

## Module 6: Case Study - Integration of DevOps Tools with Jenkins

### Problem Statement:

You have been hired as a DevOps Engineer in XYZ software company. They want to implement CI/CD pipeline in their company. You have been asked to implement this lifecycle as fast as possible. As this is a product-based company, their product is available on this GitHub link.

<https://github.com/hshar/website.git>

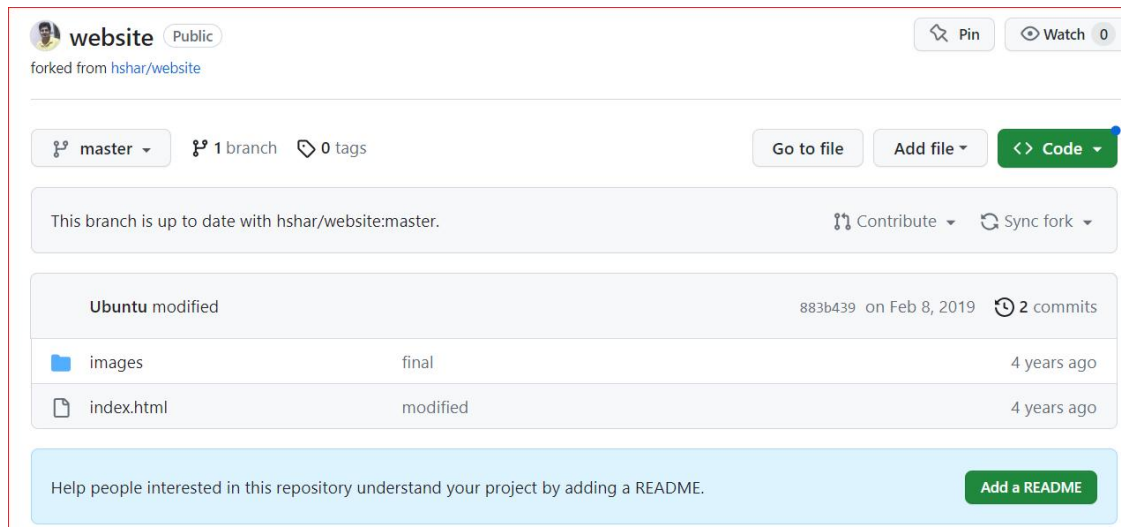
Following are the specifications of the continuous integration:

1. Git workflow has to be implemented
2. CodeBuild should automatically be triggered once a commit is made to master branch or develop branch. If a commit is made to master branch, build and publish a website on port 82. If a commit is made to develop a branch, just build the product, do not publish.
3. Create a pipeline for the above tasks
4. Create a container with Ubuntu and Apache installed in it and use that container to build the code and the code should be on '/var/www/html'.

