

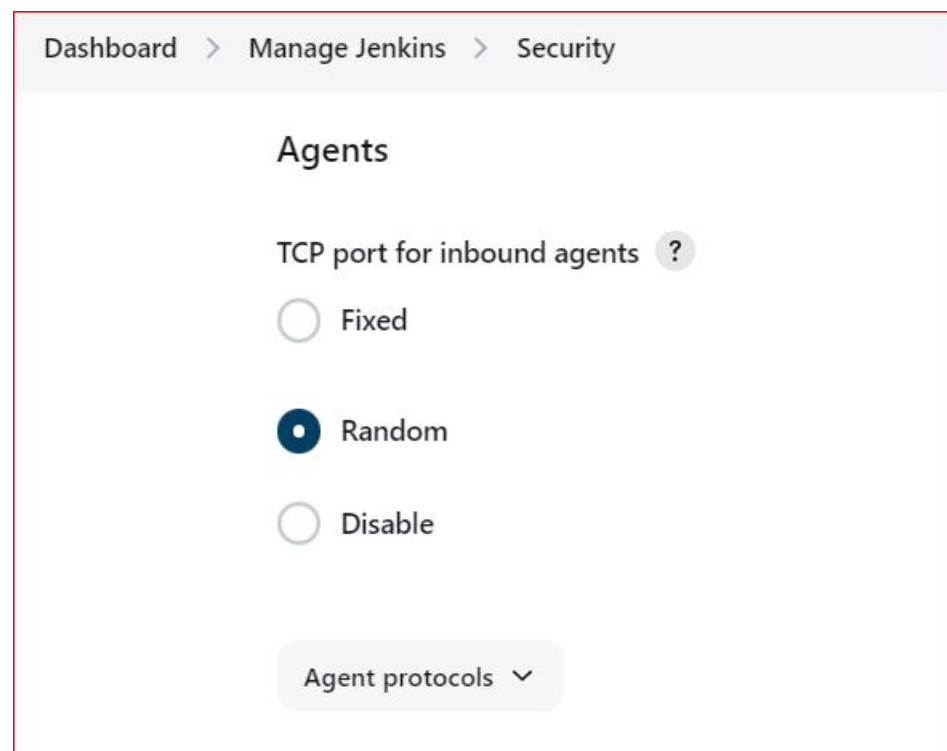
Module 6: Jenkins Assignment - 2

1. Add 2 nodes to Jenkins master
2. Create 2 jobs with the following jobs:
 - a. Push to test
 - b. Push to prod
3. Once a push is made to test branch, copy Git files to test server
4. Once a push is made to master branch, copy Git files to prod server.

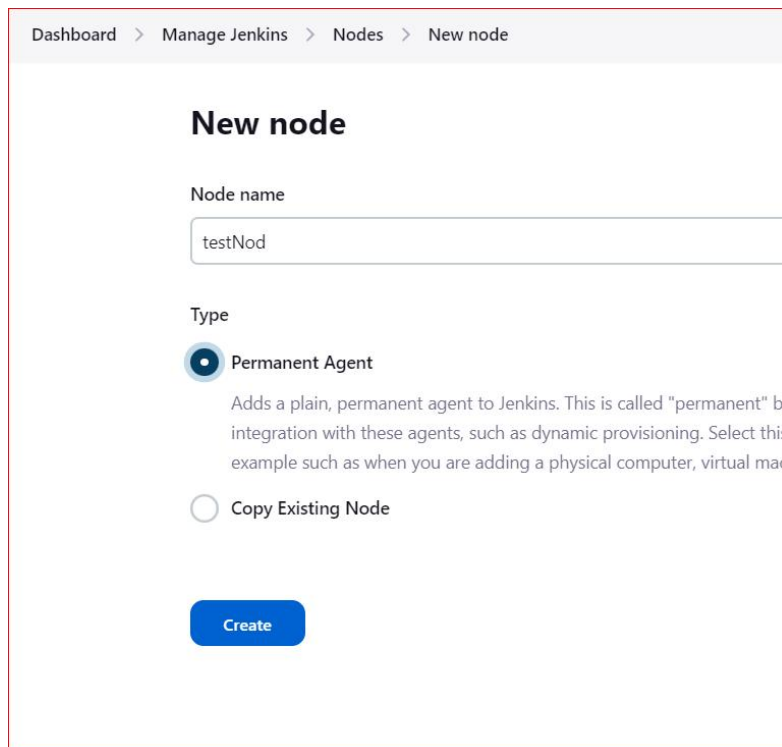
Solution:-

* Create two more machines for test and prod.

