

Sakshi Shirure(11-10-2024)

## Maven Job





### 1)Create maven job(demo\_Maven) in jenkins

Dashboard > All > New Item

### New Item

Enter an item 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.
-  **Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
-  **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.
-  **Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments.

OK

### 2)add url of your repository

Dashboard > demo\_Maven > Configuration

### Configure

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

#### Source Code Management

☐ None

☒ **Git** ?

Repositories ?

Repository URL ?

Credentials ?

- none -

+ Add

Advanced ▾

3)provide branch (dev) it means when we do any changes in dev branch job will be run

Dashboard > demo\_Maven > Configuration

Configure

General

Source Code Management

Build Triggers

Build Environment

Pre Steps

Build

Post Steps

Build Settings

Post-build Actions

dev

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add ▾

Build Triggers

☒ Build whenever a SNAPSHOT dependency is built ?

☐ Schedule build when some upstream has no successful builds ?

☐ Trigger builds remotely (e.g., from scripts) ?

☐ Build after other projects are built ?

☐ Build periodically ?

☒ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

4)give gols as “clean package” –it will compile and package code

Dashboard > demo\_Maven > Configuration

Configure

General

Source Code Management

Build Triggers

Build Environment

Pre Steps

Build

Post Steps

Build Settings

Post-build Actions

Build

Root POM ?

pom.xml

Goals and options ?

