



# Jenkins

Jeeva S. Chelladhurai  
CEO, Comorin Consulting Services  
+91 97319 77222  
[jeeva@comorin.co](mailto:jeeva@comorin.co)



## 6. Jenkins with Git

- Git Integration with Jenkins
- Integrating Jenkins with GitHub
- Integration with GitHub & Pull Request Configuration
  - Personal access token creation
- GitHub Repo Webhook Configuration





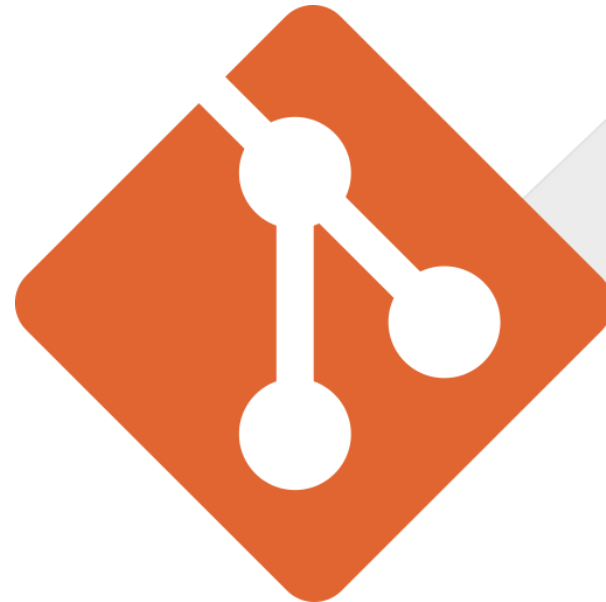
# Git Integration with Jenkins



To integrate Git with Jenkins we need to install Git plugin

**Steps:**

- Go to → Manage Jenkins → Manage plugins
- Click on Available Section → Select Git plugin & install



