

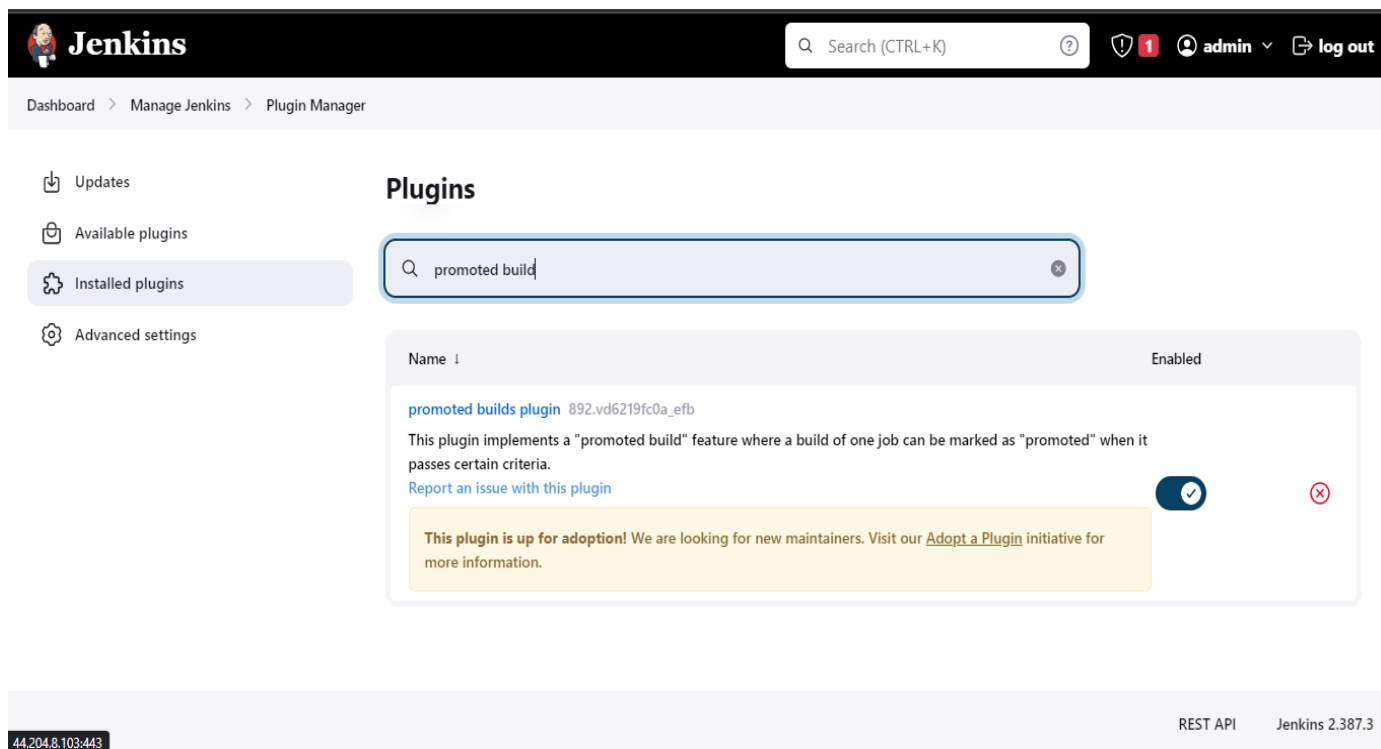
1. Promotion build plugin - scenario

The "Promoted Builds" plugin in Jenkins is used to add manual or automatic promotion processes to your build jobs. It allows you to define criteria for promoting a build, such as passing specific tests, meeting certain quality metrics, or going through a manual approval process.

Here's how you can use the Promoted Builds plugin in Jenkins:

Step1:

Install the Promoted Builds plugin: Go to Jenkins Dashboard -> Manage Jenkins -> Manage Plugins -> Available tab -> Search for "Promoted Builds" -> Select the checkbox next to it -> Click on "Install without restart" or "Download now and install after restart".



