Jenkins

Jeeva S. Chelladhurai
CEO, Comorin Consulting Services
+91 97319 77222
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

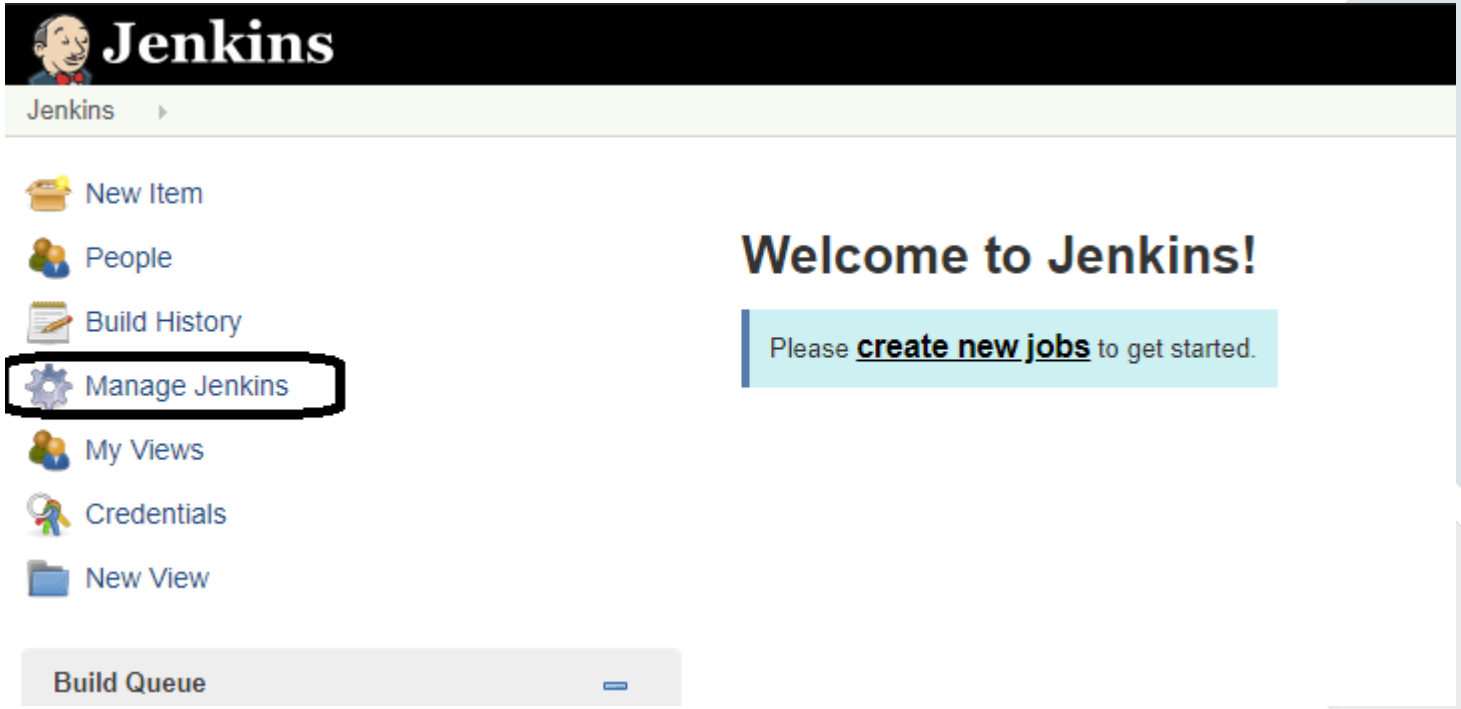
Steps:

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

Cont'd…

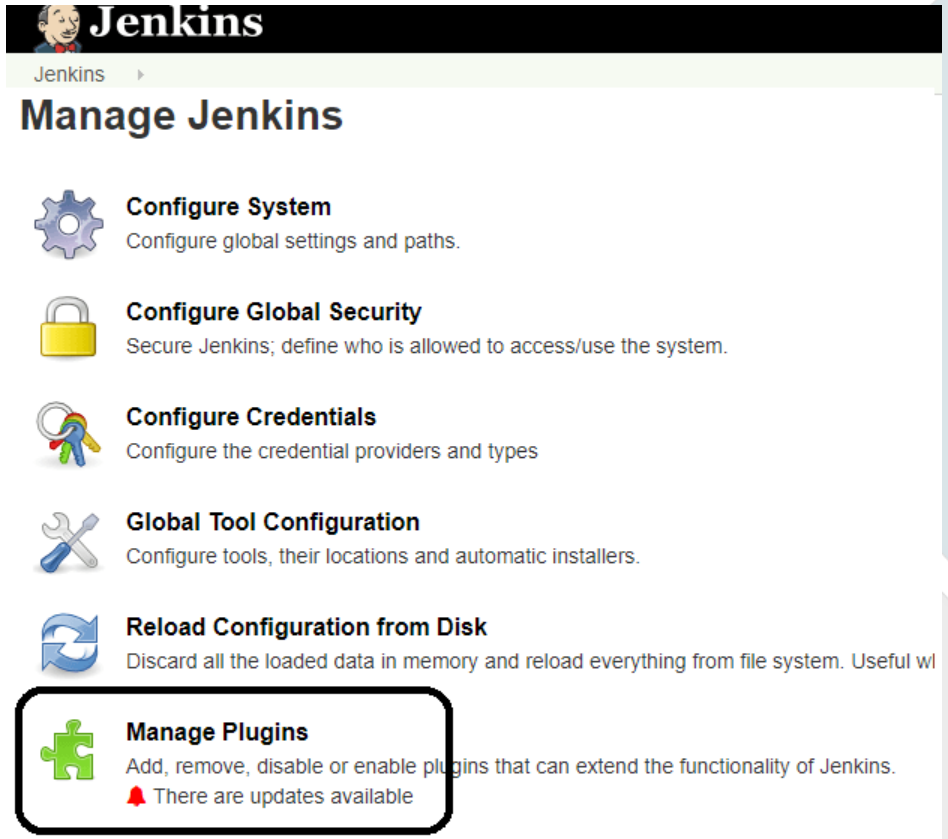
Step 1

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



Cont'd…

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

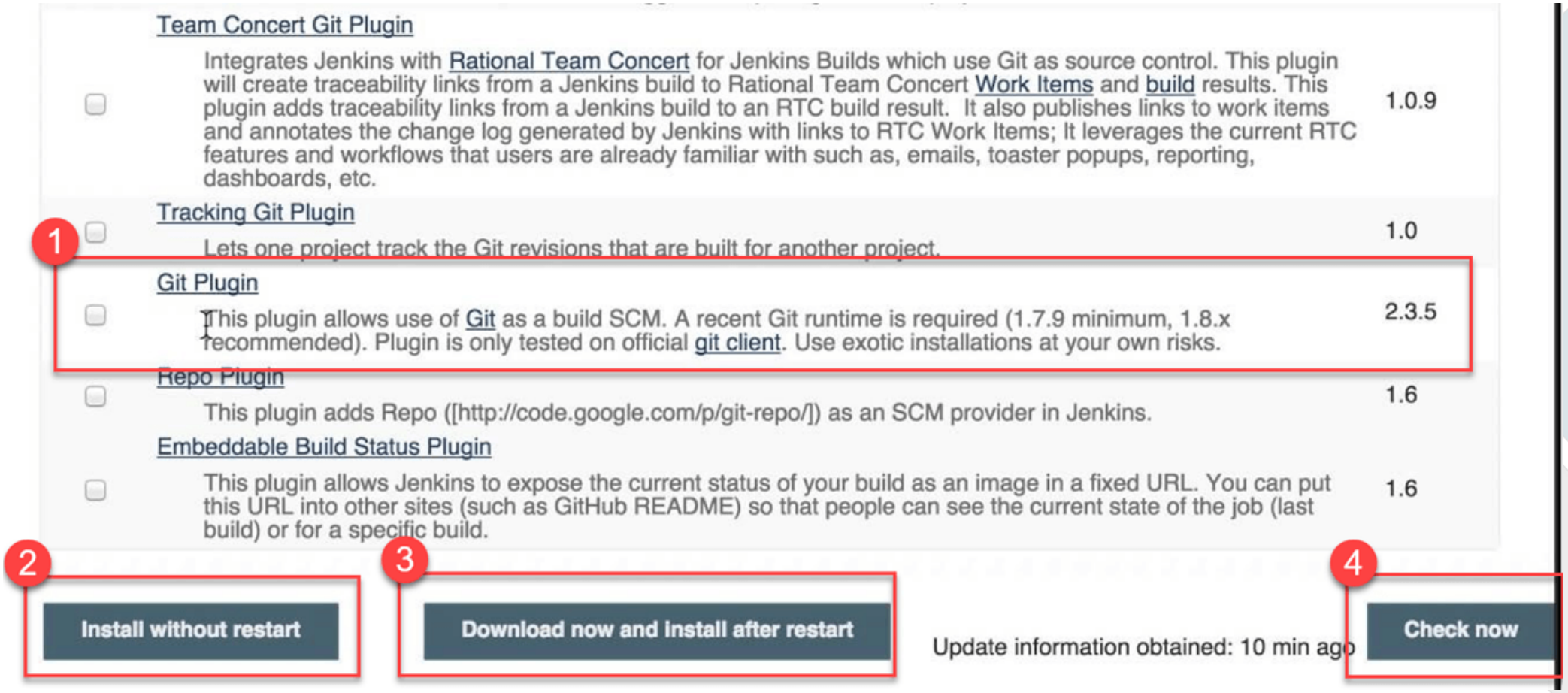


Cont'd…

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

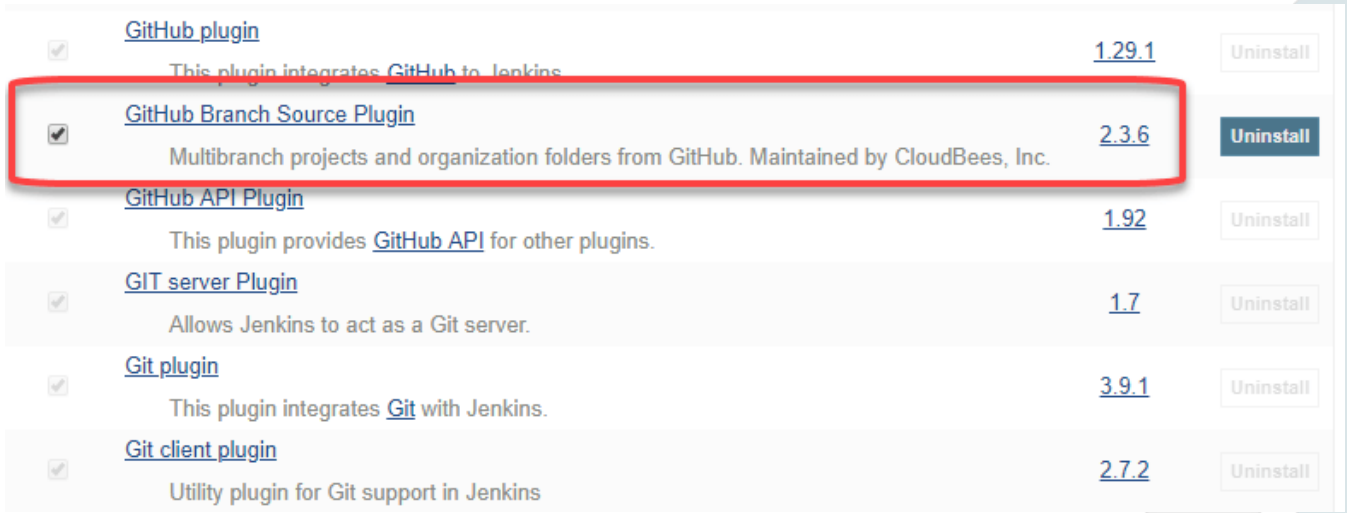
Cont'd…


Team Concert Git Plugin
Integrates Jenkins with Rational Team Concert for Jenkins Builds which use Git as source control. This plugin will create traceability links from a Jenkins build to Rational Team Concert Work Items and build 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
Tracking Git Plugin
Lets one project track the Git revisions that are built for another project.
1.0
1
Git Plugin
This plugin allows use of Git as a build SCM. A recent Git runtime is required (1.7.9 minimum, 1.8.x recommended). Plugin is only tested on official git client. Use exotic installations at your own risks.
2.3.5
Repo Plugin
This plugin adds Repo ([http://code.google.com/p/git-repo/]) as an SCM provider in Jenkins.
1.6
Embeddable Build Status Plugin
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
Update information obtained: 10 min ago
4
Check now

Cont'd…

Step 4:

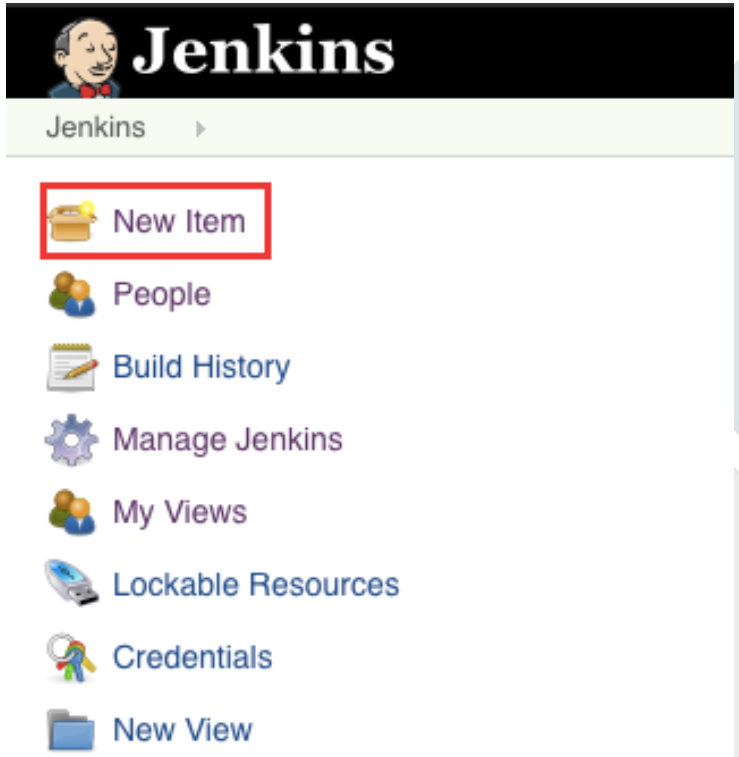
- Once the plugins have been installed, go to **Manage Jenkins** on your Jenkins dashboard
- You will see your plugins listed among the rest