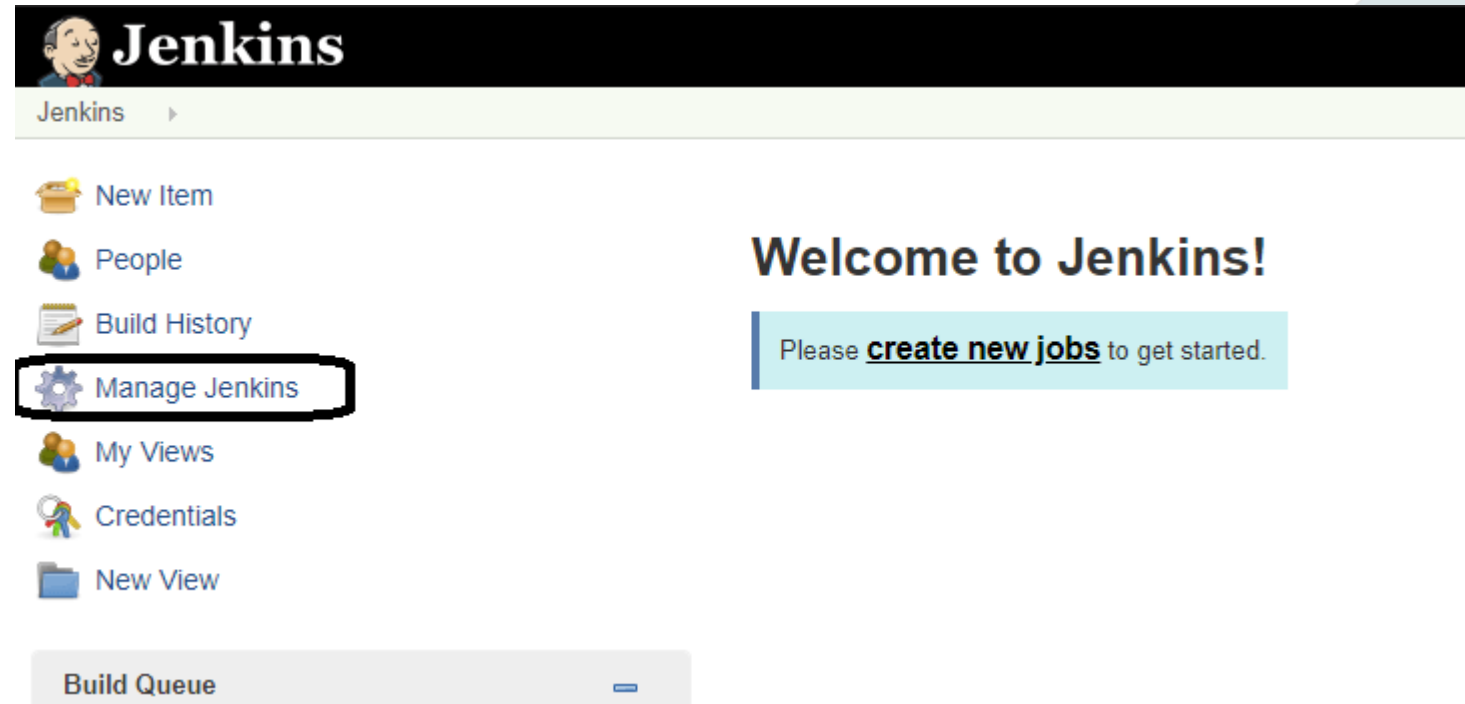


# Cont'd...



## Step 1

- Click on the **Manage Jenkins** button on your Jenkins dashboard







# Cont'd...

- **Step 2**
- Click on **Manage Plugins:**

**Jenkins**

Jenkins ▶

## Manage Jenkins

-  **Configure System**  
Configure global settings and paths.
-  **Configure Global Security**  
Secure Jenkins; define who is allowed to access/use the system.
-  **Configure Credentials**  
Configure the credential providers and types
-  **Global Tool Configuration**  
Configure tools, their locations and automatic installers.
-  **Reload Configuration from Disk**  
Discard all the loaded data in memory and reload everything from file system. Useful w
-  **Manage Plugins**  
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.  
🔔 There are updates available



## Cont'd...



### Step 3: In the Plugins Page

1. Select the GIT Plugin
2. Click on **Install without restart**
  - The plugin will take a few moments to finish downloading depending on your internet connection, and will be installed automatically
3. Select the option **Download now and Install after restart** button
  - In which plugin is installed after restart
4. You will be shown a "**No updates available**" message if you already have the Git plugin installed



# Cont'd...



<input type="checkbox"/>	<b>Team Concert Git Plugin</b> Integrates Jenkins with <a href="#">Rational Team Concert</a> for Jenkins Builds which use Git as source control. This plugin will create traceability links from a Jenkins build to Rational Team Concert <a href="#">Work Items</a> and <a href="#">build</a> results. This plugin adds traceability links from a Jenkins build to an RTC build result. It also publishes links to work items and annotates the change log generated by Jenkins with links to RTC Work Items; It leverages the current RTC features and workflows that users are already familiar with such as, emails, toaster popups, reporting, dashboards, etc.	1.0.9
<div>1</div> <input type="checkbox"/>	<b>Tracking Git Plugin</b> Lets one project track the Git revisions that are built for another project.	1.0
<input type="checkbox"/>	<b>Git Plugin</b> This plugin allows use of <a href="#">Git</a> as a build SCM. A recent Git runtime is required (1.7.9 minimum, 1.8.x recommended). Plugin is only tested on official <a href="#">git client</a> . Use exotic installations at your own risks.	2.3.5
<input type="checkbox"/>	<b>Repo Plugin</b> This plugin adds Repo ([ <a href="http://code.google.com/p/git-repo/">http://code.google.com/p/git-repo/</a> ]) as an SCM provider in Jenkins.	1.6
<input type="checkbox"/>	<b>Embeddable Build Status Plugin</b> This plugin allows Jenkins to expose the current status of your build as an image in a fixed URL. You can put this URL into other sites (such as GitHub README) so that people can see the current state of the job (last build) or for a specific build.	1.6

2

Install without restart

3

Download now and install after restart

4

Check now

Update information obtained: 10 min ago



# Cont'd...



## Step 4:

- Once the plugins have been installed, go to **Manage Jenkins** on your Jenkins dashboard
- You will see your plugins listed among the rest

<input checked="" type="checkbox"/>	<a href="#">GitHub plugin</a> This plugin integrates <a href="#">GitHub</a> to Jenkins.	<a href="#">1.29.1</a>	Uninstall
<input checked="" type="checkbox"/>	<a href="#">GitHub Branch Source Plugin</a> Multibranch projects and organization folders from GitHub. Maintained by CloudBees, Inc.	<a href="#">2.3.6</a>	Uninstall
<input checked="" type="checkbox"/>	<a href="#">GitHub API Plugin</a> This plugin provides <a href="#">GitHub API</a> for other plugins.	<a href="#">1.92</a>	Uninstall
<input checked="" type="checkbox"/>	<a href="#">GIT server Plugin</a> Allows Jenkins to act as a Git server.	<a href="#">1.7</a>	Uninstall
<input checked="" type="checkbox"/>	<a href="#">Git plugin</a> This plugin integrates <a href="#">Git</a> with Jenkins.	<a href="#">3.9.1</a>	Uninstall
<input checked="" type="checkbox"/>	<a href="#">Git client plugin</a> Utility plugin for Git support in Jenkins	<a href="#">2.7.2</a>	Uninstall



