

Big Project

Tasks needed to performed to create the project.

- Create a Dockerfile.
- Docker file should to be placed on git.
- Create job in Jenkins, integrate the job with git using webhook.
- Run the job manually first from Jenkins and check if image is created and downloaded.
- Check whether downloaded image is pushed in Docker hub.
- Check if the image is downloaded from Docker Hub and creates container from it.
- Now webhook will come in picture, any changes in the source code on the GitHub repository, then Jenkins job should run and download the image from Docker Hub.
- Check If container is created then delete the container and create a new one with the changes.

Before performing the above tasks, we need to create some pre-requisites.

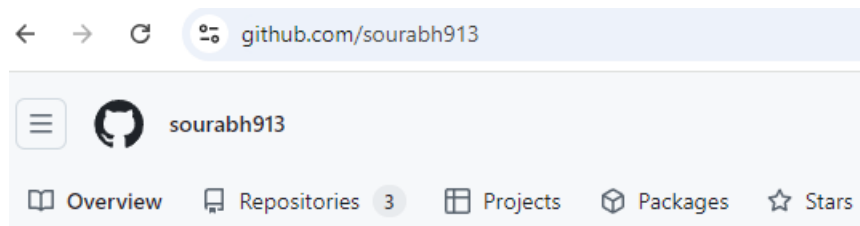
- We need to have an account in GitHub with a public repository created in it. This required so that we can put our DockerFile.
- An EC2 instance which has jenkins and docker installed on it.
- Docker Plugins should be installed & configured on Jenkins.
- Add jenkins user to docker group to execute docker commands from jenkins.
- A Docker Hub account with a public repository created in it. This is required for pushing and pulling images from Docker Hub.

Pre-requisites are performed as below:

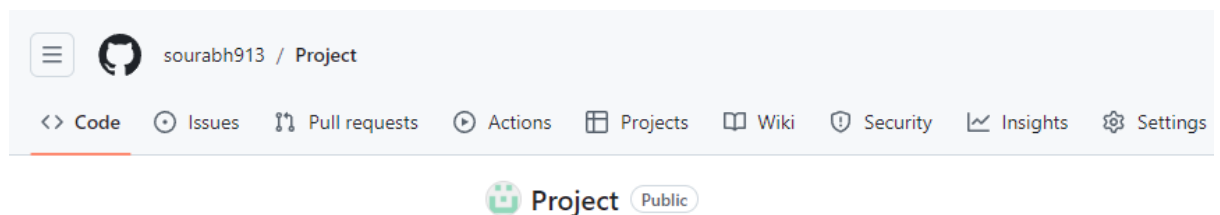
- ***We need to have an account in GitHub with a public repository created in it. This required so that we can put our Dockerfile.***

In my case, I have a GitHub Account and public repository created in it with the name “Project”. Below is the link for the repository.

GitHub Account



Project Repository



GitHub Repository link

<https://github.com/sourabh913/Project.git>

- ***An EC2 instance which has jenkins and docker installed on it.***



Docker is installed through predefined commands specified in user data.

```
#!/bin/bash

apt update -y && apt upgrade -y

apt install docker.io -y
```

EC2 Instance

| <input checked="" type="checkbox"/> | Name  | Instance ID | Instance state | Instance type |
|-------------------------------------|--|---------------------|---|---------------|
| <input checked="" type="checkbox"/> | JenDoc | i-0c2777ac3ab664e3b |  Running | t2.medium |

Docker Installed

```
root@ip-172-31-80-153:/home/ubuntu#
root@ip-172-31-80-153:/home/ubuntu# docker --version
Docker version 24.0.7, build 24.0.7-0ubuntu4.1
root@ip-172-31-80-153:/home/ubuntu#
```

Jenkins is installed using the below shell script on the same EC2 instance. We will need to make necessary changes in the script as per our need, like Jenkins URL, username, password, full name and email id



Jenkins_Installation_C
onfiguration_script.sh

<http://184.73.109.231:8080> (Jenkins URL)

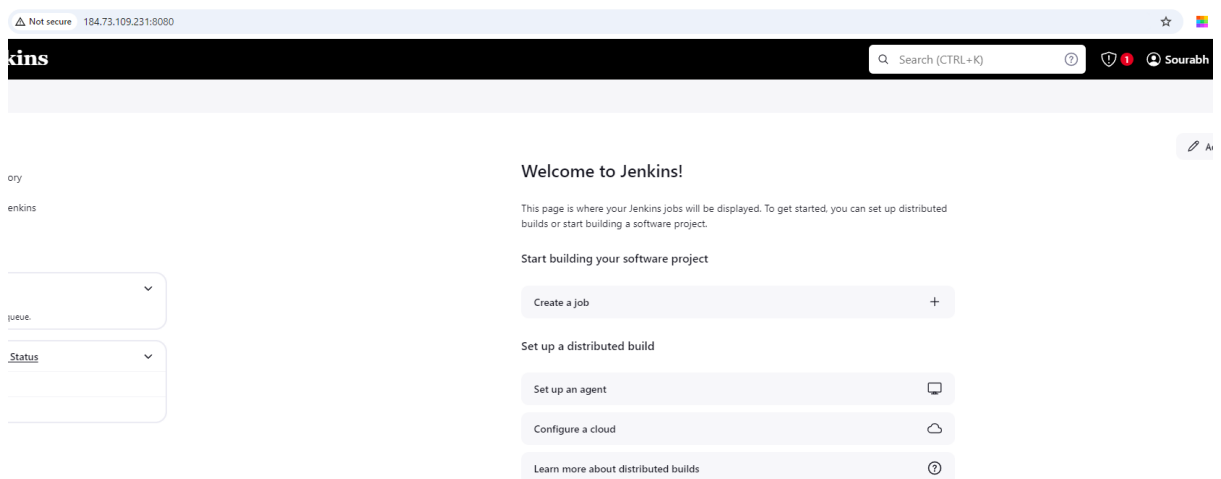
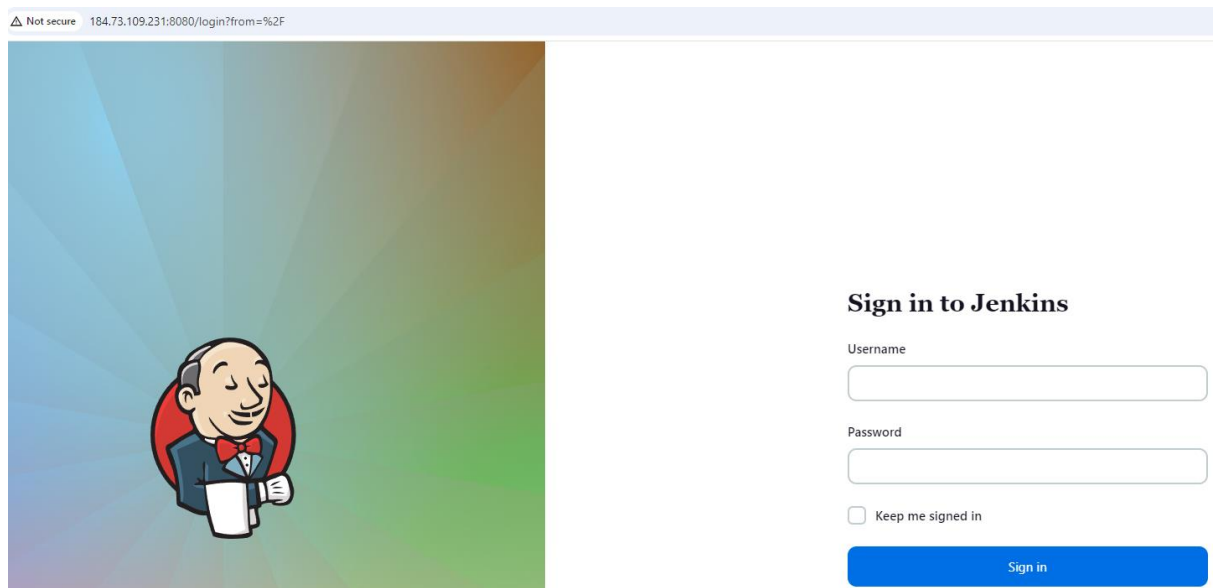
user – test

password – abc123

Full Name - Sourabh

email id – a@a.com

Jenkins Installed successfully.



