

comment-installer-harbor-docker-image-registry-sur-ubuntu-22-04

Harbor is an open-source Docker image registry for cloud-native environments. As an image registry, the harbor is used to store and distribute container images. For securing artifacts, harbor provides security protection via policies, user role-based access control, a security scanner for images to ensure images are free from vulnerabilities, and image signing that ensures the users pull images from the trusted registry.

Harbor is a CNCF graduate project and an enterprise-grade image registry. Deliver compliance and high performance that help you consistently and securely manage artifacts across cloud-native environments such as Kubernetes and Docker.

In this guide, I will show you how to install Harbor Image Registry using Docker on an Ubuntu 22.04 server. This guide includes the Docker CE installation, basic harbor installation with SSL enabled, and the basic usage and administration of the harbor as an image registry.

Prerequisites

To go over this guide, you will need the following requirements:

- An Ubuntu 22.04 server - This example uses the generic and fresh Ubuntu server with the hostname 'harbor-server'.
- A non-root user with root/administrator privileges.
- A domain name or local domain that will be used by the harbor.

Installing Docker CE (Community Edition)

The harbor image registry provides multiple versions that can be installed on different types of environments. You can deploy harbor to Kubernetes via the Helm chart or install harbor with Docker.

This example you will install harbor via the Docker engine. So you will now install the Docker CE (Community Edition) to your Ubuntu server via the official Docker repository.

Run the apt command below to install some basic dependencies.

```
sudo apt install \
ca-certificates \
curl \
gnupg \
lsb-release
```

Input y when prompted for confirmation. Then press ENTER to proceed.

After dependencies are installed, run the following command to download the GPG key for the Docker repository, and then add the official Docker repository to your system.

```
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Now refresh and update your package index via the apt command below.

```
sudo apt update
```

You will receive the output that the Docker CE repository is added.

Next, use the following apt command to install Docker CE and Docker Compose to your system.

```
sudo apt install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

Input y to confirm the installation, then press ENTER to proceed.

```

root@harbour-server:~#
root@harbour-server:~# sudo apt install docker-ce docker-ce-cli containerd.io docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
containerd.io is already the newest version (1.6.9-1).
docker-ce-cli is already the newest version (5:20.10.21-3-0-ubuntu-jammy).
docker-ce is already the newest version (5:20.10.21-3-0-ubuntu-jammy).
docker-compose-plugin is already the newest version (2.12.2-ubuntu-jammy).
0 upgraded, 0 newly installed, 0 to remove and 38 not upgraded.
root@harbour-server:~#
root@harbour-server:~#

```

After Docker is installed, run the following systemctl command to verify the current status of the Docker service.

```

sudo systemctl is-enabled docker
sudo systemctl status docker

```

You should now receive the output like the following screenshot - the Docker service is running and enabled. The Docker service will be run automatically at system boot.

```

root@harbour-server:~#
root@harbour-server:~# sudo systemctl is-enabled docker
enabled
root@harbour-server:~# sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2022-10-31 06:56:43 CET; 1min 39s ago
 TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
   Main PID: 859 (dockerd)
   Tasks: 25
   Memory: 91.9M

```

After the Docker engine is installed, you will next start the harbor installation by downloading the harbor installer package.

Downloading Harbor Installer

You will need to download the harbor installer package for installing the harbor. And there are two types of harbor installers, the offline version, and the online version.

This guide uses the harbor offline installer for installing the harbor. So you will now download the harbor offline installer to your system.

Move the working directory to '/tmp' and download the harbor offline installer via the curl command below.

```

cd /tmp
curl -s https://api.github.com/repos/goharbor/harbor/releases/latest | grep browser_download_url | cut -d '"' -f 4 | grep '\.tgz$' |
wget -i -

```

Now extract the harbor offline installer via the tar command below. After the installer is extracted, you should see the new directory 'harbor'.

```

tar -xzf harbor-offline-installer-v2.6.1.tgz

```

Move the 'harbor' directory to '/opt', which means your harbor installation will be '/opt/harbor'.

```

sudo mv harbor /opt/

```

```

root@harbour-server:~#
root@harbour-server:~# tar xvf harbor-offline-installer-v*.tgz
harbor/harbor.v2.6.1.tar.gz
harbor/prepare
harbor/LICENSE
harbor/install.sh
harbor/common.sh
harbor/harbor.yml.tpl
root@harbour-server:~#
root@harbour-server:~# ls
harbor  harbor-offline-installer-v2.6.1.tgz  harbor-online-installer-v2.6.1.tgz
root@harbour-server:~#
root@harbour-server:~# mv harbor /opt/

```

Configuring Harbor Installation

After downloaded the harbor offline installer, you will now be setting up the harbor installation via the configuration file 'harbor.yml' that is included in the harbor package.