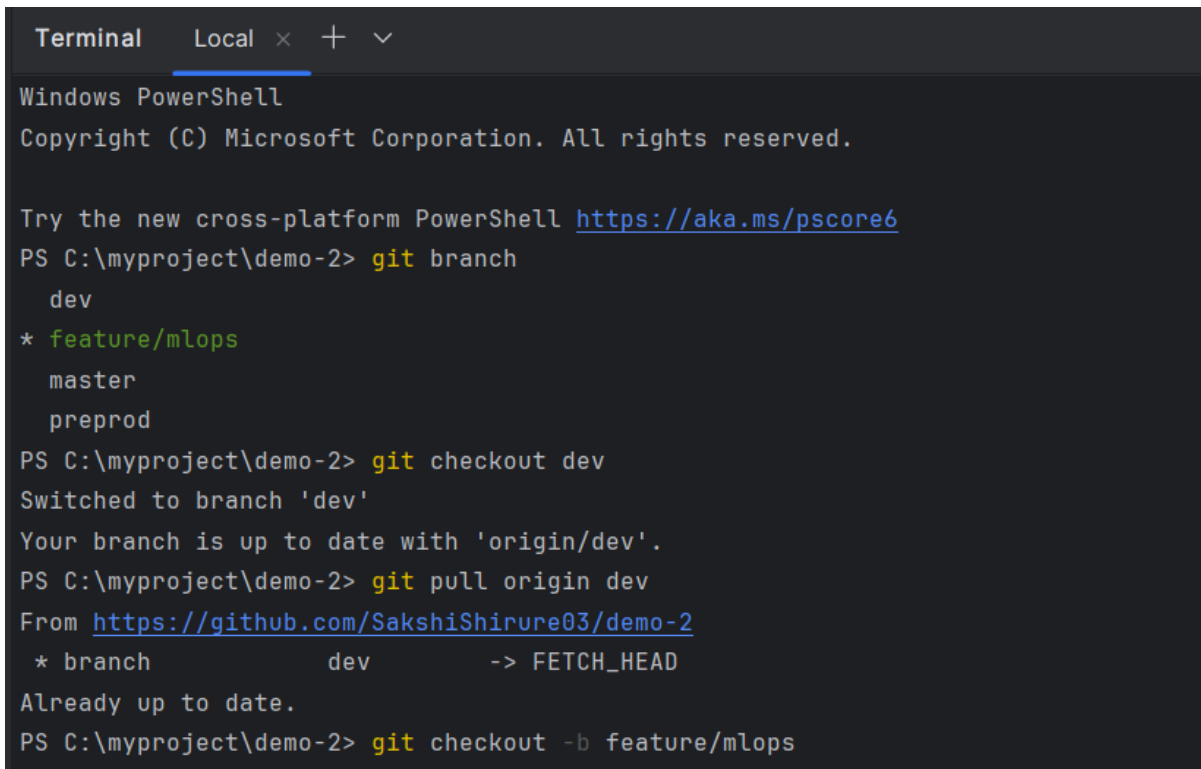
clean package

Advanced ▾

Post Steps

☐ Run only if build succeeds

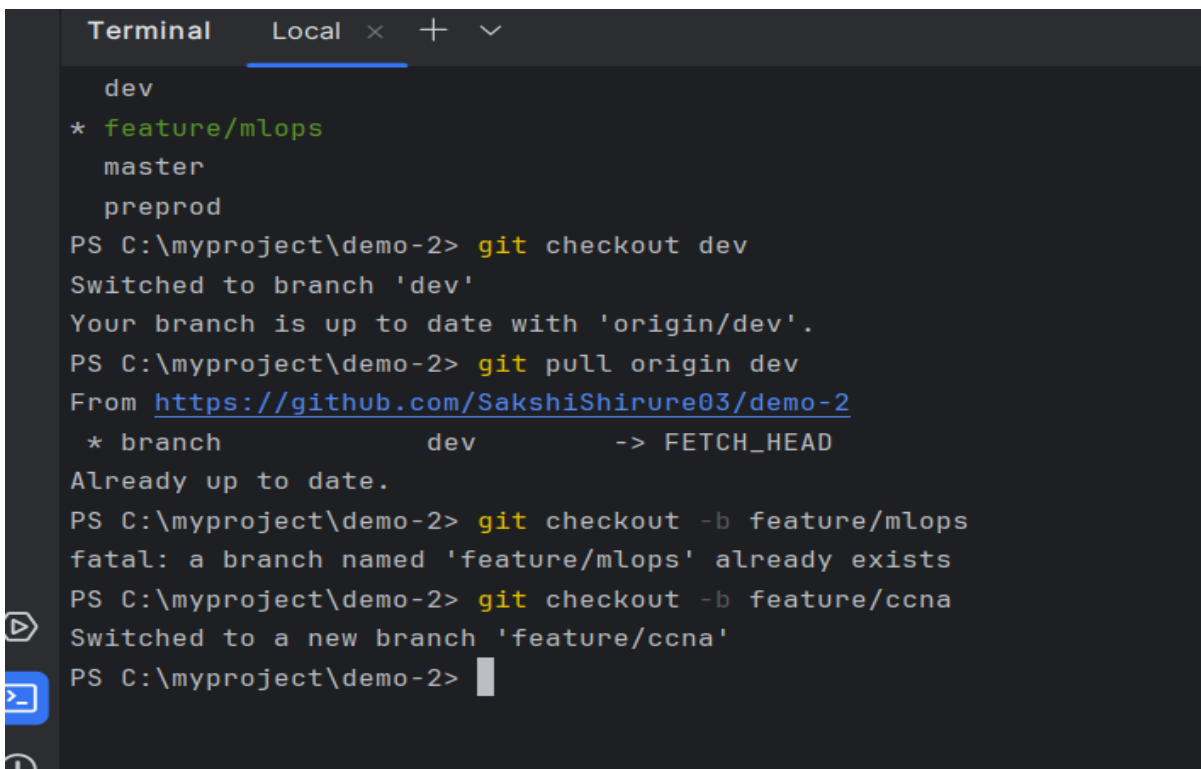
5) Now, open IntelliJ and check in which branch we are currently move to dev branch

A terminal window titled 'Terminal' with tabs 'Local', 'x', '+', and 'v'. The window shows the output of several git commands in a Windows PowerShell environment. The commands and their outputs are: 'git branch' showing 'dev', 'feature/mlops' (highlighted), 'master', and 'preprod'; 'git checkout dev' switching to the 'dev' branch; 'git pull origin dev' pulling from the remote repository, showing it is up to date; and 'git checkout -b feature/mlops' creating a new branch.

```
Terminal Local x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

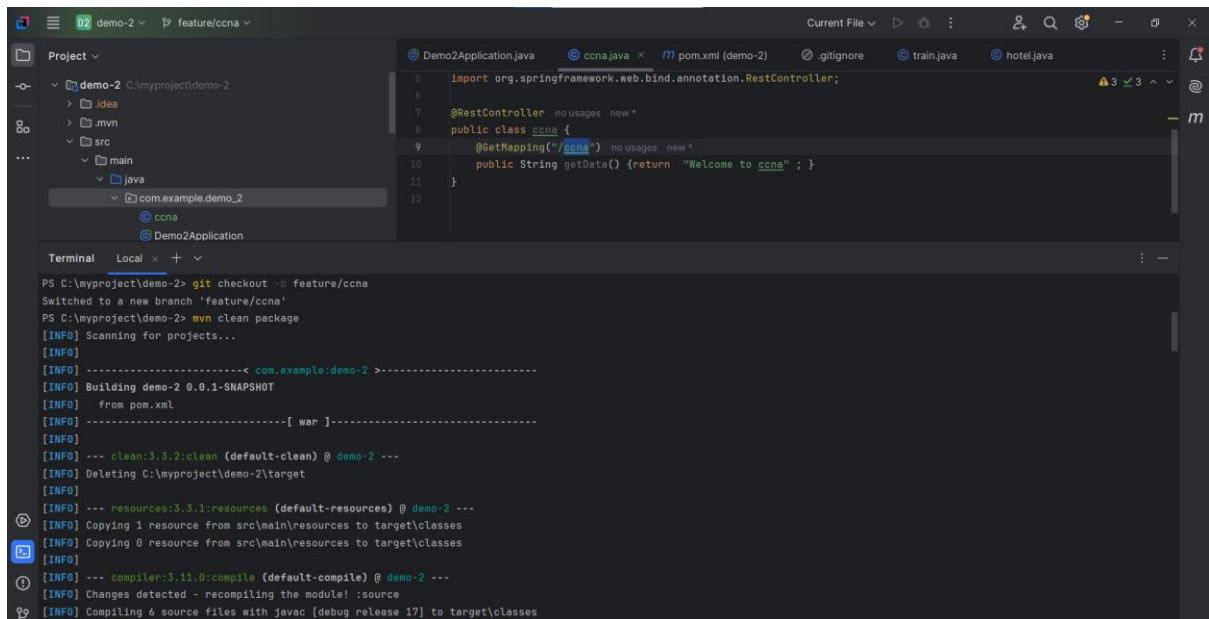
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\myproject\demo-2> git branch
dev
* feature/mlops
master
preprod
PS C:\myproject\demo-2> git checkout dev
Switched to branch 'dev'
Your branch is up to date with 'origin/dev'.
PS C:\myproject\demo-2> git pull origin dev
From https://github.com/SakshiShirure03/demo-2
* branch dev -> FETCH_HEAD
Already up to date.
PS C:\myproject\demo-2> git checkout -b feature/mlops
```

