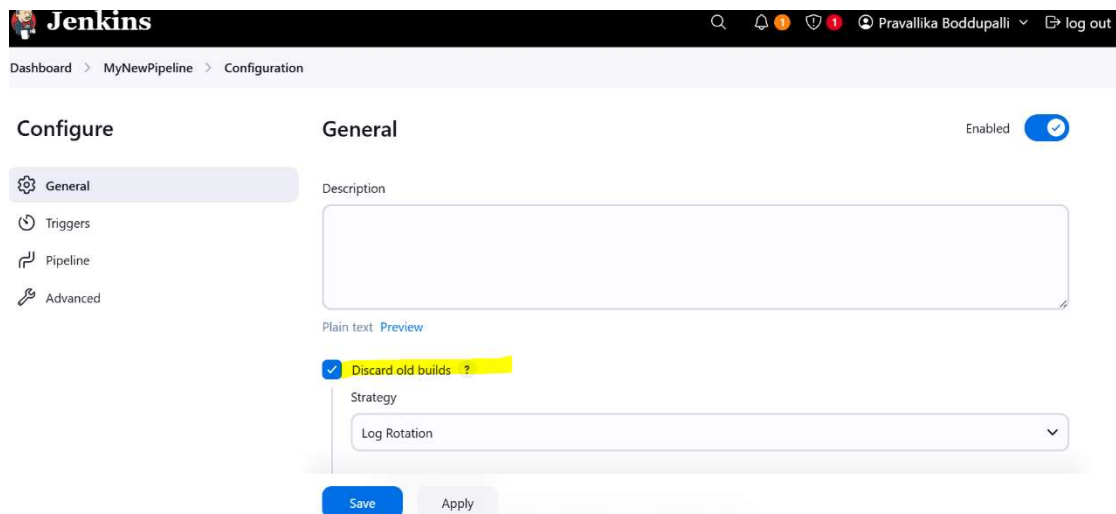
CREATING JENKINS PIPELINE

Create a New Pipeline Project

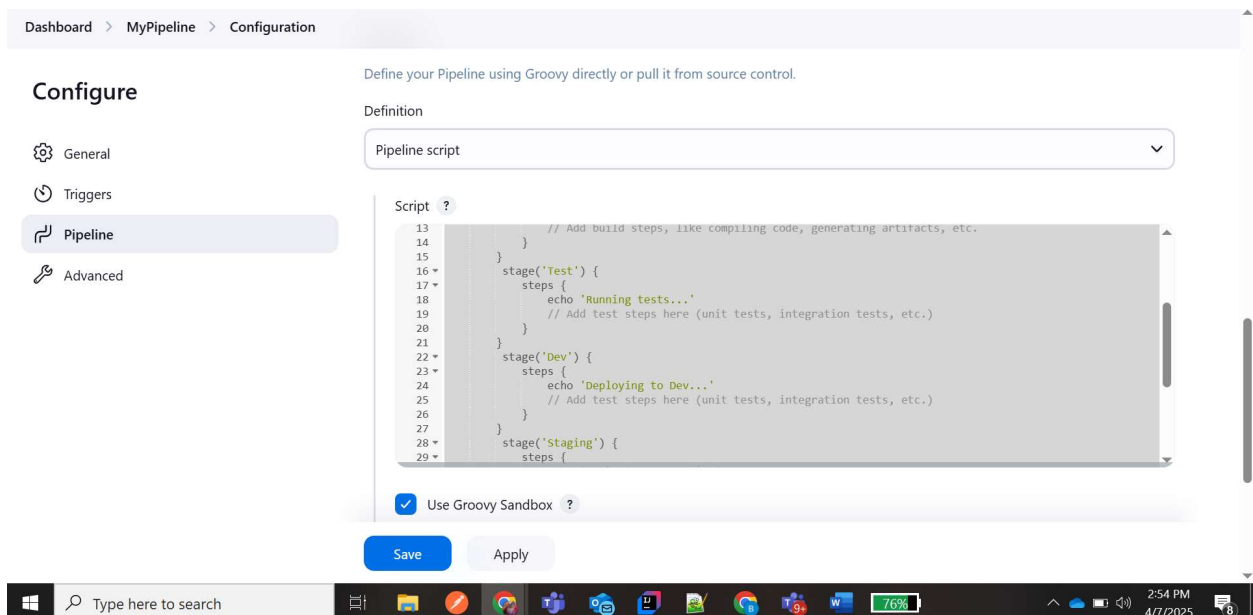
- Go to Jenkins → **New Item**.
- Select "Pipeline" and give it a name (e.g., MyPipeline).
- Choose "Pipeline" under "Project" and click "OK."