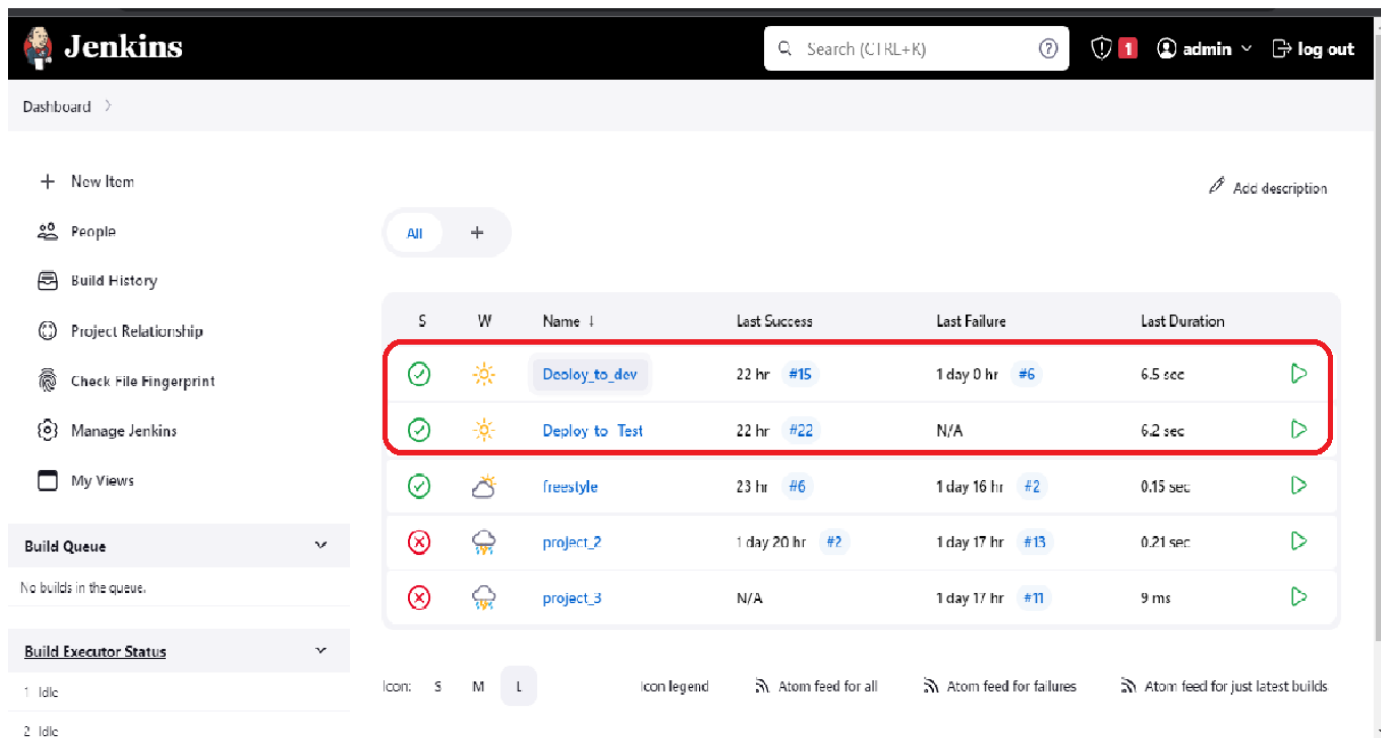
The screenshot shows the Jenkins Plugin Manager interface. The top navigation bar includes the Jenkins logo, a search bar, and user information (admin). The left sidebar shows navigation options: Updates, Available plugins, Installed plugins (selected), and Advanced settings. The main content area is titled 'Plugins' and contains a search bar with 'promoted build' entered. Below the search bar, a table lists the installed plugins. The 'promoted builds plugin' is shown with its ID '892.vd6219fc0a_efb'. The description states: 'This plugin implements a "promoted build" feature where a build of one job can be marked as "promoted" when it passes certain criteria.' There is a link to 'Report an issue with this plugin'. A yellow warning box indicates: 'This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information.' The plugin is marked as 'Enabled' with a toggle switch and a checkmark icon. The bottom status bar shows the IP address '44.204.8.103:443', the REST API endpoint, and the Jenkins version '2.387.3'.

Name	Enabled
promoted builds plugin 892.vd6219fc0a_efb This plugin implements a "promoted build" feature where a build of one job can be marked as "promoted" when it passes certain criteria. Report an issue with this plugin	<input checked="" type="checkbox"/>

Step2:

Configure a job with promotion criteria: Open the Jenkins job you want to configure promotions for.

So, I have already created 2 free style jobs "Deploy_to _dev and Deploy_to _Test".



The screenshot shows the Jenkins Dashboard. The top navigation bar includes the Jenkins logo, a search bar, and user information (admin) with a log out button. The left sidebar contains a menu with options: New Item, People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, and My Views. The main content area displays a table of build jobs. The first two rows, 'Deploy_to_dev' and 'Deploy_to Test', are highlighted with a red border. Below the table, there are sections for 'Build Queue' (showing no builds) and 'Build Executor Status' (showing two idle executors). At the bottom, there are links for icon legend and Atom feeds.

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀	Deploy_to_dev	22 hr #15	1 day 0 hr #6	6.5 sec
✓	☀	Deploy to Test	22 hr #22	N/A	6.2 sec
✓	☁	freestyle	23 hr #6	1 day 16 hr #2	0.15 sec
✗	☁	project_2	1 day 20 hr #2	1 day 17 hr #13	0.21 sec
✗	☁	project_3	N/A	1 day 17 hr #11	9 ms

Step3:

Add the promotion process: Scroll down to the "Post-build Actions" section and click on "Add post-build action". Select "Promote builds when..." from the dropdown menu.

The image displays two screenshots of the Jenkins configuration interface for a job named 'Deoloy_to_dev'.

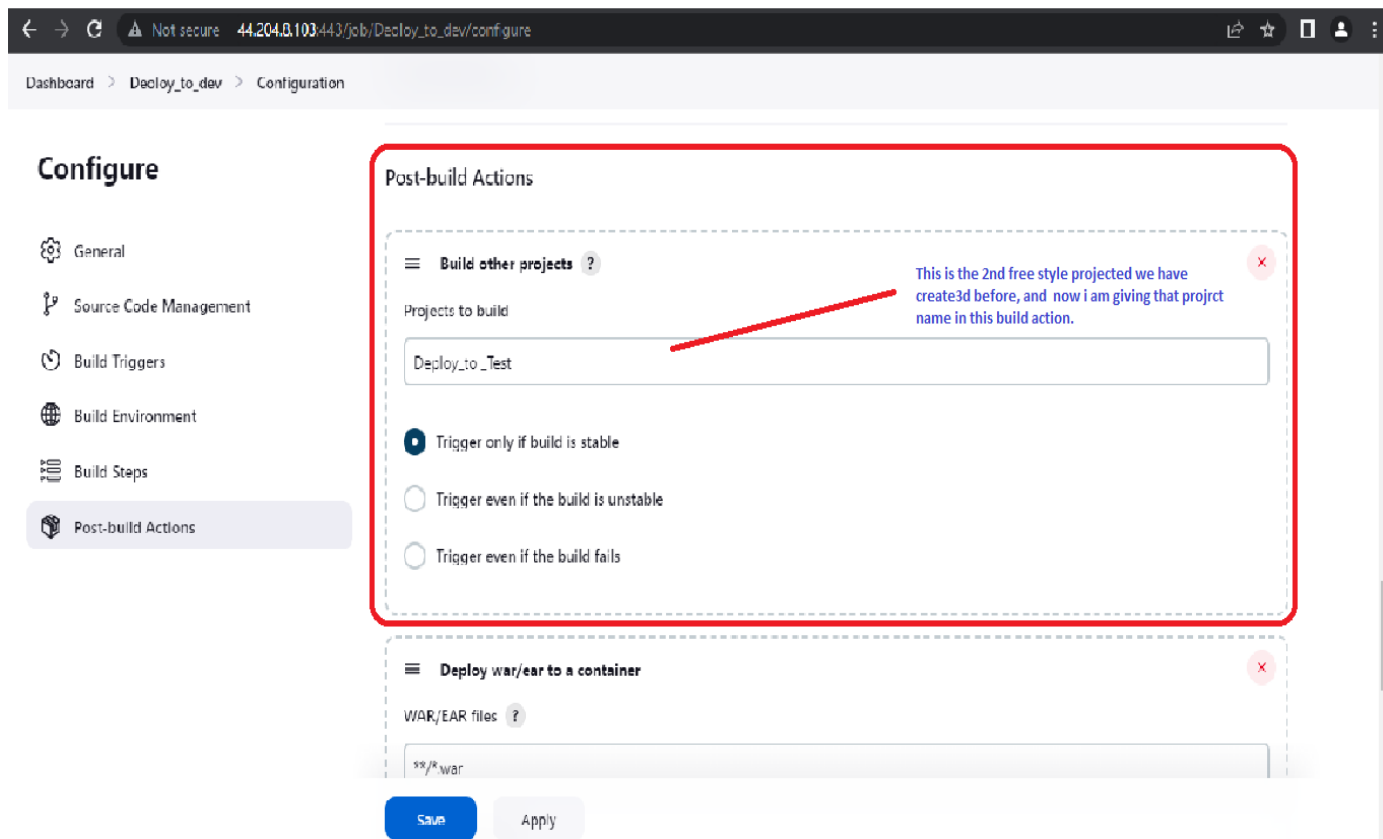
Top Screenshot: The 'Configure' page is shown with the 'Post-build Actions' tab selected. A dropdown menu is open, listing various post-build actions. The option 'Build other projects' is highlighted. Below the menu, there are 'Save' and 'Apply' buttons.

