

Implementing the Jenkins-GitHub Integration

This lab will show the integration of the github with the jenkins using the personal access and setting up Continuous integration pipeline project using webhooks trigger.

Step 1: Generate a GitHub Personal Access Token

1. Go to your **GitHub Account > Settings > Developer Settings > Personal Access Tokens > Tokens (classic)**.
2. Click on **Generate new token (classic)**, select scopes:
 - **repo** for accessing private repositories.
 - **admin:repo_hook** for managing webhooks.

What's this token for?

Expiration *
30 days ▼ The token will expire on Tue, Dec 31 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
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input type="checkbox"/> admin:org	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"/> manage_runners:org	Manage org runners and runner groups
<input type="checkbox"/> admin:public_key	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
<input type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input type="checkbox"/> write:repo_hook	Write repository hooks
<input type="checkbox"/> read:repo_hook	Read repository hooks

3. Copy the token and paste it somewhere.

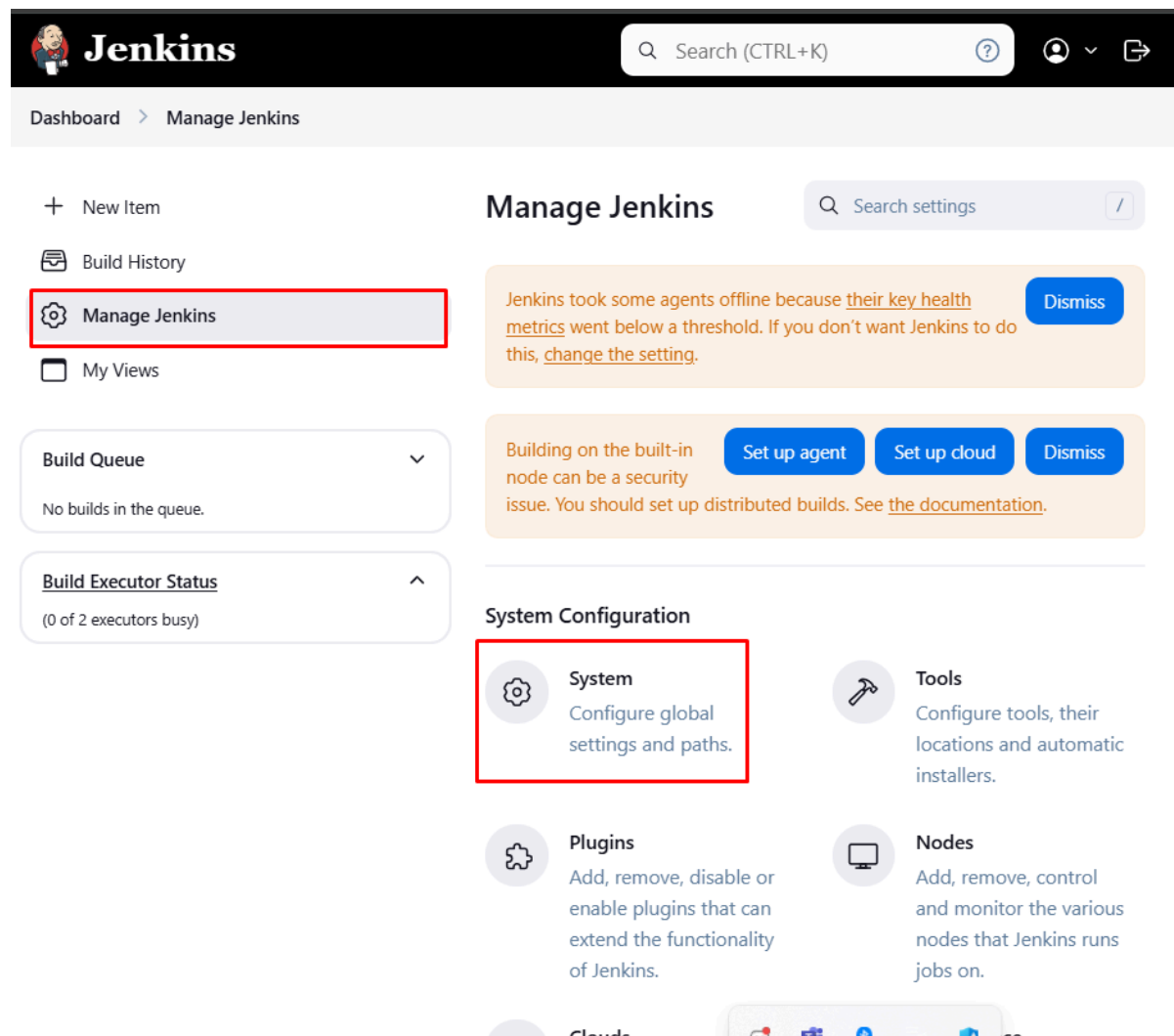
Step 2: Set Up Jenkins(install the Jenkins on AWS EC2)

1. Install Necessary Plugins

- Go to **Manage Jenkins > Plugins > Available plugins**.
- You can first even go to the installed plugins and search if they are already installed for you
- Install the following plugins:
 - **Git Plugin**
 - **GitHub Integration Plugin**
 - **GitHub Branch Source Plugin**

2. Configure Global Settings

- Navigate to **Manage Jenkins > System**.