2) Select discard old builds.



3) Define Pipeline script



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

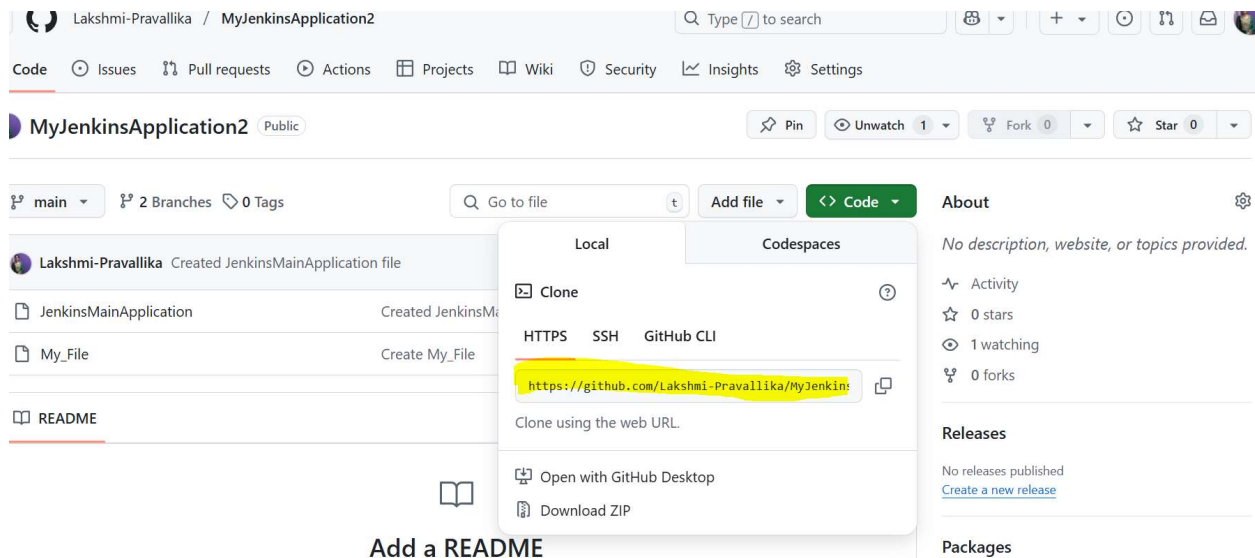
1) Click on Pipeline Syntax



2) Select GIT and enter repository URL

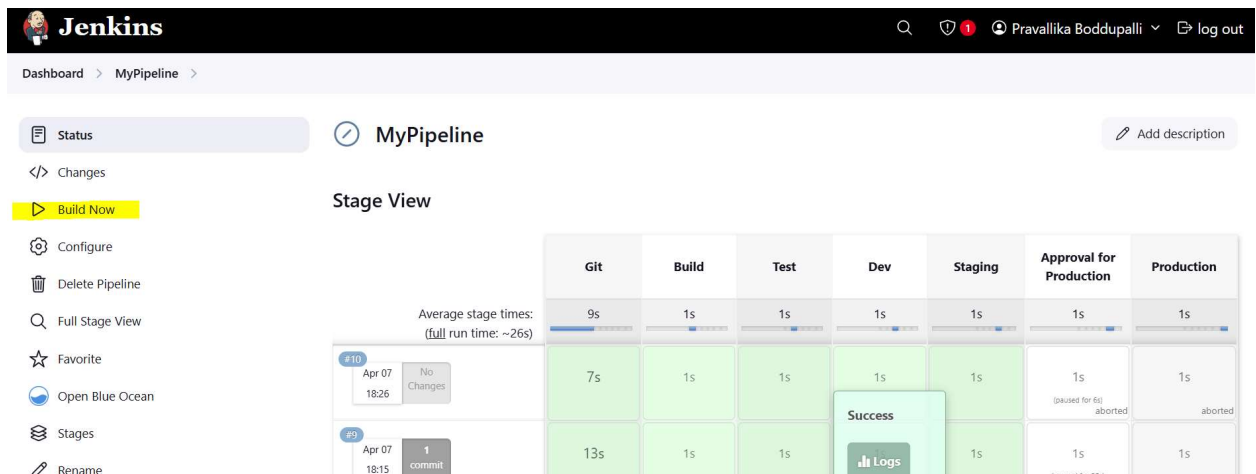
The screenshot shows the Jenkins Pipeline Snippet Generator interface. On the left is a sidebar with navigation links: 'Snippet Generator', 'Active Directive Generator', 'Active Online Documentation', 'Reference', 'Variables Reference', 'Documentation', 'Files Reference', and 'IDEA GDSL'. The main area is titled 'OVERVIEW' and contains a description of the Snippet Generator. Below this, under the 'Steps' section, there is a 'Sample Step' dropdown menu set to 'git: Git'. Below the dropdown are three input fields: 'Repository URL' with the value 'https://github.com/Lakshmi-Pravallika/MyJenkinsApplication2.git', and 'Branch' with the value 'main'.

