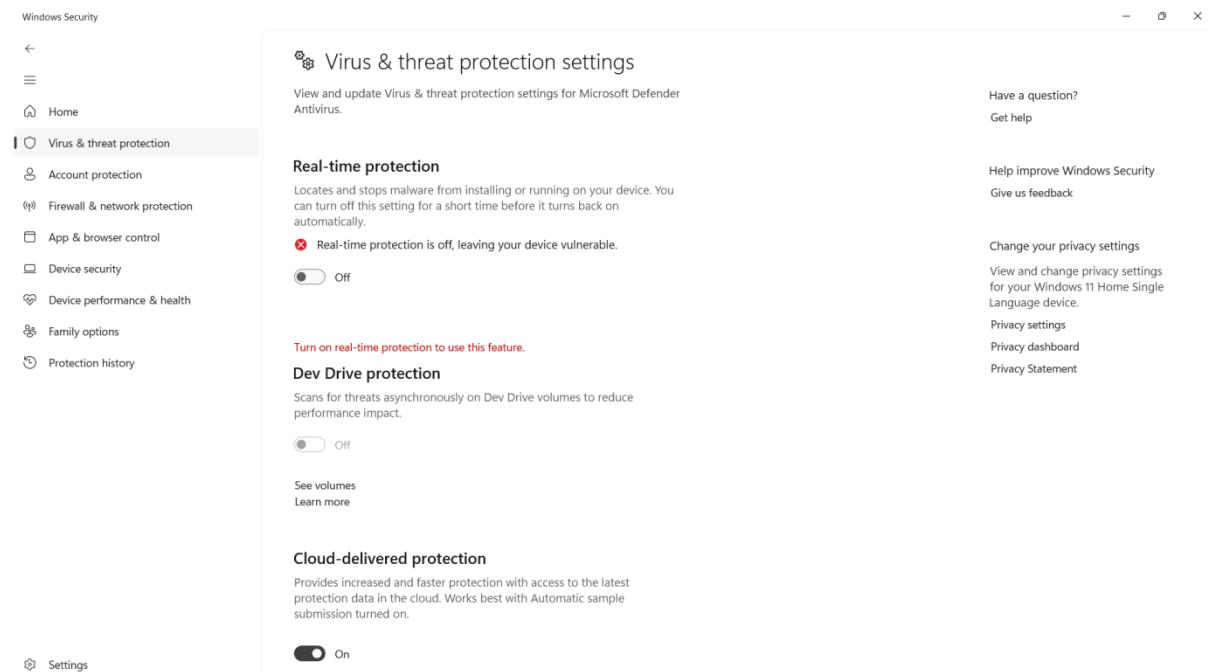


## WEEK 11

### Working on windows 11

#### Exercise 1: Jenkins CI/CD using Git Webhook

Step-1: To install ngroks Go->settings->privacy and security -> windows security -> off antivirus ->

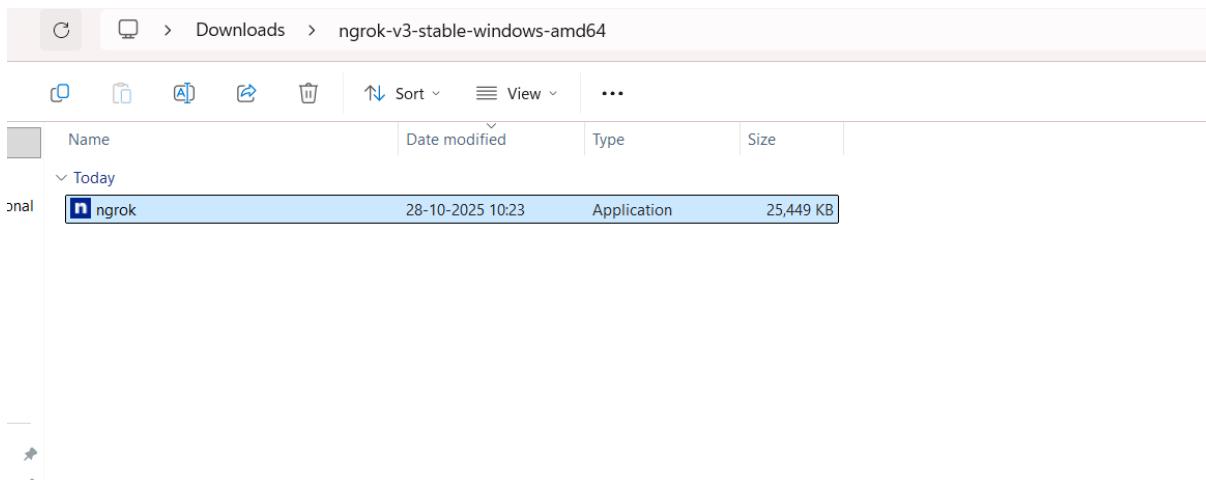


Step-2: Go -> <https://ngrok.com> and signup by giving your name ,email and password of atleast 10 charaters