Integrating Jenkins with GitHub

Step 1:

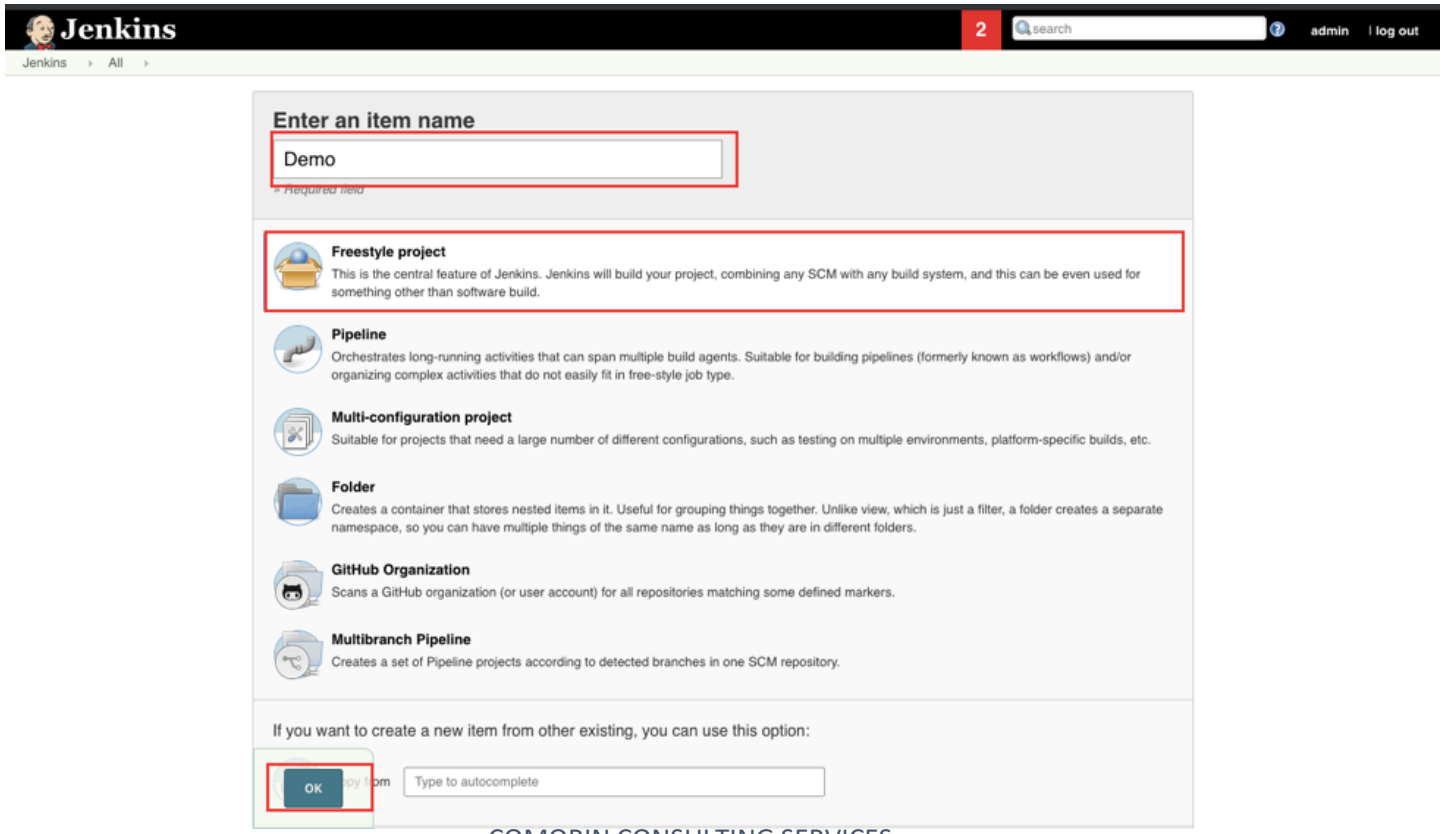
- Create a new job in Jenkins, open the Jenkins dashboard with your Jenkins URL. For example, http://localhost:8080/
- Click on **create new jobs**



Cont'd…

Step 2:

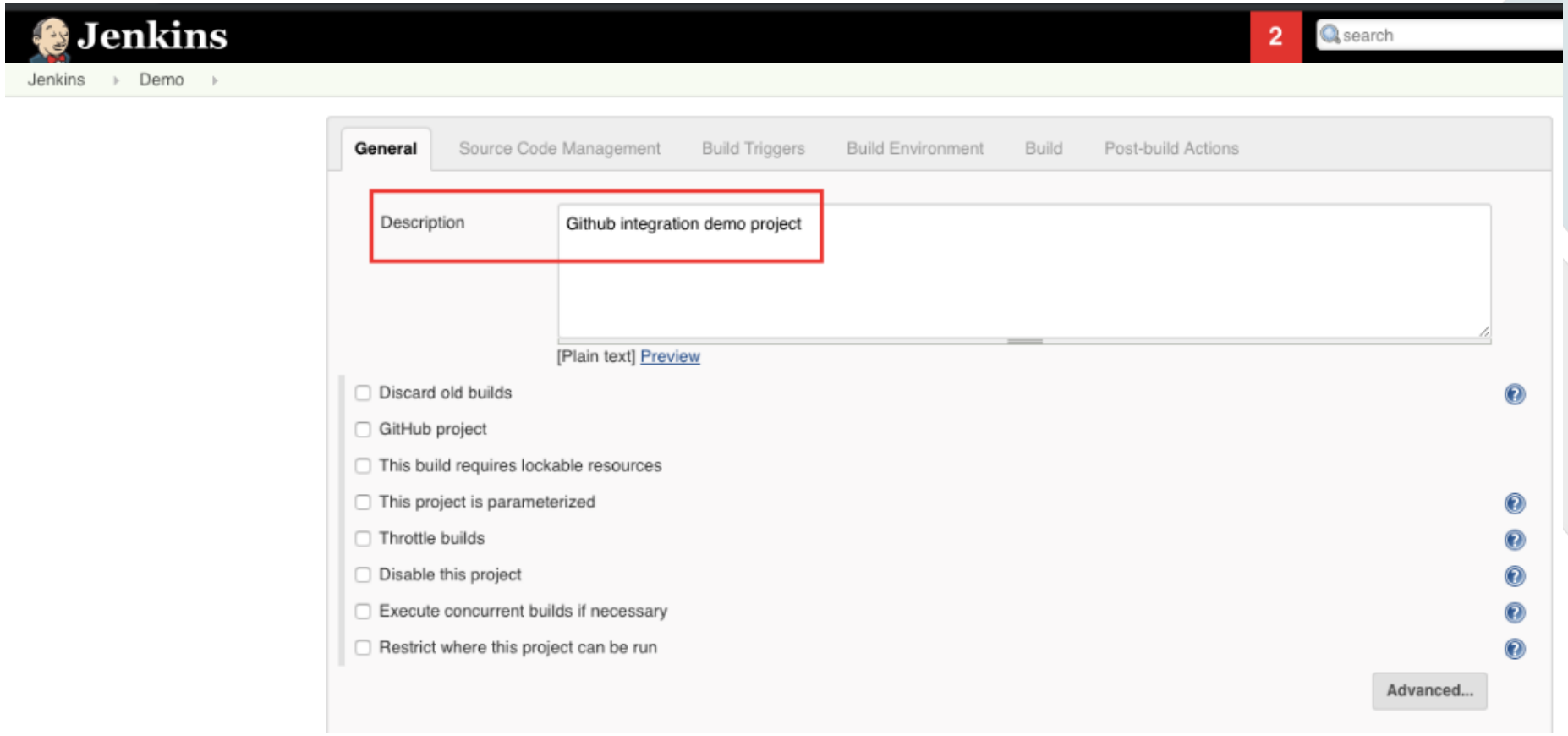
- Enter the item name, select job, type & click **OK**
- We shall create a Freestyle project as an example



