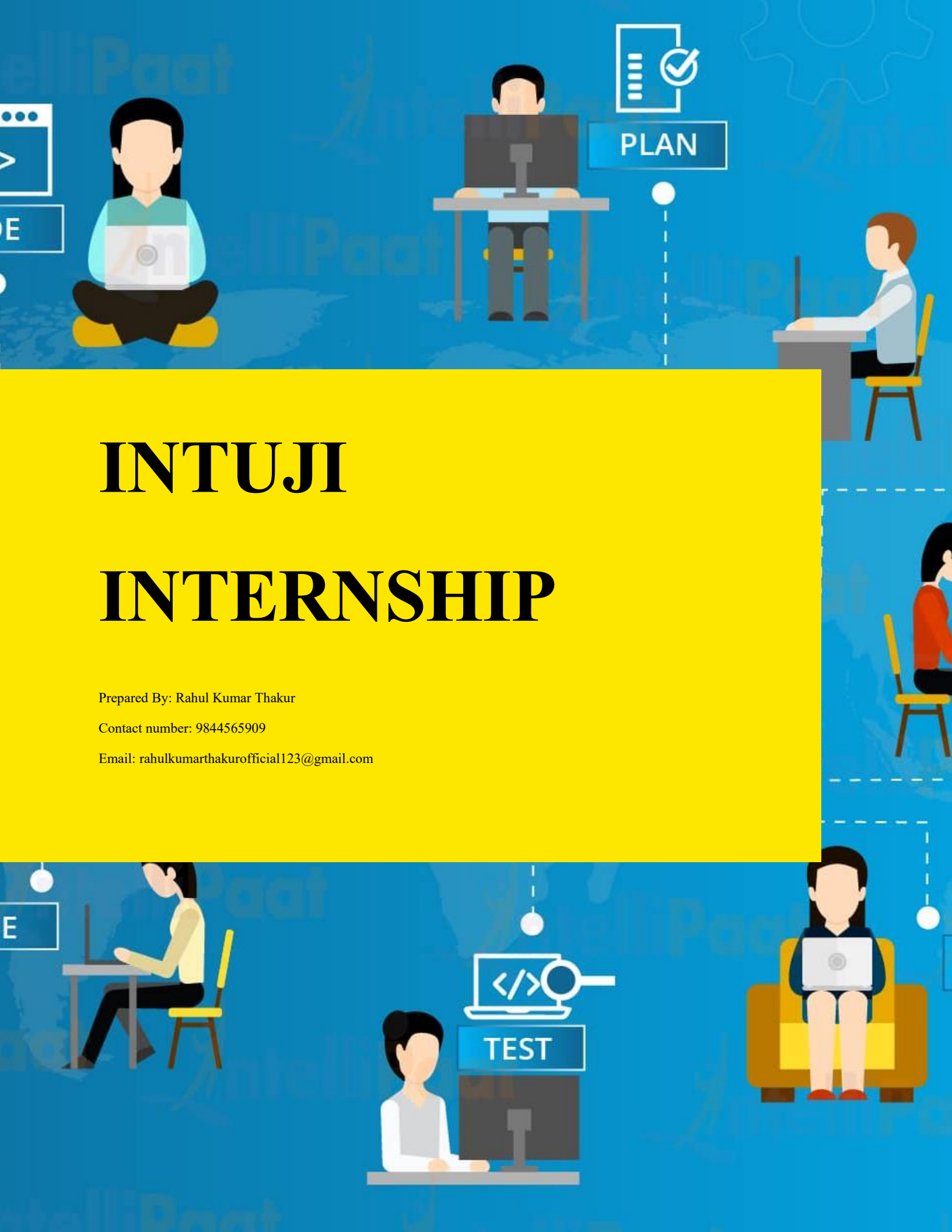



# INTUJI

# INTERNSHIP

Prepared By: Rahul Kumar Thakur

Contact number: 9844565909

Email: rahulkumarthakurofficial123@gmail.com

## Table of Contents

**CI/CD Pipeline Setup with Docker and Jenkins for php app**

To build a complete CI/CD (Continuous Integration/Continuous Deployment) pipeline, we use **Docker** containers for packaging and running applications and **Jenkins** for automating builds and deployments. Docker is an open platform that packages applications and their dependencies into portable containers[1]. This ensures *consistent, isolated environments* across development, testing, and production. As the Docker documentation explains, containers are ideal for CI/CD workflows because developers can share standardized environments and quickly deploy code by simply updating container images[1]. For example, developers write code locally, build a Docker image, test it in containers, and when it's ready, push the updated image to production – all with minimal differences between environments. Docker uses a **client–server** model[2]. The Docker CLI client (docker) sends commands (like build, run, etc.) to the Docker **daemon** (dockerd), which performs the heavy lifting of building, running, and managing containers[2]. Both client and daemon can run on the same machine or on separate machines over a network. Docker also provides **Docker Compose** – a tool for defining and running multi-container applications (mentioned as *another Docker client* in the docs)[2]. In this architecture, a central registry (such as Docker Hub) stores images, and developers push/pull images via the registry.