A screenshot of the ngrok website. At the top, there's a navigation bar with links for Platform, Use cases, Blog, Resources, Docs, Pricing, Get ngrok, and a sign-in/sign-up button. The main heading is 'Download ngrok'. Below it, a sub-headline reads 'ngrok is your app's front door—and the fastest way to put anything on the internet.' A large 'Windows' section features the Windows logo and an 'Agent' icon. It includes an 'Installation' section with links to Microsoft Store, WinGet via Microsoft Store, Scoop, and Download. A note below says 'For the most secure experience, use the Microsoft Store to install ngrok. This type of installation is fully supported by Windows, ensuring automatic updates and compatibility with app management tools.' A 'Help' button is at the bottom right.

Step-3: After sign in up ,it will show below screen with your name in the top left .now click on download for windows (64Bit) to download ngork

#### Step4: After downloading ,Extract the file and click on ngrok.exe



Ngrok command prompt appears as below

```
C:\Users\sampa\Downloads\ngrok-v3-stable-windows-amd64> + ^

tcp      start a TCP tunnel
tls     start a TLS endpoint
update   update ngrok to the latest version
version  print the version string

EXAMPLES:
# forward http traffic from assigned public URL to local port 80
ngrok http 80
# port 8080 available at baz.ngrok.dev
ngrok http --url baz.ngrok.dev 8080
# forward arbitrary TCP traffic to port 22
ngrok tcp 22
# secure your app with auth
ngrok http 80 --oauth=google --oauth-allow-email=foo@foo.com

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/choose-a-plan to access paid features

Upgrade your account at https://dashboard.ngrok.com/billing/choose-a-plan to access paid features

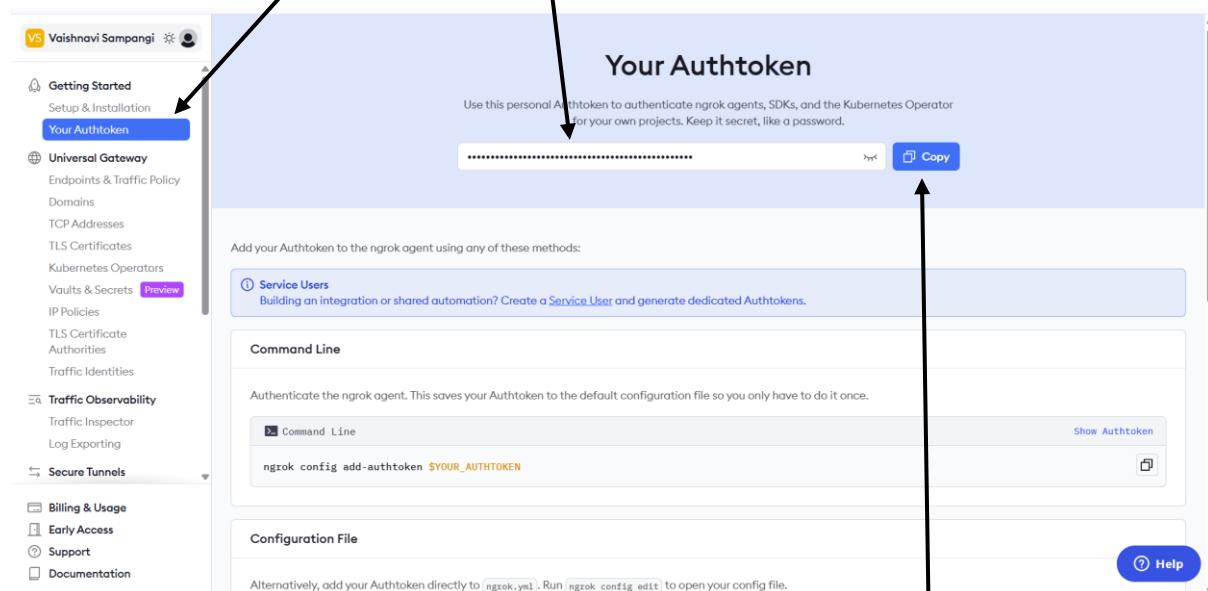
Flags:
  -h, --help      help for ngrok

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\sampa\Downloads\ngrok-v3-stable-windows-amd64>
```