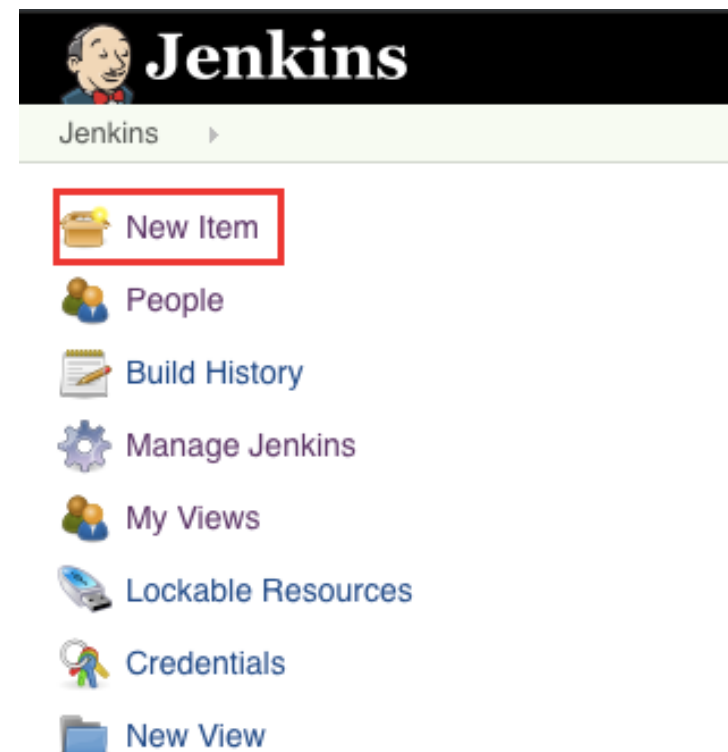


# Integrating Jenkins with GitHub



## Step 1:

- Create a new job in Jenkins, open the Jenkins dashboard with your Jenkins URL. For example, <http://localhost:8080/>
- Click on **create new jobs**





# Cont'd...



## Step 2:

- Enter the item name, select job, type & click **OK**
- We shall create a Freestyle project as an example

The screenshot shows the Jenkins 'Create New Item' interface. At the top, the Jenkins logo and navigation links are visible. The main section is titled 'Enter an item name' and contains a text input field with the value 'Demo'. Below this, there is a list of project types: 'Freestyle project', 'Pipeline', 'Multi-configuration project', 'Folder', 'GitHub Organization', and 'Multibranch Pipeline'. The 'Freestyle project' option is selected and highlighted with a red box. At the bottom, there is a section titled 'If you want to create a new item from other existing, you can use this option:' which contains an 'OK' button and a 'Copy from' dropdown menu. The 'OK' button is also highlighted with a red box.



# Cont'd...



## Step 3:

- Enter description as like below

The screenshot shows the Jenkins web interface. At the top, there's a black header with the Jenkins logo and a red square containing the number '2'. Below the header is a light green breadcrumb trail showing 'Jenkins > Demo >'. The main content area has a tabbed interface with 'General' selected. The 'Description' field is highlighted with a red box and contains the text 'Github integration demo project'. Below the field, there's a '[Plain text] [Preview](#)' link. A list of checkboxes is visible, all of which are unchecked:

- ☐ Discard old builds
- ☐ GitHub project
- ☐ This build requires lockable resources
- ☐ This project is parameterized
- ☐ Throttle builds
- ☐ Disable this project
- ☐ Execute concurrent builds if necessary
- ☐ Restrict where this project can be run

At the bottom right, there's an 'Advanced...' button and a vertical stack of help icons (question marks).