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



3) Click on Save.

4) Click on Build Now



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.

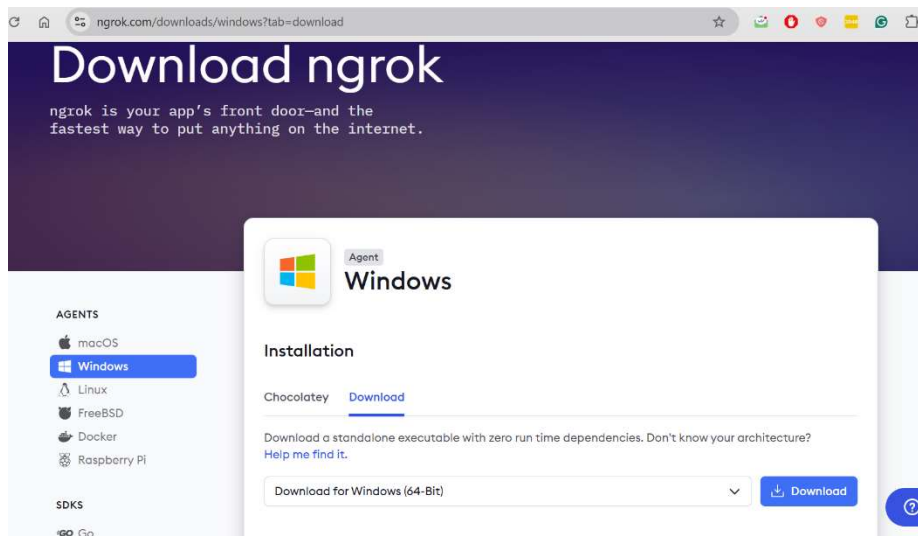
INTEGRATING GIT WITH JENKINS FOR AUTO BUILD TRIGGER

- 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.



- 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.