## Step-5: Connect Your ngrok Account (optional but useful)

- Go to ngrok gives you an auth token.
- Then go to your Authtoken click here
- 



CREATE AUTHENTICATOR [<https://dashboard.ngrok.com/get-started/your-authtoken>]  
Run this command in ngrok command prompt:(replace <your\_token> with yours):

**ngrok config add-authtoken <your\_token> // syntax:**

**Example command**

**ngrok config add-authtoken 34ejJvRYKqtjvBUEYEnxudIpbp7\_33i6pYx8BJ2cZ1TpwasWq**

```
C:\Users\sampa\Downloads\ngrok-v3-stable-windows-amd64>ngrok config add-authtoken 34ejJvRYKqtjvBUEYEnxudIpbp7_33i6pYx8BJ2cZ1TpwasWq
Authtoken saved to configuration file: C:\Users\sampa\AppData\Local\ngrok/ngrok.yml
```

```
C:\Users\sampa\Downloads\ngrok-v3-stable-windows-amd64>
```

## Step-6

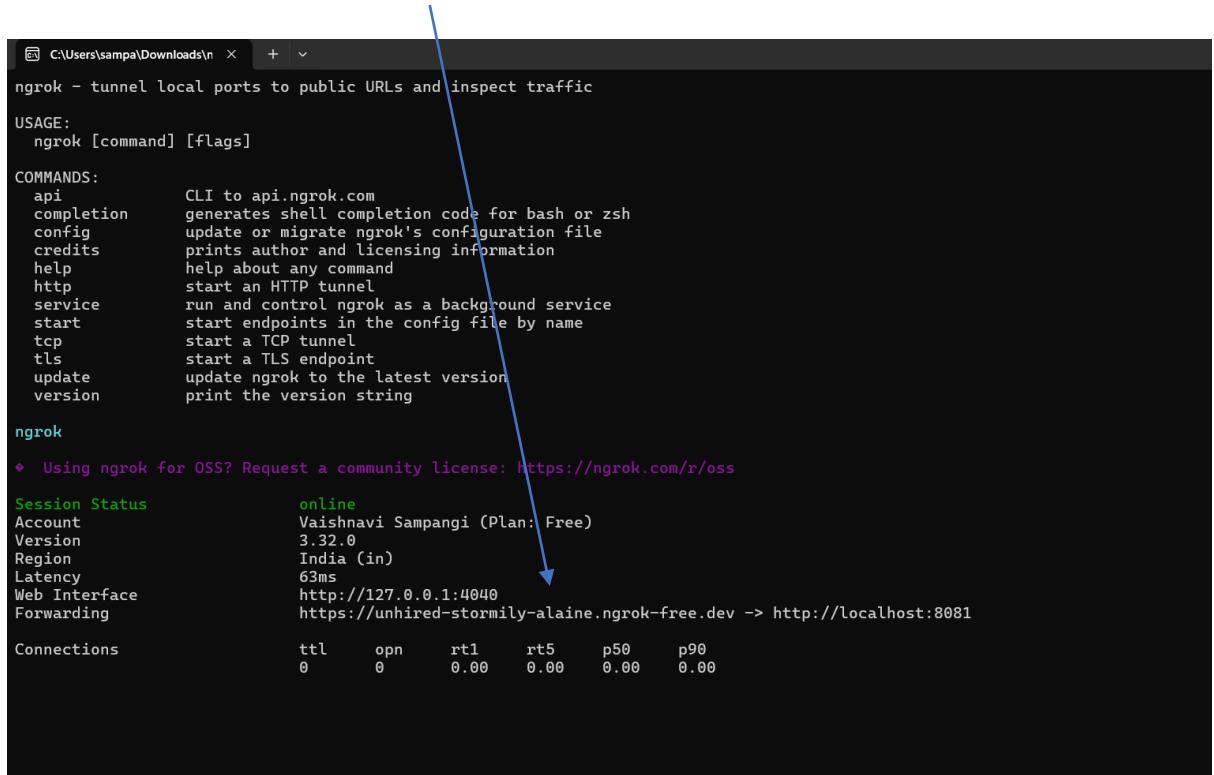
### Start a Tunnel for Jenkins

- Check on which port is your Jenkins running . for this give in browser or url localhost:8081  
For me Jenkins is running on 8081
- Go to ngrok command prompt and type below command
- ngrok http 8081 //Always use this command to start a tunnel for jenkins .