# Cont'd...



## Step 4:

- Next click on Source code management section
- Select Git & enter Git URL as shown
- Save & Apply

Jenkins > Demo >

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

**Source Code Management**

☐ None  
☒ Git

Repositories

Repository URL

Credentials

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

Add Branch

Repository browser

Additional Behaviours

☐ Subversion

**Build Triggers**

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

☐ Build after other projects are built

☐ Build periodically



# Cont'd...



## Step 5:

- To trigger build automatically, need to configure job as shown
- Testing Purpose: Set poll SCM to check repo every minute
- After every minute it will scan repo for changes, if found it triggers build

The screenshot shows the Jenkins configuration interface for a job named 'Subversion'. The 'Build Triggers' tab is selected. Under 'Build Triggers', the 'Poll SCM' option is checked, while others are unchecked. The 'Schedule' field contains '\*\*\*\*'. A warning message is displayed below the schedule field, advising that '\*\*\*\*' means every minute and suggesting 'H \*\*\*\*' for once per hour. The warning also shows the last and next run times as Sunday, August 4, 2019 2:51:04 PM GMT. At the bottom, the 'Ignore post-commit hooks' option is unchecked.

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

☐ Subversion

**Build Triggers**

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

☐ Build after other projects are built

☐ Build periodically

☐ GitHub hook trigger for GITScm polling

☒ Poll SCM

Schedule

\*\*\*\*

Do you really mean "every minute" when you say "\*\*\*\*"? Perhaps you meant "H \*\*\*\*" to poll once per hour

Would last have run at Sunday, August 4, 2019 2:51:04 PM GMT; would next run at Sunday, August 4, 2019 2:51:04 PM GMT.

Ignore post-commit hooks ☐



# Console Output



Jenkins ▸ Demo ▸ #2

Back to Project

Status

Changes

**Console Output**

View as plain text

Edit Build Information


Delete build '#2'

Polling Log

Git Build Data

No Tags

Previous Build

 **Console Output**

Started by an SCM change  
Running as SYSTEM  
Building on master in workspace /var/jenkins\_home/workspace/Demo  
No credentials specified

```
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/dhanushreemc/node-js-sample.git # timeout=10
Fetching upstream changes from https://github.com/dhanushreemc/node-js-sample.git
> git --version # timeout=10
> git fetch --tags --force --progress https://github.com/dhanushreemc/node-js-sample.git +refs/heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 05c52080f7bee8bc501ffca6b5bdd9005dellc77 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 05c52080f7bee8bc501ffca6b5bdd9005dellc77
Commit message: "Update Jenkinsfile"
> git rev-list --no-walk blaafc0d3f71cee5f2f64f16ac1d2bb8292dac8b9 # timeout=10
Finished: SUCCESS
```

**Note:** we need to provide GitHub credentials if the repository is private



# Integration with GitHub & Pull Request Configuration

---

- Building projects based on pull request is something you cannot avoid in CI/CD Pipelines
- Every team does several deployments/operations per day & lots of builds have to happen in this process
- The teams work on the same repo collaborating code require faster code integrations
- Better to have an automated build process that kicks off the CI/CD pipeline on a pull request rather than manually triggering the jobs



# Cont'd...



## **Step1:** Install GitHub Plugin

- Go to Manage Jenkins → Manage Plugins → Click on available plugins
- Search for GitHub plugin
- Select the plugin using checkbox and click on install without restart.

## **Step 2:**

- After restart we should see GitHub plugin in installed section as shown
- After successful installation select the restart checkbox as shown





# Cont'd...

## Step 3:

- Go to Manage Jenkins → Configure system
- Go to GitHub section & add credentials

GitHub

GitHub Servers

GitHub Server

Name

Personal\_Access\_Token\_User

API URL

https://api.github.com

Credentials

Personal\_Access\_Token

Add

Credentials verified for user dhanushreemc, rate limit: 4998

Manage hooks

☐

Test connection

Advanced...

Delete

Add GitHub Server

Save

Apply



# Cont'd...



## Step 4:

- Click on add credentials
- Select credentials type as secret text
- Copy paste GitHub personal\_access\_token in secret section as shown

Jenkins > Credentials > System > Global credentials (unrestricted)

[Back to credential domains](#)

[Add Credentials](#)

Kind: **Secret text**

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret:

ID:

Description:

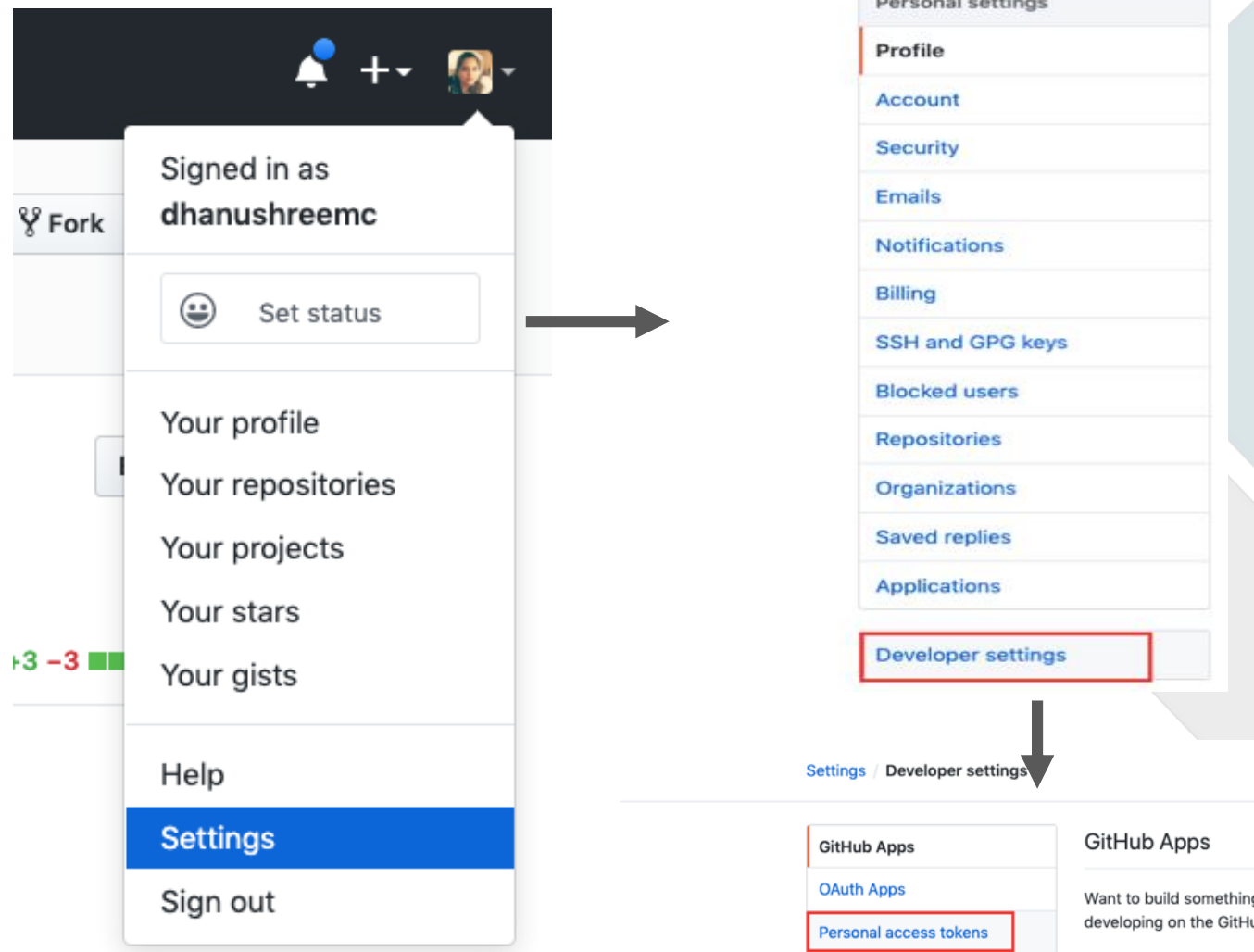
**OK**



# Personal access token creation

## Steps

- Go to GitHub account
- Click on ***Settings***
- Select ***Developer settings***
- Select **Personal access token**





# Cont'd...

- Add a personal access token as shown below

Settings / Developer settings

[GitHub Apps](#)  
[OAuth Apps](#)  
**Personal access tokens**

## Personal access tokens

Tokens you have generated that can be used to access the [GitHub API](#).

**git-jenkins** — *admin:gpg\_key, admin:org, admin:org\_hook, admin:public\_key, admin:repo\_hook, gist, notifications, read:packages, repo, user, write:discussion, write:packages* Last used within the last week

[Delete](#)

[Generate new token](#) [Revoke all](#)



# Cont'd...



- Enter for what purpose token is being created
- Select repo
- Click on save as shown

Settings / Developer settings

GitHub Apps

OAuth Apps

Personal access tokens

## New personal access token

Personal access tokens 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

personal access token

What's this token for?

### Select scopes

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

<input checked="" type="checkbox"/> <b>repo</b>	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> <b>admin:org</b>	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> <b>admin:public_key</b>	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys



# Cont'd...



- A token will be generated
- Copy it & save it as it will be generated one time

Settings / Developer settings

GitHub Apps

OAuth Apps

Personal access tokens

### Personal access tokens

Generate new token Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your new personal access token now. You won't be able to see it again!