- Under **GitHub**, click **Add GitHub Server**.
 - Provide a name which can be any name , keep the api url as it is.
 - Under the credentials click on add.

- Make sure to select global for the domain and for kind select secret text. And scope it global and then paste the secret that you copied in step1 and name your credential in id.

notification URL

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Global credentials (unrestricted)

Kind

Secret text

Scope ?

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

Secret

.....

ID ?

test

Description ?

connection

Step 3: Create a Jenkins Job for the Repository

1. Go to Jenkins Dashboard and click **New Item**.
2. Choose **Freestyle project** or **Pipeline**, then enter a name and click **OK**.
3. Under **Source Code Management**:
 - Select **Git**.
 - Provide your repository URL.

Advanced ▾

Configure

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

Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://github.com/gurpinder2023/testing_model.git

Credentials ?

- none - ▾

+ Add

Advanced ▾

Add Repository

-
- Then put the branch you want to build from which is in your github, mine is main so i select main

Dashboard > testing > Configuration

Configure

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

Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://github.com/gurpinder2023/testing_model.git

Credentials ?

- none - ▾

+ Add

Advanced ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

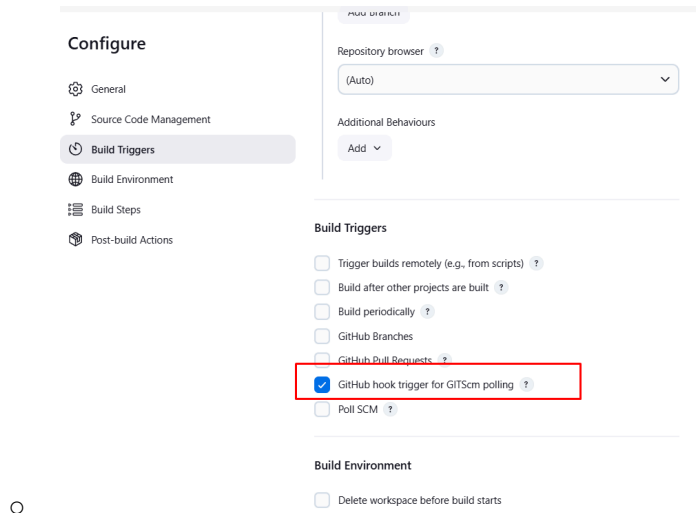
*/main

Add Branch

-
- Use **Credentials** if required (username, password, or PAT). but as we already configured the credential in the system no need to do that.

4. Under Build Triggers:

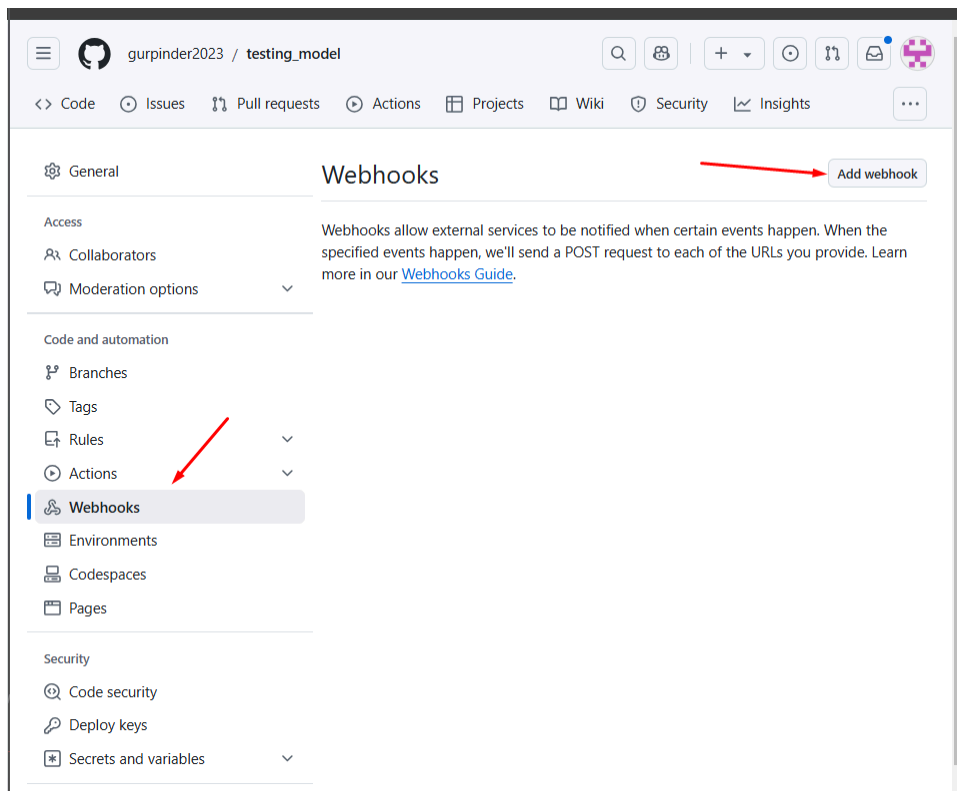
- Click on the Github hook trigger for GITScm polling.