Cont'd…

Step 3:

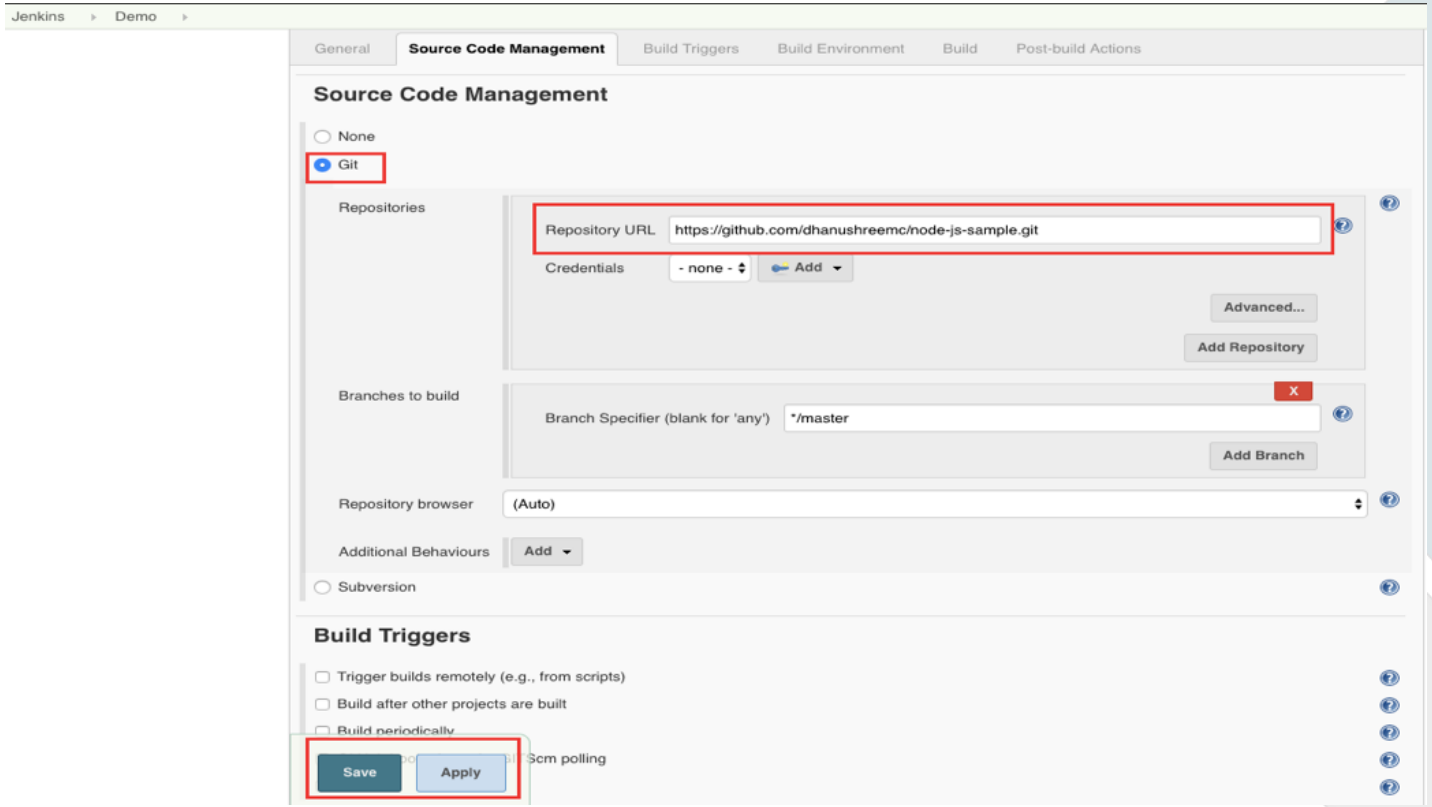
- Enter description as like below



Cont'd…

Step 4:

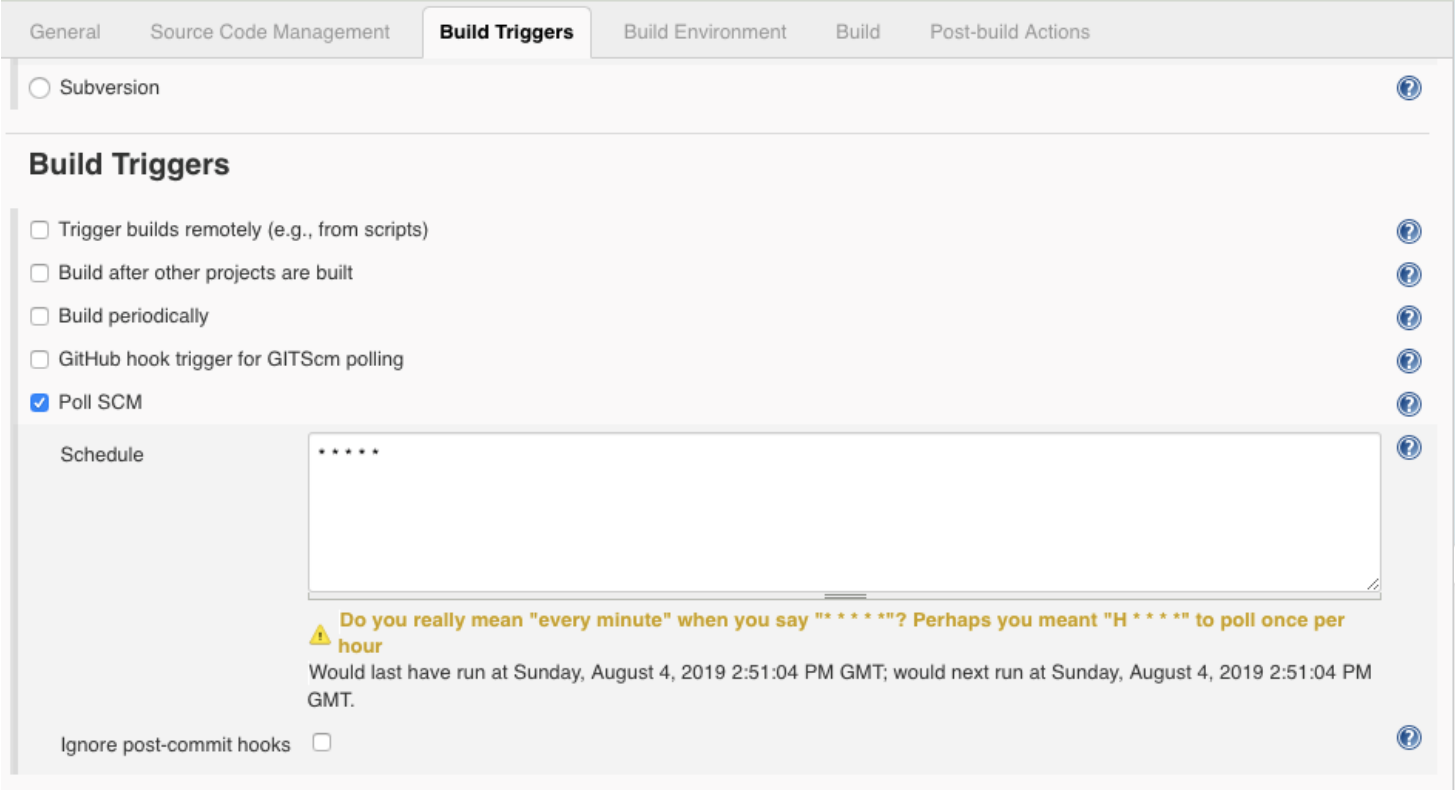
- Next click on Source code management section
- Select Git & enter Git URL as shown
- Save & Apply