**Environment Preparation and Virtualization**

Before installing software, prepare a suitable environment. The user's host is **Kali Linux** on a local machine. It's recommended to run container services in a dedicated virtual machine (VM) or VM-like environment for isolation. For example, you could use VirtualBox or KVM on Kali to create a Debian/Ubuntu VM, or even run Kali in a VM. This way, you keep the host system clean and can snapshot or roll back easily. Ensure virtualization (VT-x/AMD-V) is enabled in BIOS. Then, install a lightweight Linux VM (such as Debian 12 or Ubuntu 22.04 LTS) to run Docker and Jenkins.

Within the VM (or on the Kali host if preferred), update the system packages and install any prerequisites. It's a good practice to run:

```
Sudo apt-get update
Sudo apt-get upgrade -y
```

to ensure the system is up-to-date. You may also install Git (e.g., sudo apt-get install -y git) to enable cloning repositories and any other tools you need.

**Installing Docker with a Bash Script**

Docker is installed differently on Debian-based systems. Kali Linux already provides Docker as the docker.io package. A simple Bash script can automate the installation. For example:

```
#!/bin/bash
# Install Docker on Debian/Kali Linux

echo "Updating package index..."
sudo apt-get update

echo "Installing Docker (docker.io package)..."
sudo apt-get install -y docker.io

echo "Starting Docker service..."
sudo systemctl start docker

echo "Enabling Docker to start on boot..."
sudo systemctl enable docker

echo "Adding current user to 'docker' group (no sudo needed for Doc
ker):"
sudo usermod -aG docker $USER

echo "Docker installation complete. Please log out and log back in
to refresh group membership."
```

This script (adapted from Kali Linux documentation) runs `apt-get update`, then installs `docker.io`, and finally starts/enables the Docker service[3]. Adding the user to the `docker` group (`sudo usermod -aG docker $USER`) lets you run `docker` commands without `sudo`. After adding the user to the group, log out and back in for it to take effect.

**Cloning the github repository**

Next, clone the PHP application source code. We use the GitHub repository . In a terminal on the VM, run:

```
git clone https://github.com/silarhi/php-hello-world.git
cd php-hello-world
```

This creates a local copy of the repo. The official GitHub docs describe this process: "Type git clone, and then paste the URL ... For example: git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY"[4]. After cloning, you have the application's files in the php-hello-world directory.

**Creating Dockerfile**

We need to containerize the PHP application so it can run on a web server. We can use one of the official PHP-Apache images (e.g. `php:8.1-apache` or a later version). This base image includes PHP and the Apache HTTP server.

Create a file named `Dockerfile` in the project root (the cloned repo directory). A simple example Dockerfile might be:

```
┌──(rahul㉿rahul)-[~/php-hello-world]
└─$ cat Dockerfile
FROM php:8.2-apache
RUN docker-php-ext-install mysqli
# Enable Apache modules
RUN a2enmod rewrite
# Set working directory
WORKDIR /var/www/html
# Copy all files
COPY . .
# Install Composer
RUN apt-get update && apt-get install -y unzip curl \
 && curl -sS https://getcomposer.org/installer | php \
 && mv composer.phar /usr/local/bin/composer \
 && composer install
EXPOSE 80
```

**Creating** *docker-compose.yml* **File**

Docker Compose allows you to define and run multi-container applications with a simple YAML file. According to Docker docs, Docker Compose "lets you work with applications consisting of a set of containers"[2]. For our PHP app (which currently only needs the web server), the compose file can be simple. In the repository directory, create `docker-compose.yml`:

```
┌──(rahul㉿rahul)-[~/php-hello-world]
└─$ cat docker-compose.yml
version: '3.8'
services:
  app:
    image: mrethicalhackerofnight/php-hello-world-app:latest
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "8081:80"
    container_name: php_hello_app
```

To run the service, use:

```
docker compose up -d
```

This command builds (if necessary) and starts the `web` service in detached mode. The Docker Compose reference confirms that `docker compose up` builds, (re)creates, and starts containers for defined services[8]. After this, the PHP app will be running in a container, and visiting the VM's IP on port 8080 should show the web page. Docker Compose greatly simplifies orchestration of multiple services; while we only have one service now, in the future you could add databases or other services easily in the same `docker-compose.yml`.