* Created two nodes:- dashboard -> manage jenkins -> configure global security -> tcp port for inbound agents -> random -> save

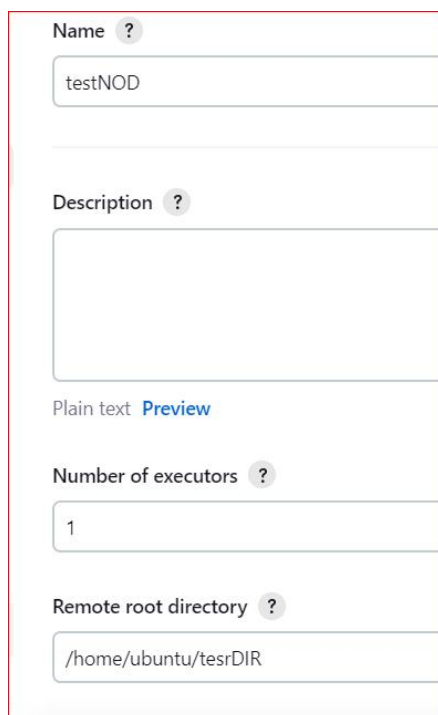


* Give a name:- manage nodes and clouds -> new node -> name
-> permanent agent -> create



The screenshot shows the Jenkins 'New node' configuration page. The breadcrumb trail at the top is 'Dashboard > Manage Jenkins > Nodes > New node'. The main heading is 'New node'. Under 'Node name', the text 'testNod' is entered. Under 'Type', the 'Permanent Agent' radio button is selected. Below this, there is explanatory text: 'Adds a plain, permanent agent to Jenkins. This is called "permanent" because of its integration with these agents, such as dynamic provisioning. Select this option for example such as when you are adding a physical computer, virtual machine, or container.' The 'Copy Existing Node' radio button is unselected. At the bottom, there is a blue 'Create' button.

* Insert remote root path:- remote root directory ->
/home/ubuntu/testDIR



This screenshot shows the lower portion of the Jenkins 'New node' configuration page. The 'Name' field contains 'testNOD'. The 'Description' field is empty. Below the description is a 'Plain text' section with a 'Preview' link. The 'Number of executors' field contains the value '1'. The 'Remote root directory' field contains the path '/home/ubuntu/testDIR'.

* Insert these info:- launch method -> launch agents by a ssh -> host -> paste private ip of node not server-> credential -> add -> domain-> global credential (unrestricted) -> kind -> ssh username with private key -> scope -> global (jenkins, nodes, items, all child items, etc.) -> username -> Jenkins-Node -> private key -> enter directly -> copy and paste key of node-> add -> host key verification strategy -> non verifying verification strategy -> Save -> refresh status

Launch method ?

Launch agents via SSH

Host ?

172.31.40.203

Credentials ?

ubuntu/***** (ubuntupasswordtest)

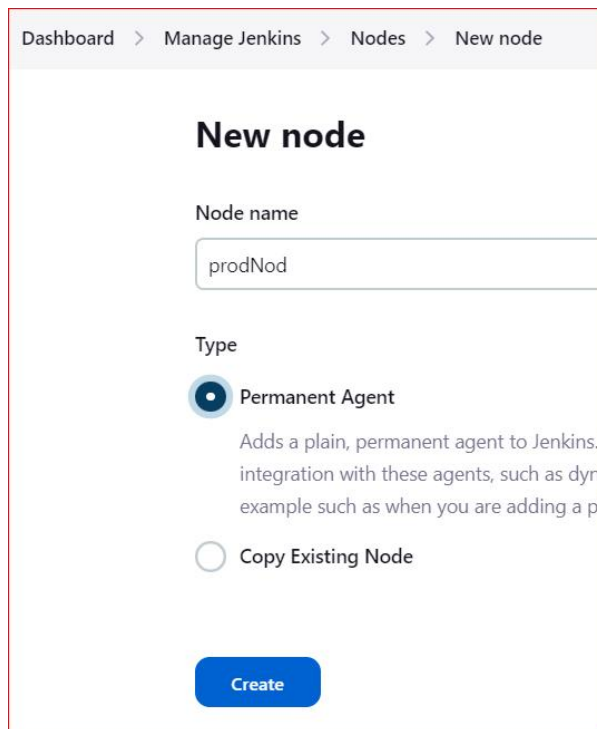
Add ▾

Host Key Verification Strategy ?