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



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 05c52080f7bee8bc501ffca6b5bdd9005de11c77 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 05c52080f7bee8bc501ffca6b5bdd9005de11c77
Commit message: "Update Jenkinsfile"
 > git rev-list --no-walk b1afc0d3f71cee5f2f64f16ac1d2bb8292dac8b9 # 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 of 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.

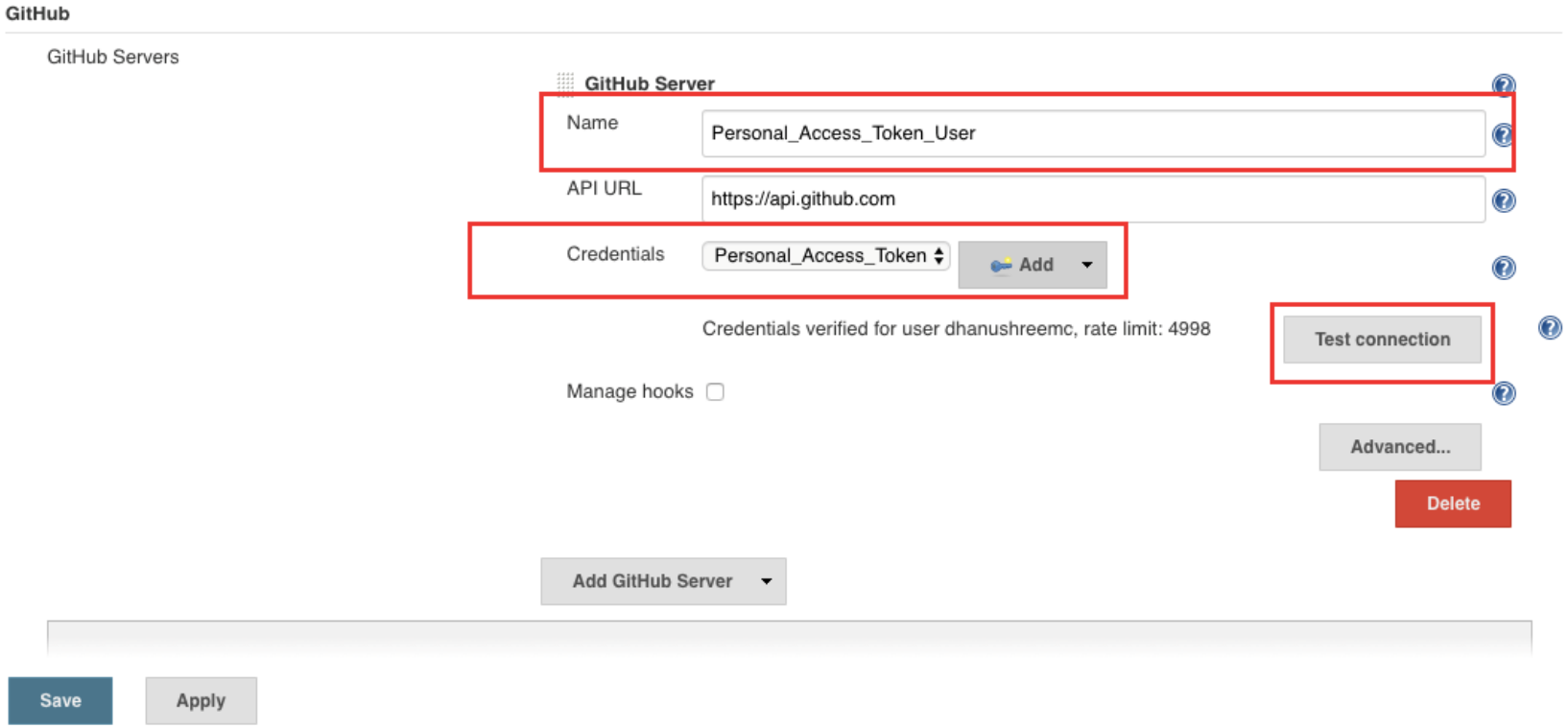
Step 2:

- 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:

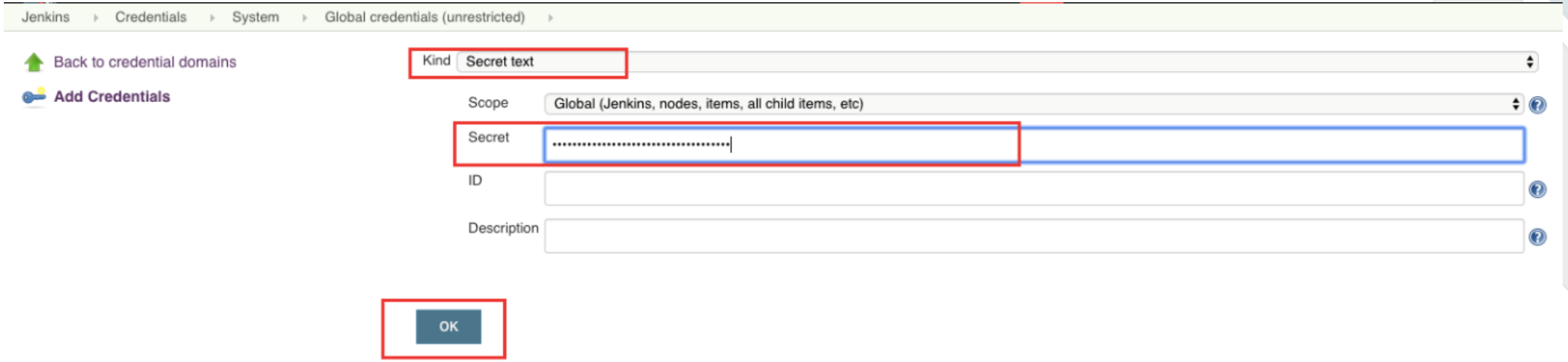
- Go to Manage Jenkins → Configure system
- Go to GitHub section & add credentials



Cont'd…

Step 4:

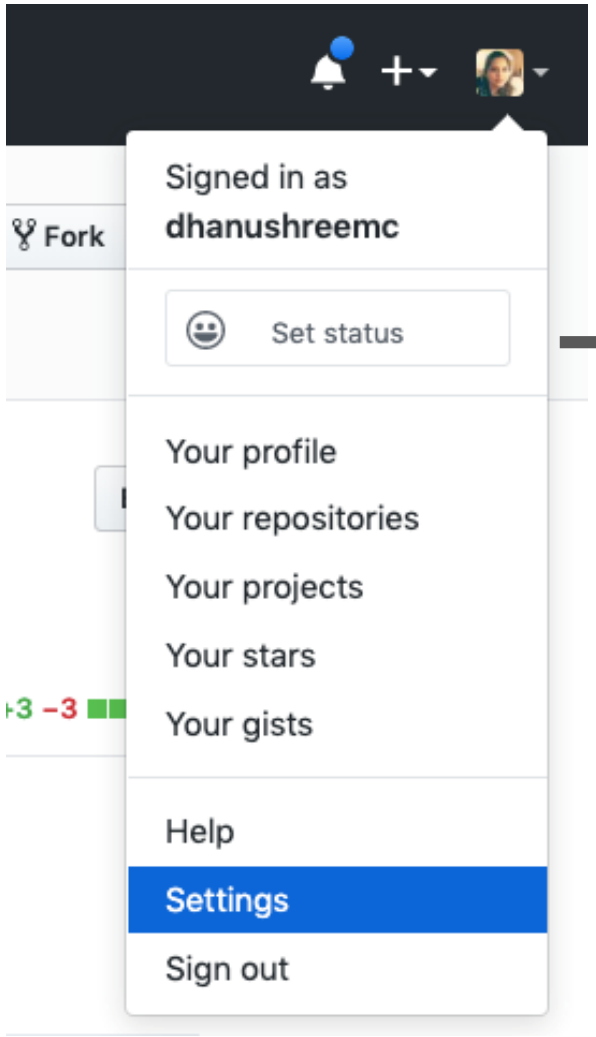
- Click on add credentials
- Select credentials type as secret text
- Copy paste GitHub personal_access_token in secret section as shown

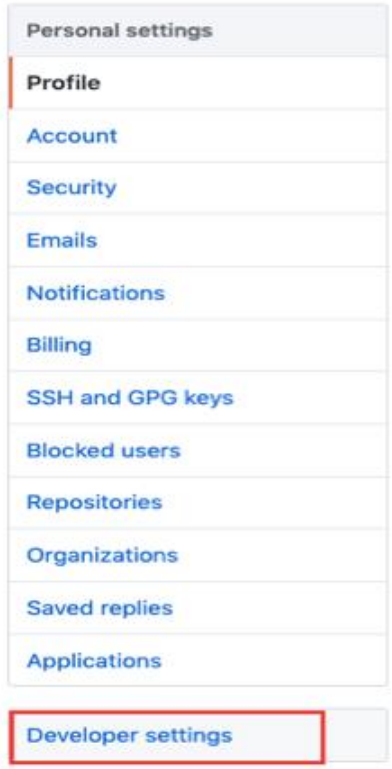


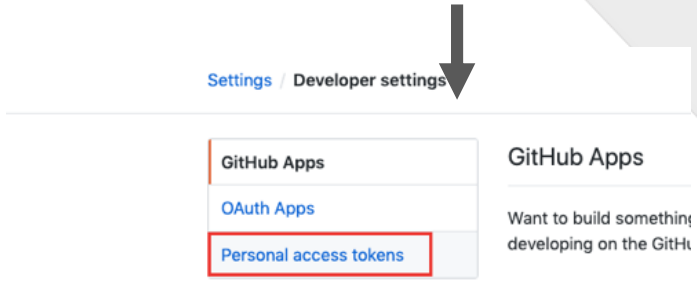
Personal access token creation

Steps

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

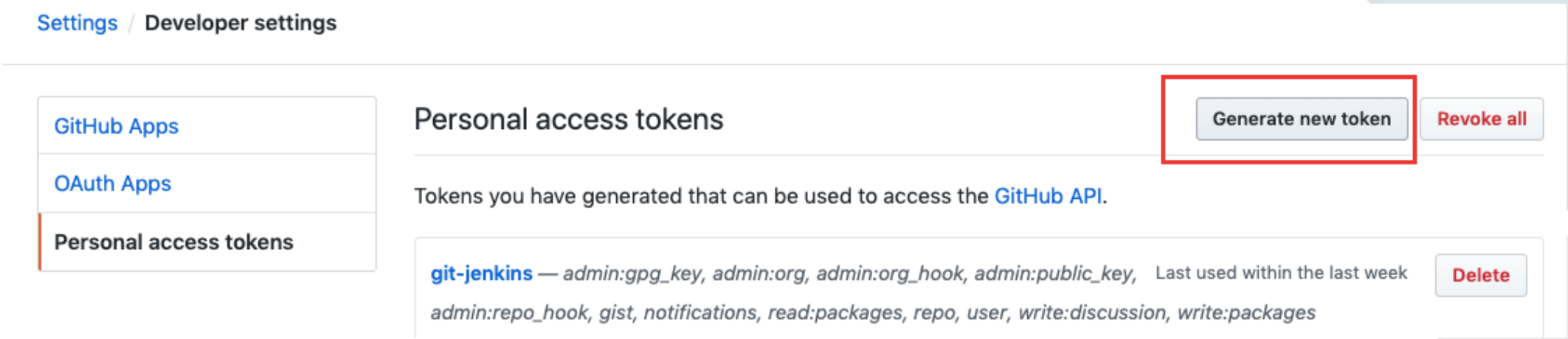






Cont'd…

- Add a personal access token as shown below



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?