### Solution:-

\* Fork this repo to your git hub account:-

<https://github.com/hshar/website.git>



```
$ which git
$ mkdir Git
$ cd Git
$ sudo git init
$ ls -al
```

```
ubuntu@Jenkins-Master:~$ which git
/usr/bin/git
ubuntu@Jenkins-Master:~$ mkdir Git
ubuntu@Jenkins-Master:~$ cd Git/
ubuntu@Jenkins-Master:~/Git$ sudo git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/ubuntu/Git/.git/
ubuntu@Jenkins-Master:~/Git$ ls -al
total 12
drwxrwxr-x 3 ubuntu ubuntu 4096 Aug 27 07:12 .
drwxr-x--- 6 ubuntu ubuntu 4096 Aug 27 07:10 ..
drwxr-xr-x 7 root    root   4096 Aug 27 07:12 .git
ubuntu@Jenkins-Master:~/Git$
```

```
$ docker -v
```

```
root@Jenkins-Master:~# docker -v
Docker version 24.0.5, build ced0996
root@Jenkins-Master:~#
```

```
$ git clone https://github.com/github-amit-  
repository/website.git
```

```
ubuntu@Jenkins-Master:~/Git$ git clone https://github.com/github-amit-repository/website.git  
Cloning into 'website'...  
remote: Enumerating objects: 8, done.  
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 8  
Receiving objects: 100% (8/8), 82.69 KiB | 9.19 MiB/s, done.  
Resolving deltas: 100% (1/1), done.
```

```
$ cd website  
$ vi dockerfile  
$ cat dockerfile  
FROM ubuntu  
RUN apt get update  
RUN apt get install apache2 -y  
ADD . /var/www/html  
ENTRYPOINT apachectl -D FOREGROUND
```

```
ubuntu@Jenkins-Master:~/Git$ cd website  
ubuntu@Jenkins-Master:~/Git/website$ sudo vi dockerfile  
ubuntu@Jenkins-Master:~/Git/website$ cat dockerfile  
FROM ubuntu  
RUN apt get update -y  
RUN apt get install apache2 -y  
ADD . /var/www/html  
ENTRYPOINT apachectl -D FOREGROUND
```

```
$ git add .  
$ git commit -m "dockerfile should be committed"
```

```
ubuntu@Jenkins-Master:~/Git/website$ git add .  
ubuntu@Jenkins-Master:~/Git/website$ git commit -m "dockerfile should be committed"  
[master 87eb8d1] dockerfile should be committed  
1 file changed, 5 insertions(+)  
create mode 100644 dockerfile
```


```
$ git branch && git branch develop && git checkout  
develop && ls && git checkout master && git push --all
```

```

ubuntu@Jenkins-Master:~/Git/website$ git branch && git branch develop && git checkout
t develop && ls && git checkout master && git push --all
* master
Switched to branch 'develop'
dockerfile  images  index.html
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
Username for 'https://github.com': github-amit-repository
Password for 'https://github-amit-repository@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 420 bytes | 420.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/github-amit-repository/website.git
  883b439..87eb8d1  master -> master
* [new branch]      develop -> develop




```

\* Now in the github repo we can see files:-



**website**
Public

forked from [hshar/website](#)

---

 master
  2 branches
  0 tags

---



**Your master branch isn't protected**




Protect this branch from force pushing or deletion, or require status checks before merging

---

This branch is **1 commit ahead** of hshar:master.

---


**amitamit1234** dockerfile should be committed

 images	final
 dockerfile	dockerfile should be committed
 index.html	modified

Prod:-

```
$ sudo apt install docker.io -y
```

```

ubuntu@prod-server:~$ sudo apt install docker.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:

```

```
$ docker -v
```

```
ubuntu@prod-server:~$ docker -v
Docker version 20.10.25, build 20.10.25-0ubuntu1~22.04.2
ubuntu@prod-server:~$
```

\* Create new node:- dashboard->manage jenkins -> nodes -> new node -> prod-node-> select permanent agent->create

Dashboard > Manage Jenkins > Nodes > New node

## New node

Node name

Type

☒ Permanent Agent

Adds a plain, permanent agent to Jenkins integration with these agents, such as dynamic scaling. This is the most common example such as when you are adding a permanent agent.

Create

\* Give details for node:- Remote Root Directory -> /home/ubuntu/jenkins

Dashboard > Manage Jenkins > Nodes >

Name ?

prod-node

Description ?

Plain text [Preview](#)

Number of executors ?

1

Remote root directory ?

/home/ubuntu/jenkins

\* Give:- Labels-> prod-node->launch method-> launch agent via SSH-> host-> give credentials

Remote root directory ?

/home/ubuntu/jenkins

Labels ?

prod-node

Usage ?

Use this node as much as possible

Launch method ?

Launch agents via SSH

Host ?