**Apendix:**

*Figure1.composer update:*

```
┌──(rahul㉿rahul)-[~/php-hello-world]
└─$ composer update
Loading composer repositories with package information
Updating dependencies
Lock file operations: 5 installs, 23 updates, 7 removals
  - Removing phpdocumentor/reflection-common (2.0.0)
  - Removing phpdocumentor/reflection-docblock (5.0.0)
  - Removing phpdocumentor/type-resolver (1.0.1)
  - Removing phpspec/prophecy (v1.10.2)
  - Removing phpunit/php-token-stream (4.0.0)
  - Removing symfony/polyfill-ctype (v1.14.0)
  - Removing webmozart/assert (1.7.0)
  - Upgrading doctrine/instantiator (1.3.0 => 1.5.0)
  - Upgrading myclabs/deep-copy (1.9.5 => 1.13.3)
  - Locking nikic/php-parser (v5.5.0)
  - Upgrading phar-io/manifest (1.0.3 => 2.0.4)
  - Upgrading phar-io/version (2.0.1 => 3.2.1)
  - Upgrading phpunit/php-code-coverage (8.0.1 => 9.2.32)
  - Upgrading phpunit/php-file-iterator (3.0.0 => 3.0.6)
  - Upgrading phpunit/php-invoker (3.0.0 => 3.1.1)
  - Upgrading phpunit/php-text-template (2.0.0 => 2.0.4)
  - Upgrading phpunit/php-timer (3.0.0 => 5.0.3)
  - Upgrading phpunit/phpunit (9.0.1 => 9.6.23)
  - Locking sebastian/cli-parser (1.0.2)
  - Locking sebastian/code-unit (1.0.8)
  - Upgrading sebastian/code-unit-reverse-lookup (2.0.0 => 2.0.3)
  - Upgrading sebastian/comparator (4.0.0 => 4.0.8)
  - Locking sebastian/complexity (2.0.3)
  - Upgrading sebastian/diff (4.0.0 => 4.0.6)
  - Upgrading sebastian/environment (5.0.1 => 5.1.5)
  - Upgrading sebastian/exporter (4.0.0 => 4.0.6)
  - Upgrading sebastian/global-state (4.0.0 => 5.0.7)
  - Locking sebastian/lines-of-code (1.0.4)
  - Upgrading sebastian/object-enumerator (4.0.0 => 4.0.4)
  - Upgrading sebastian/object-reflector (2.0.0 => 2.0.4)
  - Upgrading sebastian/recursion-context (4.0.0 => 4.0.5)
  - Upgrading sebastian/resource-operations (3.0.0 => 3.0.4)
  - Upgrading sebastian/type (2.0.0 => 3.2.1)
  - Upgrading sebastian/version (3.0.0 => 3.0.2)
  - Upgrading theseer/tokenizer (1.1.3 => 1.2.3)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 28 installs, 0 updates, 0 removals
  - Downloading sebastian/version (3.0.2)
  - Downloading sebastian/type (3.2.1)
```

*Figure.composer installation:*

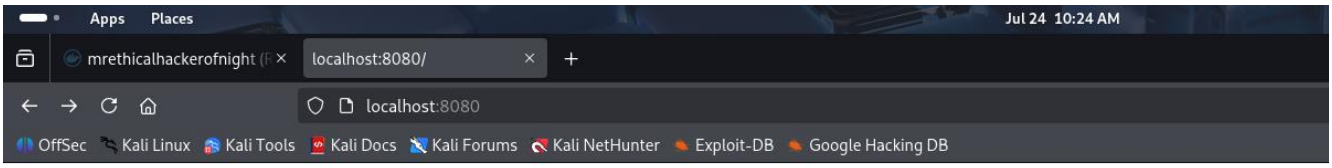
```
┌──(rahul㉿rahul)-[~/php-hello-world]
└─$ composer install
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
Nothing to install, update or remove
Generating autoload files
26 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
```