✓ dbd9deb896d3f258e5f9acb6014e156515f787c3	Delete
<b>git-jenkins</b> — <i>admin:gpg_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, gist, notifications, read:packages, repo, user, write:discussion, write:packages</i>	Last used within the last week Delete
<b>create_version</b> — <i>repo</i>	Last used within the last 3 months Delete
<b>gitpush</b> — <i>repo</i>	Last used within the last week Delete
<b>Jenkins</b> — <i>repo</i>	Last used within the last week Delete

Personal access tokens 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](#).



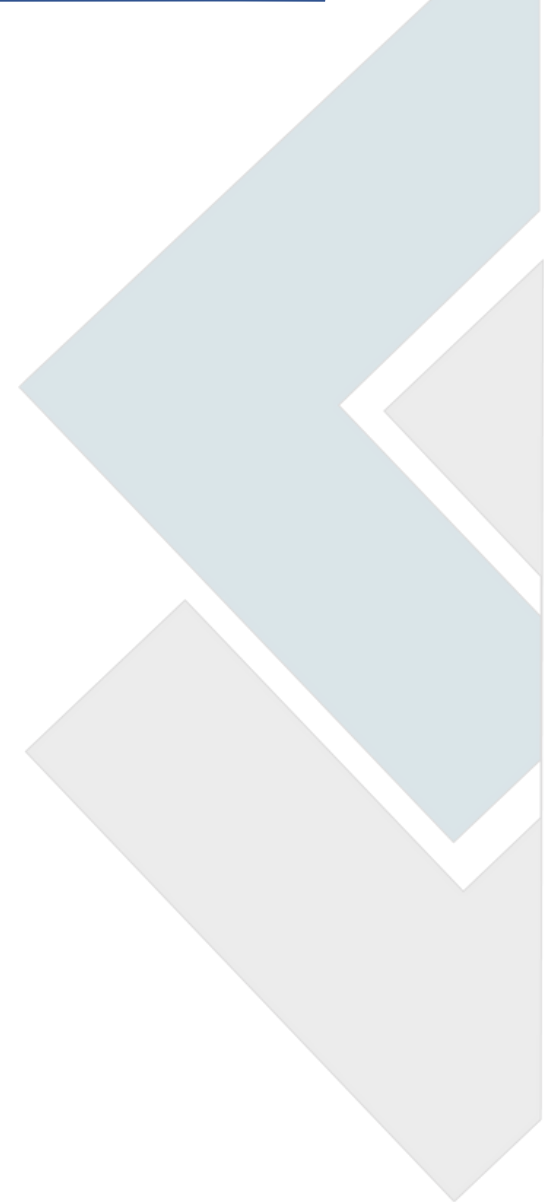
# Cont'd...

---



## Step 5:

- Select personal access token as GitHub credentials test for the connection as mentioned above
- Click on save & apply





# GitHub Repo Webhook Configuration



- For Jenkins to receive PR events through pull request plugin need to add the Jenkins pull request builder payload URL in the GitHub repository settings

## Steps:

- Go to GitHub repository settings
- Under **webhooks**, add the Jenkins pull request builder payload URL
- Following format

<http://<Jenkins-IP>:<port>/github-webhook/>

- Go to **Repository settings** in GitHub
- Select **webhooks**
- Click on add **webhook**
- Enter the webhook **URL** as shown





# Cont'd...



Okay, that hook was successfully created. We sent a ping payload to test it out! Read more about it at <https://developer.github.com/webhooks/#ping-event>. ✕

dhanushreemc / sample ≡

<> Code

! Issues 0

🔗 Pull requests 0

Z ZenHub

📁 Projects 0

📖 Wiki

🛡 Security

📊 Insights

⚙ Settings

Options

Collaborators

Branches

**Webhooks**

Notifications

Integrations & services

Deploy keys

Moderation

Interaction limits

## 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://hotspot.com/jenkins/github-webhook/> (push)

Edit

Delete

Add webhook

COMORIN CONSULTING SERVICES



# Cont'd...

- Check how Jenkins builds pull requests
- Check for the repo where Jenkins file is
- Configure a multi-branch or repository job for the repo

The screenshot shows the Jenkins 'Branch Sources' configuration page for a repository named 'Demo-GitOps'. The page is divided into several sections:

- General:** Includes fields for 'Credentials' (set to 'dhanushreemc/\*\*\*\*\*'), 'User' (dhanushreemc), 'Owner' (dhanushreemc), and 'Repository' (Demo-GitOps). There is an 'Add' button next to the Credentials field.
- Behaviors:** This section contains three expandable panels:
  - Discover branches:** Strategy is set to 'All branches'.
  - Discover pull requests from origin:** Strategy is set to 'The current pull request revision'.
  - Discover pull requests from forks:** Strategy is set to 'The current pull request revision' and Trust is set to 'From users with Admin or Write permission'. A note below states: 'May not be supported on older versions of GitHub Enterprise. See help button.'