➤ ***Below mentioned docker plugins should be installed.***

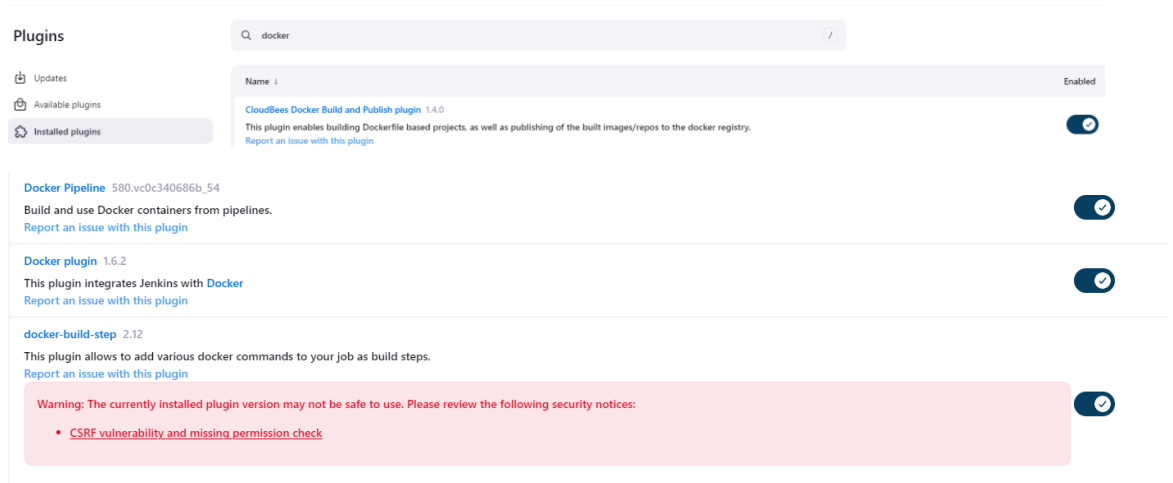
❖ Docker

❖ Docker-build-step

❖ CloudBees Docker Build and Publish

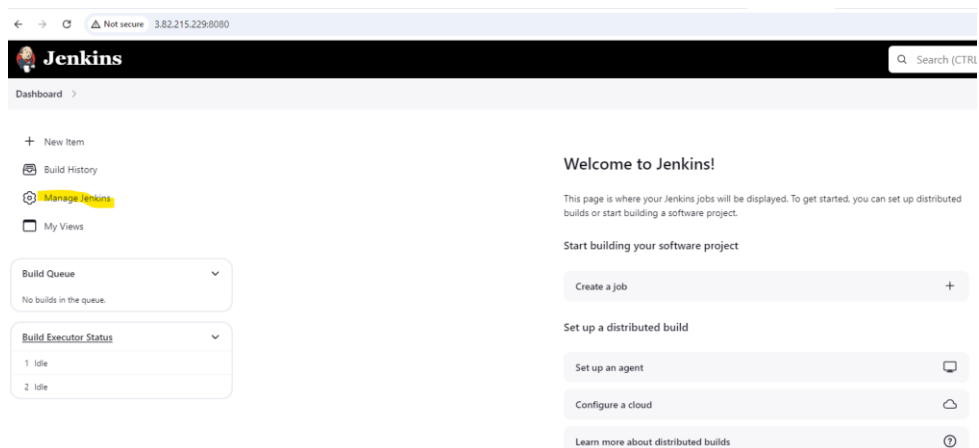
❖ Docker pipeline

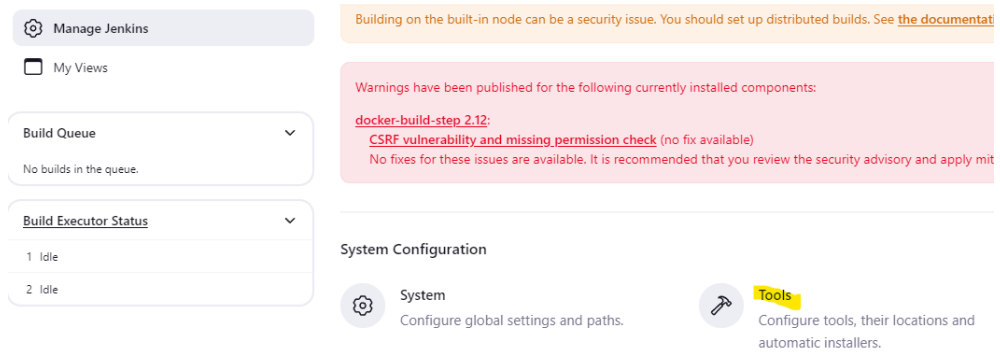
Above mentioned plugins have been installed.



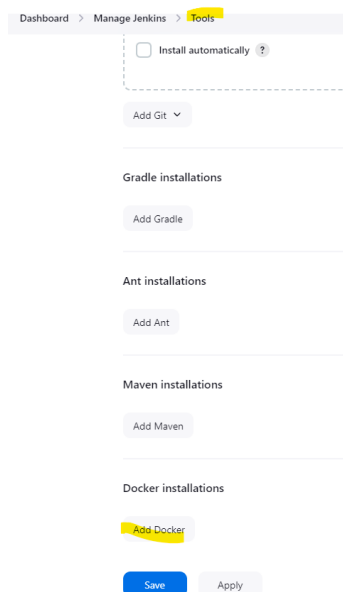
We need to configure Jenkins to use the existing docker installed on EC2 instance. Steps are below:

Go to Manage Jenkins >Tools





Under Tools > Scroll down to the bottom > Add Docker



Under Docker, type a name, in our case, we gave “Docker” only, we can leave Installation root as blank and it will take the default installation path. Click Apply and Save.

≡ Docker

Name

Docker

Installation root ?

☐ Install automatically ?

Add Docker

Save Apply

- ***Add jenkins user to docker group to execute docker commands from jenkins.***

This can be done by running the below command on EC2 instance.

usermod -a -G docker jenkins

```
root@ip-172-31-80-153:/home/ubuntu/Project# usermod -a -G docker jenkins
root@ip-172-31-80-153:/home/ubuntu/Project#
```

User addition and Plugins installation requires jenkins restart to take affect by using the below commands.

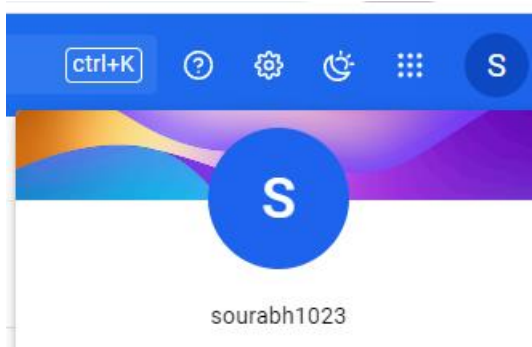
systemctl restart jenkins

```
root@ip-172-31-80-153:/home/ubuntu/Project# systemctl restart jenkins
root@ip-172-31-80-153:/home/ubuntu/Project#
root@ip-172-31-80-153:/home/ubuntu/Project#
```

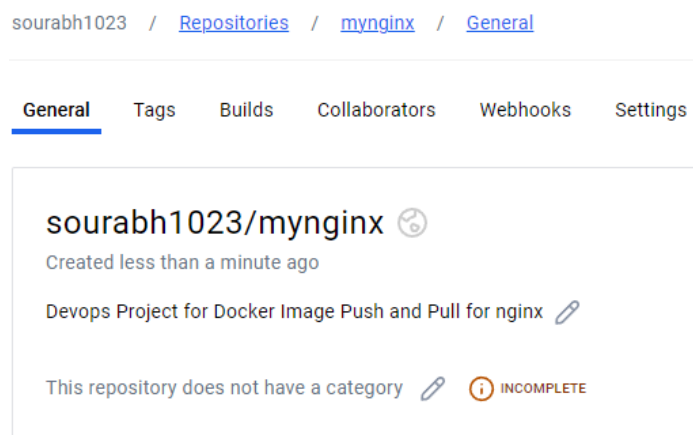

- ***A Docker Hub account with a public repository created in it. This is required for pushing and pulling images from Docker Hub.***

In my case, I have a Docker Hub Account and public repository created in it with the name “sourabh1018”. Below is the link for the repository.

Docker Hub Account



mynginx repository



Pre-requisites have been completed.

Tasks performed are as below:

- **Create a docker file.**

In our case we are creating a Nginx dockerfile.

To create a dockerfile for nginx, we need to first create index.html file.

vi index.html

```
root@ip-172-31-80-153:/home/ubuntu# vi index.html
root@ip-172-31-80-153:/home/ubuntu#
```

Contents of index.html file:

<!DOCTYPE html>

<html>

<body>

<h1>Welcome to Nginx Web Server</h1>

<p>This server is running through Docker container</p>

</body>

</html>

```
<!DOCTYPE html>
<html>
<body>
<h1>Welcome to Nginx Web Server</h1>
<p>This server is running through Docker container</p>
</body>
</html>
```

Now we will create the Nginx Dockerfile

vi Dockerfile

```
root@ip-172-31-80-153:/home/ubuntu#
root@ip-172-31-80-153:/home/ubuntu# vi Dockerfile
```

Contents of Dockerfile:

FROM nginx

LABEL maintainer="Sourabh Singh"

COPY index.html /usr/share/nginx/html

```
root@ip-172-31-81-142:/home/ubuntu
FROM nginx
LABEL maintainer="Sourabh Singh"
COPY index.html /usr/share/nginx/html
```

- Docker file to be placed on git.

To put the Dockerfile and index.html on GitHub, we will need to first clone the repository on our EC2 instance.