6) Create one new branch as “feature/ccna” and create one java class for same.

A terminal window titled 'Terminal' with tabs 'Local', 'x', '+', and 'v'. The window shows the output of several git commands. The commands and their outputs are: 'git checkout dev' switching to the 'dev' branch; 'git pull origin dev' pulling from the remote repository, showing it is up to date; 'git checkout -b feature/mlops' which results in a 'fatal: a branch named 'feature/mlops' already exists' error; and 'git checkout -b feature/ccna' which successfully creates a new branch.

```
Terminal Local x + v
dev
* feature/mlops
master
preprod
PS C:\myproject\demo-2> git checkout dev
Switched to branch 'dev'
Your branch is up to date with 'origin/dev'.
PS C:\myproject\demo-2> git pull origin dev
From https://github.com/SakshiShirure03/demo-2
* branch dev -> FETCH_HEAD
Already up to date.
PS C:\myproject\demo-2> git checkout -b feature/mlops
fatal: a branch named 'feature/mlops' already exists
PS C:\myproject\demo-2> git checkout -b feature/ccna
Switched to a new branch 'feature/ccna'
PS C:\myproject\demo-2> 
```

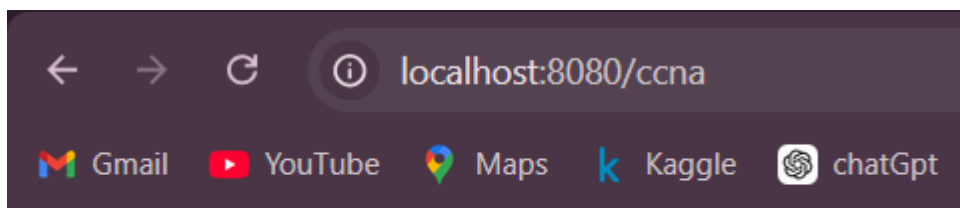
7) Check git status and do mvn clean package command, It will compile successfully



The screenshot shows an IDE with a project named 'demo-2'. The 'Project' view on the left shows the directory structure: 'demo-2' (C:\myproject\demo-2) containing 'idea', 'mvn', 'src', 'main' (containing 'java' with 'com.example.demo.2'), 'ccna', and 'Demo2Application'. The 'ccna.java' file is open in the editor, showing a REST controller with a single endpoint: `@GetMapping("/")` returning "Welcome to ccna". The 'Terminal' view at the bottom shows the execution of `git checkout -b feature/ccna` and `mvn clean package`. The Maven build output indicates a successful compilation and packaging of the application into a WAR file.

```
PS C:\myproject\demo-2> git checkout -b feature/ccna
Switched to a new branch 'feature/ccna'
PS C:\myproject\demo-2> mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.example.demo-2 >-----
[INFO] Building demo-2 0.0.1-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ war ]-----
[INFO]
[INFO] --- clean:3.3.2:clean (default-clean) @ demo-2 ---
[INFO] Deleting C:\myproject\demo-2\target
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ demo-2 ---
[INFO] Copying 1 resource from src\main\resources to target\classes
[INFO] Copying 0 resource from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.11.0:compile (default-compile) @ demo-2 ---
[INFO] Changes detected - recompiling the module! :source
[INFO] Compiling 6 source files with javac [debug release 17] to target\classes
```