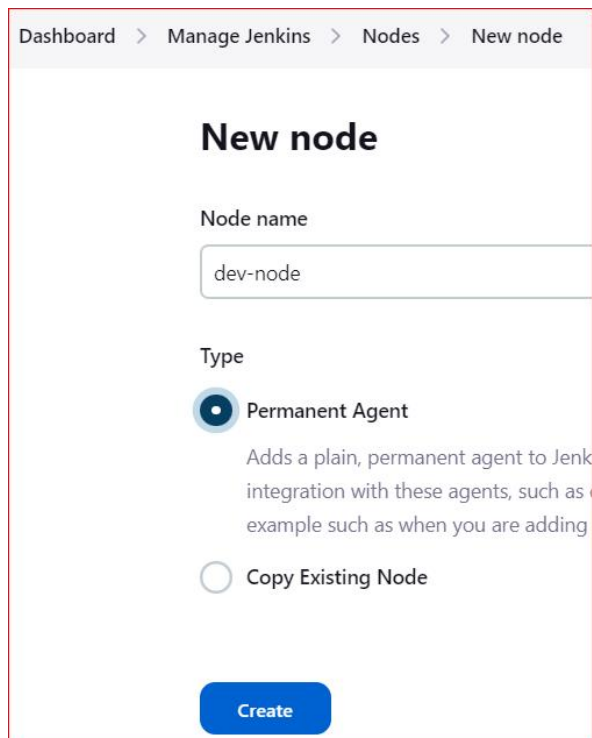
172.31.39.81

Credentials ?

ubuntu/\*\*\*\*\*

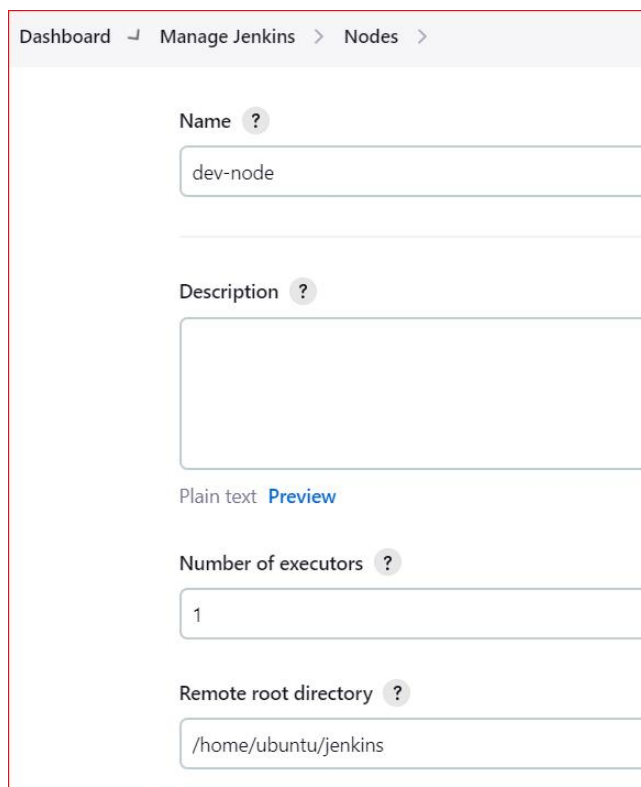
Add ▾

\* Now create dev-node nodes:- dashboard->manage jenkins-> nodes->new nodes-> select permanent agent->create



The screenshot shows the 'New node' configuration page in Jenkins. The breadcrumb navigation at the top reads 'Dashboard > Manage Jenkins > Nodes > New node'. The main heading is 'New node'. There is a text input field for 'Node name' containing 'dev-node'. Below it, under the 'Type' section, there are two radio button options: 'Permanent Agent' (which is selected) and 'Copy Existing Node'. A descriptive text for 'Permanent Agent' is partially visible: 'Adds a plain, permanent agent to Jenk... integration with these agents, such as... example such as when you are adding...'. At the bottom, there is a blue 'Create' button.

\* Insert:- Remote root directory-> /home/ubuntu/jenkins



This screenshot shows the same 'New node' configuration page but with more fields filled out. The 'Name' field contains 'dev-node'. The 'Description' field is empty. Below the description, there is a 'Plain text' label and a 'Preview' link. The 'Number of executors' field contains the value '1'. The 'Remote root directory' field contains the path '/home/ubuntu/jenkins'. Each field has a help icon (a question mark in a circle) to its right.

\* Give:- Labels-> prod-node->launch method-> launch agent via SSH-> host-> give credentials  
-> host key verification strategy->Non verifying verification strategy

Dashboard > Manage Jenkins > Nodes >

Labels ?

dev-node

Usage ?

Use this node as much as possible

Launch method ?

Launch agents via SSH

Host ?

172.31.45.149

Credentials ?

ubuntu/\*\*\*\*\*

Add ▾

Host Key Verification Strategy ?