Non verifying Verification Strategy

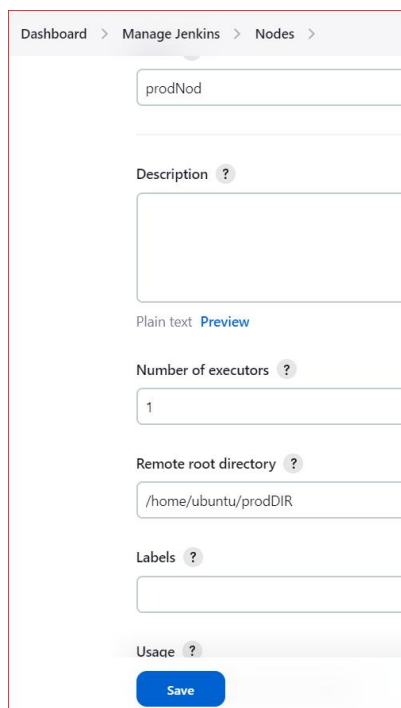
Advanced ▾

* Give a name:- manage nodes and clouds -> new node -> name
-> permanent agent -> create



The screenshot shows the 'New node' page in Jenkins. The breadcrumb navigation at the top is 'Dashboard > Manage Jenkins > Nodes > New node'. The main heading is 'New node'. There is a 'Node name' input field containing 'prodNod'. Below it, under the 'Type' section, the 'Permanent Agent' radio button is selected. A description for 'Permanent Agent' is visible: 'Adds a plain, permanent agent to Jenkins. integration with these agents, such as dynamic example such as when you are adding a ph'. The 'Copy Existing Node' radio button is unselected. At the bottom, there is a blue 'Create' button.

* Insert remote root path:- remote root directory ->
/home/ubuntu/prodDIR



This screenshot shows the same 'New node' page but with more fields visible. The 'Node name' field still contains 'prodNod'. Below it is a 'Description' field with a help icon (?). The 'Number of executors' field contains '1'. The 'Remote root directory' field contains '/home/ubuntu/prodDIR'. There is also a 'Labels' field and a 'Usage' field, both with help icons (?). A blue 'Save' button is at the bottom.

* Insert these info:- launch method -> launch agents by a ssh -> host -> paste private ip of node not server-> credential -> add -> domain-> global credential (unrestricted) -> kind -> ssh username with private key -> scope -> global (jenkins, nodes, items, all child items, etc.) -> username -> Jenkins-Node -> private key -> enter directly -> copy and paste key of node-> add -> host key verification strategy -> non verifying verification strategy -> Save -> refresh status

Dashboard > Manage Jenkins > Nodes >

Usage ⓘ

Use this node as much as possible

Launch method ⓘ

Launch agents via SSH

Host ⓘ

172.31.38.240

Credentials ⓘ

ubuntu/***** (ubuntupasswordprod)

Add ▾

Host Key Verification Strategy ⓘ

Non verifying Verification Strategy

Advanced ▾

Availability ⓘ

Save

* List of Nodes present in jenkins:-

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	4.42 GB	<div><div></div>0 B</div>	4.42 GB	0ms
	Node1	Linux (amd64)	In sync	4.94 GB	<div><div></div>0 B</div>	4.94 GB	31ms
	prodNod	Linux (amd64)	In sync	5.00 GB	<div><div></div>0 B</div>	5.00 GB	55ms
	testNod	Linux (amd64)	In sync	4.94 GB	<div><div></div>0 B</div>	4.94 GB	12ms
Data obtained		0.53 sec	0.53 sec	0.53 sec	0.53 sec	0.53 sec	0.53 sec

\$ ls

```
ubuntu@test-server:~/tesrDIR$ ls
remoting  remoting.jar
ubuntu@test-server:~/tesrDIR$
```

\$ ls

```
ubuntu@prod-server:~/prodDIR$ ls
remoting  remoting.jar
ubuntu@prod-server:~/prodDIR$
```

* Create job:- Jenkins -> new items -> name -> free style project -> ok

Enter an item name

testNOD-project

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will do anything you want for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple machines and/or organizing complex activities that do not fit a simple build.