8) We can see output here

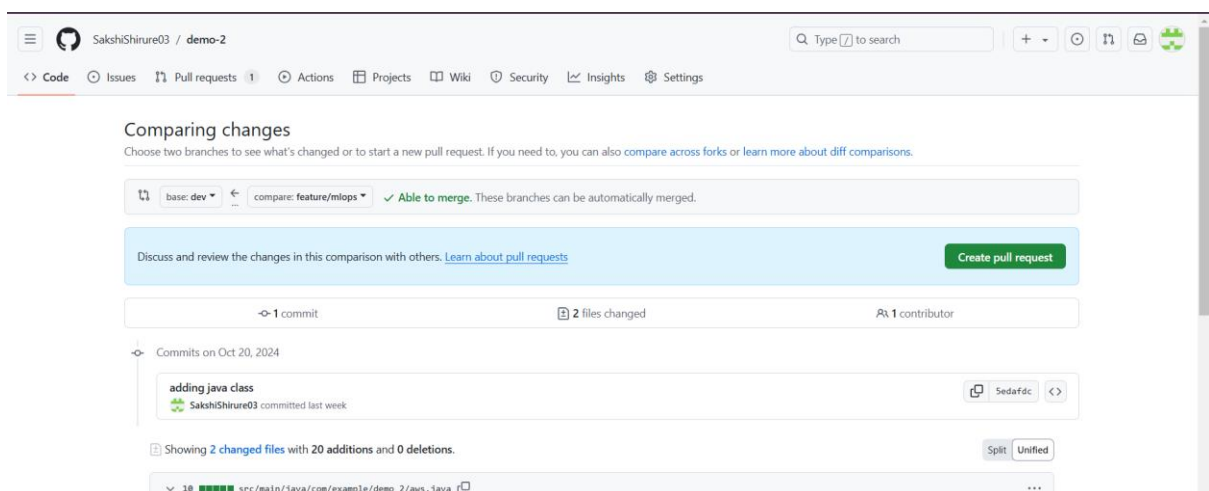


Welcome to ccna

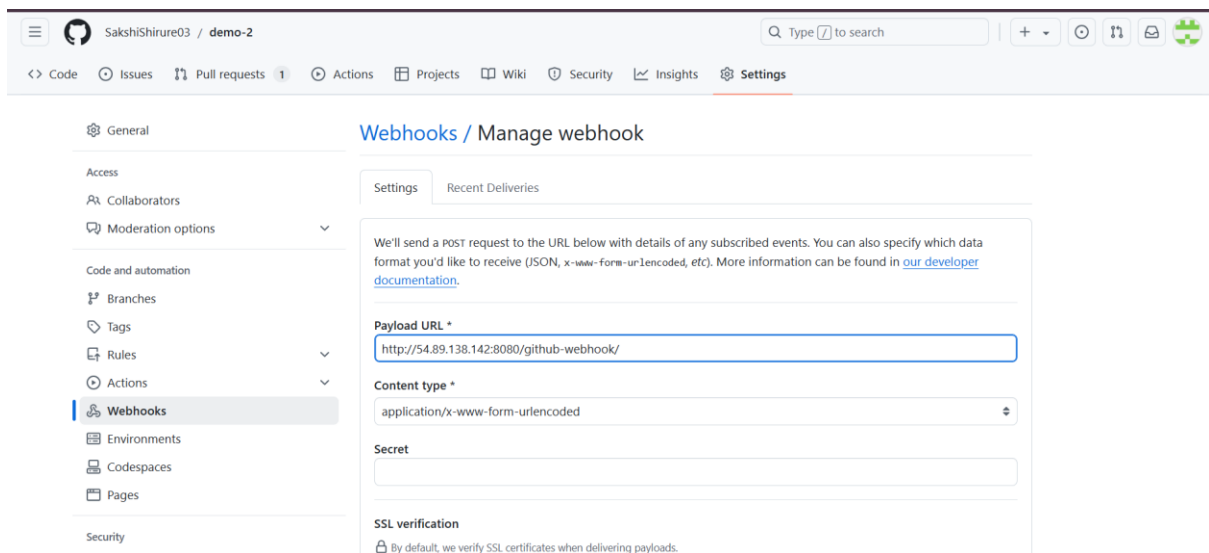
9) Now pass one message and push code to “feature/ccna”

```
Terminal Local x + v
931)
2024-10-29T00:29:34.997+05:30 INFO 1868 --- [demo-2] [nio-8080-exec-1] o.a.c.c.␣.␣.[Tomcat].␣.[localhost].␣.[/]
2024-10-29T00:29:34.997+05:30 INFO 1868 --- [demo-2] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
2024-10-29T00:29:34.997+05:30 INFO 1868 --- [demo-2] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 03:20 min
[INFO] Finished at: 2024-10-29T00:32:21+05:30
[INFO] -----
Terminate batch job (Y/N)?
Terminate batch job (Y/N)? y
PS C:\myproject\demo-2>
PS C:\myproject\demo-2> git add --all
PS C:\myproject\demo-2> git commit -m "adding ccna class"
[feature/ccna 0302680] adding ccna class
2 files changed, 13 insertions(+)
create mode 100644 src/main/java/ccna.java
create mode 100644 src/main/java/com/example/demo_2/ccna.java
PS C:\myproject\demo-2>
```

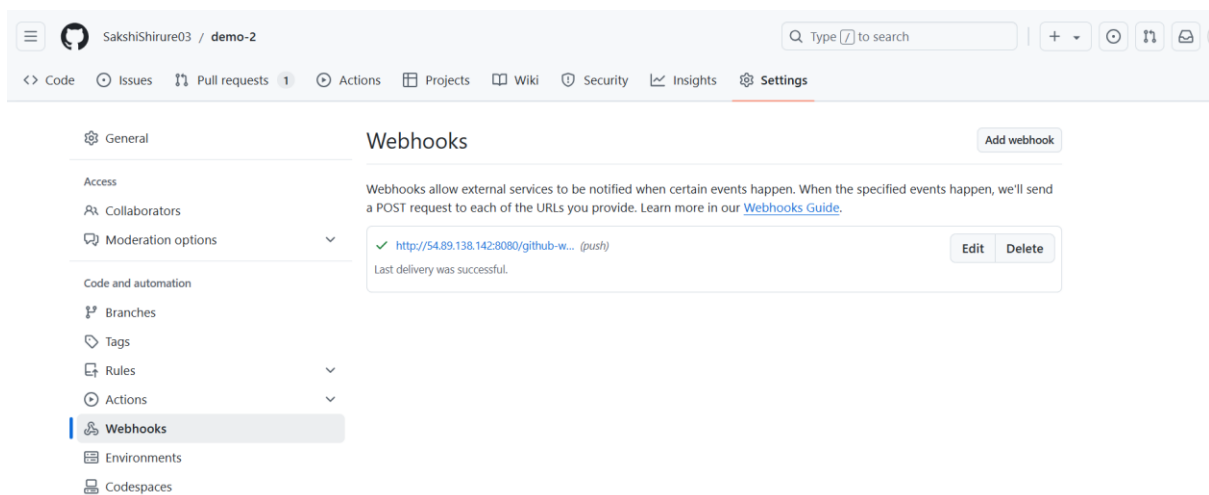
10) Here we can get popup and create pull request of (dev → feature/mlogs)



## 11) Now, Create workbooks to access code in Jenkins application

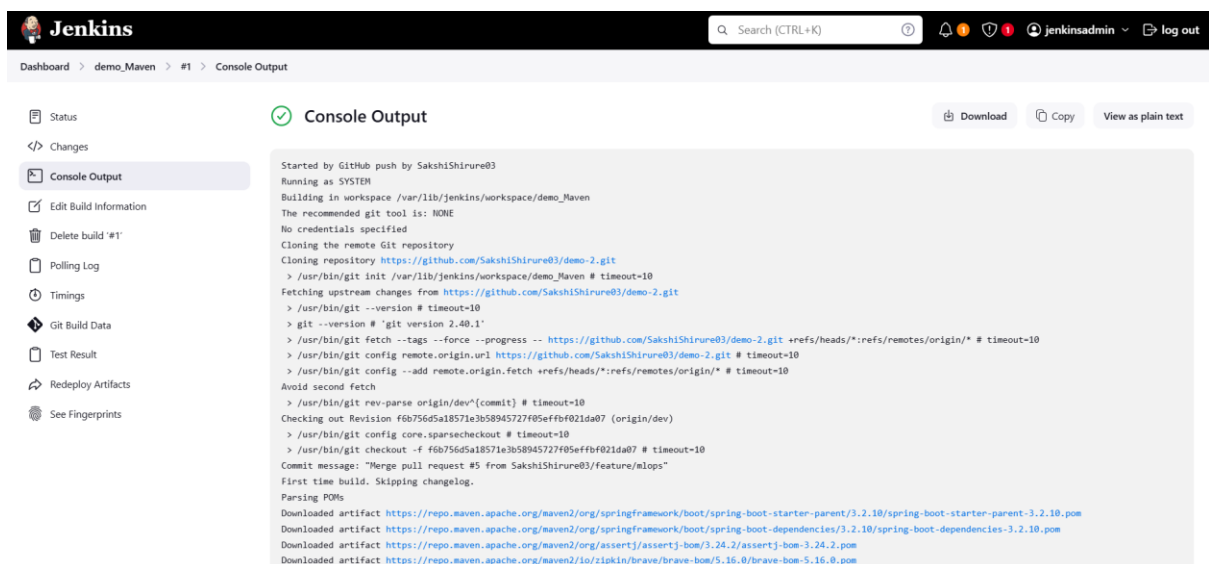


The screenshot shows the Jenkins 'Webhooks / Manage webhook' configuration page. The left sidebar contains a menu with options like General, Access, Collaborators, Moderation options, Code and automation, Branches, Tags, Rules, Actions, Webhooks (selected), Environments, Codespaces, Pages, and Security. The main content area has two tabs: 'Settings' and 'Recent Deliveries'. The 'Settings' tab is active, showing a form to configure a webhook. The 'Payload URL' field is filled with 'http://54.89.138.142:8080/github-webhook/'. The 'Content type' is set to 'application/x-www-form-urlencoded'. There is a 'Secret' field and an 'SSL verification' checkbox which is checked. A note at the bottom states: 'By default, we verify SSL certificates when delivering payloads.'