Type in ngrok command prompt:

```
at this terminal prompt to expose port 80.
C:\Users\sampa\Downloads\ngrok-v3-stable-windows-amd64>ngrok http 8081
```

Next it shows this public jenkins URL generated by ngrok that can be pasted into github repo for Webhooks.



```
C:\Users\sample\Downloads\n  +  v
ngrok - tunnel local ports to public URLs and inspect traffic

USAGE:
  ngrok [command] [flags]

COMMANDS:
  api      CLI to api.ngrok.com
  completion  generates shell completion code for bash or zsh
  config    update or migrate ngrok's configuration file
  credits   prints author and licensing information
  help     help about any command
  http     start an HTTP tunnel
  service   run and control ngrok as a background service
  start    start endpoints in the config file by name
  tcp      start a TCP tunnel
  tls     start a TLS endpoint
  update   update ngrok to the latest version
  version  print the version string

ngrok

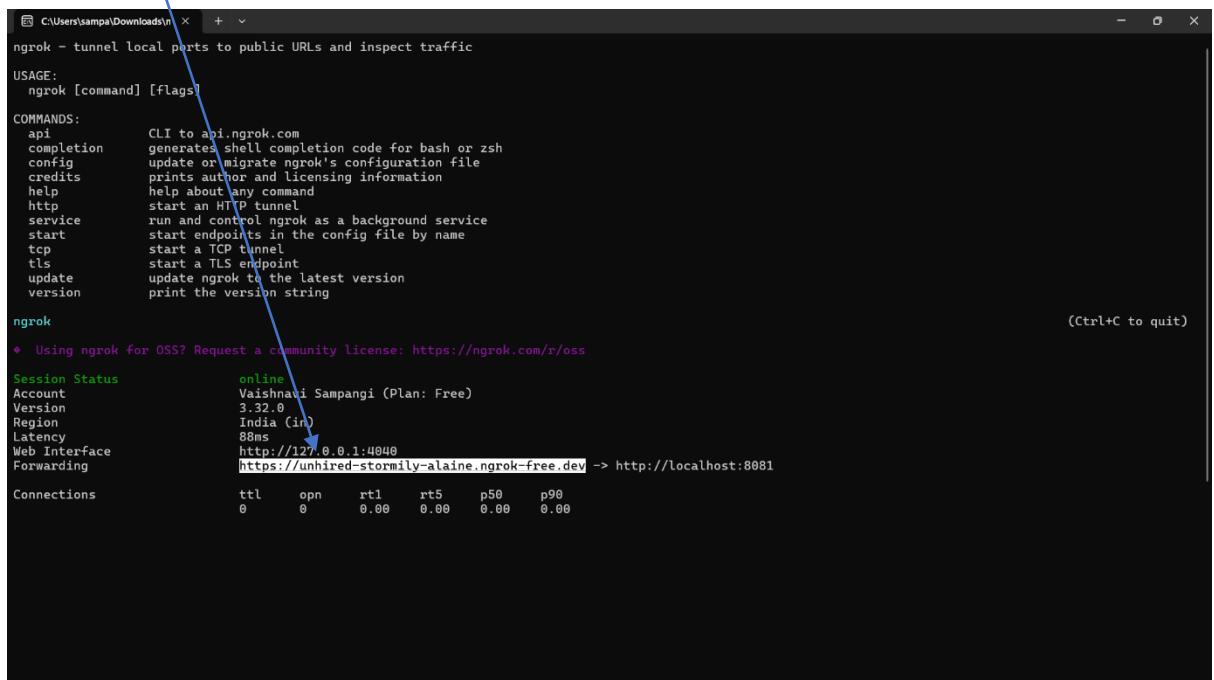
♦ Using ngrok for OSS? Request a community license: https://ngrok.com/r/oss

Session Status          online
Account                 Vaishnavi Sampangi (Plan: Free)
Version                3.32.0
Region                 India (in)
Latency                63ms
Web Interface          http://127.0.0.1:4040
Forwarding             https://unhired-stormily-alaine.ngrok-free.dev -> http://localhost:8081

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

(Ctrl+C to quit)
```