Bottom Screenshot: The 'Post-build Actions' configuration is shown. A new action, 'Build other projects', has been added. The configuration for this action includes a text field for 'Projects to build' containing 'dep', a list of project names including 'Deploy_to_Test', and three radio buttons for triggering conditions: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. 'Save' and 'Apply' buttons are at the bottom.

Step4:

Define promotion criteria: In the promotion configuration, you can specify the criteria that need to be met for the build to be promoted. For example, you can add conditions like "Build stability" to check if the build is stable or "Conditional steps (single)" to add custom conditions using Groovy script.

Save the configuration: Click on "Save" to apply the promotion configuration to the job.



Now, when the build completes, Jenkins will evaluate the defined criteria, and if they are met, it will trigger the configured promotion actions. The promotion process can be automated or require manual approval, depending on how you configure it.

You can also view the promotion history and manually trigger promotions for specific builds from the Jenkins job page.

Dashboard >

+ New Item

✎ Add description

👤 People

All +

📅 Build History

🔗 Project Relationship

🔍 Check File Fingerprint

⚙️ Manage Jenkins

📌 My Views

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 **Deoloy to dev** #20

Icon: S M L

Icon legend

📡 Atom feed for all

📡 Atom feed for failures

📡 Atom feed for just latest builds

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀️	Deoloy_to_dev	50 sec #18	1 day 1 hr #6	6.4 sec
✓	☁️	Deploy_to_Test	38 sec #26	25 min #23	6.2 sec
✓	☁️	freestyle	1 day 1 hr #6	1 day 18 hr #2	0.15 sec
✗	☁️	project_2	1 day 21 hr #2	1 day 19 hr #13	0.21 sec
✗	☁️	project_3	N/A	1 day 19 hr #11	9 ms

Dashboard > Deploy_to_Test >

</> Changes

📁 Workspace

▶ Build Now

⚙️ Configure

🗑️ Delete Project

✎ Rename

🔍 Build History

trend

🔍 Filter builds...

#29
(pending—in the quiet period. Expires in 4 sec)

#28 Jun 1, 2023, 7:05 AM

#26 Jun 1, 2023, 7:03 AM

#25 Jun 1, 2023, 7:02 AM

#24 Jun 1, 2023, 6:39 AM

#23 Jun 1, 2023, 6:38 AM

Upstream Projects

✓ Deoloy_to_dev

Permalinks

- Last build (#28), 3 min 36 sec ago
- Last stable build (#28), 3 min 36 sec ago
- Last successful build (#28), 3 min 36 sec ago
- Last failed build (#23), 37 min ago
- Last unsuccessful build (#23), 37 min ago
- Last completed build (#28), 3 min 36 sec ago

✎ Add description

Disable Project

After building the "devlop to Dev", automatically we can see here "Deploy to Test" is in pending, and it is ready to build