Cloning the Project Repository on the EC2 instance.

git clone <https://github.com/sourabh913/Project.git>

```
root@ip-172-31-80-153:/home/ubuntu# git clone https://github.com/sourabh913/Project.git
Cloning into 'Project'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 11 (delta 2), reused 9 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (11/11), done.
Resolving deltas: 100% (2/2), done.
root@ip-172-31-80-153:/home/ubuntu#
root@ip-172-31-80-153:/home/ubuntu#
```

Verify the Repository is cloned

cd Project/

```
root@ip-172-31-80-153:/home/ubuntu# cd Project/
root@ip-172-31-80-153:/home/ubuntu/Project#
root@ip-172-31-80-153:/home/ubuntu/Project#
```

Copying the index.html and Dockerfile on Project Repository.

cp index.html Dockerfile Project/

```
root@ip-172-31-80-153:/home/ubuntu# cp index.html Dockerfile Project/  
root@ip-172-31-80-153:/home/ubuntu#
```

Verify the contents have been copied on the repository.

ls -ltr

```
root@ip-172-31-80-153:/home/ubuntu/Project# ls -ltr  
total 8  
-rw-r--r-- 1 root root 145 Sep 11 07:11 index.html  
-rw-r--r-- 1 root root  82 Sep 11 07:11 Dockerfile  
root@ip-172-31-80-153:/home/ubuntu/Project#
```

Now we have to push the changes to our repository by following the below steps.

Run the below command on Project repository to check the current git status

git status (command to check git status)

```
root@ip-172-31-80-153:/home/ubuntu/Project# git status  
On branch main  
Your branch is up to date with 'origin/main'.  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
    Dockerfile  
    index.html  
  
nothing added to commit but untracked files present (use "git add" to track)  
root@ip-172-31-80-153:/home/ubuntu/Project#
```

Above screenshot shows that there are untracked files on git, so to add them we will run the below command.

git add . (this command will include all the files in the current folder for tracking)

```
root@ip-172-31-80-153:/home/ubuntu/Project# git add .  
root@ip-172-31-80-153:/home/ubuntu/Project#
```

Again, we will check git status to see if the files have been added for tracking or not.

git status (command to check git status)

```
root@ip-172-31-80-153:/home/ubuntu/Project# git status  
On branch main  
Your branch is up to date with 'origin/main'.  
  
Changes to be committed:  
  (use "git restore --staged <file>..." to unstage)  
    new file:   Dockerfile  
    new file:   index.html  
  
root@ip-172-31-80-153:/home/ubuntu/Project#
```

We can see in the above screenshot that the files are added for tracking and they need to be committed to apply the changes by using the below command.

Before running commit, we need to set the global username and user email for GitHub account on our EC2 instance as it takes user credentials of EC2 locally not the GitHub account details by using the below commands.

Setting Global user name

git config --global user.name "sourabh913" (This is for my case and it may differ for others)

```
root@ip-172-31-80-153:/home/ubuntu/Project# git config --global user.name "sourabh913"
```

Setting Global user email

git config --global user.email "ssourabh139@gmail.com"

```
root@ip-172-31-80-153:/home/ubuntu/Project# git config --global user.email "ssourabh139@gmail.com"
root@ip-172-31-80-153:/home/ubuntu/Project#
```

git commit -m "Nginx Docker" (Command is used to commit the files that have been added for tracking with -m for giving any message before committing)

```
root@ip-172-31-80-153:/home/ubuntu/Project# git commit -m "Nginx Docker"
[main 4e8ab50] Nginx Docker
2 files changed, 11 insertions(+)
create mode 100644 Dockerfile
create mode 100644 index.html
root@ip-172-31-80-153:/home/ubuntu/Project#
```

Now we will push the changes in git repository.

Before pushing the changes, we need to run the below command to check if there are any files already in the git repo which are not there in the cloned directory. If there is nothing then we are good to push our changes.

git pull (Command to pull any changes that are already there on the git repo)

```
root@ip-172-31-80-153:/home/ubuntu/Project# git pull
Already up to date.
root@ip-172-31-80-153:/home/ubuntu/Project#
```

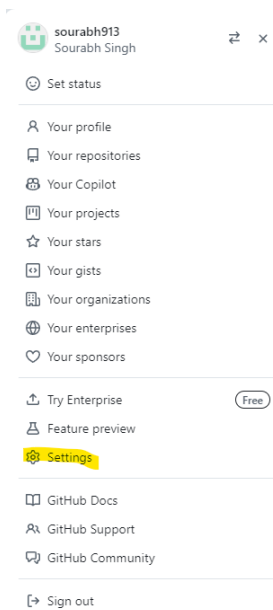
To push the files on our git repo, we need to generate PAT (personal access tokens to use as password)

Steps to generate personal access tokens are as follows:

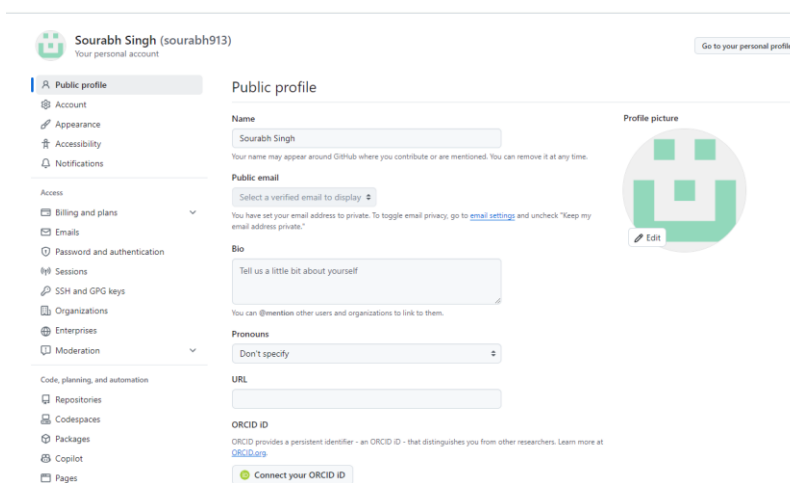
On the GitHub, dashboard, Click on profile picture.

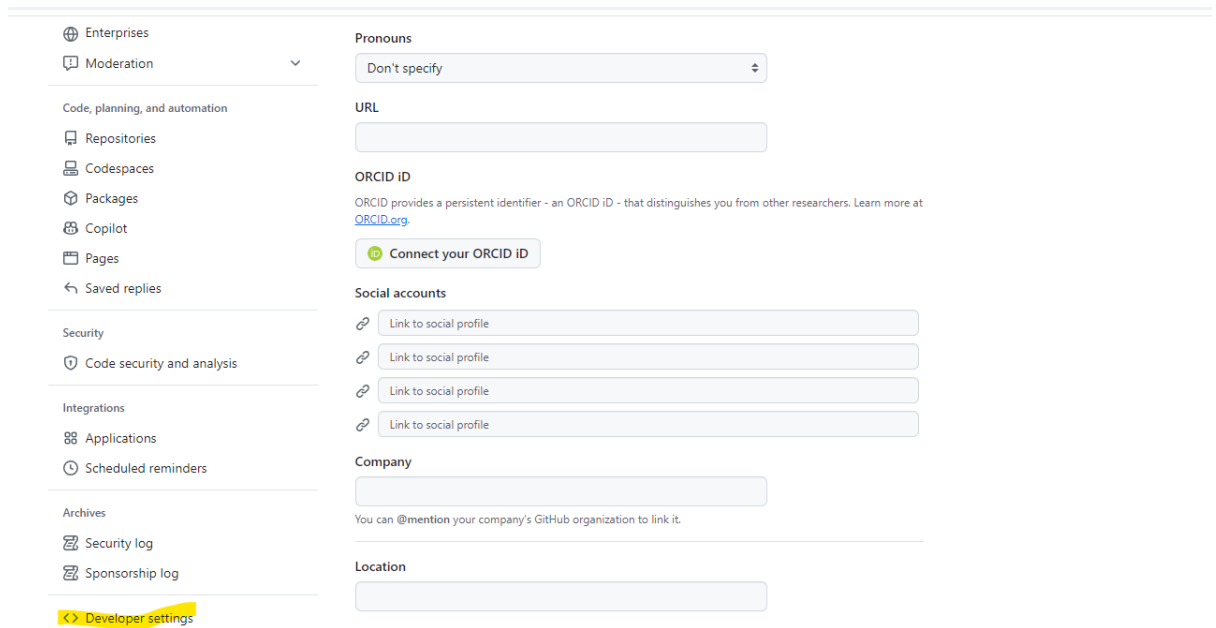


Click on Settings

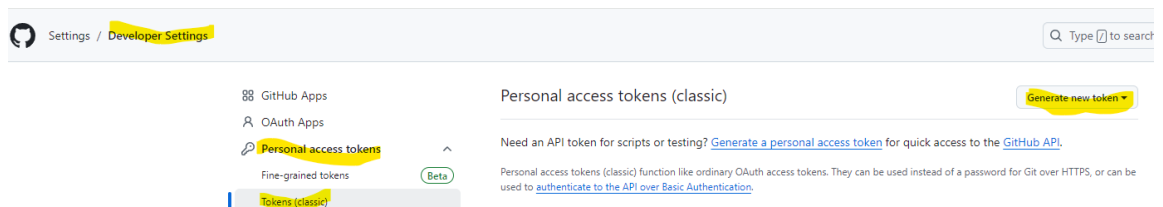


Scroll down to the bottom and click on Developer settings.