5. Save the project

Step 4: Configure GitHub Webhook

1. Go to your GitHub repository.
2. Navigate to **Settings > Webhooks > Add Webhook**.



3. In the **Payload URL**, enter your Jenkins URL followed by `/github-webhook/`, e.g., `http://<jenkins-server-ip>:8080/github-webhook/`.
4. Click on let me select individual events. And select pull and pushes.

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

Security

Code security

Deploy keys

Secrets and variables

Integrations

GitHub Apps

Webhooks / Add webhook

We'll send a post request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

`http://ec2-52-42-2-122.us-west-2.compute.amazonaws.com:8080/github-webhook/`

Content type *

application/x-www-form-urlencoded

Secret

SSL verification

By default, we verify SSL certificates when delivering payloads.

☒ Enable SSL verification ☐ Disable (not recommended)

Which events would you like to trigger this webhook?

☐ Just the push event.

☐ Send me everything.

☒ Let me select individual events.

- ☐ **Pull request reviews**
Pull request review submitted, edited, or dismissed.
- ☒ **Pushes**
Git push to a repository.
- ☒ **Pull requests**
Pull request assigned, auto merge disabled, auto merge enabled, closed, converted to draft, demilestoned, dequeued, edited, enqueued, labeled, locked, milestoned, opened, ready for review, reopened, review request removed, review requested, synchronized, unassigned, unlabeled, or unlocked.
- ☐ **Registry packages**
Registry package published or updated in a repository.
- ☐ **Repositories**

5. Save the webhook.

Step 5: Test the Integration

1. Push a change to your GitHub repository
2. Go to Jenkins and see if a build was triggered automatically.
3. The after the build success ful you will see the changes in the logs

Dashboard > newproject > #2

Status #2 (2 Dec 2024, 04:57:42) [Add description](#) [Keep this build forever](#)

Started 1 hr 12 min ago
Took 0.48 sec

Started by GitHub push by gurpinder2023

This run spent:

- 1 min 5 sec waiting;
- 0.48 sec build duration;
- 1 min 6 sec total from scheduled to completion.

git **Revision:** 7b4863b98f14d0a2564252e033597062fa2db919
Repository: https://github.com/gurpinder2023/Advice_generator_frontend.git

- refs/remotes/origin/main

Changes

1. Update admin.html ([details](#) / [githubweb](#))

Step 6 : (optional if want to deploy) Create a Jenkins Job for the Repository

1. If you want to deploy the simple nginx html files to test.
2. Install the nginx server on the instance using following commands:
 - `sudo yum update -y`
 - `sudo yum install nginx -y`
 - `sudo systemctl start nginx`
 - `sudo systemctl enable nginx`
3. Choose **Freestyle project** or **Pipeline**, and follow all the above steps from 1 to 5
4. Under the **Build Steps**: Select execute shell and type the following.

Navigate to the Jenkins workspace

```
cd $WORKSPACE
```

Copy files to the Nginx web directory

```
sudo cp -r * /usr/share/nginx/html/
```

Restart Nginx to apply changes

```
sudo systemctl restart nginx
```

5. Build might fail because Jenkins can't run sudo commands because it does not have access to sudo so go to the git bash and type

- sudo visudo
- Add the following line at the end of the file:

```
bash
```

Copy code

```
jenkins ALL=(ALL) NOPASSWD: /bin/cp, /bin/systemctl
```

This allows the `jenkins` user to run `cp` and `systemctl` commands without being prompted for a password.

6. Deploy the project again and then access the page using `http:// public-ip-address:80` at port 80.
7. Now if you make any changes it will be visible in deployed.

TROUBLESHOOTING GUIDE:

1. Github integration error:

Make sure you install the git in the instance by using the commands like

```
sudo yum update -y
```

```
sudo yum install git -y
```

```
git --version
```

This will avoid the errors like the one shown below

Source Code Management

The screenshot shows the Jenkins 'Source Code Management' configuration page. The 'Git' option is selected. The 'Repository URL' is set to 'https://github.com/gurpinder2023/Advice_generator_frontend.git'. Below this, a red error message is displayed: 'Failed to connect to repository : Error performing git command: git ls-remote -h https://github.com/gurpinder2023/Advice_generator_frontend.g HEAD'. The 'Credentials' dropdown is set to '- none -'. There are buttons for '+ Add' and 'Advanced'.

2. Unable to build:

Sometimes when you build the project it continuously shows you in process then check the node

Go to the **Manage Jenkins > nodes** and see if the node is online and have **more than 1 executor** and if it is offline make it online then you will see that your project will be started building