At the bottom, there is a 'Property strategy' dropdown set to 'All branches get the same properties', and 'Save' and 'Apply' buttons.



# Cont'd...

- Check the job, Jenkins performed repository scan first time

Jenkins > sandbox > gitops > [ENABLE AUTO REFRESH](#)

[Up](#)  
[Status](#)  
[Configure](#)  
[Scan Repository Now](#)  
[Scan Repository Log](#)  
[Repository Events](#)  
[Delete Repository](#)  
[People](#)  
[Build History](#)  
[Project Relationship](#)  
[Check File Fingerprint](#)

## gitops

**Branches (3)** Pull Requests (0)

S	W	Name ↓	Last Success	Last Failure	Last Duration	Robot Results
		<a href="#">develop</a>	1 day 7 hr - <a href="#">#8</a>	N/A	3.1 sec	
		<a href="#">feature</a>	1 day 10 hr - <a href="#">#7</a>	N/A	3.1 sec	
		<a href="#">master</a>	1 day 11 hr - <a href="#">#1</a>	N/A	4 sec	

Icon: [S](#) [M](#) [L](#)

[Legend](#) [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)



# Cont'd...

- Now we will create a pull request from develop to master branch
- As soon as the pull request created the pull request got build by Jenkins

Jenkins > sandbox > gitops > [ENABLE AUTO REFRESH](#)

Up  
Status  
Configure  
Scan Repository Now  
Scan Repository Log  
Repository Events  
Delete Repository  
People  
Build History  
Project Relationship

### gitops

Branches (3) Pull Requests (1)

S	W	Name ↓	Last Success	Last Failure	Last Duration	Robot Results
		<a href="#">develop</a>	1 day 7 hr - <a href="#">#8</a>	N/A	3.1 sec	
		<a href="#">feature</a>	1 min 43 sec - <a href="#">#1</a>	N/A	2.9 sec	
		<a href="#">master</a>	1 min 43 sec - <a href="#">#1</a>	N/A	3.3 sec	

Icon: [S](#) [M](#) [L](#)

[Legend](#) [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

Jenkins > sandbox > gitops > Pull Requests (1) > [ENABLE AUTO REFRESH](#)

Up  
Status  
Configure  
Scan Repository Now  
Scan Repository Log  
Repository Events  
People  
Build History

### gitops

Branches (3) Pull Requests (1)

S	W	Name ↓	Last Success	Last Failure	Last Duration	Robot Results
		<a href="#">PR-12</a>	4 min 14 sec - <a href="#">#1</a>	N/A	4.1 sec	

Icon: [S](#) [M](#) [L](#)

[Legend](#) [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)



# Console



- Observe the pull request build has triggered by what branch indexing
- Each pull request treated as a new branch

```
Jenkins > sandbox > gitops > Pull Requests (1) > PR-12 > #1

Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete Build
Git Build Data
No Tags
Restart from Stage
Replay
Pipeline Steps

Console Output

Branch indexing
19:06:57 Connecting to https://api.github.com using dhanushreemc/*****
Obtained Jenkinsfile from ef3da26956c3c124d1c7521e047d16ba5b6fd7e5
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] node
Running on build-server in /home/ubuntu/workspace/sandbox_gitops_PR-12
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Cloning the remote Git repository
Cloning with configured refspecs honoured and without tags
Fetching without tags
Checking out Revision ef3da26956c3c124d1c7521e047d16ba5b6fd7e5 (PR-12)
Commit message: "updated"
First time build. Skipping changelog.
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (checkout)
[Pipeline] checkout
Fetching changes from the remote Git repository
Fetching without tags
Cloning repository https://github.com/dhanushreemc/Demo-GitOps.git
> git init /home/ubuntu/workspace/sandbox_gitops_PR-12 # timeout=10
Fetching upstream changes from https://github.com/dhanushreemc/Demo-GitOps.git
> git --version # timeout=10
using GIT_ASKPASS to set credentials
> git fetch --no-tags --progress https://github.com/dhanushreemc/Demo-GitOps.git +refs/pull/12/head:refs/remotes/origin/PR-12
> git config remote.origin.url https://github.com/dhanushreemc/Demo-GitOps.git # timeout=10
> git config --add remote.origin.fetch +refs/pull/12/head:refs/remotes/origin/PR-12 # timeout=10
> git config remote.origin.url https://github.com/dhanushreemc/Demo-GitOps.git # timeout=10
Fetching upstream changes from https://github.com/dhanushreemc/Demo-GitOps.git
using GIT_ASKPASS to set credentials
```