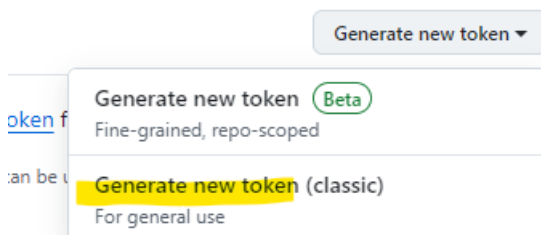





Now Click on Personal Access Tokens > Token Classic >
Click Generate new token




Under Generate new token > Drop down to Generate
new token (classic)



Enter your GitHub Credentials



Confirm access

 Signed in as @sourabh913

Password

.....

[Forgot password?](#)

Confirm

Tip: You are entering [sudo mode](#). After you've performed a sudo-protected action, you'll only be asked to re-authenticate again after a few hours of inactivity.

Now you give any description, we have selected write: packages, which will server our purpose for push and pull in Git

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens Beta

Tokens (classic)

New personal access token (classic)

Personal access tokens (classic) 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

What's this token for?

Expiration *

30 days ⌵ The token will expire on Thu, Oct 10 2024

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

| | |
|---|--|
| <input checked="" type="checkbox"/> repo | Full control of private repositories |
| <input checked="" type="checkbox"/> repo:status | Access commit status |
| <input checked="" type="checkbox"/> repo_deployment | Access deployment status |
| <input checked="" type="checkbox"/> public_repo | Access public repositories |
| <input checked="" type="checkbox"/> repo:invite | Access repository invitations |
| <input checked="" type="checkbox"/> security_events | Read and write security events |
| <input type="checkbox"/> workflow | Update GitHub Action workflows |
| <input checked="" type="checkbox"/> write:packages | Upload packages to GitHub Package Registry |
| <input checked="" type="checkbox"/> read:packages | Download packages from GitHub Package Registry |

Once the above is done, we will click Generate token

Generate token

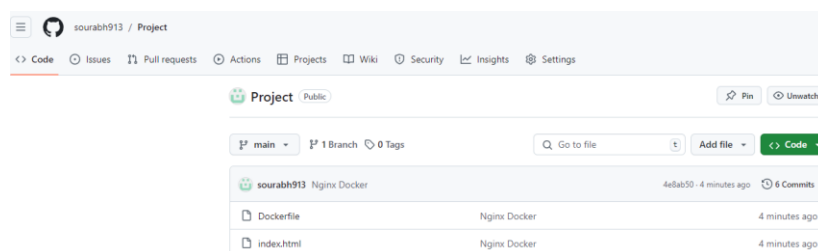
Personal access tokens (classic) Generate new token Revoke all

1. *Journal of the American Medical Association*, 2000; 283: 2689-2695.

Now we will push our changes in git using our GitHub credentials

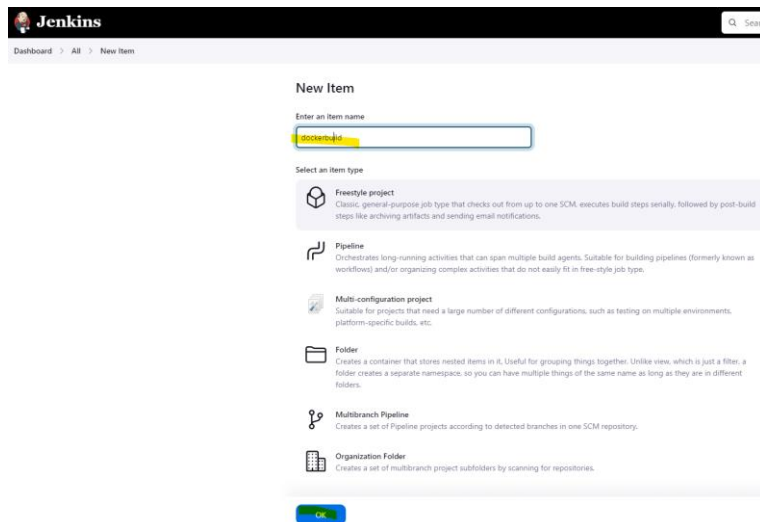
git push origin main (Command to push the changes to GitHub Repo)

```
root@ip-172-31-80-153:/home/ubuntu/Project# git push origin main
Username for 'https://github.com': sourabh913
Password for 'https://sourabh913@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 481 bytes | 481.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/sourabh913/Project.git
 397bf3e..4e8ab50  main -> main
```



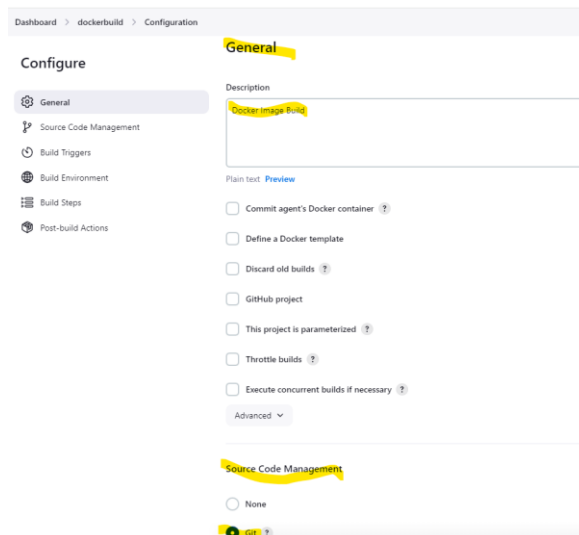
- **Create job in Jenkins, integrate the job with git**

Create a freestyle Job named “dockerbuild”



Give the General Description

Under Source Code Management > Select Git



Under Repository URL > Type the URL

<https://github.com/sourabh913/Project.git>

Dashboard > dockerbuild > Configuration

Configure

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

☐ Throttle builds ?

☐ Execute concurrent builds if necessary ?

Advanced ▾

Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://github.com/sourabh913/Project.git

Under Branch we will write */main since our GitHub repo has main branch

Branches to build ?

Branch Specifier (blank for 'any') ?

*/main

Add Branch

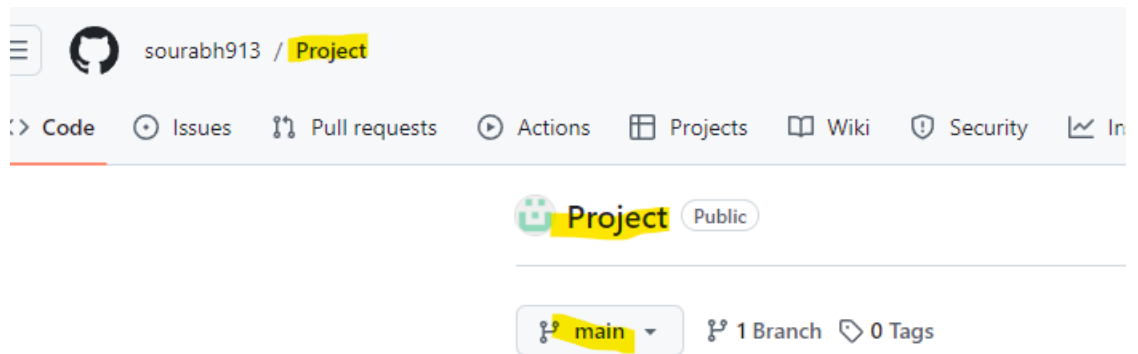
Repository browser ?