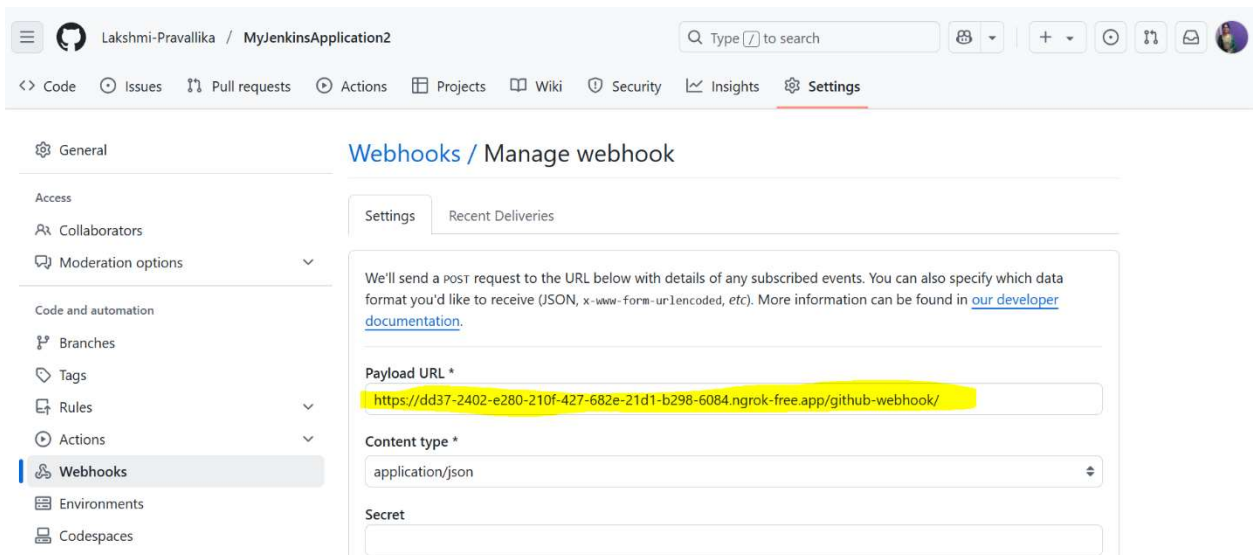
ngrok is a command line application, try typing 'ngrok.exe http 80'
at this terminal prompt to expose port 80.
C:\Users\bnagalakshmi\OneDrive - Concentrix Corporation\setup_files\ngrok-v3-stable-windows-amd64 (1)>
```

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

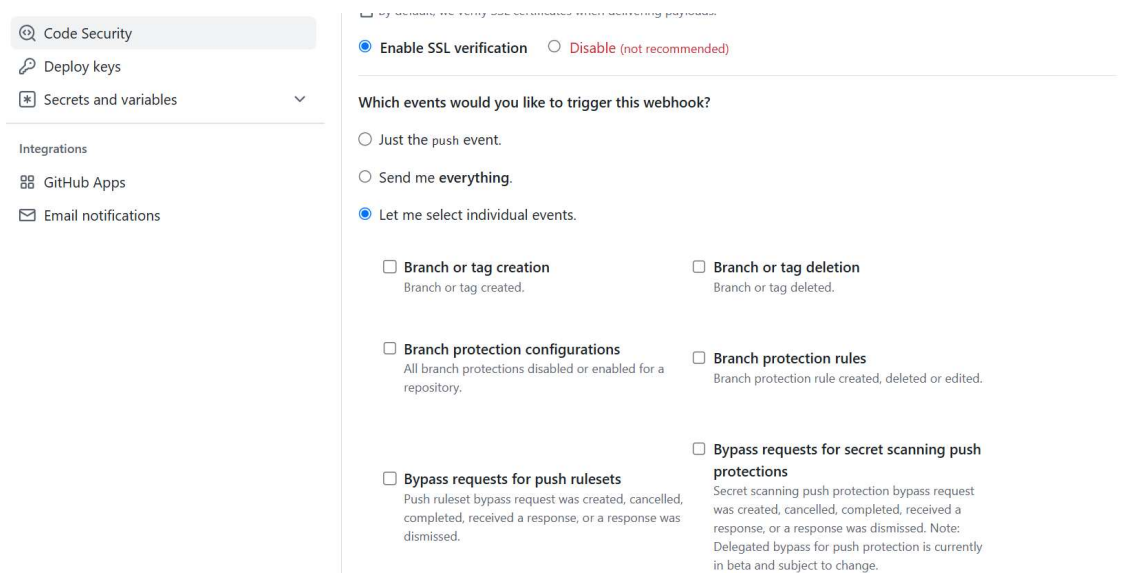
- Start the endpoint on the port on which Jenkins is running.

```
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.



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



RUNNING THE PIPELINE BY CLICKING ON BUILD NOW

 **Jenkins**

Dashboard > MyPipeline >

Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Favorite

Open Blue Ocean

Stages


MyPipeline

Add description

Stage View

Average stage times:
(full run time: ~27s)

	Git	Build	Test	Dev	Staging	Approval for Production	Production
#15 23:21 No Changes	8s	1s	1s	1s	1s	1s	1s

 **Jenkins**

Dashboard > MyPipeline >

Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Favorite

Open Blue Ocean

Stages


MyPipeline

Add description

Stage View

Average stage times:
(full run time: ~27s)

	Git	Build	Test	Dev	Staging	Approval for Production	Production
#15 23:21 No Changes	9s	1s	1s	1s	1s	1s	1s