# Cont'd...

- Merge the pull request

**Open** **Develop #12**  
dhanushreemc wants to merge 15 commits into `master` from `develop`

Merge pull request #11 from dhanushreemc/feature  
updated  
resolved conflicts  
updated

Verified ✓ 3bf8c6e  
ee46352  
✓ b28ba7e  
✓ ef3da26

Releases  
Not inside a Release

Epics  
Not inside an Epic

Notifications [Customize](#)  
[Unsubscribe](#)  
You're receiving notifications because you authored the thread.

1 participant

[Lock conversation](#)

[Move Issue](#)

Add more commits by pushing to the **develop** branch on **dhanushreemc/Demo-GitOps**.

**All checks have passed** [Hide all checks](#)  
3 successful checks

- ✓ **ci/jenkins/build-status** — Build succeeded [Details](#)
- ✓ **continuous-integration/jenkins/branch** — This commit looks good [Details](#)
- ✓ **continuous-integration/jenkins/pr-head** — This commit looks good [Details](#)

**This branch has no conflicts with the base branch**  
Merging can be performed automatically.

**Merge pull request** [You can also open this in GitHub Desktop or view command line instructions.](#)

**Connect this pull request with an existing issue** [Connect with an issue](#)

**Write** **Preview** [AA](#) [B](#) [i](#) [“](#) [<>](#) [🔗](#) [☰](#) [☰](#) [☑](#) [@](#) [📌](#) [↩](#)

Leave a comment



# Console



- Observe after pull request no-12 PR-12 merged to master
- Master branch triggered a new build, which we can observe in resent build console

The screenshot shows the Jenkins console output for a build. The left sidebar contains navigation links: Back to Project, Status, Changes, Console Output (selected), View as plain text, Edit Build Information, Delete Build, Git Build Data, No Tags, Restart from Stage, Replay, Pipeline Steps, and Previous Build. The main area displays the console output for the build. The output shows the build starting on the 'build-server' in the '/home/ubuntu/workspace/sandbox\_gitops\_master' directory. It then proceeds to checkout the SCM, clone the remote Git repository, and fetch changes from the remote repository. The output is as follows:

```
Push event to branch master
19:15:52 Connecting to https://api.github.com using dhanushreemc/*****
Obtained Jenkinsfile from b7768118409de6c2868b62ab0890b3c983c46cea
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] node
Running on build-server in /home/ubuntu/workspace/sandbox_gitops_master
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Cloning the remote Git repository
Cloning with configured refspecs honoured and without tags
Fetching without tags
Checking out Revision b7768118409de6c2868b62ab0890b3c983c46cea (master)
Commit message: "Merge pull request #12 from dhanushreemc/develop"
[Pipeline] {
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (checkout)
[Pipeline] checkout
Fetching changes from the remote Git repository
Fetching without tags
Cloning repository https://github.com/dhanushreemc/Demo-GitOps.git
> git init /home/ubuntu/workspace/sandbox_gitops_master # timeout=10
Fetching upstream changes from https://github.com/dhanushreemc/Demo-GitOps.git
> git --version # timeout=10
using GIT_ASKPASS to set credentials
> git fetch --no-tags --progress https://github.com/dhanushreemc/Demo-GitOps.git +refs/heads/master:refs/remotes/origin/master
> git config remote.origin.url https://github.com/dhanushreemc/Demo-GitOps.git # timeout=10
> git config remote.origin.url https://github.com/dhanushreemc/Demo-GitOps.git # timeout=10
Fetching upstream changes from https://github.com/dhanushreemc/Demo-GitOps.git
using GIT_ASKPASS to set credentials
> git fetch --no-tags --progress https://github.com/dhanushreemc/Demo-GitOps.git +refs/heads/master:refs/remotes/origin/master
```