Copy this URL only highlighted part



```
C:\Users\sample\Downloads\n  +  v
ngrok - tunnel local ports to public URLs and inspect traffic

USAGE:
  ngrok [command] [flags]

COMMANDS:
  api      CLI to api.ngrok.com
  completion  generates shell completion code for bash or zsh
  config    update or migrate ngrok's configuration file
  credits   prints author and licensing information
  help     help about any command
  http     start an HTTP tunnel
  service   run and control ngrok as a background service
  start    start endpoints in the config file by name
  tcp      start a TCP tunnel
  tls     start a TLS endpoint
  update   update ngrok to the latest version
  version  print the version string

ngrok

♦ Using ngrok for OSS? Request a community license: https://ngrok.com/r/oss

Session Status          online
Account                 Vaishnavi Sampangi (Plan: Free)
Version                3.32.0
Region                 India (in)
Latency                88ms
Web Interface          http://127.0.0.1:4040
Forwarding             https://unhired-stormily-alaine.ngrok-free.dev -> http://localhost:8081

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

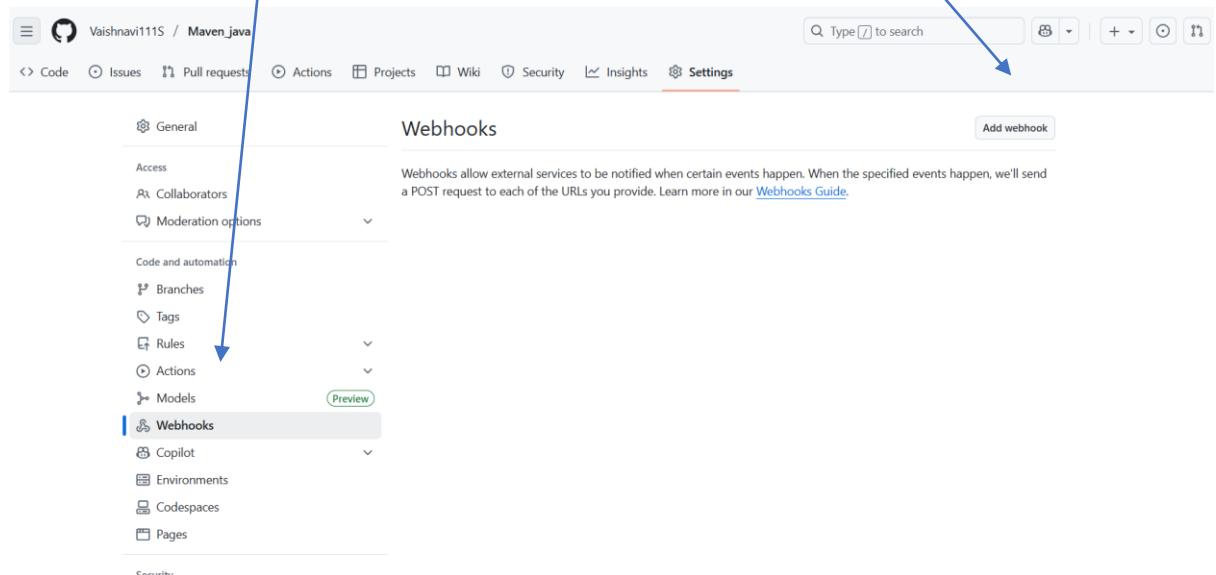
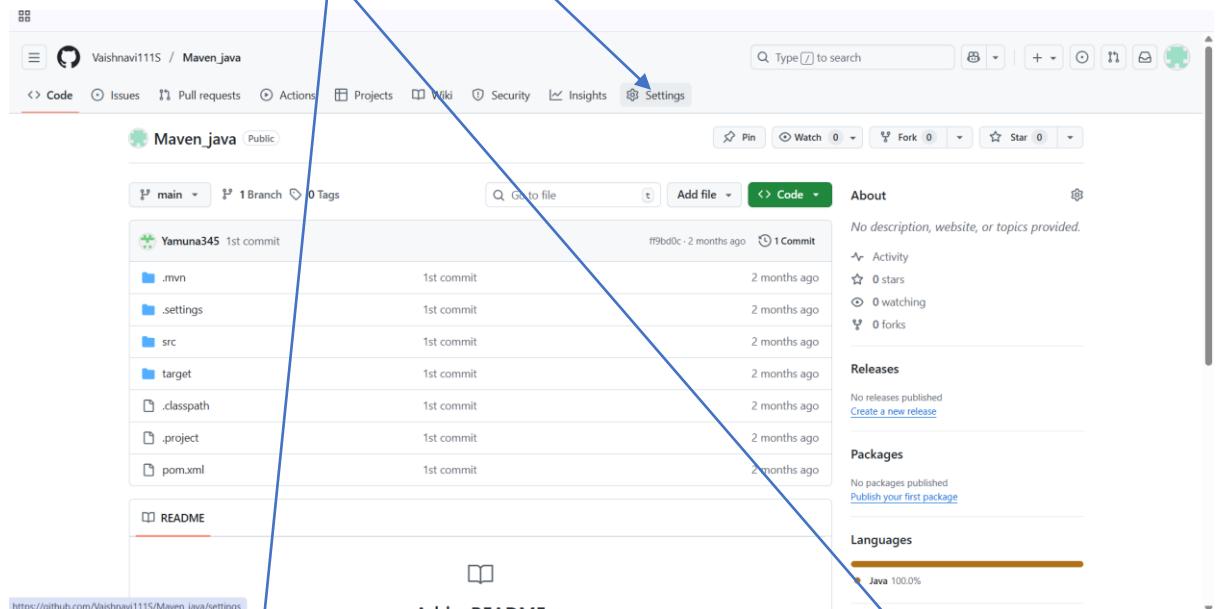
(Ctrl+C to quit)
```

