

CI CD Pipeline on Jenkins

Problem statement:

Need to implement CI CD pipeline on jenkins as per below instructions

—

- Get the project from GIT : <https://github.com/asif-ops/asif-ops.github.io.git>
- Configure jenkins thus it Install the project website from git at slave-1 testing server on docker first, if successful then it should built the project website on slave2-production server.
- Create pipeline view and Trigger the job using git web-hooks , whenever any change at git repository then jenkins should notify it and jenkins will automatically start build process.

Prescription:





1. Check git repository
2. Install docker at slave nodes
3. Configure build job for Testing server
4. Configure build job for Production server
5. Configure job; after Test job done success it will build Prod job
6. Create Pipeline view to run the job
7. Create Web hook to initiate job when commit has made at GIT repository

Implementation:

First you need to setup Jenkins Master slave architecture at AWS, how to setup Master slave installation please follow this github link to do the installation <https://github.com/asif-ops/AWS-jenkins-master-slave-setup>

1. Check git repository:

<https://github.com/asif-ops/asif-ops.github.io.git> directory where we can see the website source code and a dockerfile exists as showing below :

 src	update
 Dockerfile	Dockerfile
 docker_check.sh	docker_check
 index.html	modi

2. Install docker at slave nodes:

To install the project on slaves nodes need to install the docker first, run below commands at slave nodes

```
sudo apt-get update
```

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

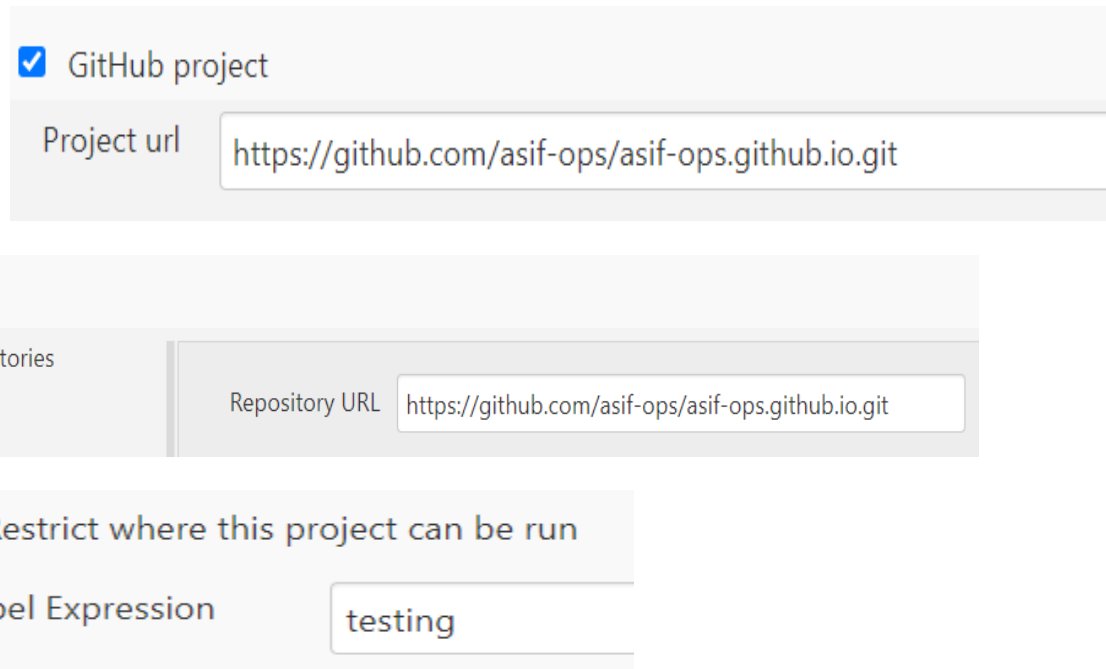
3. Configure build job for Testing server

First create job for testing server, following steps:

Jenkins GUI dashboard-> Create a job -> Freestyle project -> Enter a item name i.e slave_testing -> Ok -> GitHub project -> Project URL = <https://github.com/asif-ops/asif-ops.github.io.git> -> Source Code Management -> Git -> Repository URL = <https://github.com/asif-ops/asif-ops.github.io.git> -> Restrict where this project can be run -> Label expression = testing (as this project will run at testing so restricting to testing server only by this option) -> Save -> So

now we can check connections status between jenkins master and slave

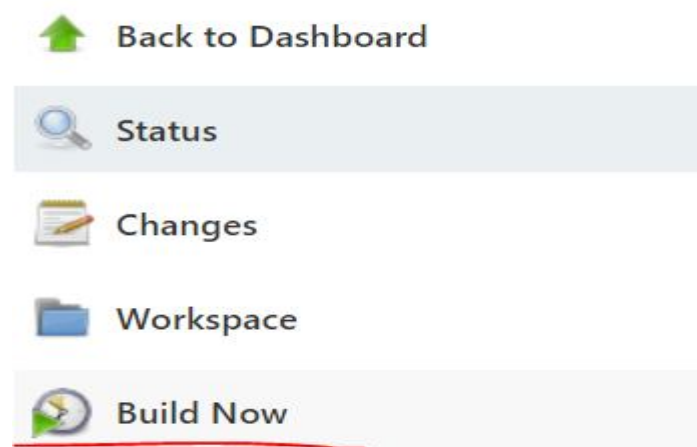
Below is the screenshot of the important steps below



The screenshot shows three configuration steps in Jenkins:

- Step 1:** A checkbox labeled "GitHub project" is checked. Below it, the "Project url" is set to `https://github.com/asif-ops/asif-ops.github.io.git`.
- Step 2:** The "Git" radio button is selected. Under the "Repositories" section, the "Repository URL" is set to `https://github.com/asif-ops/asif-ops.github.io.git`.
- Step 3:** A checkbox labeled "Restrict where this project can be run" is checked. Below it, the "Label Expression" is set to `testing`.

Now click on “Build Now” and check whether build step success or not , if the build button show blue color then build step is success



After build now we see that build process is successful as build process is showing blue color

Now login to testing server and check that whether build step is success or not. We see that at under
/home/ubuntu/jenkins/workspace/testing_slave our remote git repository is created as showing below :

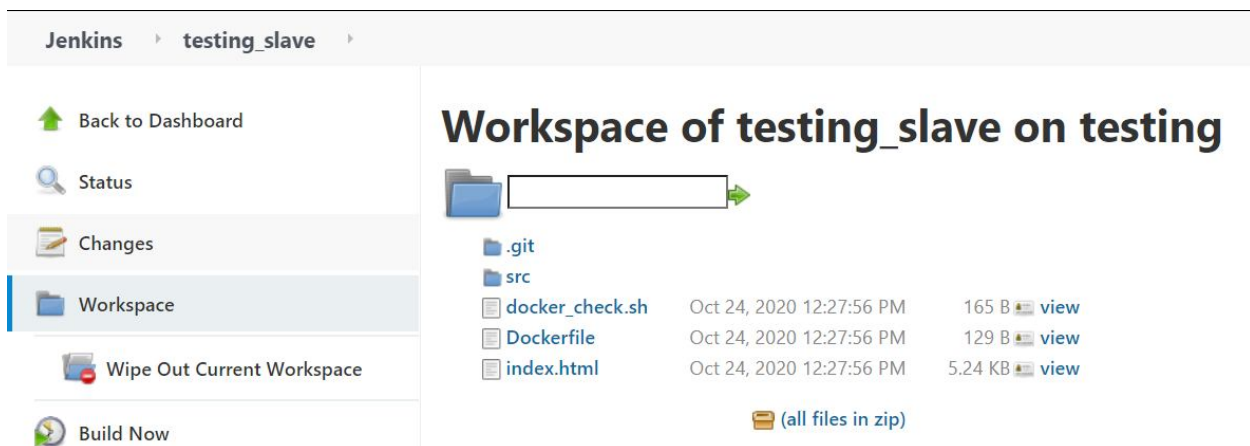
```
ubuntu@ip-172-31-12-142:~/jenkins/workspace/testing_slave$ pwd
```

```
/home/ubuntu/jenkins/workspace/testing_slave
```

```
ubuntu@ip-172-31-12-142:~/jenkins/workspace/testing_slave$ ls
```

```
Dockerfile  docker_check.sh  index.html  src
```

if I go to workspace of testing sever at jenkins GUI then it also shows content of the git repository . So that means from our master Jenkins we can successfully can connect to our slave node and apply job successfully



The screenshot displays the Jenkins web interface. The breadcrumb navigation at the top shows 'Jenkins' > 'testing_slave'. On the left sidebar, the 'Workspace' option is selected. The main panel is titled 'Workspace of testing_slave on testing' and shows a file explorer view of the workspace. It contains a list of files and directories: '.git', 'src', 'docker_check.sh' (165 B, view), 'Dockerfile' (129 B, view), and 'index.html' (5.24 KB, view). All files were last modified on 'Oct 24, 2020 12:27:56 PM'. A download icon and the text '(all files in zip)' are visible at the bottom right of the file list.