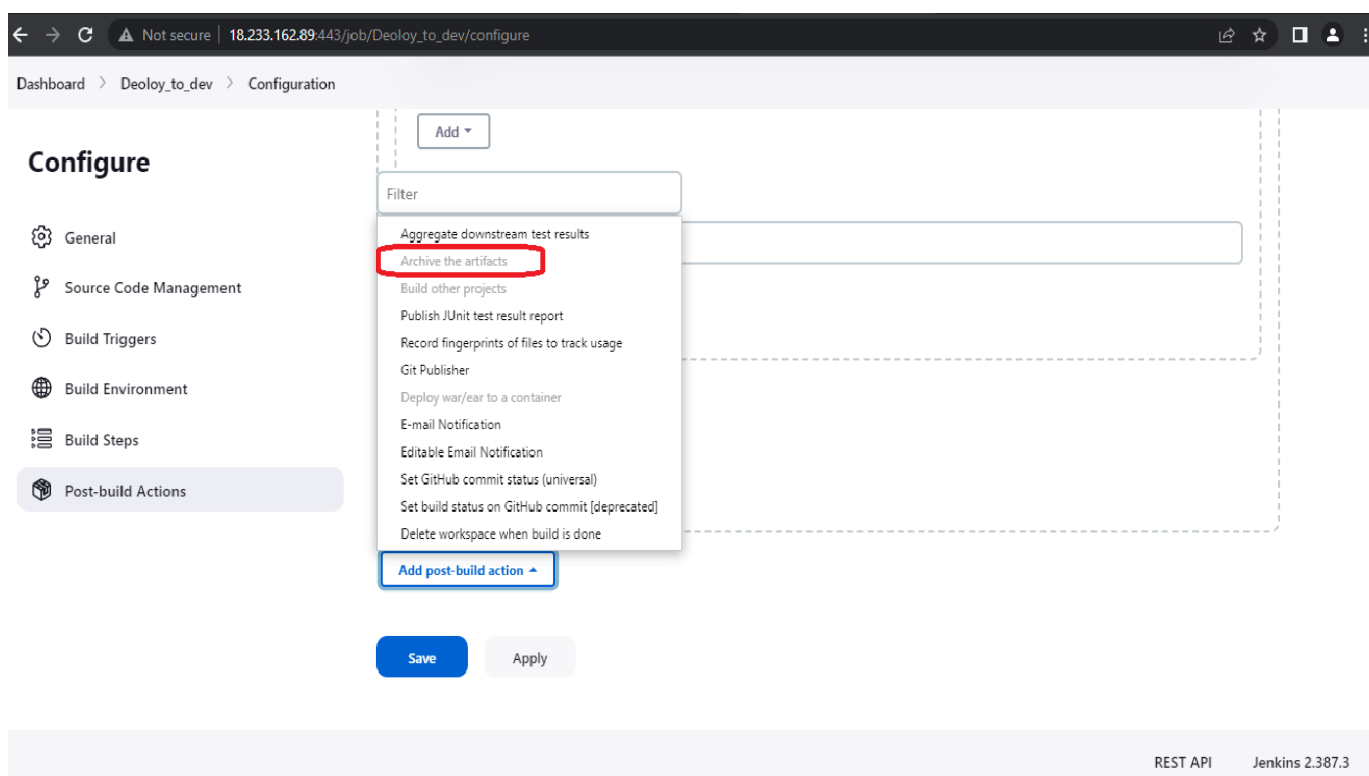
This is how **Promoted Build Plugin** Works...

2. Using Jenkins deploying the code and executed the code through Dev to Test and Test to Production environment.

As we have done till now.....In the before task, same thing to be followed and from now we need to add some post build action, then we ll get the required output.

Step1:

Go to “deploy_to_dev” and configure options, then at the bottom we find **Add Post-Build Action**, we need to add the **Archive the Anticraft..**



Dashboard > Deoloy_to_dev > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

Advanced ▾

Add build step ▾

Post-build Actions

Archive the artifacts ?

Files to archive ?

**/*.war

Advanced ▾

We need to give war file path in this

Build other projects ?

Projects to build

Save Apply

Need to give the Content Path..

Dashboard > Deoloy_to_dev > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

Deploy war/ear to a container

WAR/EAR files ?

**/*.war

Context path ?

devlop

Containers

Tomcat 9.x Remote

Credentials

uday/*****

Add ▾

Tomcat URL ?

Step2:

We need to do the same thing in the other project “Deploy_to_Test”. Giving war file Path . And need to give the content path “Test”..

Dashboard > Deploy_to_Test > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

Post-build Actions

Deploy war/ear to a container

WAR/EAR files ?

**/*.war

Context path ?

Test

Containers

Tomcat 9.x Remote

Credentials

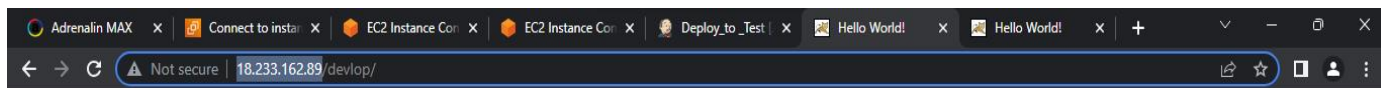
uday/*****

Add

Save Apply

Step3:

Now build both the projects..

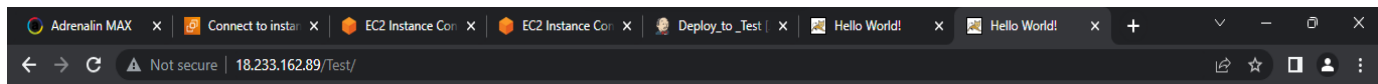


welcome to DevOps!

It is now Thu Jun 01 09:19:32 UTC 2023

You are coming from 106.51.122.2





Hello World!