Multi-configuration project
Suitable for projects that need a large number of builds, etc.

Folder
Creates a container that stores nested items in its own namespace, so you can have multiple items with the same name.

OK

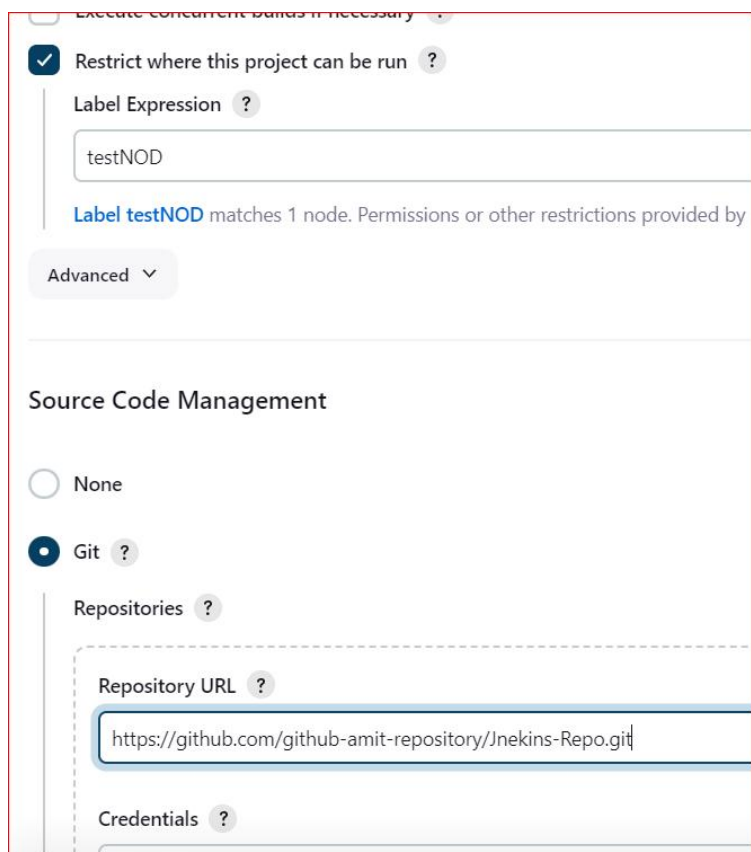
Cancel

* Insert github url:- Github project -> project url-> paste repo url
-> restrict where this project can be run



The screenshot shows the 'GitHub project' configuration section in Jenkins. It features a checked checkbox labeled 'GitHub project'. Below it is a 'Project url' field with a question mark icon, containing the text 'https://github.com/github-amit-repository/Jnekins-Repo.git'. At the bottom of this section is an 'Advanced' dropdown menu.

* Give node name:- Label expression -> testNOD->source code
managemnt -> git -> repo url



The screenshot shows two sections of the Jenkins configuration. The top section, 'Restrict where this project can be run', has a checked checkbox and a 'Label Expression' field containing 'testNOD'. Below the field, a message states: 'Label testNOD matches 1 node. Permissions or other restrictions provided by'. An 'Advanced' dropdown menu is also present. The bottom section, 'Source Code Management', has two radio buttons: 'None' (unselected) and 'Git' (selected). Below the 'Git' option is a 'Repositories' section with a 'Repository URL' field containing 'https://github.com/github-amit-repository/Jnekins-Repo.git' and a 'Credentials' field.

* Create credential and insert branch name:- branch -> */testBranch -> build triggers

https://github.com/github-amit-repository/Jenkins-Repo.git

Credentials ?

ubuntu/***** (ubuntupasswordtest)

Add ▾

Advanced ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/testBranch

Add Branch

Repository browser ?

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

Dashboard > testNOD-project > Configuration

Configure

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

☐ Build after other projects are built ?

☐ Build periodically ?

☒ GitHub hook trigger for GIT SCM polling ?

☐ Poll SCM ?