(Auto) ▾

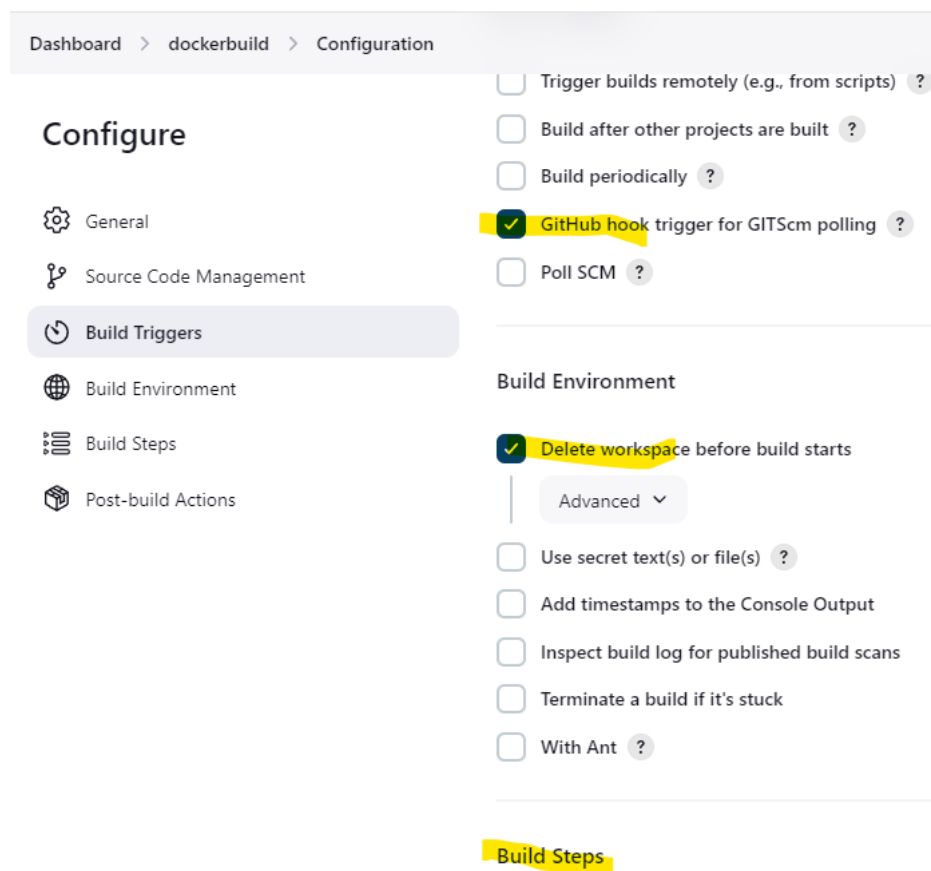
Additional Behaviours

Add ▾

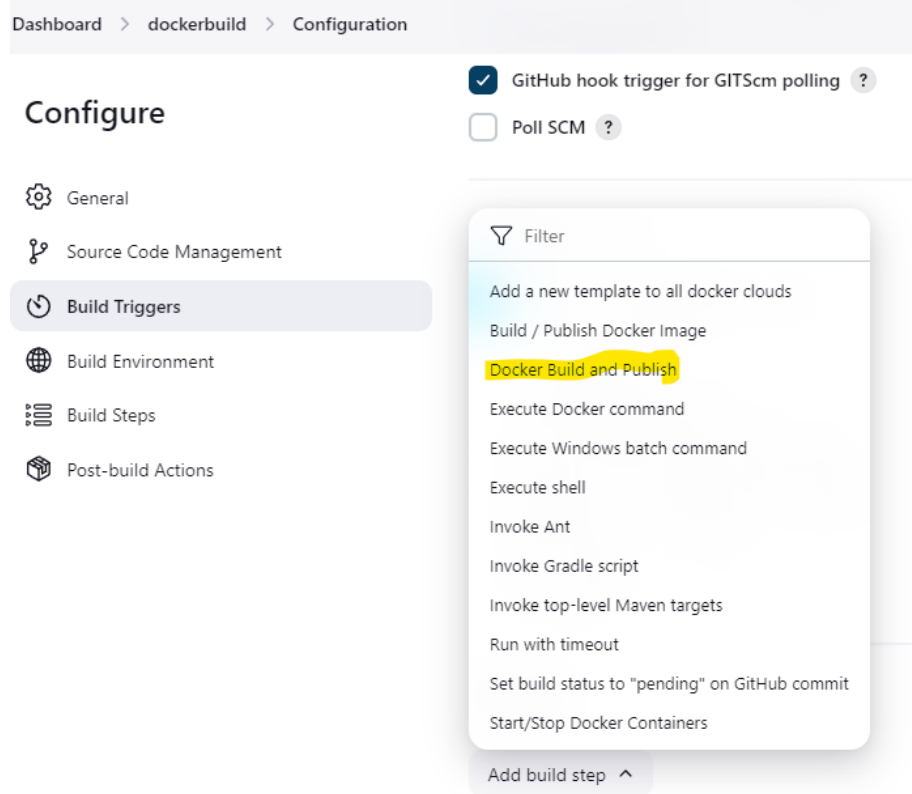
Verifying the same on our GitHub Repo.



Now we will select the option of GitHub hook trigger for GITScm polling and under Build Environment > Select Delete Workspace before build starts



Under Build Steps > Add Build Step and then drop down/up to Docker Build and Push

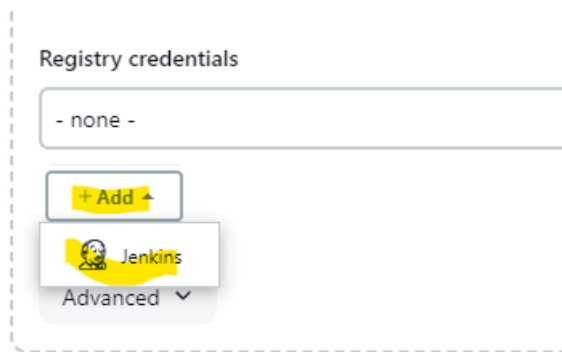


Under Build Steps>Repository name> we will give our repository details created on Docker Hub i.e.

sourabh1023/mynginx



Now we will add the registry credentials i.e. our Docker Hub Credentials. Click Add > Jenkins

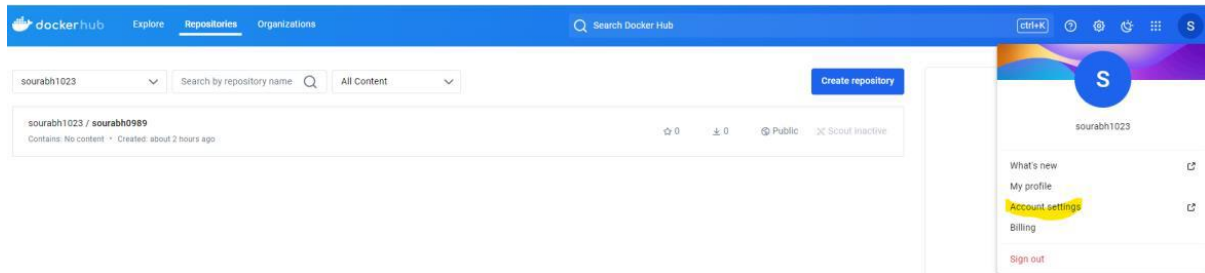


We will give our username as below.

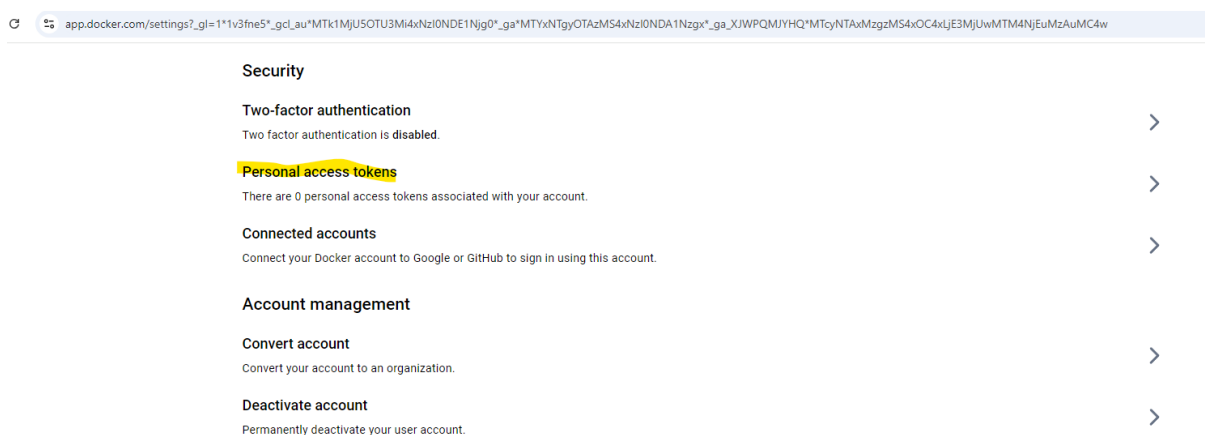
A screenshot of the Jenkins 'Add Credentials' form. The form is titled 'Jenkins Credentials Provider: Jenkins'. It has a section 'Add Credentials' with the following fields: 'Domain' (dropdown menu with 'Global credentials (unrestricted)' selected), 'Kind' (dropdown menu with 'Username with password' selected), 'Scope' (dropdown menu with 'Global (Jenkins, nodes, items, all child items, etc)' selected), 'Username' (text input field with a yellow highlight), 'Treat username as secret' (checkbox), and 'Password' (text input field with a yellow highlight).

To put password for the Docker hub in Jenkins, we need to generate a personal access token on the Docker hub by using the below steps

On your docker hub, Go to Account Settings



Under Account Settings > Personal access tokens



There will be new access token creation request.

Fill the token details

Under access and permissions select **Read & Write**

click generate

[Settings](#) / [Personal access tokens](#) / New access token

Create access token

A personal access token is similar to a password except you can have many tokens and revoke access to each one at any time. [Learn more](#)

Access token description

test

Access permissions

Read & Write

Read & Write tokens allow you to push images to any repository managed by your account.

Cancel

Generate

You will get the below screen with token details. Copy the token as password in the jenkins registry credentials screen

Copy access token

Use this token as a password when you sign in from the Docker CLI client. [Learn more](#)

Make sure you copy your personal access token now. Your personal access token is only displayed once. It isn't stored and can't be retrieved later.