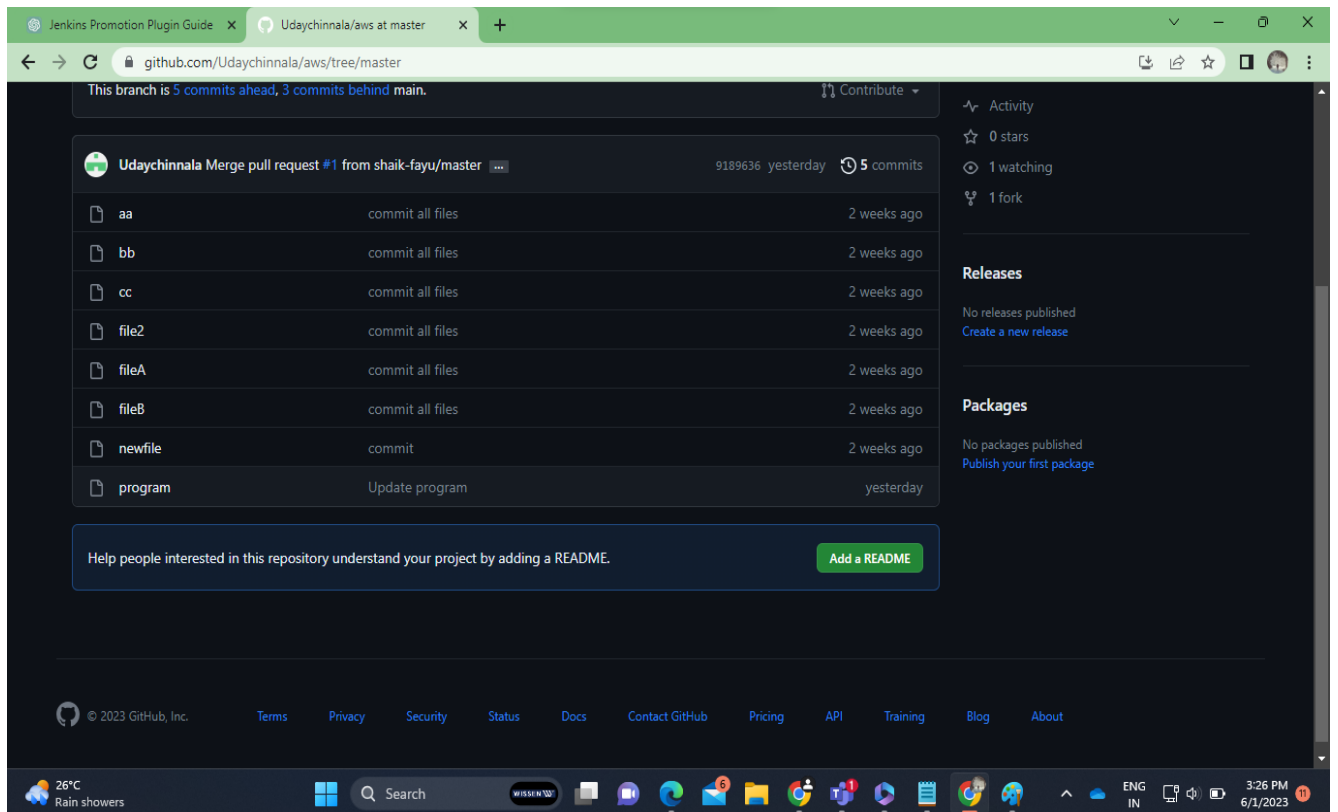
It is now Thu Jun 01 09:48:18 UTC 2023

You are coming from 106.51.122.2

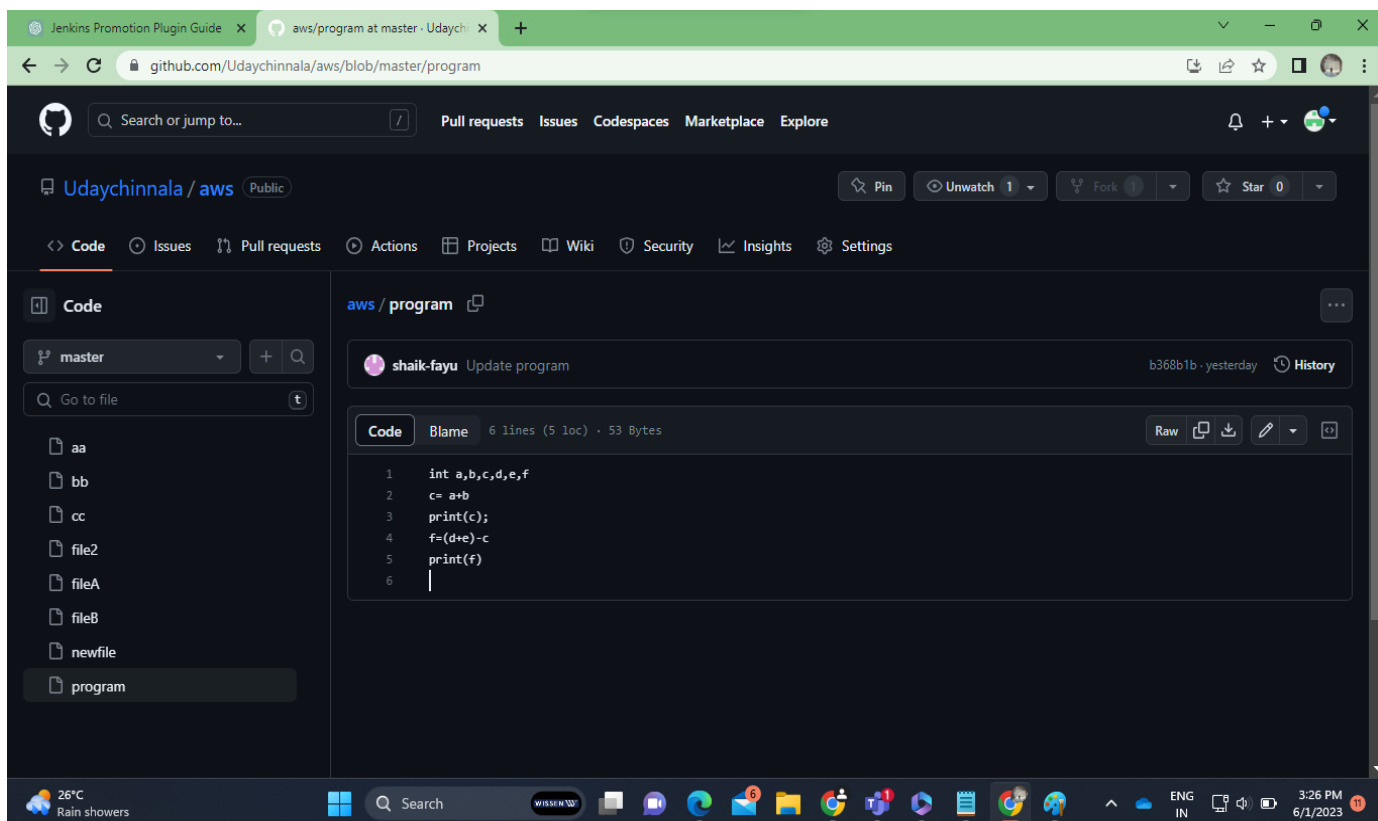


3.PULL Request