 **Jenkins**

Dashboard > MyPipeline >

Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Favorite

Open Blue Ocean

Stages

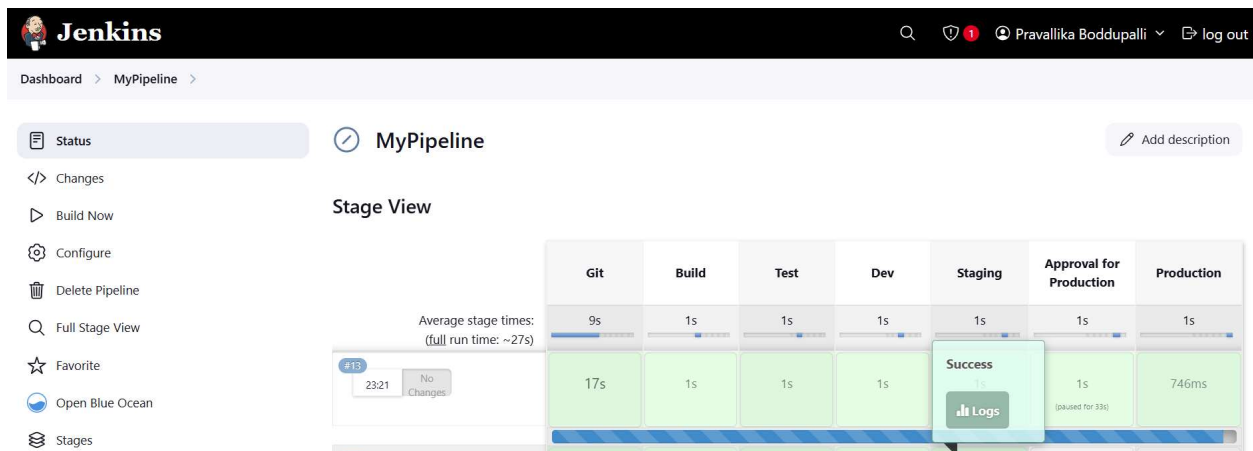
MyPipeline

Add description

Stage View

Average stage times:
(full run time: ~27s)

	Git	Build	Test	Dev	Staging	Approval for Production	Production
#15 23:21 No Changes	9s	1s	1s	1s	1s	462ms (skipped for 13s)	1s



PUSHING CODE TO MAIN BRANCH TO TRIGGER AUTO BUILD THROUGH WEBHOOK

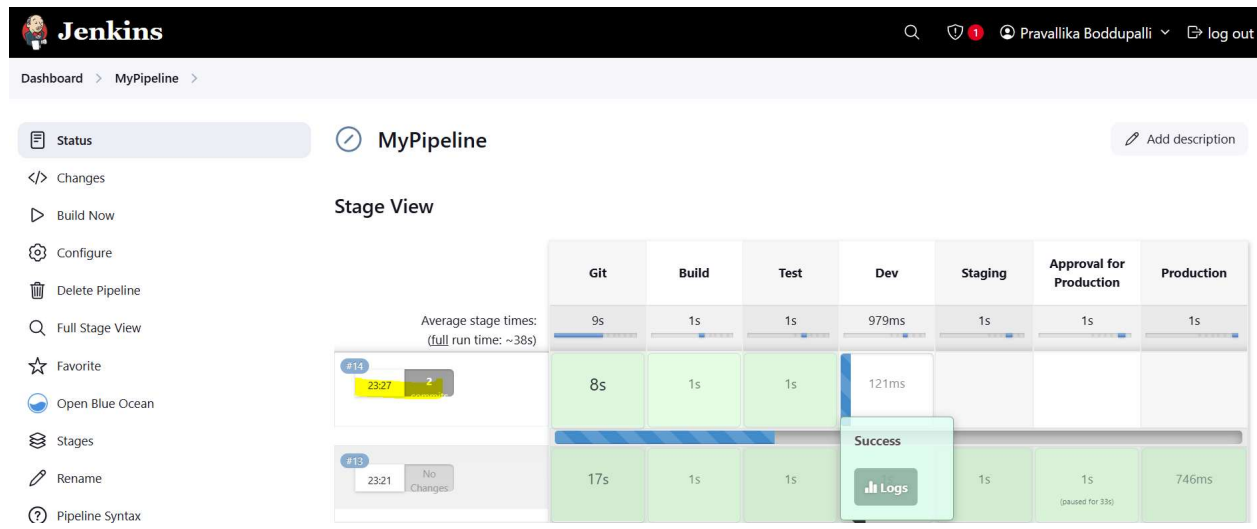
- 1) Make a commit into the main branch.
- 2) Created Cart Service file.