Access token description

test

Access permissions

Read & Write

To use the access token from your Docker CLI client:

1. Run

```
$ docker login -u sourabh1023
```

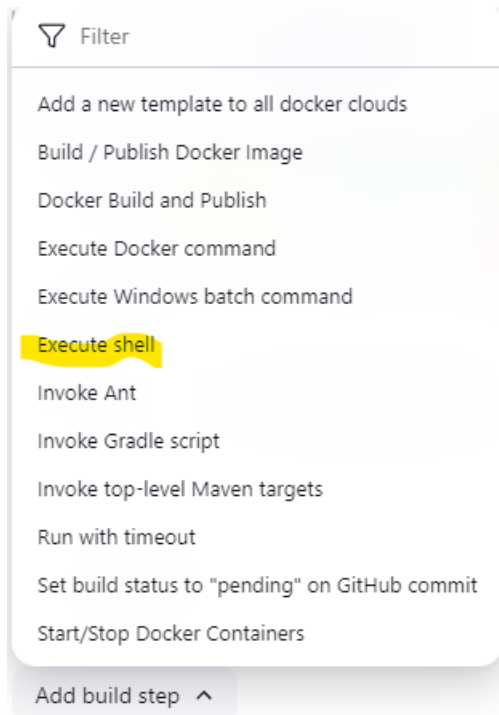
Copy

2. At the password prompt, enter the personal access token.

```
docker_login: Dd31upM1ndHusjM-B8Dn...K9Y
```

Copy

Now again we will click on Add build step and select Execute Shell.



In the shell, we will write the below shell script, which will check if there are any containers matching our given name, if there are any containers running with same name, it will stop and delete the container and create a new one.

```
container_name="my-nginx-container"
```

```
if [ "$(docker ps -q -f name=$container_name)" ]; then
```

```
    echo "Container $container_name is running. It will be  
stopped ... "
```

```
docker stop $container_name
```

```
docker rm $container_name
```

```
else
```

```
echo "Container $container_name is not running."
```

```
if [ "!(docker container inspect $container_name > /dev/null 2>&1)" ]; then
```

```
    # cleanup
```

```
    echo "removing stopped container $(docker rm $container_name)"
```

```
fi
```

```
fi
```

```
docker run --name $container_name -d -p 8081:80  
sourabh1023/mynginx
```

≡ Execute shell ?

Command

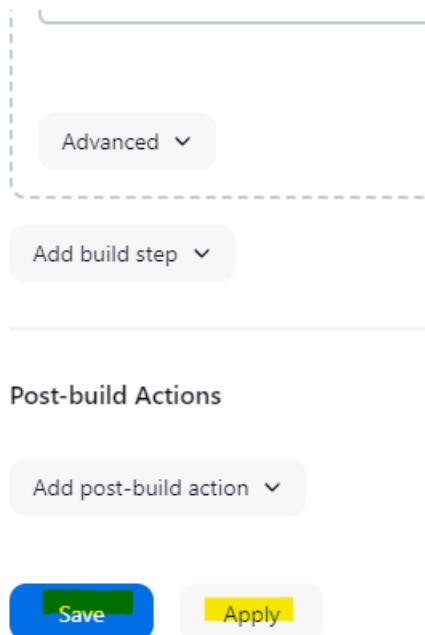
See [the list of available environment variables](#)

```
container_name="my-nginx-container"

if [ "$(docker ps -q -f name=$container_name)" ]; then
    echo "Container $container_name is running. It will be stopped ..."
    docker stop $container_name
    docker rm $container_name
else
    echo "Container $container_name is not running."
    if [ "!(docker container inspect $container_name > /dev/null 2>&1)" ]; then
        # cleanup
        echo "removing stopped container $(docker rm $container_name)"
    fi
fi

docker run --name $container_name -d -p 8081:80 sourabh1023/mynginx
```

Then Click Apply and Save the Build configuration.



The screenshot shows a configuration interface with a dashed border. Inside, there is a tab labeled 'Advanced' with a dropdown arrow. Below the tab is a button labeled 'Add build step' with a dropdown arrow. Underneath this is a section titled 'Post-build Actions'. Below the title is a button labeled 'Add post-build action' with a dropdown arrow. At the bottom of the interface are two buttons: a blue 'Save' button and a yellow 'Apply' button.

Now we will integrate this project with our GitHub Repo using webhook,

This is used when a job needs to be triggered when there is any change in the GitHub repo.

Steps to followed are below:

While configuring the job, Check the option of GitHub hook trigger for

GITScm polling under Build Triggers and then click Save.

Dashboard > dockerbuild > Configuration

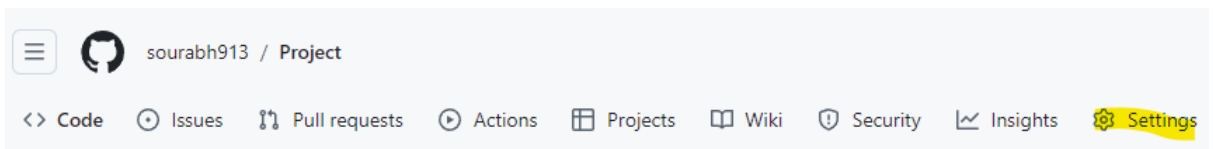
Configure

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

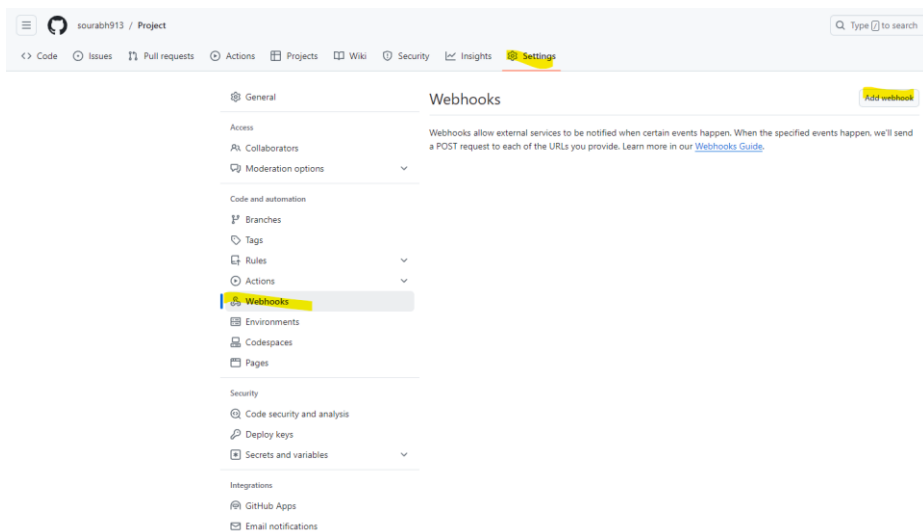
Build Triggers

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

Then on the GitHub side, Go to Settings



Select Webhook and Click on Add Webhook



Now we will give our Jenkins URL under Payload URL in the below manner.

<http://184.73.109.231:8080/github-webhook/>

Rest other settings will remain as it is on the page and will click add webhook.

Webhooks / Manage webhook

Settings

Recent Deliveries

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 *

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.

☒ **Active**

We will deliver event details when this hook is triggered.

Update webhook

Delete webhook

We see that the Webhook is created successfully.

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](#).

✓ <http://184.73.109.231:8080/github-...> (push)

Last delivery was successful.

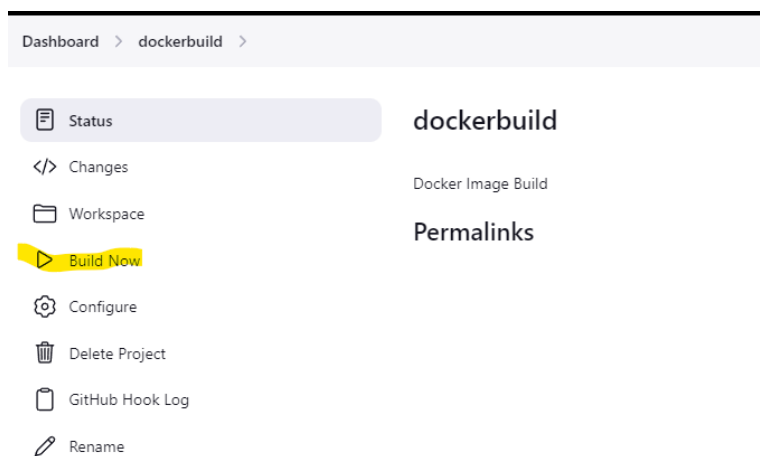
Edit

Delete

- **Run the job manually first from Jenkins and check if image is created and downloaded.**

We will run the build for the first time manually to see if gets successful.

Run Build now on the dockerbuild project



We see the build got successful. In my case I did some R&D, i.e. the Build Number is showing 22.

Dashboard > dockerbuild >

Status

</> Changes

Workspace

Build Now

Configure

Delete Project

GitHub Hook Log

Rename

✓ dockerbuild

Docker Image Build and Push

Permalinks

- Last build (#22 sourabh1023/mynginx), 2 sec ago
- Last stable build (#21 sourabh1023/mynginx:v1), 1 hr 46 min ago
- Last successful build (#21 sourabh1023/mynginx:v1), 1 hr 46 min ago
- Last failed build (#18 sourabh1023/mynginx:v1), 2 hr 1 min ago
- Last unsuccessful build (#18 sourabh1023/mynginx:v1), 2 hr 1 min ago
- Last completed build (#21 sourabh1023/mynginx:v1), 1 hr 46 min ago

Build History

trend ▾

Filter...