Now go back to configure the build job

 [Back to List](#)

 [Status](#)



 [Delete Agent](#)

 [Configure](#)

 [Build History](#)



Agent testing Projects tied to test

S	W	Name ↓
		testing_slave

Icon:

Build -> Execute shell

Execute shell will basically run shell commands that need to run to perform the build step

#this will make this script file executable. function of this script

#file is to rm any running container from the testing server if exists

sudo chmod +x docker_check.sh

sudo ./docker_check.sh > /home/ubuntu/jenkins/output.txt

#docker build

sudo docker build . -t web

#docker run the website on port 82

sudo docker run -it -d -p 82:80 web

#check docker process


sudo docker ps

#then check website on localhost on port 82

sudo curl localhost:82

Configure -> Build -> Execute shell

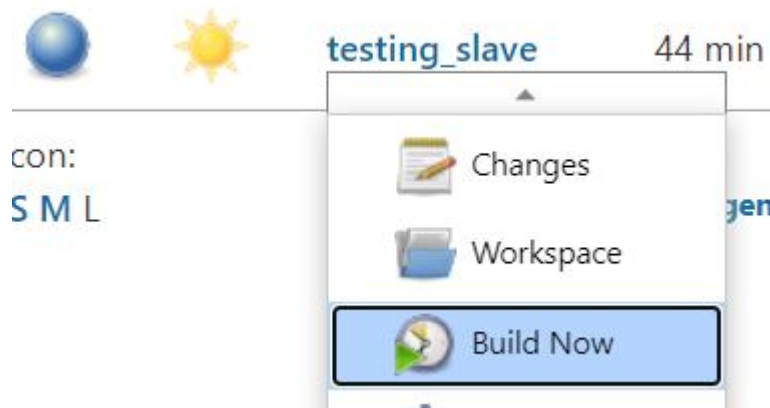
Build

 **Execute shell**

Command

```
#this will make this script file executable. function
#file is to rm any running container from the testing
sudo chmod +x docker_check.sh
sudo ./docker_check.sh
#docker build the website
sudo docker build . -t web
#docker run the website on port 82
sudo docker run -it -d -p 82:80 web
#check docker process
sudo docker ps
#then check website on localhost on port 82
sudo curl localhost:82
```

Now “Save” . Our job now has saved and next thing we need to do lets go ahead and run this job .Lets go to jenkins dash board and run “Build Now”

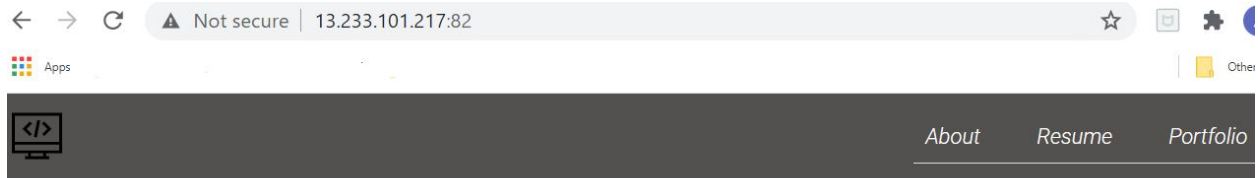


Now we see the build process is success for this step



Also we can see the console output of this build process for the log if there is any error

Finally we can browse the website at instance public IP at port 82 to see the website content as showing below . this is the project website , which tested at testing server successfully



4. Configure build job for Production server

Now configure the Production server build job same like Testing server as per below steps:

Jenkins GUI dashboard -> Free style project -> New Item -> Enter an item name: Production server -> **Jenkins GUI dashboard-> Create a job -> Freestyle project -> Enter a item name i.e slave_testing -> Ok -> GitHub project -> Project URL = <https://github.com/asif-ops/asif-ops.github.io.git> -> Source Code Management -> Git -> Repository URL = <https://github.com/asif-ops/asif-ops.github.io.git> -> Restrict where this project can be run -> Label expression = production (as this project will run at production so restricting to production server only by this option) -> Build Step -> Execute shell commands (here use the same commands as testing server) -> Save -> Run the build step and verify whether the production server has deploy the project successfully or not**

Execute shell will basically run shell commands that need to run to perform the build step

#this will make this script file executable. function of this script

#file is to rm any running container from the testing server if exists

sudo chmod +x docker_check.sh

sudo ./docker_check.sh > /home/ubuntu/jenkins/output.txt

#docker build

sudo docker build . -t web

#docker run the website on port 82

sudo docker run -it -d -p 82:80 web

#check docker process

sudo docker ps

#then check website on localhost on port 82

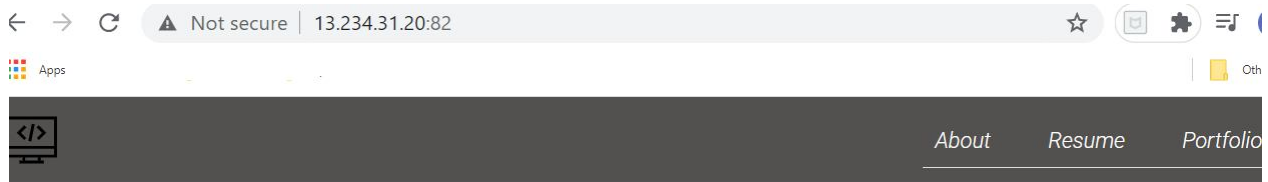
sudo curl localhost:82

Now we see the build project, found that build job is running successfully



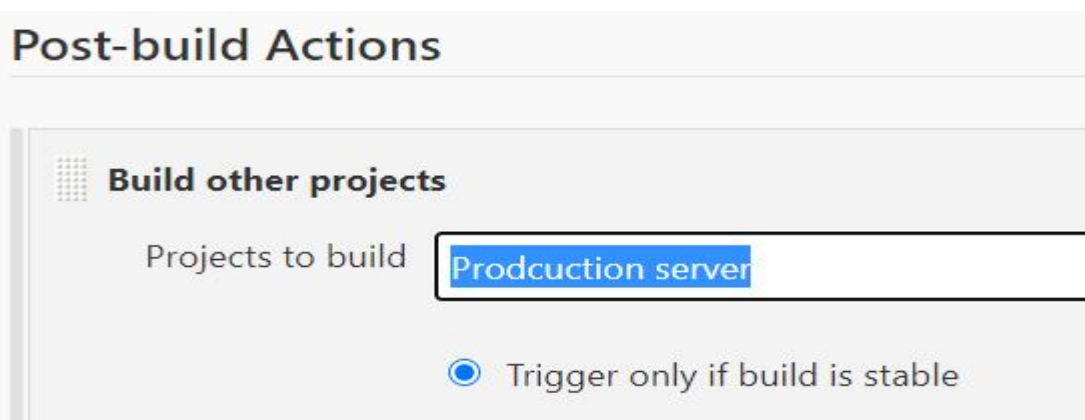
Oct 24, 2020 2:18 PM

Now we can browse the production server public IP at port and can see the website contents as showing below picture. Now testing is successfully at both production and testing server



5. Configure job; after Test job done success it will build Prod job

To do that Jenkins GUI -> testing server -> Configure -> Post-build Actions -> Select "Build other projects" from dropdown -> Projects to build = Production server -> Trigger only build is stable



Now test the build process by build now on testing serve "Builld now". After test we found that build process for testing server and after that build job production is also successful

Testing server build job



Production server build job



Browse production server ip on port 82 and found that project is website is successful l.