Non verifying Verification Strategy

\* Here we have nodes:-

Dashboard > Manage Jenkins > Nodes >

Nodes

+ New Node

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	3.93 GB	0 B	3.93 GB	0ms
	dev-node	Linux (amd64)	In sync	5.00 GB	0 B	5.00 GB	87ms
	prod-node	Linux (amd64)	In sync	4.71 GB	0 B	4.71 GB	66ms
Data obtained		0.23 sec	0.21 sec	0.19 sec	0.15 sec	0.17 sec	0.19 sec

Build Queue

No builds in the queue.

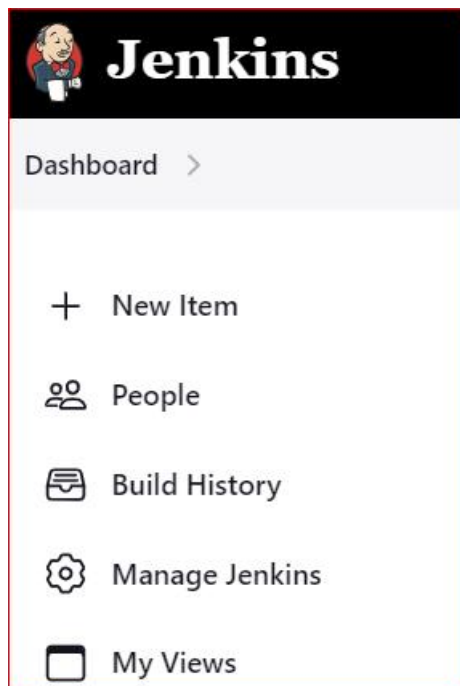
Build Executor Status

Built-In Node

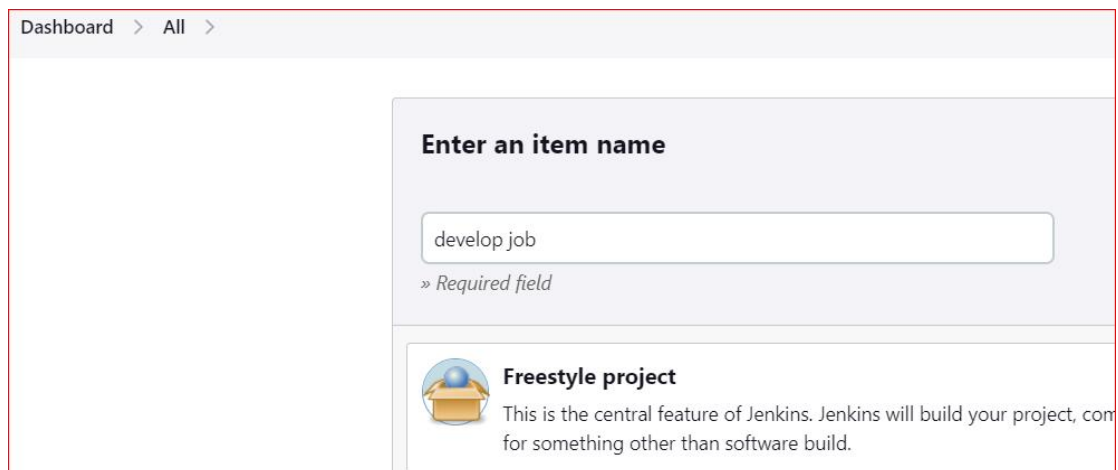


Jenkins:-

\* Go to dashboard select New Item:-



\* Create job:- new items -> name (develop job) -> free style project -> ok



\* Insert github url:- configure -> description-> For develop branch do not publish it. -> Github project -> project url ( <https://github.com/github-amit-repository/website.git> )

Dashboard > develop job > Configuration

### Configure

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

### General

Description

For develop branch do not publish it.

Plain text [Preview](#)

☐ Discard old builds ?

☒ GitHub project

Project url ?

<https://github.com/github-amit-repository/website.git>

Advanced ▾

☐ This project is parameterized ?

\* Select:- restrict where this project can be run -> Label expression -> prod-node

☐ Execute concurrent builds if necessary ?

☒ Restrict where this project can be run ?

Label Expression ?

[prod-node](#)

Label [prod-node](#) matches 1 node. Permissions or other restrict

Advanced ▾

\* Select and insert:-

source code management -> git -> repo url -> paste repo website url -> give credential means insert username and password

Git ?