*Figure.Docker build and testing:*

```
┌──(rahul㉿rahul)-[~/php-hello-world]
└─$ cd ~/php-hello-world
docker-compose build
docker-compose up

WARN[0000] /home/rahul/php-hello-world/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Building 69.4s (12/12) FINISHED                                                                                                                docker:default
 => [app internal] load build definition from Dockerfile                                                                                                        0.0s
 => => transferring dockerfile: 312B                                                                                                                            0.0s
 => [app internal] load metadata for docker.io/library/php:8.2-apache                                                                                           3.4s
 => [app internal] load .dockerignore                                                                                                                           0.0s
 => => transferring context: 2B                                                                                                                                 0.0s
 => [app 1/7] FROM docker.io/library/php:8.2-apache@sha256:6f109f7e92d08ecfd25410e701d258b55d49521385a190b501433625eb489bb5                                     0.0s
 => [app internal] load build context                                                                                                                           0.0s
 => => transferring context: 326B                                                                                                                               0.0s
 => CACHED [app 2/7] RUN a2enmod rewrite                                                                                                                        0.0s
 => CACHED [app 3/7] WORKDIR /var/www/html                                                                                                                      0.0s
 => CACHED [app 4/7] COPY src/ .                                                                                                                                0.0s
 => [app 5/7] COPY index.php ./                                                                                                                                 0.0s
 => [app 6/7] COPY composer.* ./                                                                                                                                0.0s
 => [app 7/7] RUN apt-get update &&     apt-get install -y unzip &&     curl -sS https://getcomposer.org/installer | php &&     php composer.phar install   65.6s
 => [app] exporting to image                                                                                                                                    0.2s
 => => exporting layers                                                                                                                                         0.2s
 => => writing image sha256:ea0750343cadb23fad99df404541c6cb8894546e3163fafd3cf876d5ec020815                                                                    0.0s
 => => naming to docker.io/library/php-hello-world-app                                                                                                          0.0s
WARN[0000] /home/rahul/php-hello-world/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 0/0
 ⠿ Container php_hello_app  Recreated                                                                                                                           0.1s
Attaching to php_hello_app
php_hello_app  | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.18.0.2. Set the 'ServerName' directive globally to suppress this message
php_hello_app  | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.18.0.2. Set the 'ServerName' directive globally to suppress this message
php_hello_app  | [Thu Jul 24 14:22:59.799076 2025] [mpm_prefork:notice] [pid 1:tid 1] AH00163: Apache/2.4.62 (Debian) PHP/8.2.29 configured -- resuming normal operations
php_hello_app  | [Thu Jul 24 14:22:59.799184 2025] [core:notice] [pid 1:tid 1] AH00094: Command line: 'apache2 -D FOREGROUND'


php_hello_app  | 172.18.0.1 - - [24/Jul/2025:14:23:41 +0000] "GET / HTTP/1.1" 200 249 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
```




Hello World vlatest

*Figure.Passowrd location of jenkins:*

```
┌──(rahul㉿rahul)-[~]
└─$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword

a7[redacted]3106
```

*Figure.JDK installation step for jenkins:*

```
┌──(rahul㉿rahul)-[~]
└─$ sudo apt update
sudo apt install openjdk-17-jdk -y
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/nu
ll
sudo apt update
sudo apt install jenkins -y
sudo systemctl start jenkins
sudo systemctl enable jenkins
```

*Figure.docker login for pushing image:*

```
┌──(rahul㉿rahul)-[~]
└─$ docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub.
eate one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-s
SO. Learn more at https://docs.docker.com/go/access-tokens/

Username: mrethicalhackerofnight
Password:
WARNING! Your password will be stored unencrypted in /home/rahul/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

*Figure.Docker image push to docker hub and testing of app:*

```
┌──(rahul㉿rahul)-[~]
└─$ docker pull mrethicalhackerofnight/php-hello-world-app:latest

latest: Pulling from mrethicalhackerofnight/php-hello-world-app
0bdf2c4d1714: Pull complete
e8c70bd5ff59: Pull complete
73e6a42ba33e: Pull complete
c90c3e5b2c04: Pull complete
662dec6bb076: Pull complete
614225424baf: Pull complete
e4b0ce2a63a0: Pull complete
aa244c246889: Pull complete
3ad73792eb2e: Pull complete
5987442f4c60: Pull complete
24679b88893a: Pull complete
344697a4c5f7: Pull complete
121e57d7341e: Pull complete
267bcff90bee: Pull complete
4f4fb700ef54: Pull complete
2ee0c32547bb: Pull complete
ee92dc246881: Pull complete
ab8b9c69f71b: Pull complete
c4cf5c15267b: Pull complete
Digest: sha256:70f600e356d6578093864c2211ac4797a125437f2b953711b7b026b813332dfe
Status: Downloaded newer image for mrethicalhackerofnight/php-hello-world-app:latest
docker.io/mrethicalhackerofnight/php-hello-world-app:latest