### Step-7: Configure Webhook in GitHub

1. Go to your GitHub repository.

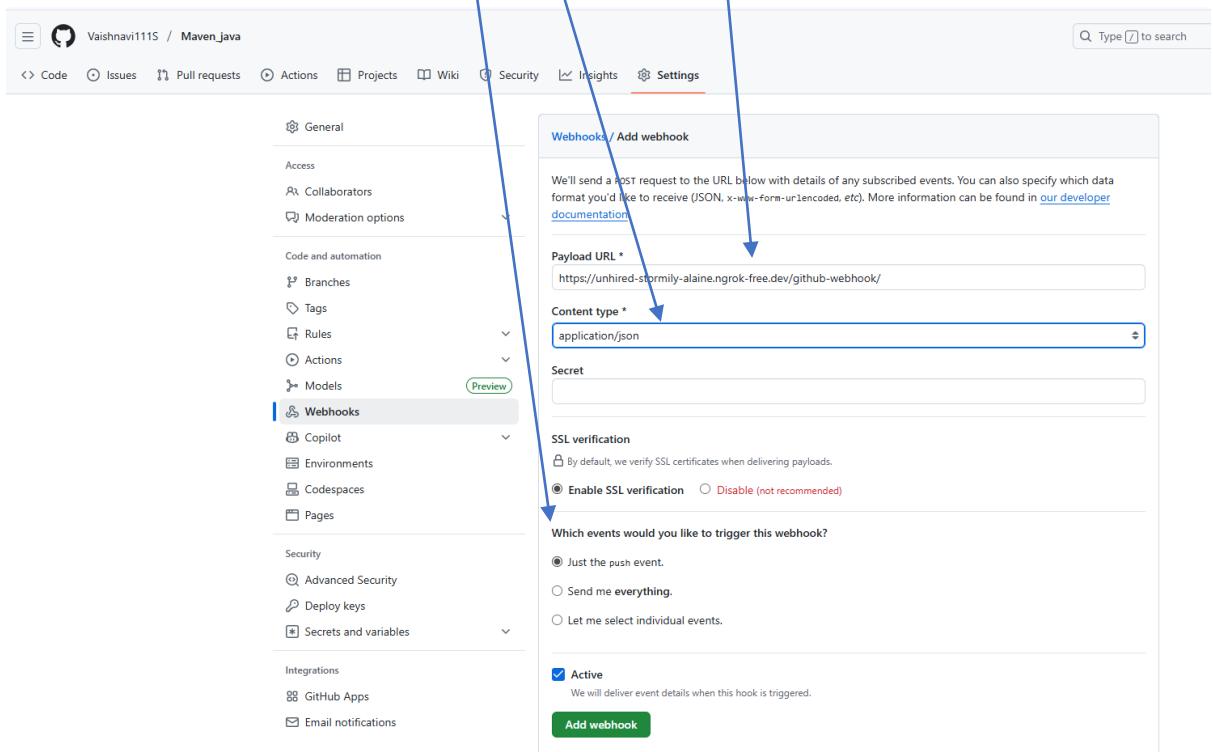
2. Navigate to Settings → Webhooks.
3. Click “Add webhook”.
4. In the Payload URL field:
  - Enter the Jenkins webhook URL in the format:  
`http://<jenkins-server-url>/github-webhook/`  
Ex: `https://unhired-stormily-alaine.ngrok-free.dev/github-webhook/`

Note: If Jenkins is running on localhost, GitHub cannot access it directly



### Step-8:

- Add url <https://unhired-stormily-alaine.ngrok-free.dev/github-webhook/>
- Set content Type to application/json
- Under “Which events would you like to trigger this webhook?”, select:  
Just the push event.
  - Click “Add webhook” to save.