The GitHub repository view shows the 'main' branch. The commit history is displayed, with the most recent commit being 'Created Cart Service in main branch' by Lakshmi-Pravallika, authored now. The commit is marked as 'Verified' and has a commit ID of 'cd12443'. The commit message is highlighted in yellow.

The webhook has triggered successfully.

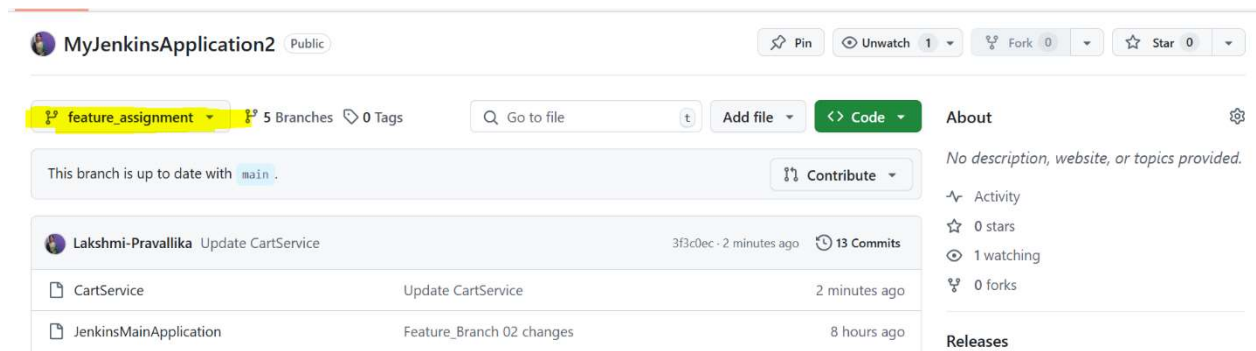
The GitHub Webhooks configuration page shows the 'Add webhook' button. Below the button, the text states: 'Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).' The 'Add webhook' button is highlighted in yellow. Below the text, a list of webhooks is shown. The first webhook is 'https://e0d1-2402-e280-210f-427-6...' (pull_request and push). The 'Last delivery was successful.' message is highlighted in yellow. The 'Edit' and 'Delete' buttons are visible next to the webhook.

Build automatically started with commits

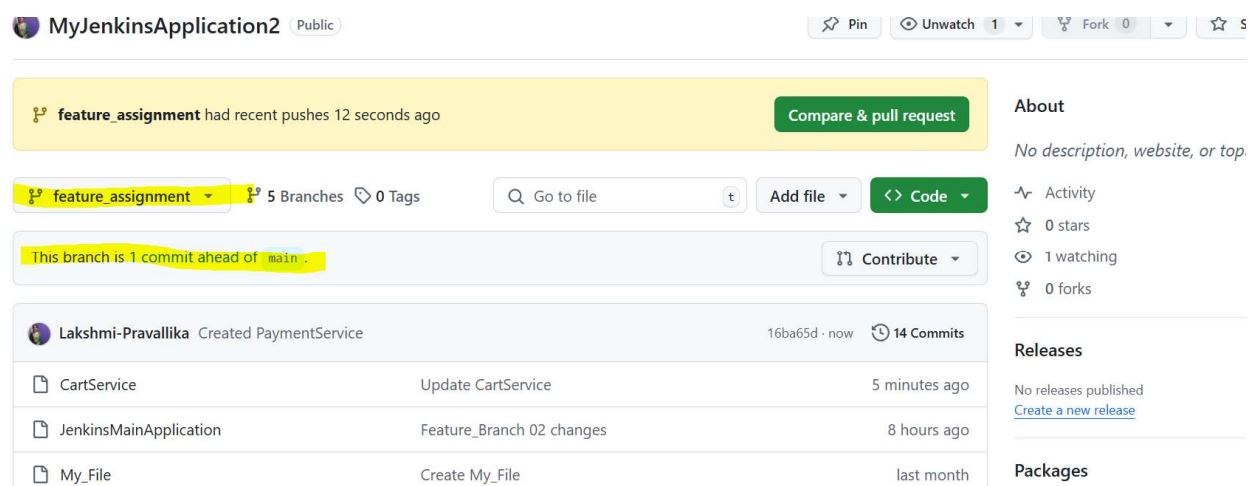


CREATING A NEW FEATURE BRANCH AND MERGING IT TO MAIN TO TRIGGER AUTO BUILD THROUGH WEBHOOK

1) Created a new feature branch named “feature_assignment”