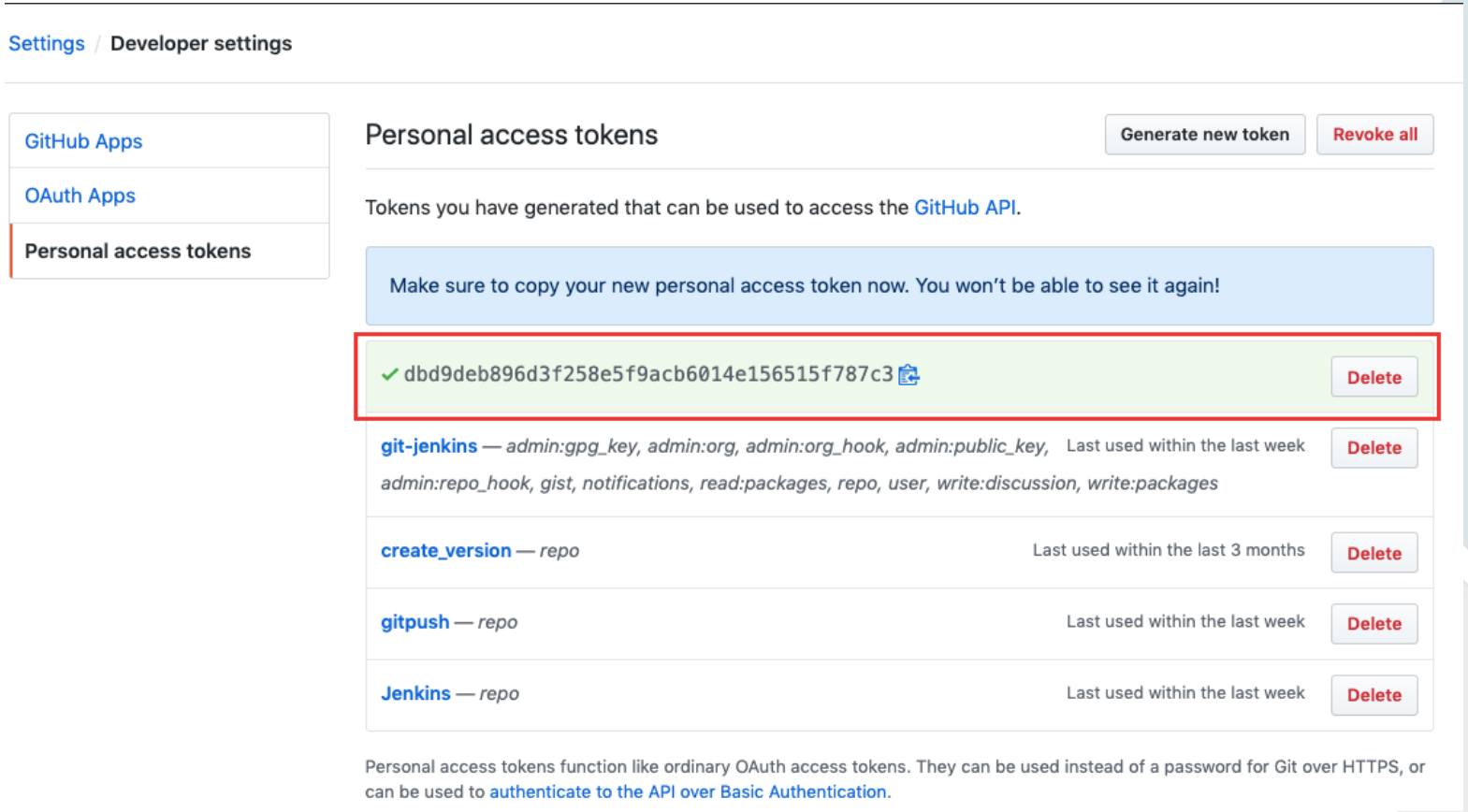
Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

☑ **repo**	Full control of private repositories
☑ repo:status	Access commit status
☑ repo_deployment	Access deployment status
☑ public_repo	Access public repositories
☑ repo:invite	Access repository invitations
☐ **admin:org**	Full control of orgs and teams, read and write org projects
☐ write:org	Read and write org and team membership, read and write org projects
☐ read:org	Read org and team membership, read org projects
☐ **admin:public_key**	Full control of user public keys
☐ write:public_key	Write user public keys
☐ 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



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

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | ZenHub | Projects 0 | Wiki | Security | Insights | Settings

- Options
- Collaborators
- Branches
- Webhooks
- Notifications
- Integrations & services
- Deploy keys

Moderation

- Interaction limits

Webhooks

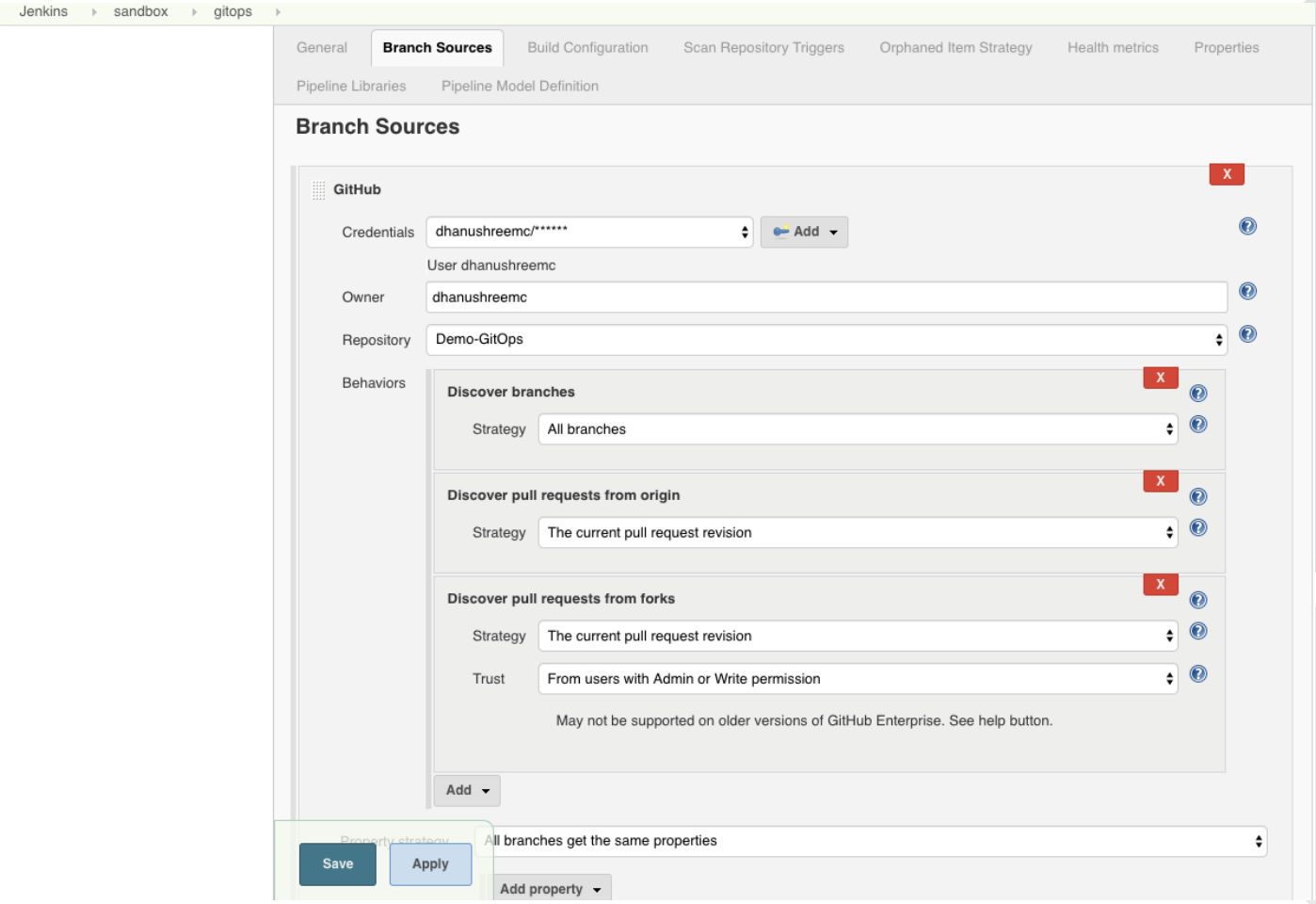
Add webhook

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

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



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)