Repositories ?

Repository URL ?

https://github.com/github-amit-repository/website.git

Credentials ?

ubuntu/\*\*\*\*\*

Add ▾

Advanced ▾

Kind

Username with password

Scope ?

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

Username ?

ubuntu

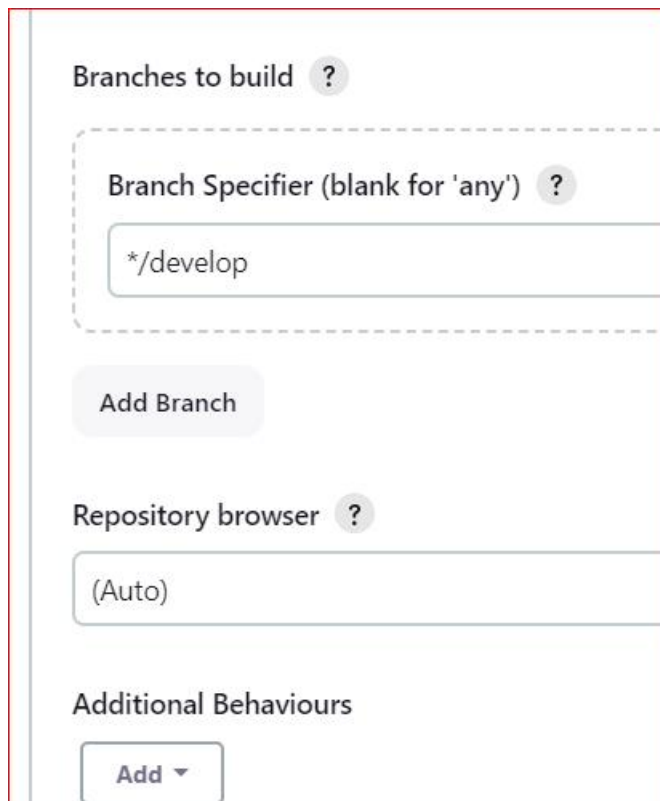
☐ Treat username as secret ?

Password ?

.....

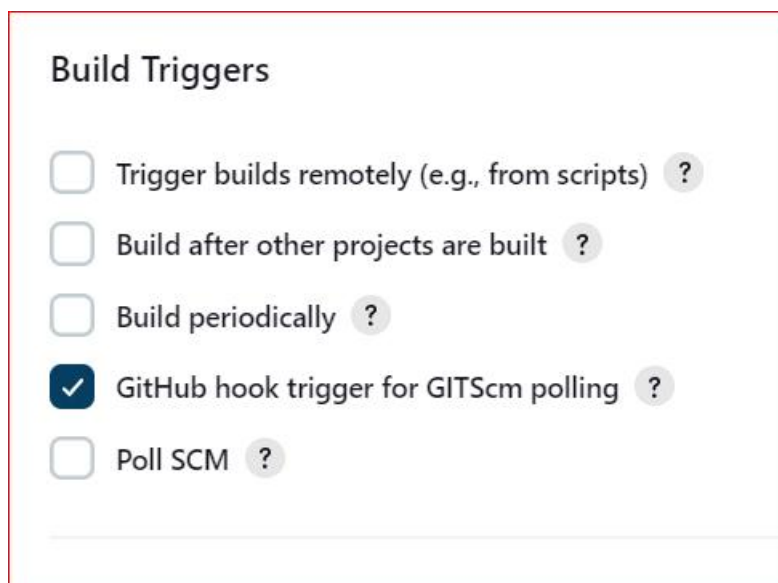
ID ?

\* Give branch:- \*/develop



The screenshot shows a configuration panel titled "Branches to build" with a help icon. Inside, there is a dashed box containing a "Branch Specifier (blank for 'any')" label and a text input field containing "\*/develop". Below this is an "Add Branch" button. Further down is a "Repository browser" label and a dropdown menu currently showing "(Auto)". At the bottom is an "Additional Behaviours" section with an "Add" button and a downward arrow.