The screenshot shows the Jenkins 'Webhooks' list page. The left sidebar is the same as the previous screenshot. The main content area has a title 'Webhooks' and an 'Add webhook' button. Below the title, there is a list of webhooks. One webhook is listed with the URL 'http://54.89.138.142:8080/github-w...' and the event '(push)'. Below the URL, it says 'Last delivery was successful.' There are 'Edit' and 'Delete' buttons next to the webhook entry.

## 13) after that when we merge branch job should run/trigger



The screenshot shows the Jenkins 'Console Output' page for a job named 'demo\_Maven'. The left sidebar contains a menu with options like Status, Changes, Console Output (selected), Edit Build Information, Delete build '#1', Polling Log, Timings, Git Build Data, Test Result, Redeploy Artifacts, and See Fingerprints. The main content area shows the console output of the build. The output starts with 'Started by GitHub push by SakshiShirure03' and 'Running as SYSTEM'. It shows the process of cloning the repository, fetching upstream changes, and checking out the revision. The build is successful, and the console output ends with 'Parsing POMs' and 'Downloaded artifact' links.

## 14) We can see in Jenkins your job have run

The screenshot shows the Jenkins dashboard for a project named 'demo\_Maven'. The top navigation bar includes the Jenkins logo and a search bar. Below the navigation bar, the breadcrumb trail reads 'Dashboard > demo\_Maven >'. The main content area is titled 'Maven project demo\_Maven'. On the left sidebar, there are several menu items: 'Status' (selected), 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Maven project', 'Modules', 'GitHub Hook Log', and 'Rename'. The main content area displays the 'Latest Test Result (no failures)' with a document icon. Below this, there is a section for 'Permalinks' with a list of links: 'Last build (#1), 5 min 19 sec ago', 'Last stable build (#1), 5 min 19 sec ago', 'Last successful build (#1), 5 min 19 sec ago', and 'Last completed build (#1), 5 min 19 sec ago'. At the bottom, there is a 'Build History' section with a 'trend' dropdown and a 'Filter...' input field. The build history shows a single build, '#1', dated 'Oct 28, 2024, 7:21 PM'. Below the build history, there are two links: 'Atom feed for all' and 'Atom feed for failures'.

## 15) see logs of output

The screenshot shows the Jenkins console output for a build. The breadcrumb trail reads 'Dashboard > demo\_Maven > #1 > Console Output'. The main content area is titled 'Console Output' and shows the following log:

```
Started by GitHub push by SakshiShirure03
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/demo_Maven
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/SakshiShirure03/demo-2.git
> /usr/bin/git init /var/lib/jenkins/workspace/demo_Maven # timeout=10
Fetching upstream changes from https://github.com/SakshiShirure03/demo-2.git
> /usr/bin/git --version # timeout=10
> git --version # 'git version 2.40.1'
> /usr/bin/git fetch --tags --force --progress -- https://github.com/SakshiShirure03/demo-2.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> /usr/bin/git config remote.origin.url https://github.com/SakshiShirure03/demo-2.git # timeout=10
> /usr/bin/git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> /usr/bin/git rev-parse origin/dev^{commit} # timeout=10
Checking out Revision f6b756d5a18571e3b58945727f05effbf021da07 (origin/dev)
> /usr/bin/git config core.sparsecheckout # timeout=10
> /usr/bin/git checkout -f f6b756d5a18571e3b58945727f05effbf021da07 # timeout=10
Commit message: "Merge pull request #5 from SakshiShirure03/feature/nlops"
First time build. Skipping changelog.
Parsing POMs
Downloaded artifact https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-starter-parent/3.2.10/spring-boot-starter-parent-3.2.10.pom
Downloaded artifact https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-dependencies/3.2.10/spring-boot-dependencies-3.2.10.pom
Downloaded artifact https://repo.maven.apache.org/maven2/org/assertj/assertj-bom/3.24.2/assertj-bom-3.24.2.pom
Downloaded artifact https://repo.maven.apache.org/maven2/io/zipkin/brave/brave-bom/5.16.0/brave-bom-5.16.0.pom
Downloaded artifact https://repo.maven.apache.org/maven2/org/zipkin/reporter2/zipkin-reporter-bom/2.16.3/zipkin-reporter-bom-2.16.3.pom
Downloaded artifact https://repo.maven.apache.org/maven2/com/datastax/oss/java-driver-bom/4.17.0/java-driver-bom-4.17.0.pom
Downloaded artifact https://repo.maven.apache.org/maven2/io/dropwizard/metrics/metrics-bom/4.2.27/metrics-bom-4.2.27.pom
```