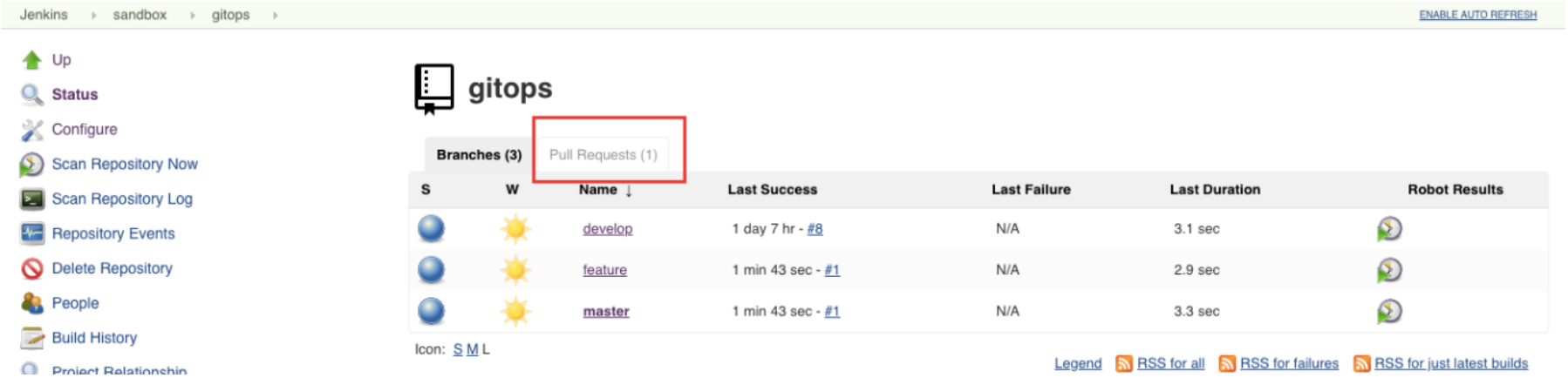
S	W	Name ↓	Last Success	Last Failure	Last Duration	Robot Results
		develop	1 day 7 hr - #8	N/A	3.1 sec	
		feature	1 day 10 hr - #7	N/A	3.1 sec	
		master	1 day 11 hr - #1	N/A	4 sec	

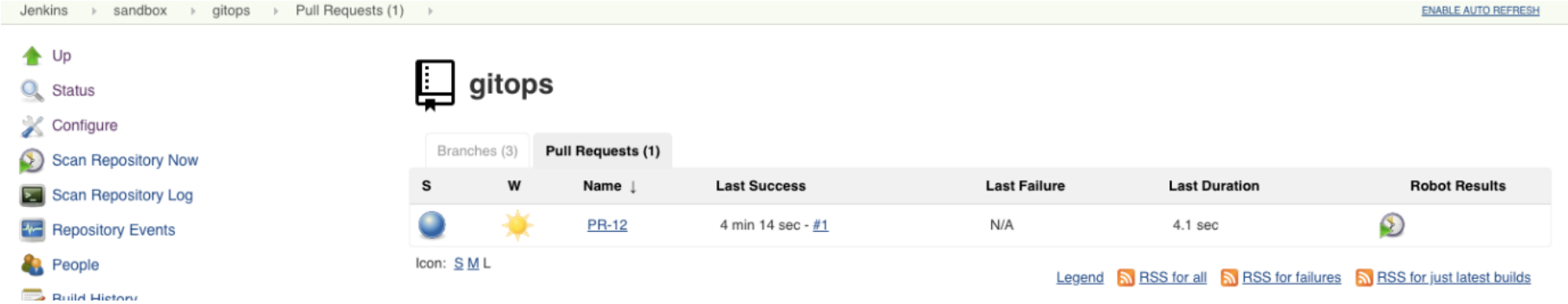
Icon: S M L

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





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

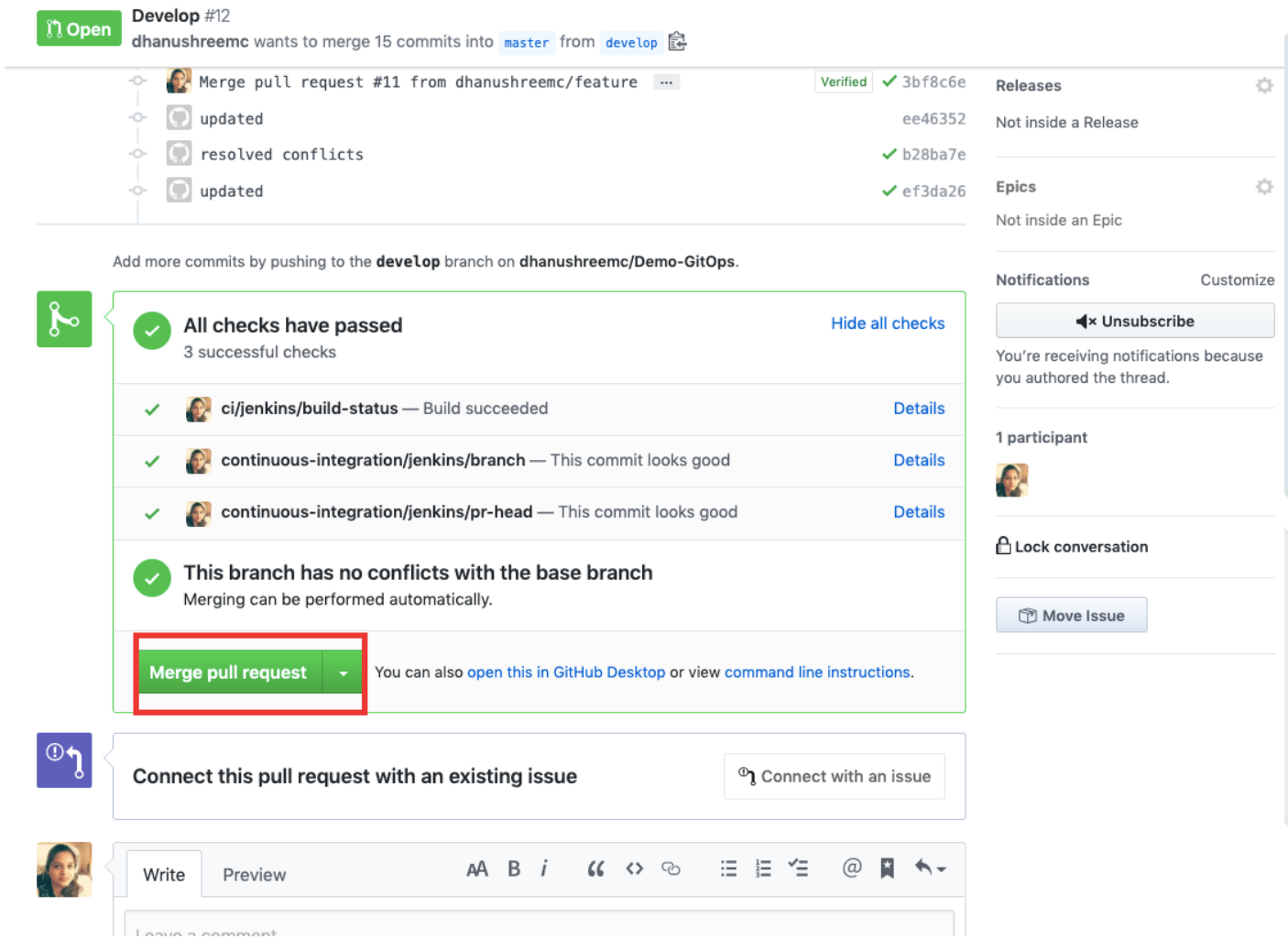
- Back to Project
- Status
- Changes
- Console Output
  - View as plain text
- Edit Build Information
- Delete Build
- Git Build Data
- No Tags
- Restart from Stage
- Replay
- Pipeline Steps

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



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

Jenkins › sandbox › gitops › master › #2

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
Previous Build

Console Output

```
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 --add remote.origin.fetch +refs/heads/master:refs/remotes/origin/master # 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
```