Build Environment

☐ Delete workspace before build starts

☐ Use secret text(s) or file(s) ?

☐ Add timestamps to the Console Output


☐ Inspect build log for published build scans

* Create job:- Jenkins -> new items -> name -> free style project -> ok


Enter an item name

prodNOD-project


» Required field



Freestyle project
This is the central feature of Jenkins. Je
for something other than software build



Pipeline
Orchestrates long-running activities th
and/or organizing complex activities th



Multi-configuration project
Suitable for projects that need a large

* Insert github url:- Github project -> project url-> paste repo url

☐ Discard old builds ?

☒ GitHub project

Project url ?

https://github.com/github-amit-repository/Jnekins-Repo.git

Advanced ▾

☐ This project is parameterized ?

☐ Throttle builds ?

* Give node name:- Label expression -> prodNOD->source code managemnt -> git -> repo url

☒ Restrict where this project can be run ?

Label Expression ?

prodNod

Label prodNod matches 1 node. Permissions or other restrictions pr

Advanced ▾

Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://github.com/github-amit-repository/Jnekins-Repo.git

Credentials ?

* Create credential and insert branch name:- branch -> */prodBranch

Repositories ?

Repository URL ?

https://github.com/github-amit-repository/Jenkins-Repo.git

Credentials ?

ubuntu/***** (ubuntupasswordprod)

Add ▾

Advanced ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/prodBranch

* Select gitscm:- build triggers -> github hook trigger for gitscm polling -> apply -> save

Dashboard > prodNOD-project > Configuration

Configure

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

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 ?

Build Environment

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s) ?
- ☐ Add timestamps to the Console Output
- ☐ Inspect build log for published build scans
- ☐ Terminate a build if it's stuck

\$ git checksum testBranch

```
ubuntu@Jenkins-Master:~/Git$ git checkout testBranch
Switched to branch 'testBranch'
```




\$ git add puch_to_test
\$ git commit -m "puch to test"

```
ubuntu@Jenkins-Master:~/Git$ git add puch_to_test
ubuntu@Jenkins-Master:~/Git$ git commit -m "puch to test"
[testBranch 13c04af] puch to test
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 puch_to_test
```


```
$ git push origin testBranch
```





```
ubuntu@Jenkins-Master:~/Git$ git push origin testBranch
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% (2/2), done.
Writing objects: 100% (3/3), 271 bytes | 271.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
```

* Now in github repo we can see files are added:-

 testBranch ▾  4 branches  0 tags

This branch is [1 commit ahead](#) of develop.

 **github-amit-repository** push to test

 develop_file	commit develop file
 master_file	master file
 puch_to_test	puch to test
 push_by_webhook	auto push by using webhook

Help people interested in this repository understand your project by adding a README.

```
$ ls
$ cd workspace/testNOD-project
$ ls
```

```
ubuntu@test-server:~/tesrDIR$ ls
remoting  remoting.jar
ubuntu@test-server:~/tesrDIR$ ls
remoting  remoting.jar  workspace
ubuntu@test-server:~/tesrDIR$ cd workspace/testNOD-project
ubuntu@test-server:~/tesrDIR/workspace/testNOD-project$ ls
develop_file  master_file  puch_to_test  push_by_webhook
ubuntu@test-server:~/tesrDIR/workspace/testNOD-project$
```