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



3) Create a Pull Request


Created PaymentService #2 Edit <>


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 Revert

 Pull request successfully merged and closed
You're all set — the `feature_assignment` branch can be safely deleted. Delete branch

Reviewers
No reviews
Still in progress? [Convert to draft](#)

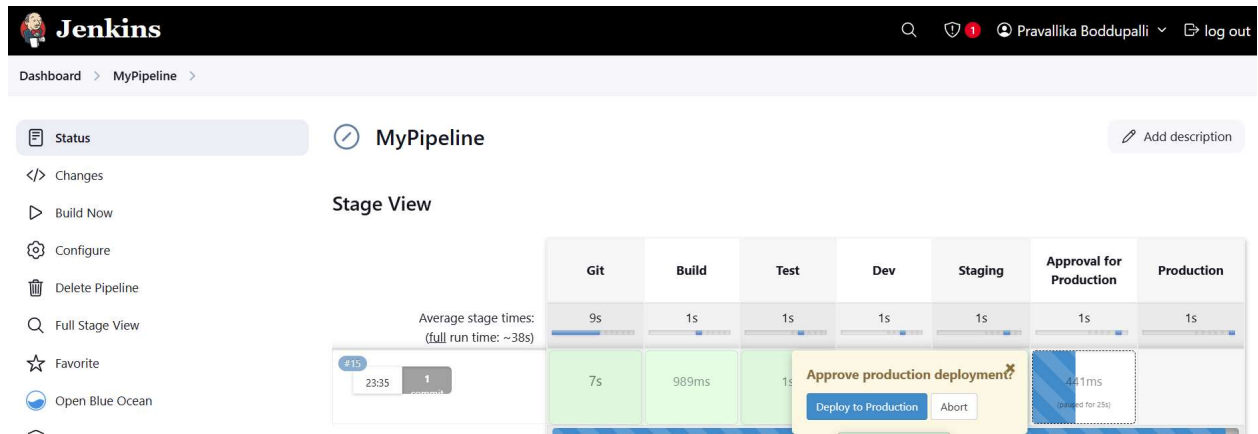
Assignees
No one—[assign yourself](#)

Labels
None yet

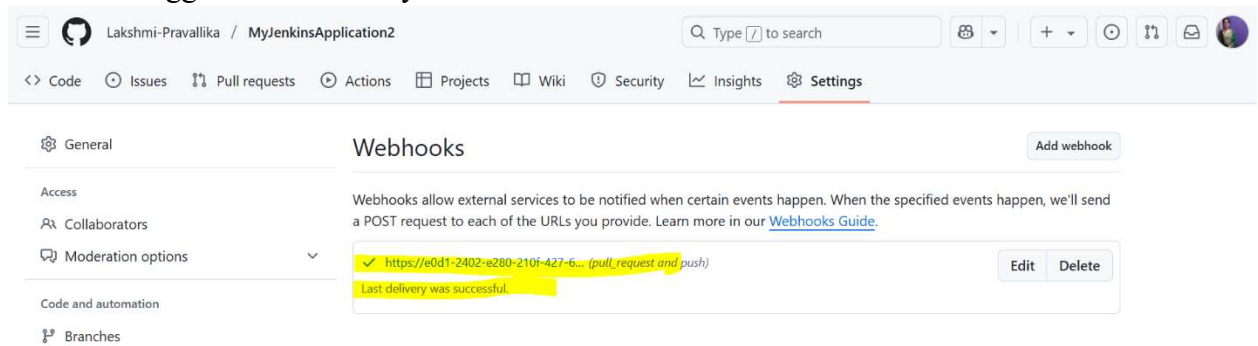
Projects
None yet

Milestone

5) Automatic build started



6) Webhook triggered successfully



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.