✓ #22 sourabh1023/mynginx

Sep 11, 2024, 12:12 PM

We will see the console output and it shows the details that took place.

Status

</> Changes

Console Output

Edit Build Information

Delete build '#22 sourabh1023/mynginx'

Git Build Data

Docker Fingerprints

Previous Build

✓ Console Output

Started by user Sourabh

Running as SYSTEM

Building in workspace /var/lib/jenkins/workspace/dockerbuild

[MS-CLEANUP] Deleting project workspace...

[MS-CLEANUP] Deferred wipeout is used...

[MS-CLEANUP] Done

The recommended git tool is: NONE

No credentials specified

Cloning the remote Git repository

Cloning repository <https://github.com/sourabh913/Project.git>

> git init /var/lib/jenkins/workspace/dockerbuild # timeout=10

Fetching upstream changes from <https://github.com/sourabh913/Project.git>

> git --version # timeout=10

> git --version # 'git version 2.43.0'

> git fetch --tags --force --progress -- <https://github.com/sourabh913/Project.git> +refs/heads/*:refs/remotes/origin/* # timeout=10

> git config remote.origin.url <https://github.com/sourabh913/Project.git> # timeout=10

> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10

Avoid second fetch

> git rev-parse refs/remotes/origin/main^{commit} # timeout=10

Checking out Revision 6d5385a24945038db8df861593035752a945baf0 (refs/remotes/origin/main)

> git config core.sparsecheckout # timeout=10

> git checkout -f 6d5385a24945038db8df861593035752a945baf0 # timeout=10

Commit message: "Update index.html"

> git rev-list --no-walk 6d5385a24945038db8df861593035752a945baf0 # timeout=10

[dockerbuild] \$ docker build -t sourabh1023/mynginx --pull=true /var/lib/jenkins/workspace/dockerbuild

DEPRECATED: The legacy builder is deprecated and will be removed in a future release.

Install the builds component to build images with BuildKit:

<https://docs.docker.com/go/buildx/>

Sending build context to Docker daemon 95.74kB

```
Step 1/3 : FROM nginx
latest: Pulling from library/nginx
a2318d6c47ec: Pulling fs layer
095d327c79ae: Pulling fs layer
bbfaa25db775: Pulling fs layer
7bb6fb0cfb2b: Pulling fs layer
0723edc10c17: Pulling fs layer
24b3fdc4d1e3: Pulling fs layer
3122471704d5: Pulling fs layer
7bb6fb0cfb2b: Waiting
0723edc10c17: Waiting
24b3fdc4d1e3: Waiting
3122471704d5: Waiting
bbfaa25db775: Verifying Checksum
bbfaa25db775: Download complete
7bb6fb0cfb2b: Verifying Checksum
7bb6fb0cfb2b: Download complete
0723edc10c17: Verifying Checksum
0723edc10c17: Download complete
24b3fdc4d1e3: Verifying Checksum
24b3fdc4d1e3: Download complete
3122471704d5: Verifying Checksum
3122471704d5: Download complete
a2318d6c47ec: Verifying Checksum
a2318d6c47ec: Download complete
095d327c79ae: Verifying Checksum
095d327c79ae: Download complete
a2318d6c47ec: Pull complete
095d327c79ae: Pull complete
bbfaa25db775: Pull complete
7bb6fb0cfb2b: Pull complete
0723edc10c17: Pull complete
24b3fdc4d1e3: Pull complete
3122471704d5: Pull complete
Digest: sha256:04ba374043ccd2fc5c593885c0eacddebabd5ca375f9323666f28dfd5a9710e3
Status: Downloaded newer image for nginx:latest
```

```
---> 39286ab8a5e1
Step 2/3 : LABEL maintainer="Sourabh Singh"
---> Running in ff2c0f4dee1b
Removing intermediate container ff2c0f4dee1b
---> 63f69ee7a88f
Step 3/3 : COPY index.html /usr/share/nginx/html
---> ffeed4bd5fd6
Successfully built ffeed4bd5fd6
Successfully tagged sourabh1023/mynginx:latest
[dockerbuild] $ docker inspect ffeed4bd5fd6
[dockerbuild] $ docker push sourabh1023/mynginx
Using default tag: latest
The push refers to repository [docker.io/sourabh1023/mynginx]
6836a26f9fc6: Preparing
11de3d47036d: Preparing
16907864a2d0: Preparing
2bdf51597158: Preparing
0fc6bb94eec5: Preparing
eda13eb24d4c: Preparing
67796e30ff04: Preparing
8e2ab394fabf: Preparing
eda13eb24d4c: Waiting
67796e30ff04: Waiting
8e2ab394fabf: Waiting
11de3d47036d: Mounted from library/nginx
16907864a2d0: Mounted from library/nginx
0fc6bb94eec5: Mounted from library/nginx
2bdf51597158: Mounted from library/nginx
67796e30ff04: Mounted from library/nginx
8e2ab394fabf: Mounted from library/nginx
eda13eb24d4c: Mounted from library/nginx
6836a26f9fc6: Pushed
latest: digest: sha256:e29601464d1d1fa62da9200ff8a2242fc2f1d64c8196e035d86eb41af18fc913 size: 1985
[dockerbuild] $ /bin/sh -xe /tmp/jenkins14227242351297693328.sh
+ container_name=my-nginx-container
+ docker ps -q -f name=my-nginx-container
+ [ ]
+ echo Container my-nginx-container is not running.
Container my-nginx-container is not running.
+ docker container inspect my-nginx-container
```

```
+ [ ! ]
+ docker rm my-nginx-container
Error response from daemon: No such container: my-nginx-container
+ echo removing stopped container
removing stopped container
+ docker run --name my-nginx-container -d -p 8081:80 sourabh1023/mynginx
a45bd18ba149a6bf5f3f4e7f2e4bc69c4717c7714fd6b71f89e261c0eb8587d2
Finished: SUCCESS
```

To verify that our image is downloaded on the EC2 instance, we will run the below command.

docker images

```
root@ip-172-31-80-153:/home/ubuntu/Project# docker images
REPOSITORY          TAG       IMAGE ID       CREATED        SIZE
sourabh1023/mynginx latest    ffeed4bd5fd6   6 minutes ago  188MB
nginx                latest    39286ab8a5e1   3 weeks ago   188MB
root@ip-172-31-80-153:/home/ubuntu/Project#
```


The above screenshot confirms that image is created and downloaded on the EC2 instance.

- **Check whether downloaded image is pushed in Docker hub.**



Now we will check whether image is pushed on docker Hub.



Below Screenshot confirms that image is pushed on Docker Hub.

General Tags Builds Collaborators Webhooks Settings

sourabh1023/mynginx 



Last pushed 13 minutes ago

This repository does not have a description   INCOMPLETE

This repository does not have a category   INCOMPLETE

Tags

This repository contains 1 tag(s).

| Tag | OS | Type | Pulled | Pushed |
|--|---|-------|--------|----------------|
|  latest |  | Image | --- | 13 minutes ago |

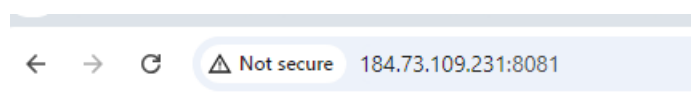
[See all](#)

- **Check if the image is downloaded from Docker Hub and creates container from it.**

Below screenshot confirms that container is created from the downloaded image.

```
root@ip-172-31-80-153:/home/ubuntu/Project# docker ps -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED        STATUS        PORTS                               NAMES
a45bd18ba149   sourabh1023/mynginx   "/docker-entrypoint..." 14 minutes ago Up 14 minutes 0.0.0.0:8081->80/tcp, :::8081->80/tcp my-nginx-container
root@ip-172-31-80-153:/home/ubuntu/Project#
```

Below is the output of our first Project run for nginx



Welcome to Nginx Web Server

This server is running through Docker container

- Now webhook will come in picture, any changes in the source code on the GitHub repository, then Jenkins job should run and download the image from Docker Hub.

In this task, we will edit the index.html file and make the below changes, commit and push those changes on the GitHub to see if the Jenkins job is triggered automatically.

vi index.html

```
root@ip-172-31-80-153:/home/ubuntu# vi index.html
root@ip-172-31-80-153:/home/ubuntu#
```

Contents of changed index.html file:

<!DOCTYPE html>

<html>

<body>

<h1>Welcome to Nginx Web Server</h1>

<p>This server is running through Sourabh Singh docker container</p>

</body>

</html>

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Welcome to Nginx Web Server</h1>
    <p>This server is running through Sourabh Singh docker container</p>
  </body>
</html>
```

Now we have to push the changes to our repository by following the below steps.

Run the below command on Project repository to check the current git status

git status (command to check git status)

```
root@ip-172-31-80-153:/home/ubuntu/Project# git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
root@ip-172-31-80-153:/home/ubuntu/Project#
```

Above screenshot shows that there are untracked files on git, so to add them we will run the below command.

git add . (this command will include all the files in the current folder for tracking)


```
root@ip-172-31-80-153:/home/ubuntu/Project# git add .  
root@ip-172-31-80-153:/home/ubuntu/Project#
```

Again, we will check git status to see if the files have been added for tracking or not.

git status (command to check git status)

```
root@ip-172-31-80-153:/home/ubuntu/Project# git status  
On branch main  
Your branch is up to date with 'origin/main'.  
  
Changes to be committed:  
  (use "git restore --staged <file>..." to unstage)  
        modified:   index.html  
  
root@ip-172-31-80-153:/home/ubuntu/Project#
```

We can see in the above screenshot that the files are added for tracking and they need to be committed to apply the changes by using the below command.

git commit -m "Index File Changes" (Command is used to commit the files that have been added for tracking with -m for giving any message before committing)

```
root@ip-172-31-80-153:/home/ubuntu/Project# git commit -m "Index File Changes"  
[main 810724f] Index File Changes  
1 file changed, 1 insertion(+), 1 deletion(-)  
root@ip-172-31-80-153:/home/ubuntu/Project#
```

Now we will push the changes in git repository.

Before pushing the changes, we need to run the below command to check if there are any files already in the git repo which are not there in the cloned directory. If there is nothing then we are good to push our changes.

git pull (Command to pull any changes that are already there on the git repo)

```
root@ip-172-31-80-153:/home/ubuntu/Project# git pull
Already up to date.
root@ip-172-31-80-153:/home/ubuntu/Project#
```

git push origin main (Command to push the changes to GitHub Repo)

```
root@ip-172-31-80-153:/home/ubuntu/Project# git push origin main
Username for 'https://github.com': sourabh913
Password for 'https://sourabh913@github.com':
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 481 bytes | 481.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/sourabh913/Project.git
   6d5385a..be17e42  main -> main
root@ip-172-31-80-153:/home/ubuntu/Project#
```

We can see that immediately after pushing the changes, Jenkins Job Ran and it got successful. Build 23 completed successfully.

Dashboard > dockerbuild >

Status

</> Changes

Workspace

Build Now

Configure

Delete Project

GitHub Hook Log

Rename

dockerbuild

Docker Image Build and Push

Permalinks

- Last build (#21 sourabh1023/mynginx:v1), 41 min ago
- Last stable build (#21 sourabh1023/mynginx:v1), 41 min ago
- Last successful build (#21 sourabh1023/mynginx:v1), 41 min ago
- Last failed build (#18 sourabh1023/mynginx:v1), 56 min ago
- Last unsuccessful build (#18 sourabh1023/mynginx:v1), 56 min ago
- Last completed build (#21 sourabh1023/mynginx:v1), 41 min ago

Build History

trend

Filter...

#23 sourabh1023/mynginx

Sep 11, 2024, 12:46 PM

Build details shows that webhook worked as it took the changes in index.html

Status

</> Changes

Console Output

Edit Build Information

Delete build '#23 sourabh1023/mynginx'

Polling Log

Git Build Data

Docker Fingerprints

Previous Build

#23 sourabh1023/mynginx (Sep 11, 2024, 12:46:44 PM)

Started by GitHub push by sourabh913

git

Revision: be17e42cc442de1641d92f6c7ffa26524630ced1

Repository: <https://github.com/sourabh913/Project.git>

refs/remotes/origin/main

</>

Changes

1. Index File Changes (details / githubweb)

Console Output gives the below output

✓ Console Output

```
Started by GitHub push by sourabh913
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/dockerbuild
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] Done
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/sourabh913/Project.git
> git init /var/lib/jenkins/workspace/dockerbuild # timeout=10
Fetching upstream changes from https://github.com/sourabh913/Project.git
> git --version # timeout=10
> git --version # 'git version 2.43.0'
> git fetch --tags --force --progress -- https://github.com/sourabh913/Project.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/sourabh913/Project.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision be17e42cc442de1641d92f6c7ffa26524630ced1 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f be17e42cc442de1641d92f6c7ffa26524630ced1 # timeout=10
Commit message: "Merge branch 'main' of https://github.com/sourabh913/Project"
> git rev-list --no-walk 6d5385a24945038db8df861593035752a945baf0 # timeout=10
[dockerbuild] $ docker build -t sourabh1023/mynginx --pull=true /var/lib/jenkins/workspace/dockerbuild
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  101.9kB
```

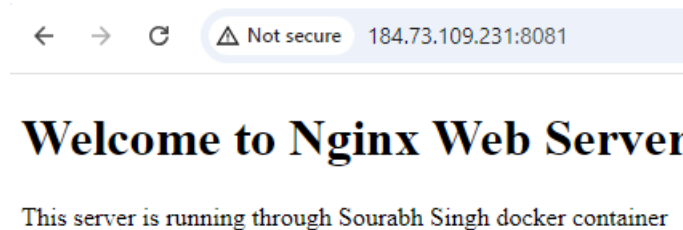
```
Sending build context to Docker daemon  101.9kB

Step 1/3 : FROM nginx
latest: Pulling from library/nginx
Digest: sha256:04ba374043ccd2fc5c593885c0eacddebbd5ca375f9323666f28dfd5a9710e3
Status: Image is up to date for nginx:latest
--> 39286ab8a5e1
Step 2/3 : LABEL maintainer="Sourabh Singh"
--> Using cache
--> 63f69ee7a88f
Step 3/3 : COPY index.html /usr/share/nginx/html
--> aaadae0e74e2
Successfully built aaadae0e74e2
Successfully tagged sourabh1023/mynginx:latest
[dockerbuild] $ docker inspect aaadae0e74e2
[dockerbuild] $ docker push sourabh1023/mynginx
Using default tag: latest
The push refers to repository [docker.io/sourabh1023/mynginx]
d2354766c4da: Preparing
11de3d47036d: Preparing
16907864a2d0: Preparing
2bdf51597158: Preparing
0fc6bb94eec5: Preparing
eda13eb24d4c: Preparing
67796e30ff04: Preparing
8e2ab394fabf: Preparing
eda13eb24d4c: Waiting
67796e30ff04: Waiting
8e2ab394fabf: Waiting
11de3d47036d: Layer already exists
2bdf51597158: Layer already exists
0fc6bb94eec5: Layer already exists
16907864a2d0: Layer already exists
67796e30ff04: Layer already exists
eda13eb24d4c: Layer already exists
8e2ab394fabf: Layer already exists
d2354766c4da: Pushed
latest: digest: sha256:49a73e93edc7ce961501b79504732ed62466f93587b7e585b71060dde526d59f size: 1985
[dockerbuild] $ /bin/sh -xe /tmp/jenkins15130552738214791667.sh
...

```

```
+ container_name=my-nginx-container
+ docker ps -q -f name=my-nginx-container
+ [ a45bd18ba149 ]
+ echo Container my-nginx-container is running. It will be stopped ...
Container my-nginx-container is running. It will be stopped ...
+ docker stop my-nginx-container
my-nginx-container
+ docker rm my-nginx-container
my-nginx-container
+ docker run --name my-nginx-container -d -p 8081:80 sourabh1023/mynginx
01f023eedd3bb26909524af173885ac6145efbdc81ceb6ed4ac3b18d4dcfe09
Finished: SUCCESS
```

Below is the output of after the index.html changes were done for nginx



- **Check If container is created then delete the container and create a new one with the changes.**

Below screenshot of console output shows that the existing container with the same name is delete and new container with same name is created with new container ID.

```

+ container_name=my-nginx-container
+ docker ps -q -f name=my-nginx-container
+ [ a45bd18ba149 ]
+ echo Container my-nginx-container is running. It will be stopped ...
Container my-nginx-container is running. It will be stopped ...
+ docker stop my-nginx-container
my-nginx-container
+ docker rm my-nginx-container
my-nginx-container
+ docker run --name my-nginx-container -d -p 8081:80 sourabh1023/mynginx
01f023eedd3bb26909524af173885ac6145efbdc81ceb6ed4ac3b18d4dcfe09
Finished: SUCCESS

```

Container running before we made changes to the index.html file.

```

root@ip-172-31-80-153:/home/ubuntu/Project#
root@ip-172-31-80-153:/home/ubuntu/Project# docker ps -a

```

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|--------------|---------------------|-------------------------|----------------|---------------|---------------------------------------|--------------------|
| a45bd18ba149 | sourabh1023/mynginx | "/docker-entrypoint..." | 14 minutes ago | Up 14 minutes | 0.0.0.0:8081->80/tcp, :::8081->80/tcp | my-nginx-container |

```

root@ip-172-31-80-153:/home/ubuntu/Project#

```

Container running after we made changes to the index.html file.

```

root@ip-172-31-80-153:/home/ubuntu/Project# docker ps -a

```

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|--------------|---------------------|-------------------------|----------------|---------------|---------------------------------------|--------------------|
| 01f023eedd3b | sourabh1023/mynginx | "/docker-entrypoint..." | 16 minutes ago | Up 16 minutes | 0.0.0.0:8081->80/tcp, :::8081->80/tcp | my-nginx-container |

```

root@ip-172-31-80-153:/home/ubuntu/Project#
root@ip-172-31-80-153:/home/ubuntu/Project#

```

All the tasks have been completed successfully.