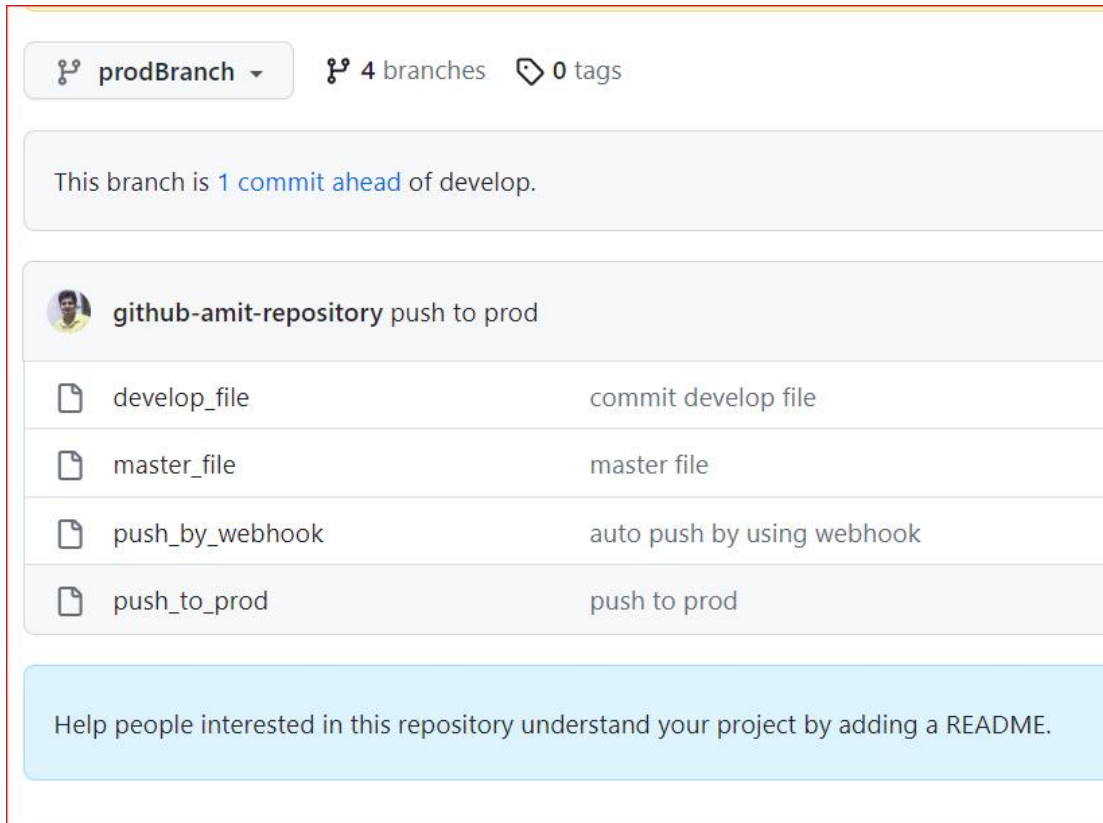
```
$ git checkout prodBranch
$ touch push_to_prod
```

```
ubuntu@Jenkins-Master:~/Git$ git checkout prodBranch
Switched to branch 'prodBranch'
ubuntu@Jenkins-Master:~/Git$ touch push_to_prod
```

```
$ git add push_to_prod
$ git commit push_to_prod -m "push to prod"
```

```
ubuntu@Jenkins-Master:~/Git$ git add push_to_prod
ubuntu@Jenkins-Master:~/Git$ git commit push_to_prod -m "push to prod"
[prodBranch acad8d9] push to prod
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 push_to_prod
ubuntu@Jenkins-Master:~/Git$ git push origin prodBranch
Username for 'https://github.com': github-amit-repository
Password for 'https://github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 277 bytes | 277.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'prodBranch' on GitHub by visiting:
remote:   https://github.com/github-amit-repository/Jnekins-Repo/pull/new/prodBranch
remote:
To https://github.com/github-amit-repository/Jnekins-Repo.git
 * [new branch]      prodBranch -> prodBranch
```


* Now in github repo we can see files are added:-



The screenshot shows a GitHub repository interface. At the top, there's a dropdown menu for 'prodBranch' with a downward arrow, followed by '4 branches' and '0 tags'. Below this, a message states 'This branch is 1 commit ahead of develop.' The main section features a commit by 'github-amit-repository' titled 'push to prod'. Below the commit, there's a table of files:

develop_file	commit develop file
master_file	master file
push_by_webhook	auto push by using webhook
push_to_prod	push to prod

At the bottom, a light blue box contains the text: 'Help people interested in this repository understand your project by adding a README.'

```
$ cd workspace
$ ls
$ cd prodNOD-project
$ ls
```

```
ubuntu@prod-server:~/prodDIR$ cd workspace/
ubuntu@prod-server:~/prodDIR/workspace$ ls
prodNOD-project  prodNOD-project@tmp
ubuntu@prod-server:~/prodDIR/workspace$ cd prodNOD-project
ubuntu@prod-server:~/prodDIR/workspace/prodNOD-project$ ls
develop_file  master_file  push_by_webhook  push_to_prod
ubuntu@prod-server:~/prodDIR/workspace/prodNOD-project$
```