\* Select git SCM:- build triggers -> github hook trigger for git SCM polling -> apply -> save



The screenshot shows a configuration panel titled "Build Triggers". It contains five checkboxes, each with a help icon to its right: "Trigger builds remotely (e.g., from scripts)", "Build after other projects are built", "Build periodically", "GitHub hook trigger for GIT SCM polling" (which is checked), and "Poll SCM".

# Project develop job

For develop branch do not publish it.

\* Create webhook connection:- Github -> copy jenkins url and paste in github website repo -> github repo -> settings -> webhooks -> add webhook -> payload url -> http://jenkins url till port/github-webhook/ -> add webhook -> webhooks/manage webhook/ recent deliveries -> save

The screenshot shows the GitHub 'Webhooks' configuration page. On the left is a sidebar with navigation links: 'Code and automation' (Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, Pages), 'Security' (Code security and analysis, Deploy keys, Secrets and variables), and 'Integrations' (GitHub Apps). The 'Webhooks' link is selected. The main content area is titled 'Payload URL \*' and contains a text input field with the value 'http://54.235.6.192:8080/github-webhook/'. Below this is a 'Content type' dropdown menu set to 'application/x-www-form-urlencoded'. A 'Secret' field is present but empty. The section 'Which events would you like to trigger this webhook?' has three radio button options: 'Just the push event.' (selected), 'Send me everything.', and 'Let me select individual events.'. At the bottom, the 'Active' checkbox is checked, with a note: 'We will deliver event details when this hook is triggered.'. A green 'Add webhook' button is at the bottom right.

Section	Item
Code and automation	Branches
	Tags
	Rules
	Actions
	<b>Webhooks</b>
	Environments
	Codespaces
	Pages
Security	Code security and analysis
	Deploy keys
	Secrets and variables
	Integrations
GitHub Apps	

**Payload URL \***

**Content type**

application/x-www-form-urlencoded

**Secret**

**Which events would you like to trigger this webhook?**

☒ Just the push event.

☐ Send me **everything**.

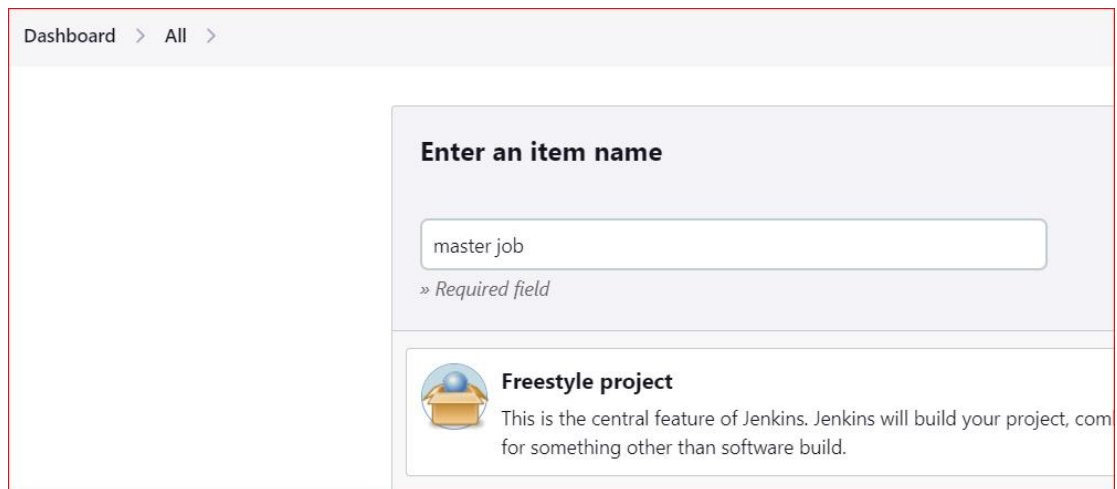
☐ Let me select individual events.

☒ **Active**

We will deliver event details when this hook is triggered.