---

### Step 10: Configure Jenkins to Accept GitHub Webhooks

1. Open Jenkins Dashboard.

2. Select the job (freestyle or pipeline) you've already created.

The screenshot shows the Jenkins dashboard with a list of jobs. The 'Maven\_java\_build' job is highlighted with a blue arrow pointing to it. The dashboard includes sections for 'Build Queue' (empty), 'Build Executor Status' (0/2), and a main table of jobs. The table columns are S, W, Name, Last Success, Last Failure, Last Duration, and F. The 'Maven\_java\_build' job has a status icon with a sun and a cloud, a name of 'Maven\_java\_build', a last success at '20 days #11', and a last failure at 'N/A'. Other jobs listed include JS1, Maven\_java\_test, Maven\_web\_build, Maven\_web\_deploy, and Maven\_web\_test.

3. Click Configure.  
 4. Scroll down to the Build Triggers section.  
 5. Check the box:  GitHub hook trigger for GITScm polling

The screenshot shows the configuration page for the 'Maven\_java\_build' job. The 'Triggers' section is open, displaying several options: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling' (which is checked), and 'Poll SCM'. Below the triggers, the 'Environment' section is partially visible, containing various build environment settings. At the bottom of the page are 'Save' and 'Apply' buttons.

6. Click Save.

#### Step 11: Test the Setup

1. Make any code update in your local repo and push it to GitHub.
2. Once pushed, GitHub will trigger the webhook.
3. Jenkins will automatically detect the change and start the build pipeline.

**Jenkins** / Maven\_java\_build / Configuration

## Configure

### 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 ?

### Environment

Configure settings and variables that define the context in which your build runs, like credentials, paths, and global parameters.

- Delete workspace before build starts
- Use secret text(s) or file(s) ?
- Add timestamps to the Console Output
- Inspect build log for published build scans
- Terminate a build if it's stuck
- With Ant ?

**Save** **Apply**

**Jenkins**

+ New Item

Build History

All	pipeline2.java	pipeline_web	+				
Build Queue	No builds in the queue.						
Build Executor Status	0/2						
S	W	Name ↓	Last Success	Last Failure	Last Duration	F	
		JS1	20 days #11	N/A	28 sec		
		Maven_java_build	1 mo 2 days #12	1 mo 2 days #10	7.1 sec		
		Maven_java_test	1 mo 2 days #13	1 mo 2 days #11	3.5 sec		
		Maven_web_build	1 mo 2 days #5	1 mo 2 days #3	25 sec		
		Maven_web_deploy	1 mo 2 days #12	1 mo 2 days #9	1.4 sec		
		Maven_web_test	1 mo 2 days #8	N/A	5.6 sec		

Icon: S M L

REST API Jenkins 2.516.3

Vaishnavi111S / Maven.java

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Maven.java / Name your file... in main

Cancel changes Commit changes...

Edit Preview

```
1 demo of webhook
```

Vaishnavi111S / Maven.java

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Maven.java / Demofile in main

Cancel changes Commit changes...

Edit Preview

```
1 webhook
```

### Commit changes

**Commit message**

commit for webhook

**Extended description**

Add an optional extended description...

Commit directly to the **main** branch  
 Create a **new branch** for this commit and start a pull request [Learn more about pull requests](#)

**Cancel** **Commit changes**

Vaishnavi111S / Maven\_Java

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main / Maven.java /

Vaishnavi111S commit for webhook

Name	Last commit message	Last commit date
.mvn	1st commit	2 months ago
.settings	1st commit	2 months ago
src	1st commit	2 months ago
target	1st commit	2 months ago
classpath	1st commit	2 months ago
.project	1st commit	2 months ago
DemoFile	commit for webhook	now
pom.xml	1st commit	2 months ago

The screenshot shows the Jenkins Pipeline interface. At the top, there's a navigation bar with 'New Item', 'Build History' (selected), 'All', 'pipeline2\_java', 'pipeline\_web', and a '+' button. Below this is a 'Build Queue' section stating 'No builds in the queue.' To the right is a table titled 'Build Executor Status' showing the following data:

S	W	Name	Last Success	Last Failure	Last Duration	F
<span style="color: green;">✓</span>	<span style="color: orange;">☀️</span>	JS1	20 days #11	N/A	28 sec	<span style="color: green;">▶</span> <span style="color: blue;">star</span>
<span style="color: green;">✓</span>	<span style="color: yellow;">☁️</span>	Maven_java_build	1 mo 2 days #12	1 mo 2 days #10	7.1 sec	<span style="color: green;">▶</span> <span style="color: blue;">star</span>
<span style="color: green;">✓</span>	<span style="color: grey;">☁️</span>	Maven_java_test	1 mo 2 days #13	1 mo 2 days #11	3.5 sec	<span style="color: green;">▶</span> <span style="color: blue;">star</span>
<span style="color: green;">✓</span>	<span style="color: yellow;">☁️</span>	Maven_web_build	1 mo 2 days #5	1 mo 2 days #3	25 sec	<span style="color: green;">▶</span> <span style="color: blue;">star</span>
<span style="color: green;">✓</span>	<span style="color: lightblue;">☁️</span>	Maven_web_deploy	1 mo 2 days #12	1 mo 2 days #9	1.4 sec	<span style="color: green;">▶</span> <span style="color: blue;">star</span>
<span style="color: green;">✓</span>	<span style="color: orange;">☀️</span>	Maven_web_test	1 mo 2 days #8	N/A	5.6 sec	<span style="color: green;">▶</span> <span style="color: blue;">star</span>

Below the table are icons for 'Icon: S M L' and search/filter buttons. The main area shows a 'Build Pipeline' visualization for 'pipeline2\_java'. It features a grey box labeled 'Pipeline #13' and two green boxes labeled '#13 Maven\_java\_build' and '#14 Maven\_java\_test'. Each green box contains a timestamp ('28-Oct-2025 11:37:13 am') and a duration ('29 sec' or '13 sec'). Above the pipeline visualization is a toolbar with 'Run', 'History', 'Configure', 'Add Step', 'Delete', and 'Manage' buttons.

## outcome

- You've successfully connected GitHub and Jenkins using webhooks.
- Every time you push code to GitHub, Jenkins will automatically start building your project without manual intervention.

## EXERCISE-2

### Setting Up Jenkins Email Notification Setup (Using Gmail with App Password)

#### Creation of app password

##### 1. Gmail: Enable App Password (for 2-Step Verification)

i. Go to: <https://myaccount.google.com>

##### ii. Enable 2-Step Verification

- Navigate to:
  - Security → 2-Step Verification
  - Turn it **ON**
  - Complete the OTP verification process (via phone/email)

##### iii. Generate App Password for Jenkins

- Go to:
  - Security → App passwords
- Select:
  - **App: Other (Custom name)**
  - **Name: Jenkins-Demo**
- Click **Generate**

- Copy the **16-digit app password**
  - Save it in a secure location (e.g., Notepad)

## 2. Jenkins Plugin Installation

### i. Open Jenkins Dashboard

### ii. Navigate to:

- Manage Jenkins → Manage Plugins

### iii. Install Plugin:

- Search for and install:
  - Email Extension Plugin

---

## 3. Configure Jenkins Global Email Settings

### i. Go to:

- Manage Jenkins → Configure System

---

### A. E-mail Notification Section

Field	Value
SMTP Server	smtp.gmail.com
Use SMTP Auth	<input checked="" type="checkbox"/> Enabled
User Name	Your Gmail ID (e.g., archanareddykmit@gmail.com)
Password	Paste the 16-digit App Password
Use SSL	<input checked="" type="checkbox"/> Enabled
SMTP Port	465
Reply-To Address	Your Gmail ID (same as above)

### ► Test Configuration

- Click: Test configuration by sending test e-mail
- Provide a valid email address to receive a test mail
- Should receive email from Jenkins

---

### B. Extended E-mail Notification Section

Field	Value
SMTP Server	smtp.gmail.com
SMTP Port	465
Use SSL	<input checked="" type="checkbox"/> Enabled

Field	Value
Credentials	Add Gmail ID and App Password as Jenkins credentials
Default Content Type	text/html or leave default
Default Recipients	Leave empty or provide default emails
Triggers	Select as per needs (e.g., Failure)

---

#### 4. Configure Email Notifications for a Jenkins Job

##### i. Go to:

- Jenkins → Select a Job → Configure
- 

##### ii. In the Post-build Actions section:

- Click: Add post-build action → **Editable Email Notification**

##### A. Fill in the fields:

Field	Value
Project Recipient List	Add recipient email addresses (comma-separated)
Content Type	Default (text/plain) or text/html
Triggers	Select events (e.g., Failure, Success, etc.)
Attachments	(Optional) Add logs, reports, etc.

---

##### iii. Click Save

---

**Now your Jenkins job is set up to send email notifications based on the build status!**

---

##### Takeaway :

Students learned how to integrate Jenkins with GitHub using webhooks to automate build triggers and configure email notifications to monitor build success or failure effectively.