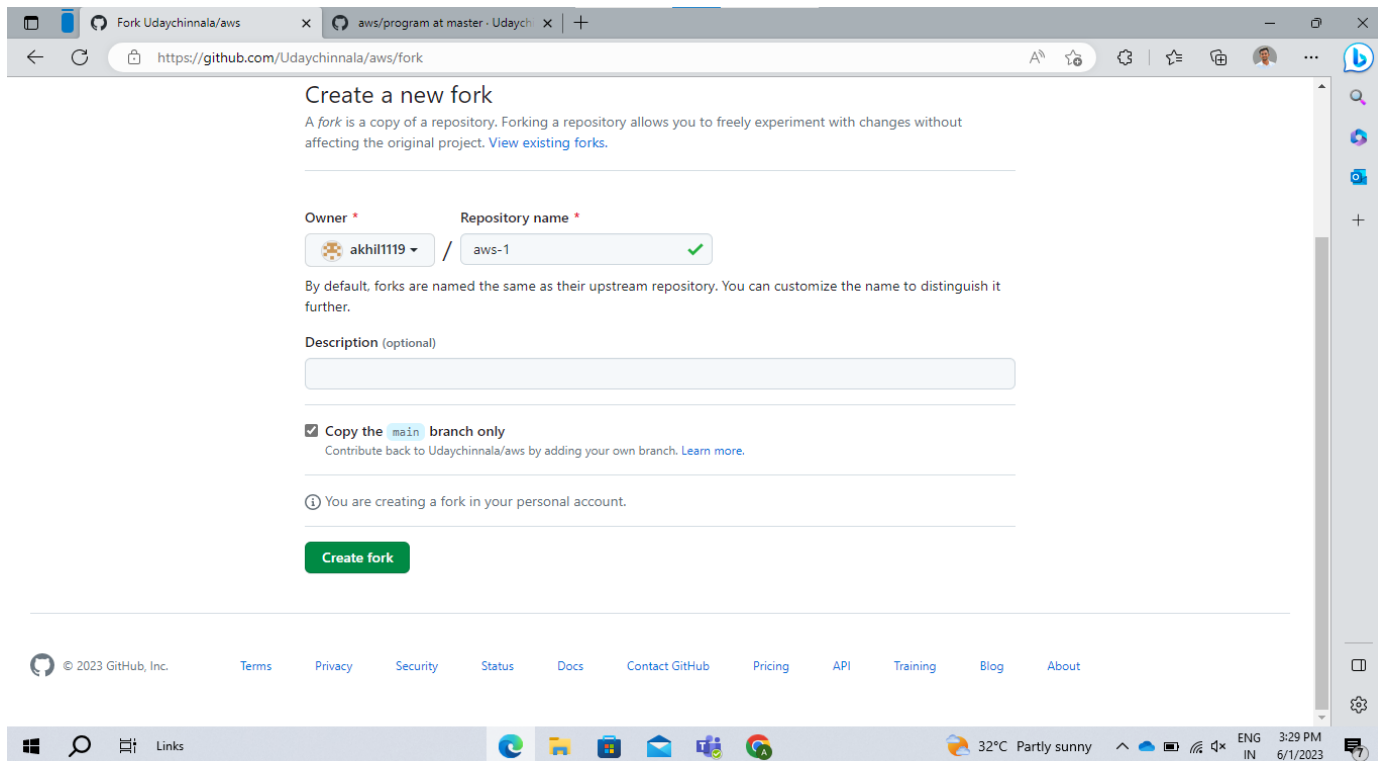
Firstly , lets create a text file, and write a code..(named as Program)..



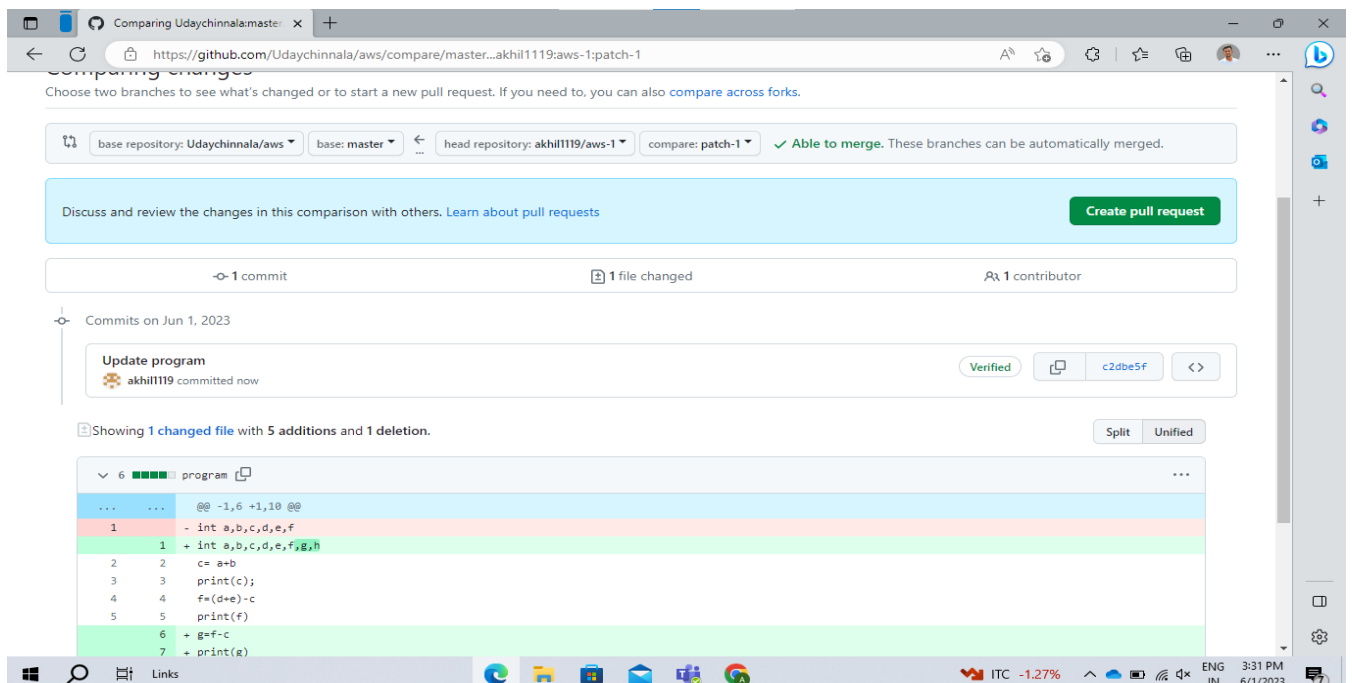
This is the code in the Program file..