6. Create Pipeline view to run the job

First install the pipeline plugin by:

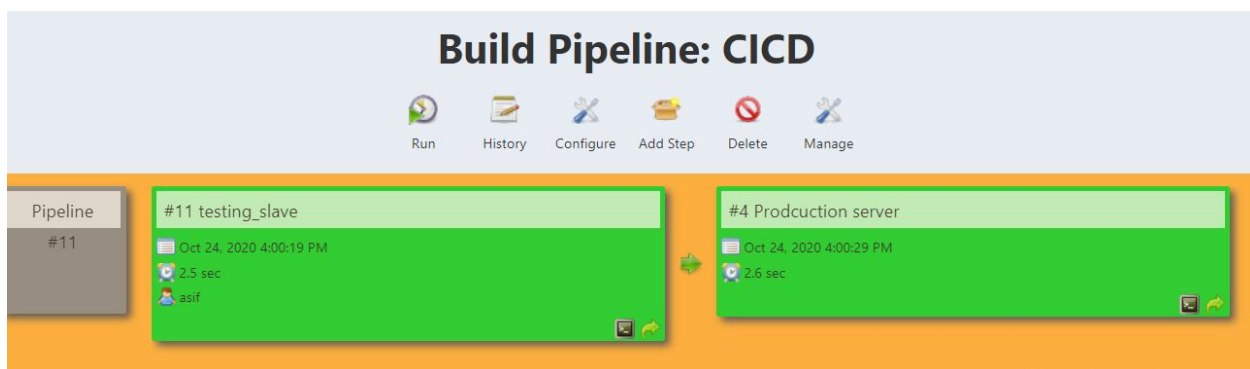
Jenkins GUI dashboard -> Manage jenkins -> Manage plugins -> Available -> search "build pipeline" -> [Build Pipeline](#) -> Install without restart

Now start create build pipeline view

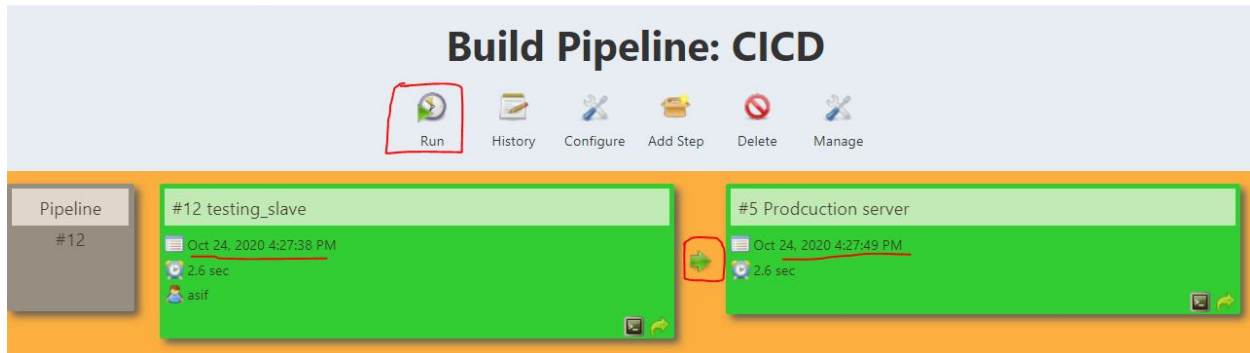
Jenkins GUI dashboard -> click + sign -> select "Build pipeline view" -> just give a view name -> ok -> Select initial Job = Testing server -> OK

Select Initial Job

Then finally Build view will look like below



Now click on “run” on same page and we found that Buile pipeline CICD has run successfully at slave node first and then it run the build at production node as showing below



7. Create Web hook to initiate job when commit has made at GIT repository

Jenkins GUI dashboard -> Testing server -> Configure -> Build Triggers -> GitHub hook trigger for GITScm polling -> Save



Now go to github repository -> settings -> Webhooks -> Add Webhook -> Payload URL = Your jenkins server IP:port - github-webhook/ (in this case <http://13.232.199.239:8080/github-webhook/>) > Add webhook

<> Code ⓘ Issues 🔑 Pull requests ▶ Actions 📁 Projects 📖 Wiki 🛡️ Security 📈 Insights ⚙️ **Settings**

Options
Manage access
Security & analysis
Branches
Webhooks

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](#).

Activate Windows

Payload URL *

`http://13.232.199.239:8080/github-webhook/`

Then if everything working fine then it will show you a tick mark as show below

Webhooks

Webhooks allow external services to be notified when certain we'll send a POST request to each of the URLs you provide. L

✓ `http://13.232.199.239:8080/githu...` (push)

Now commit some changes to your Git hub repository. In this case I did some change at index.html file and push the change to git hub . now as soon new code has been pushed to github then jenkins build starts job and job done successfully

Git hub updated:

📄 index.html index tile update 3 minutes ago

Jenkins do new build job:

