ASSESSMENT TASK

Submitted By: Aditya Jaishi

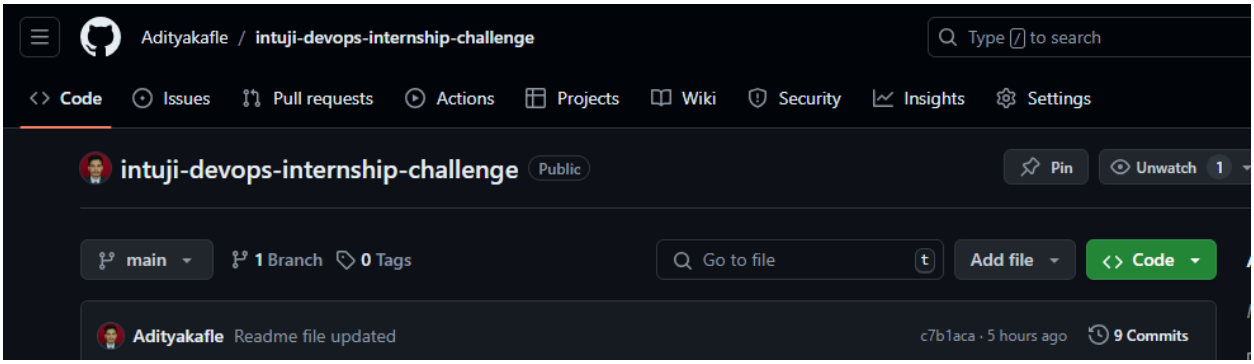
==========

==========

The challenge was to setup Dockerfiles, Docker Compose files and configuring a Jenkins CI/CD pipeline for a Simple PHP hello-world application.

Step 1: Create a GitHub repository:

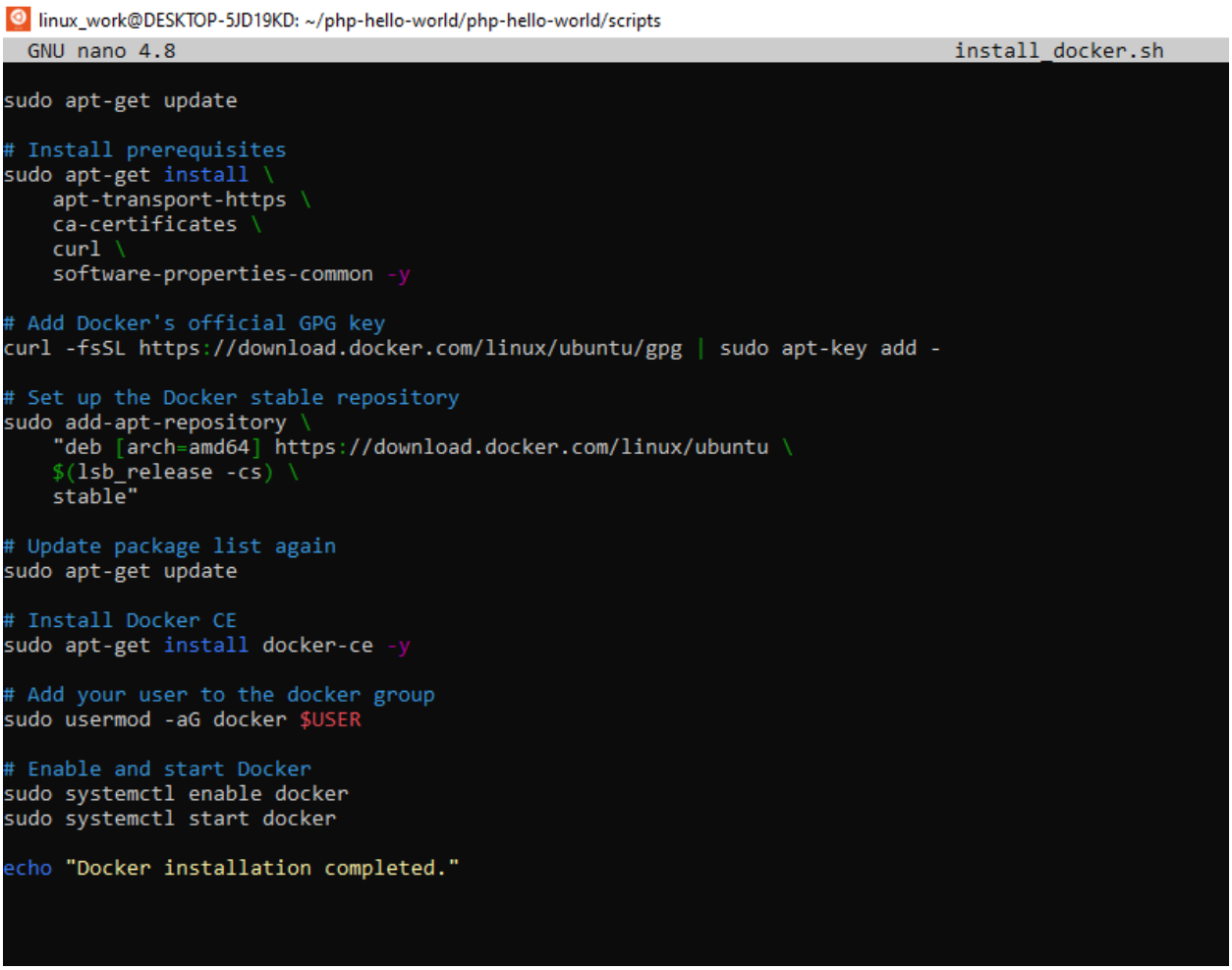
First, I created a public GitHub repository named “intuji-devops-internship-challenge.”



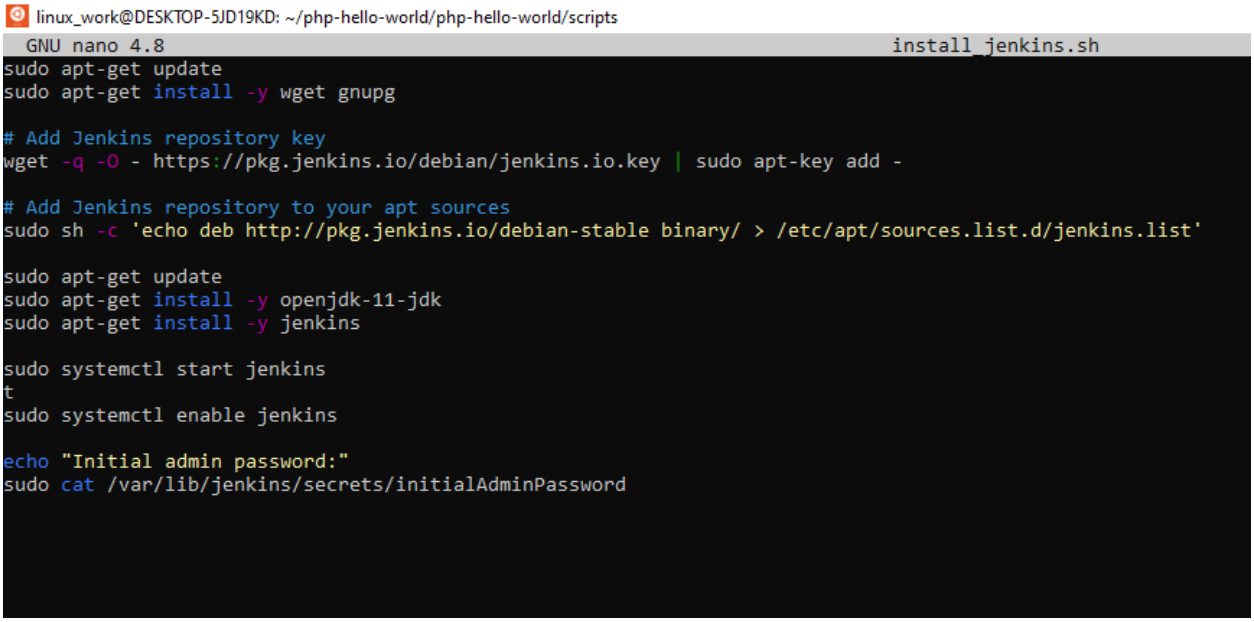
Step 2: Install Docker and Jenkins

I installed Docker and Jenkins on my Ubuntu system by running the following bash script.

Install_docker.sh



Install_jenkins.sh



Now, we have installed docker and Jenkins

Step 3: Clone the GitHub repository:

I cloned the repository “https://github.com/silarhi/php-hello-world” into the current directory.

Step 4: Create a Dockerfile

After that, I created a Dockerfile and an index.php file.

Dockerfile

linux_work@DESKTOP-5JD19KD: ~/php-hello-world/php-hello-world

GNU nano 4.8 Dockerfile

```
FROM php:7.4-apache

# Copy the project files into the container
COPY . /var/www/html/

# Set the working directory
WORKDIR /var/www/html

# Set the correct permissions
RUN chown -R www-data:www-data /var/www/html && \
    chmod -R 755 /var/www/html

# Update Apache configuration to allow access
RUN echo "<Directory /var/www/html>\n\
    Options Indexes FollowSymLinks\n\
    AllowOverride None\n\
    Require all granted\n\
</Directory>" >> /etc/apache2/apache2.conf

# Install dependencies using composer
RUN apt-get update && \
    apt-get install -y git zip unzip && \
    docker-php-ext-install pdo pdo_mysql && \
    curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer && \
    composer install && \
    composer dump-autoload


# Expose port 80
EXPOSE 80
```

Index.php

It is required to demonstrate the usage of a class named Hello from a library that is autoloaded via Composer.

linux_work@DESKTOP-5JD19KD: ~/php-hello-world/php-hello-world

GNU nano 4.8 index.php

```
<?php

require 'vendor/autoload.php';

use Silarhi\Hello;

$hello = new Hello();
echo $hello->display();
```

Step 5: Create Image using Dockerfile

```
linux_work@DESKTOP-5JD19KD:~/php-hello-world$ sudo nano Dockerfile
[sudo] password for linux_work:
linux_work@DESKTOP-5JD19KD:~/php-hello-world$ docker build -t adityaji777/php-project .
[+] Building 26.0s (10/10) FINISHED                                    docker:default
 => [internal] load build definition from Dockerfile                           0.3s
 => => transferring dockerfile: 494B                                           0.0s
 => [internal] load metadata for docker.io/library/php:7.4-apache              2.6s
 => [auth] library/php:pull token for registry-1.docker.io                     0.0s
 => [internal] load .dockerignore                                              0.3s
 => => transferring context: 2B                                                0.0s
 => [internal] load build context                                              0.4s
 => => transferring context: 2.75kB                                            0.0s
 => CACHED [1/4] FROM docker.io/library/php:7.4-apache@sha256:c9d7e608f73832   0.0s
```

```
 => [2/6] COPY . /var/www/html/                                                                                  7.4s
 => [3/6] WORKDIR /var/www/html                                                                                  2.1s
 => [4/6] RUN chown -R www-data:www-data /var/www/html &&     chmod -R 755 /var/www/html                        18.4s
 => [5/6] RUN echo "<Directory /var/www/html>\n    Options Indexes FollowSymLinks\n    AllowOverride None\n    Require all granted\n</Directory>" >> /etc/apache2  4.4s
 => [6/6] RUN apt-get update &&     apt-get install -y git zip unzip &&     docker-php-ext-install pdo pdo_mysql &&     curl -sS https://getcomposer.org/instal  102.8s
 => exporting to image                                                                                           4.3s
 => => exporting layers                                                                                          4.0s
 => => writing image sha256:9c5cdd6ff4874f71e12dec59f848becfed5e940dbb5723c760ac63db043fe852                     0.1s
 => => naming to docker.io/adityaji777/php-project                                                               0.2s
```

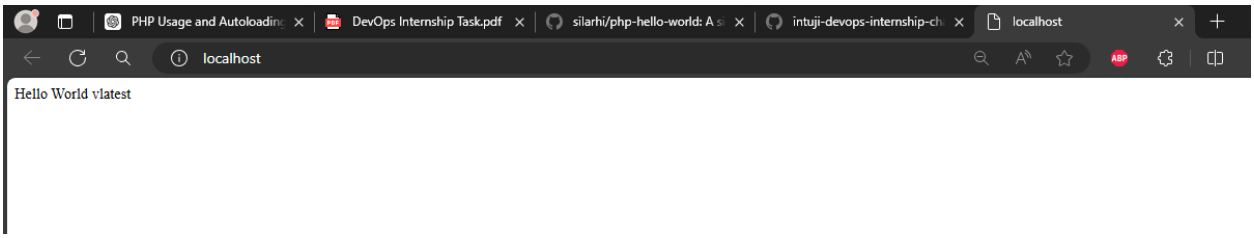
For test purpose I run one container name Aditya to obtain output

```
linux_work@DESKTOP-5JD19KD:~/php-hello-world/php-hello-world$ docker run -d -p 80:80 --name aditya adityaji777/php-project
b7dcd74ab0bb4621057d93fe66c62d4e1198e87a8dfd75875fa9fa072ff20917
linux_work@DESKTOP-5JD19KD:~/php-hello-world/php-hello-world$ docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED         STATUS         PORTS                NAMES
b7dcd74ab0bb   adityaji777/php-project   "docker-php-entrypoi…"   2 minutes ago   Up 2 minutes   0.0.0.0:80->80/tcp   aditya
linux_work@DESKTOP-5JD19KD:~/php-hello-world/php-hello-world$
```

```
linux_work@DESKTOP-5JD19KD:~/php-hello-world/php-hello-world$ docker images
REPOSITORY               TAG       IMAGE ID       CREATED         SIZE
adityaji777/php-project   latest    9c5cdd6ff487   6 minutes ago   519MB
linux_work@DESKTOP-5JD19KD:~/php-hello-world/php-hello-world$
```

Up to this point, we have created the image and tested the output by running the container.

The content can be seen on localhost:80.



Step 6 : Push the image in docker hub by login to docker hub

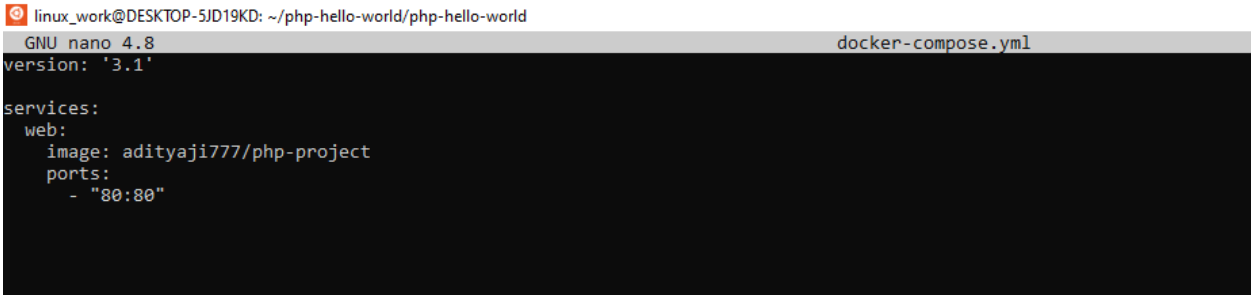
Command:

- Docker login: *docker login (*provide required credentials*)*
- Docker push: *docker push adityaji777/php-project*

```
linux_work@DESKTOP-5JD19KD:~/php-hello-world/php-hello-world$ docker push adityaji777/php-project
Using default tag: latest
The push refers to repository [docker.io/adityaji777/php-project]
fec669dba44e: Pushing [==>                                                ]  3.305MB/66.09MB
7756bc8e721b: Pushed
dcf2965cfb2f: Pushed
5f70bf18a086: Layer already exists
a89bb8ac7636: Pushed
```

Step 7: Create Docker Compose File

Created a file named docker-compose.yml with the following content:



Run the Application Using Docker Compose:

docker-compose up -d

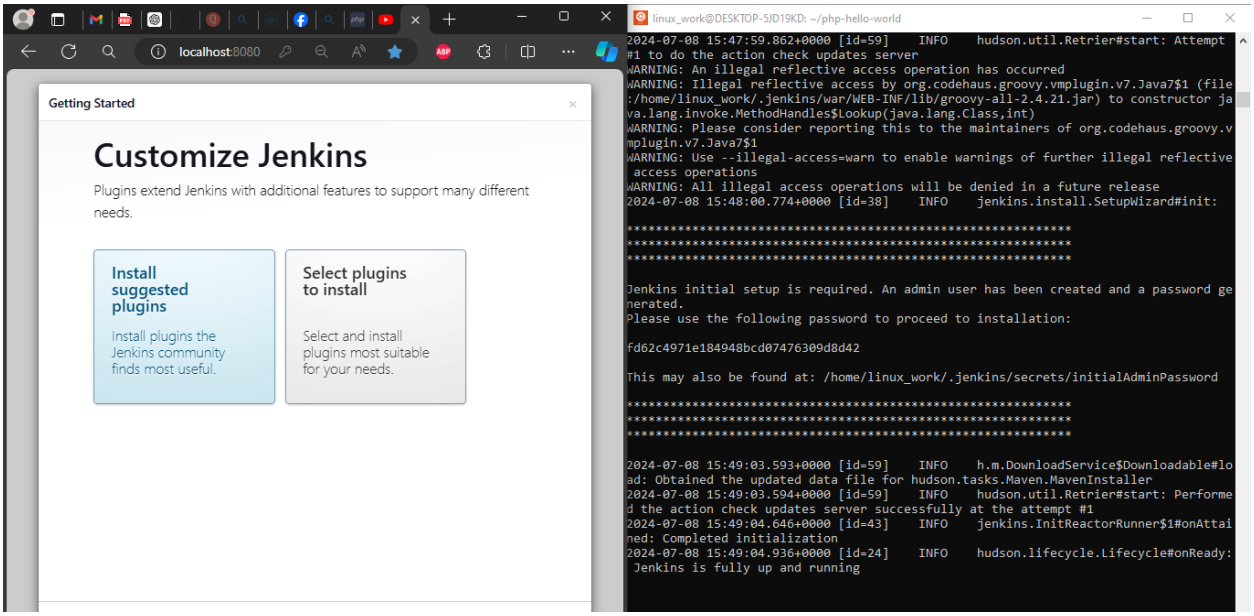
```
linux_work@DESKTOP-5JD19KD:~/php-hello-world/php-hello-world$ docker-compose up -d
[+] Running 1/2
 ⠿ Network php-hello-world_default  Created
 ⠿ Container php-hello-world-web-1  Started
linux_work@DESKTOP-5JD19KD:~/php-hello-world/php-hello-world$ docker ps
CONTAINER ID   IMAGE                     COMMAND                  CREATED          STATUS          PORTS                  NAMES
4bb5faf50cf0   adityaji777/php-project   "docker-php-entrypoi…"   15 seconds ago   Up 11 seconds   0.0.0.0:80->80/tcp   php-hello-world-web-1
linux_work@DESKTOP-5JD19KD:~/php-hello-world/php-hello-world$
```

==========

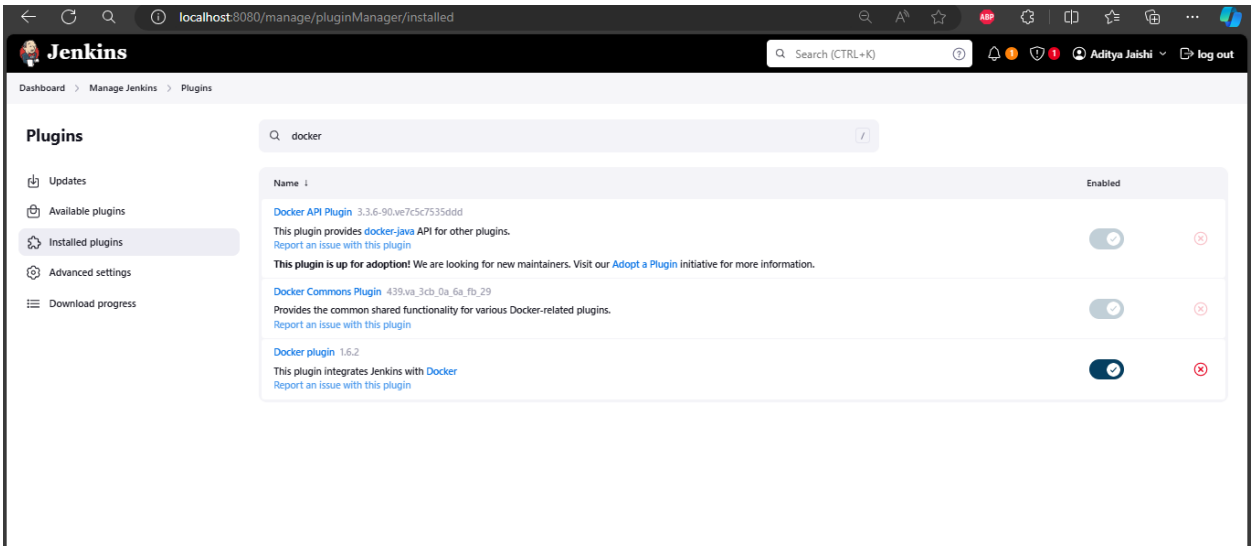
==========

Jenkins Setup

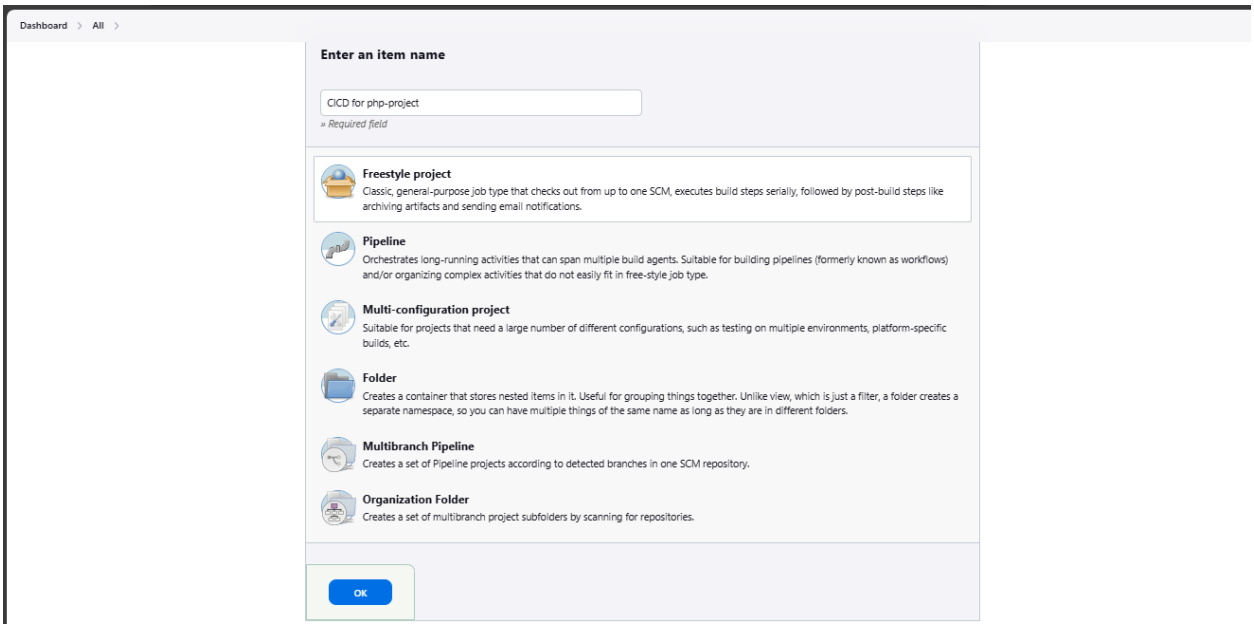
After enabling Jenkins it should start on port 8080:

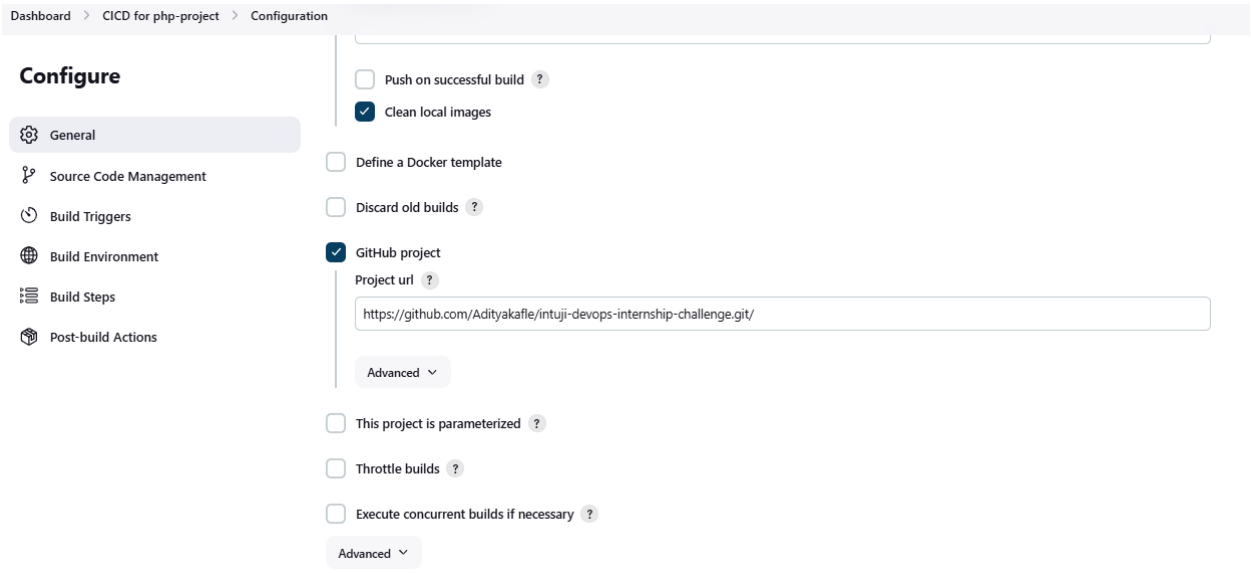


Step1: Installed necessary plugins



Step2: Create freestyle project




Dashboard > CICD for php-project > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Push on successful build
Clean local images
Define a Docker template
Discard old builds
GitHub project
Project url
https://github.com/Adityakafle/intuji-devops-internship-challenge.git/
Advanced
This project is parameterized
Throttle builds
Execute concurrent builds if necessary
Advanced


Git
Repositories
Repository URL
https://github.com/Adityakafle/intuji-devops-internship-challenge.git
Credentials
aditya/******
+ Add
Advanced
Add Repository

Poll SCM ?

Schedule ?

```
* * * * *
```

⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour

Would last have run at Wednesday, July 10, 2024 at 2:18:03 AM Nepal Time; would next run at Wednesday, July 10, 2024 at 2:18:03 AM Nepal Time.

☐ Ignore post-commit hooks ?

Dashboard › CICD for php-project › Configuration

Configure

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

Execute shell ?

Command

See the list of available environment variables

```
docker-compose down
sleep 10
DOCKER_API_VERSION=1.41 docker system prune -a -f
sleep 10
docker build -t adityaji777/php-project .
docker push adityaji777/php-project
docker-compose up -d
```

Dashboard › CICD for php-project › #2 › Console Output

```
e5d6c528cded Extracting [==================================================>]  91.46kB/91.46kB
e5d6c528cded Extracting [==================================================>]  91.46kB/91.46kB
e5d6c528cded Pull complete
355f5a3ba7c9 Extracting [==================================================>]     266B/266B
355f5a3ba7c9 Extracting [==================================================>]     266B/266B
355f5a3ba7c9 Pull complete
199a42ed7c4e Extracting [>                                                  ]  360.4kB/34.57MB
199a42ed7c4e Extracting [====>                                              ]  3.244MB/34.57MB
199a42ed7c4e Extracting [==========>                                        ]  7.569MB/34.57MB
199a42ed7c4e Extracting [=================>                                 ]  12.26MB/34.57MB
199a42ed7c4e Extracting [=========================>                         ]   17.3MB/34.57MB
199a42ed7c4e Extracting [============================>                      ]  18.74MB/34.57MB
199a42ed7c4e Extracting [============================>                      ]   19.1MB/34.57MB
199a42ed7c4e Extracting [======================================>            ]  27.03MB/34.57MB
199a42ed7c4e Extracting [==================================================>  ]  33.52MB/34.57MB
199a42ed7c4e Extracting [==================================================>]  34.57MB/34.57MB
199a42ed7c4e Pull complete
web Pulled
Network cicdforphp-project_default  Creating
Network cicdforphp-project_default  Created
Container cicdforphp-project-web-1  Creating
Container cicdforphp-project-web-1  Created
Container cicdforphp-project-web-1  Starting
Container cicdforphp-project-web-1  Started
Finished: SUCCESS
```

All set now! This setup ensures that whenever code is pushed to GitHub, Jenkins triggers a build. It then removes the old Docker image and creates a new one with the necessary changes, thus enabling Continuous Integration and Continuous Deployment (CI/CD).

THE END

==========
==========