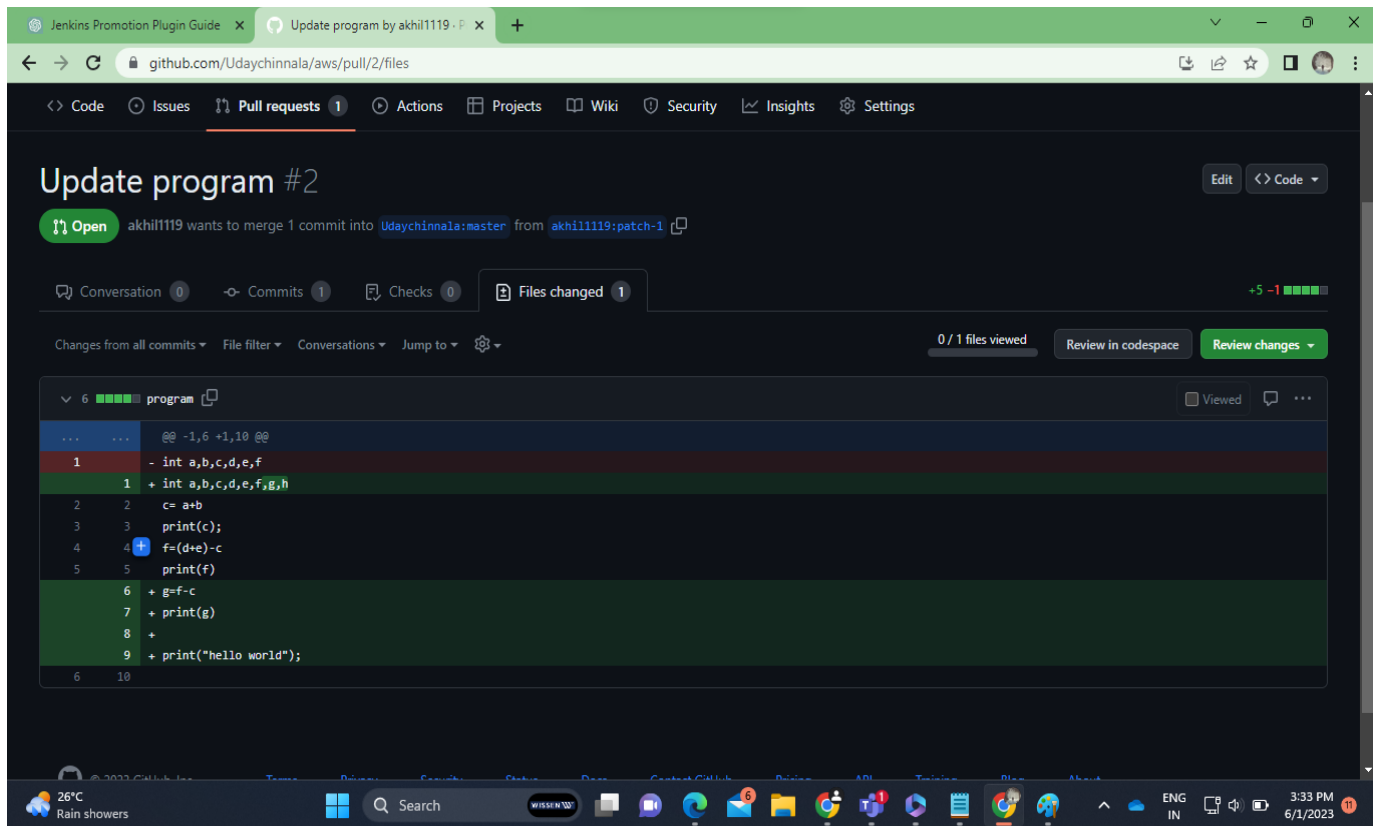
Copy the link and open with another Git-Hub Account..and fork the code into the git-hub Account .



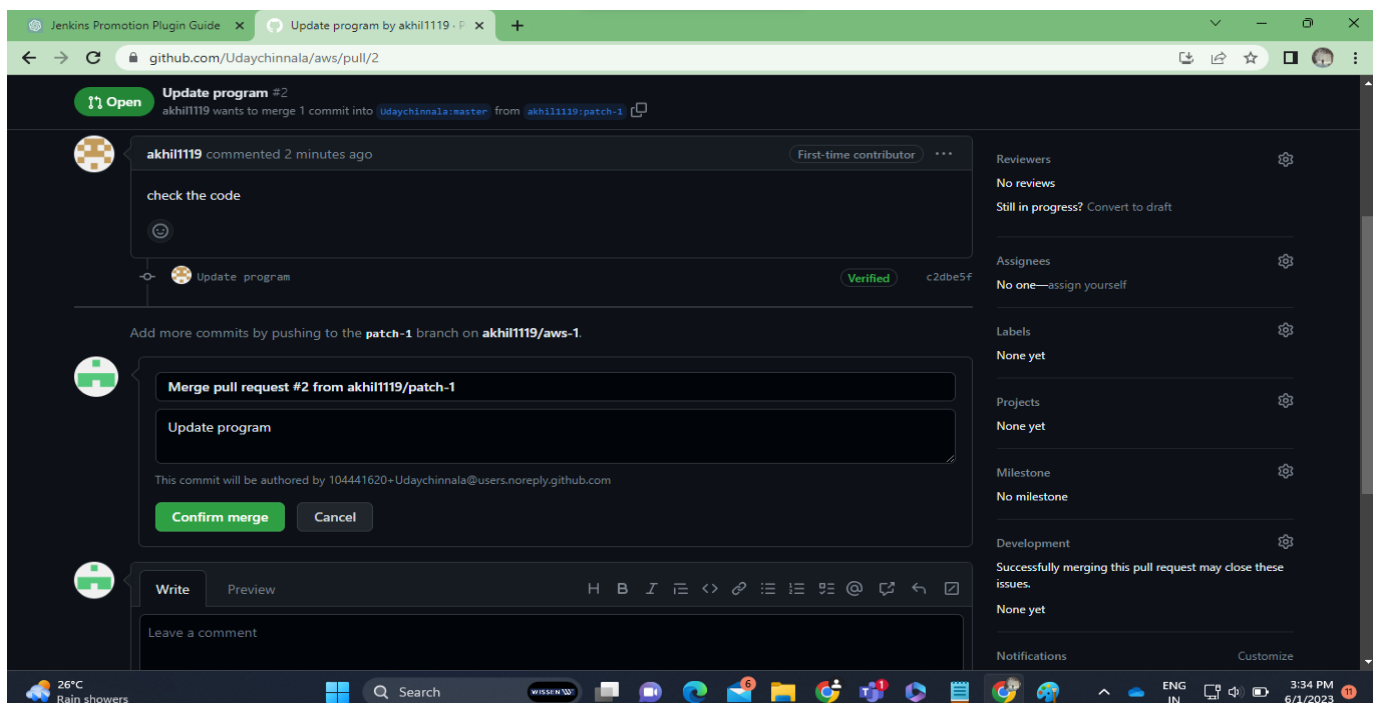
Do some changes in the file and commit the changes...then we ll get a pull request.



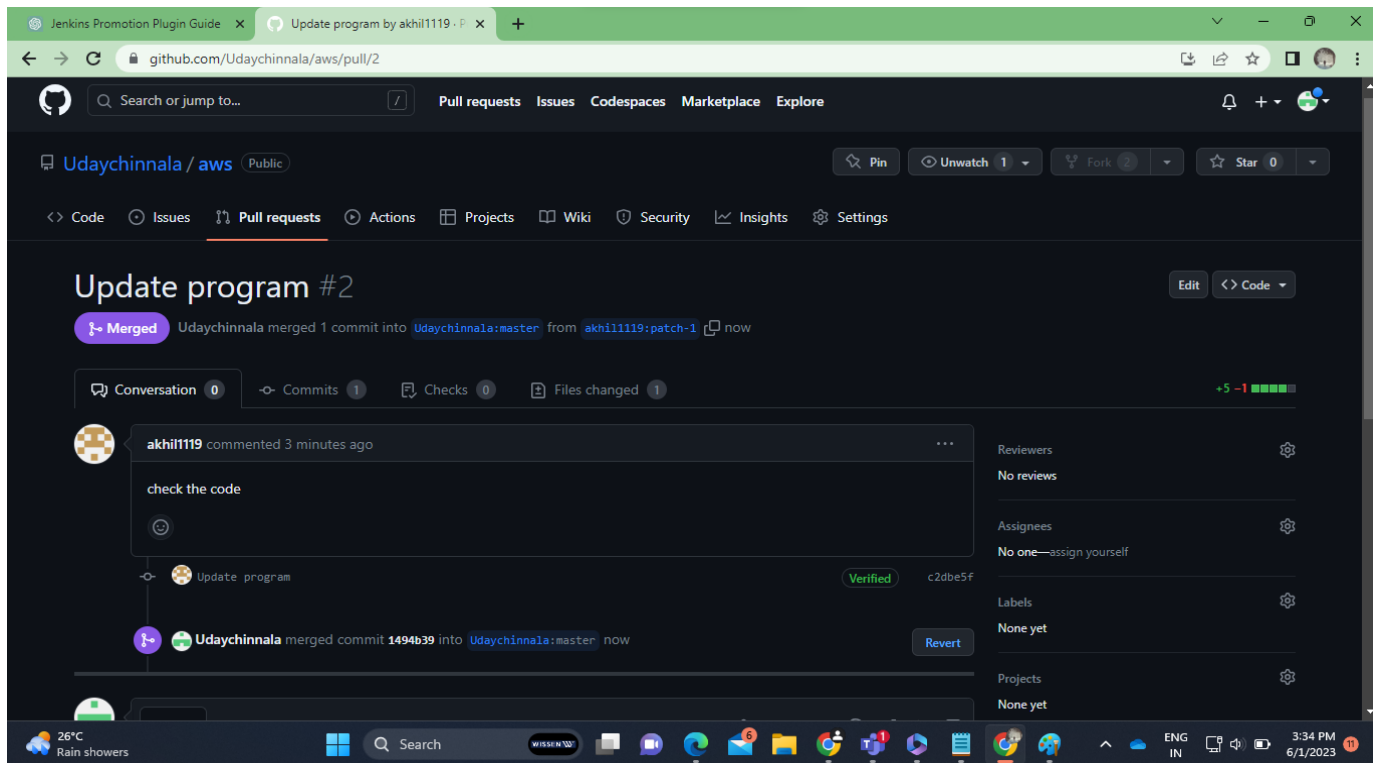
Now we can see the changes made on another account, and green colour indicates the new code commits, and red indicates what we have done..



If everything is proper, then we can merge the changes ..so for that we need to go with conform merge.



Now it has successfully merged into our code.



Here we can see the latest code that we have Merged