┌──(rahul㉿rahul)-[~]
└─$ docker run -d -p 8080:80 mrethicalhackerofnight/php-hello-world-app:latest

e05570ab89af9c0206b704cac79c05ae7296921144e959a35c41b347267b584e

┌──(rahul㉿rahul)-[~]
└─$ firefox http://localhost:8080
```

rahul@rahul: ~

localhost:8080/

localhost:8080

OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali N

Hello World vlatest

*Figure.web interface of user setup of jenkins:*

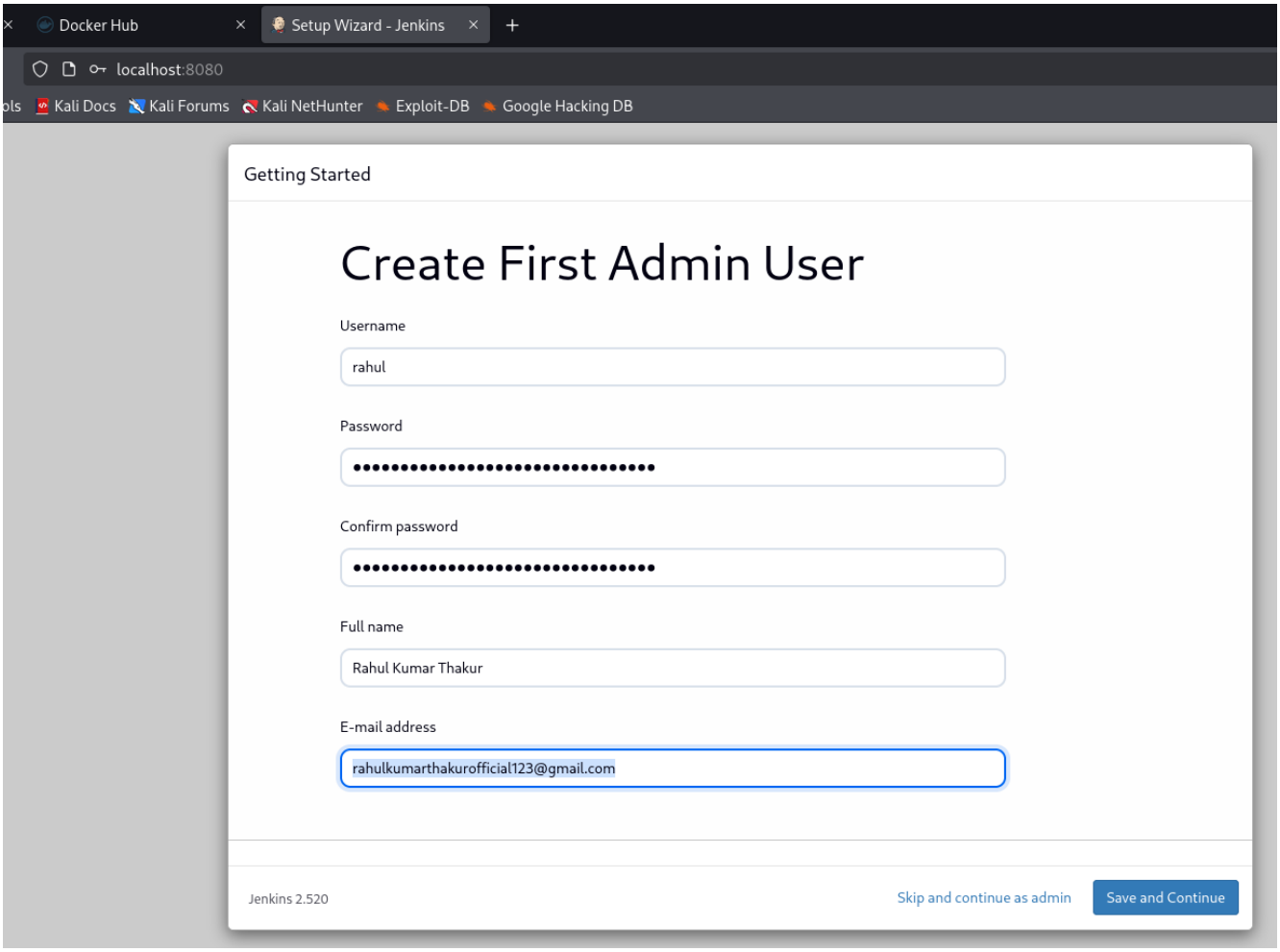


*Figure.Web dashboard of Jenkins automation tool:*