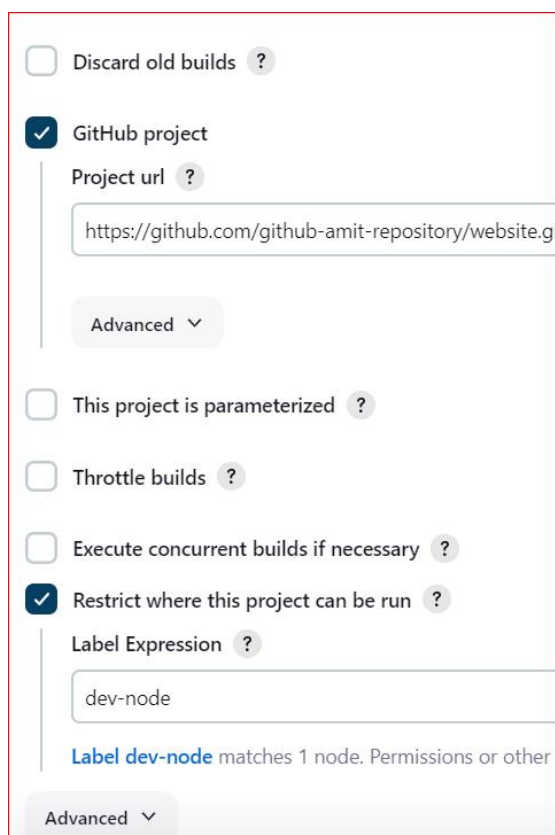
**Add webhook**

\* Create job:- new items -> name (develop job) -> free style project -> ok



The screenshot shows the Jenkins 'Enter an item name' dialog box. At the top, there is a breadcrumb trail: 'Dashboard > All >'. Below this, the title 'Enter an item name' is displayed. A text input field contains the text 'master job'. Below the input field, there is a small text label '» Required field'. Below the input field, there is a section titled 'Freestyle project' with a small icon of a box and a ball. The text below the icon reads: 'This is the central feature of Jenkins. Jenkins will build your project, com for something other than software build.'

\* Insert github url:- configure -> description-> For develop branch do not publish it. -> Github project -> project url ( <https://github.com/github-amit-repository/website.git> ) -> restrict where this project can be run -> Label expression -> dev-node



The screenshot shows the Jenkins 'Configure' page for a project. The page has a list of checkboxes and input fields. The 'Discard old builds' checkbox is unchecked. The 'GitHub project' checkbox is checked. Below it, the 'Project url' is set to 'https://github.com/github-amit-repository/website.git'. The 'Advanced' dropdown is expanded. The 'This project is parameterized' checkbox is unchecked. The 'Throttle builds' checkbox is unchecked. The 'Execute concurrent builds if necessary' checkbox is unchecked. The 'Restrict where this project can be run' checkbox is checked. Below it, the 'Label Expression' is set to 'dev-node'. A note below the label expression states: 'Label dev-node matches 1 node. Permissions or other'. The 'Advanced' dropdown is expanded.

\* Select and insert:- source code management -> git -> repo url -> paste repo website url -> give credential means insert username and password

The screenshot shows a web interface for configuring source code management. At the top, there's a search bar with 'dev-node' entered. Below it, a message says 'Label dev-node matches 1 node. Permissions or other restrictions provided by the node.' There's an 'Advanced' dropdown menu. The main section is titled 'Source Code Management'. It has two radio buttons: 'None' and 'Git'. The 'Git' option is selected. Below this, there's a 'Repositories' section with a dashed border. Inside, there's a 'Repository URL' field containing 'https://github.com/github-amit-repository/website.git' and a 'Credentials' field containing 'ubuntu/\*\*\*\*\*'. There's an 'Add' button at the bottom of the 'Repositories' section.

\* Give branch:- \*/master

The screenshot shows a web interface for configuring branches to build. It has a 'Credentials' section with a field containing 'ubuntu/\*\*\*\*\*' and an 'Add' button. Below this is an 'Advanced' dropdown menu. The main section is titled 'Add Repository'. Below that is a 'Branches to build' section with a dashed border. Inside, there's a 'Branch Specifier (blank for 'any')' field containing '\*/master'. There's an 'Add Branch' button at the bottom of the 'Branches to build' section. At the very bottom, there's a 'Repository browser' section with a question mark icon.

\* Select git SCM:- build triggers -> github hook trigger for git SCM polling -> apply -> save

### Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ GitHub hook trigger for GIT SCM polling ?
- ☐ Poll SCM ?

Now we have two jobs and now click build now:-

Dashboard >

+ New Item

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

Build Queue

No builds in the queue.