## 16) Now, we will make repository as private

demo-2 Private

Your master branch isn't protected

Protect this branch from force pushing or deletion, or require status checks before merging. [View documentation](#) [Protect this branch](#)

master 6 Branches Tags

Go to file Add file Code

SakshiShirure03 Merge pull request #6 from SakshiShirure03/dev 1395478 · 14 minutes ago 7 Commits

.mvn/wrapper	hello	3 weeks ago
src	adding java class	last week
.gitignore	hello	3 weeks ago
mvnw	hello	3 weeks ago
mvnw.cmd	hello	3 weeks ago

About

No description, website, or top

Activity

0 stars

1 watching

0 forks

Releases

No releases published

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

## 17) when we make repo as private we cannot access in Jenkins for that we need to create creds

Jenkins

Search (CTRL+K)

jenkinsadmin

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted)

Global credentials (unrestricted) [Add Credentials](#)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
This credential domain is empty. How about <a href="#">adding some credentials</a> ?			

Icon: S M L

## 18) From Github we will create one Token

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

MyGithubToken

What's this token for?

Expiration \*

30 days The token will expire on Thu, Nov 28 2024

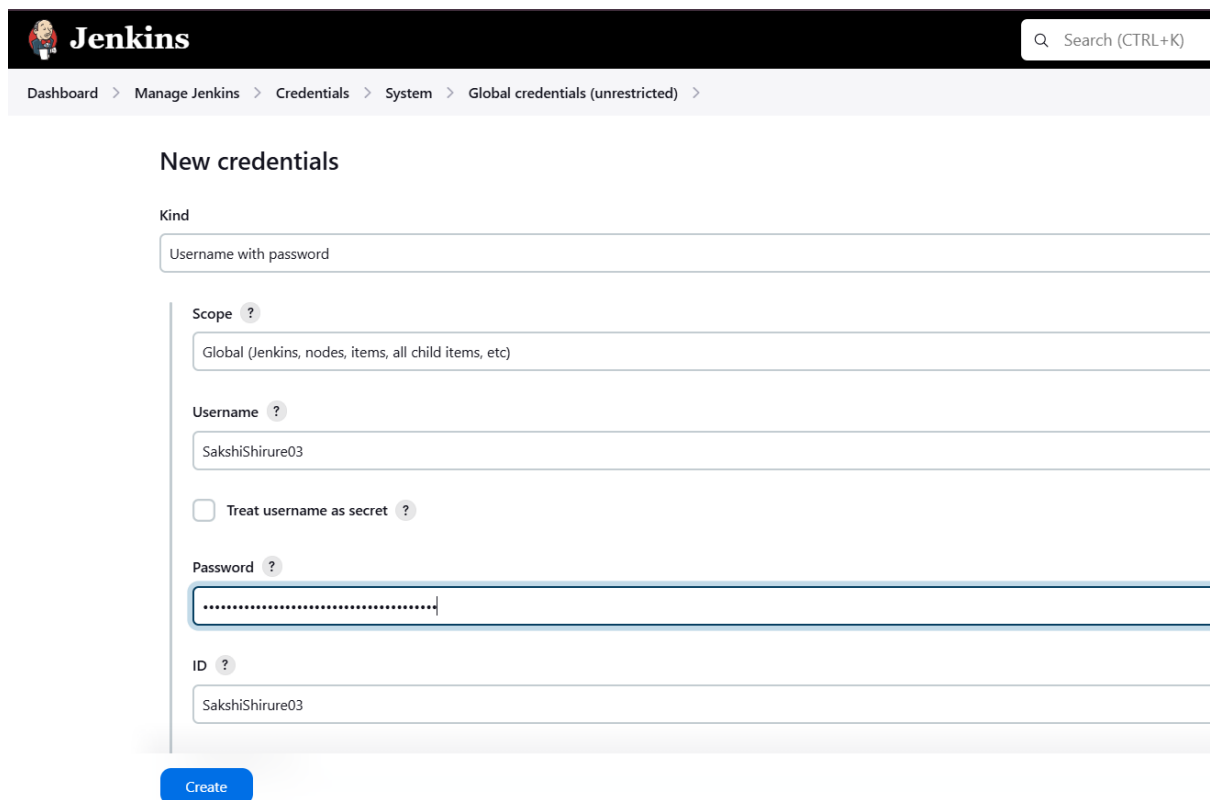
Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows



19) paste here password which we get from Token

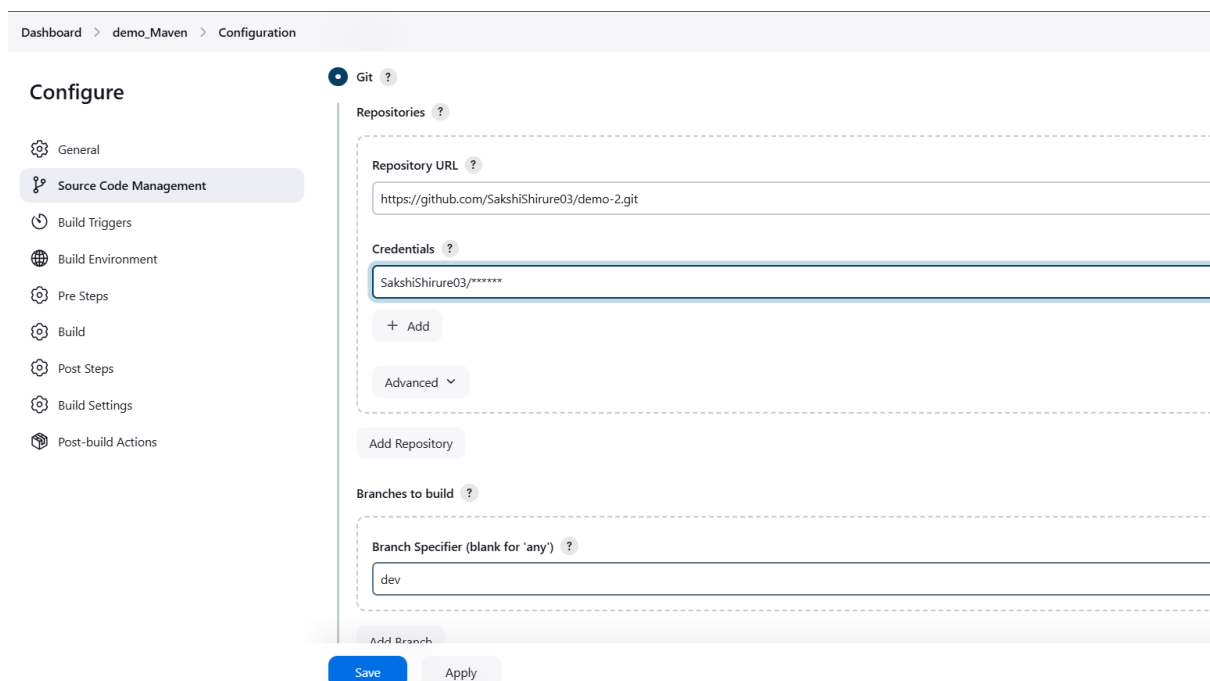


The image shows the 'New credentials' form in Jenkins. The breadcrumb trail is 'Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted)'. The form has the following fields:

- Kind:** A dropdown menu with 'Username with password' selected.
- Scope:** A dropdown menu with 'Global (Jenkins, nodes, items, all child items, etc)' selected.
- Username:** A text input field containing 'SakshiShirure03'.
- Treat username as secret:** An unchecked checkbox.
- Password:** A password input field with masked characters (dots).
- ID:** A text input field containing 'SakshiShirure03'.

A blue 'Create' button is located at the bottom left of the form.

20) Now we can add creds in Jenkins job




The image shows the 'Configuration' page for a Jenkins job named 'demo\_Maven'. The breadcrumb trail is 'Dashboard > demo\_Maven > Configuration'. The left sidebar shows the 'Configure' section with 'Source Code Management' selected. The main content area is for 'Git' and includes the following sections:

- Repositories:** A section with a 'Repository URL' field containing 'https://github.com/SakshiShirure03/demo-2.git' and a 'Credentials' dropdown menu with 'SakshiShirure03/\*\*\*\*\*' selected. There are '+ Add' and 'Advanced' buttons below.
- Branches to build:** A section with a 'Branch Specifier (blank for 'any')' field containing 'dev'.

At the bottom, there are 'Add Repository', 'Add Branch', 'Save', and 'Apply' buttons.

21) Job have been run successfully

 **Jenkins**

Dashboard > demo\_Maven > #2

Status

</> Changes

📄 Console Output

📝 Edit Build Information

🗑️ Delete build '#2'

🕒 Timings

🔗 Git Build Data

📄 Test Result

🔄 Redeploy Artifacts

👤 See Fingerprints

⬅️ Previous Build

✅ #2 (Oct 28, 2024, 7:52:31 PM)

🕒

Started by user [jenkinsadmin](#)

🕒

This run spent:

- 3 ms waiting;
- 16 sec build duration;
- 16 sec total from scheduled to completion.

🔗 git

Revision: f6b756d5a18571e3b58945727f05effbf021da07

Repository: <https://github.com/SakshiShirure03/demo-2.git>

- origin/dev

📄

[Test Result](#) (no failures)

Module Builds

✅ [demo-2](#)

11 sec

Thank you!!!