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀	develop job	15 min #3	N/A	8 sec
✓	☀	master job	15 sec #1	N/A	7.2 sec

Icon: S M L

Icon legend

Atom feed for all

Atom feed for failures

Atom feed for just latest builds

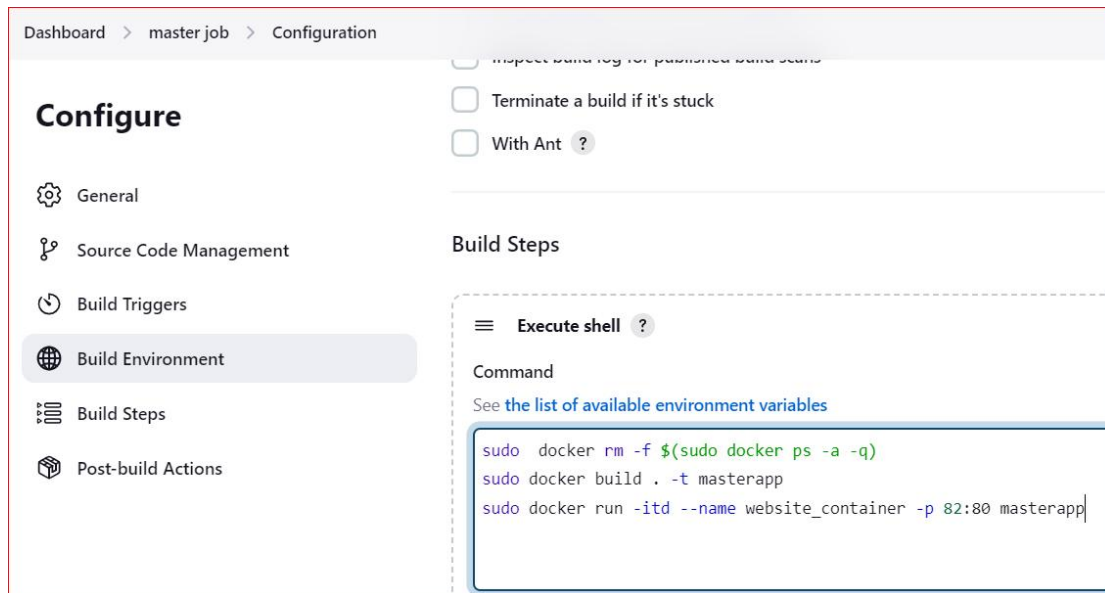
Add description

\* Now in master job add this:- Master Job->build environment->build steps-> add build step -> execute -> in execute shell -> command ->

```
sudo docker rm -f $(sudo docker ps -a -q)
sudo docker build . -t masterapp
sudo docker run -itd --name website_container -p 82:80
masterapp
```

-> apply-> save -> build now





Now we have dockerfile in both server:-

```
$ ls
$ cd workspace
$ ls
$ cd develop\ job
$ ls
```

```
ubuntu@prod-server:~/jenkins$ ls
remoting  remoting.jar
ubuntu@prod-server:~/jenkins$ ls
remoting  remoting.jar  workspace
ubuntu@prod-server:~/jenkins$ cd workspace/
ubuntu@prod-server:~/jenkins/workspace$ ls
'develop job'  'develop job@tmp'
ubuntu@prod-server:~/jenkins/workspace$ cd develop\ job
ubuntu@prod-server:~/jenkins/workspace/develop job$ ls
dockerfile  images  index.html
ubuntu@prod-server:~/jenkins/workspace/develop job$
```

```
$ ls
$ cd workspace
$ ls
$ cd master\ job
$ ls
```

```
ubuntu@test-server:~/jenkins$ ls
remoting  remoting.jar  workspace
ubuntu@test-server:~/jenkins$ cd workspace/
ubuntu@test-server:~/jenkins/workspace$ ls
'master job'  'master job@tmp'
ubuntu@test-server:~/jenkins/workspace$ cd master\ job
ubuntu@test-server:~/jenkins/workspace/master job$ ls
dockerfile  images  index.html
ubuntu@test-server:~/jenkins/workspace/master job$
```

\* Browser-> <http://34.229.44.2:82>