With the 'harbor.yml' file, you will set up the domain name for the harbor installation, enable secure HTTPS on the harbor, set up an admin password, and lastly set up the database.

Before you start, ensure that you have the domain name pointed to the server IP address and generated SSL certificates that will be used to secure harbor.

Move your working directory to the harbor installation directory `'/opt/harbor'`.

```
cd /opt/harbor
```

Copy the harbor configuration template `'harbor.yml.tpl'` to `'harbor.yml'`. Then, use the following nano editor command to edit the harbor config file `'harbor.yml'`.

```
cp harbor.yml.tpl harbor.yml
sudo nano harbor.yml
```

Change details configurations as below.

```
# The IP address or hostname to access admin UI and registry service.
# DO NOT use localhost or 127.0.0.1, because Harbor needs to be accessed by external clients.
hostname: registry.hwdomain.io
```

```
....
```

```
# https related config
https:
  # https port for harbor, default is 443
  port: 443
  # The path of cert and key files for nginx
  certificate: /etc/letsencrypt/live/registry.hwdomain.io/fullchain.pem
  private_key: /etc/letsencrypt/live/registry.hwdomain.io/privkey.pem
```

```
....
```

```
# The initial password of Harbor admin
# It only works in first time to install harbor
# Remember Change the admin password from UI after launching Harbor.
harbor_admin_password: Harbor_Docker_Ubuntu
```

```
....
```

```
# Harbor DB configuration
database:
  # The password for the root user of Harbor DB. Change this before any production use.
  password: db_pass_harbor
```

Save the file and exit the editor when finished.

Below is the detailed section of the configuration:

- **hostname** - the domain name that will be used to run the harbor image registry. In this example, the harbor will be installed on 'registry.hwdomain.io'.
- **https** - uncomment this option will enable and secure your harbor installation. But also you need to ensure that you have generated SSL certificates for your domain and input the full path of the certificate public and private key.
- **harbor_admin_password** - the initial password that will be used during the harbor installation.
- **database** - database configuration for the harbor. So ensure to change the password with a new strong password.

Installing Harbor via Installer Script and Docker Compose

Now that the harbor offline installer is downloaded and the base configuration file for 'harbor.yml' is configured, you will now be ready to start the harbor installation via the installer script 'install.sh' that available on the harbor installation directory `'/opt/harbor'`.

Before starting the harbor installation, ensure that your current working directory is `'/opt/harbor'`.

Now run the harbor installer script `'install.sh'` with sudo privileges.

```
sudo ./install.sh
```

The installer script will now check the system requirements before starting the harbor installation. The installer will

ensure that the Docker engine and Docker Compose are installed on the system.

```
root@harbour-server:~#
root@harbour-server:~# cd /opt/harbor/
root@harbour-server:/opt/harbor#
root@harbour-server:/opt/harbor# ls
common.sh harbor.v2.6.1.tar.gz harbor.yml harbor.yml.tpl install.sh LICENSE prepare
root@harbour-server:/opt/harbor#
root@harbour-server:/opt/harbor# sudo nano harbor.yml
root@harbour-server:/opt/harbor#
root@harbour-server:/opt/harbor# sudo ./install.sh

[Step 0]: checking if docker is installed ...

Note: docker version: 20.10.21

[Step 1]: checking docker-compose is installed ...

Note: Docker Compose version v2.12.2

[Step 2]: loading Harbor images ...
```

Now the installer will extract images that will be used to deploy harbor.

```
[Step 2]: loading Harbor images ...
19b3e561bd53: Loading layer [=====] 37.69MB/37.69MB
b1c55ad746b8: Loading layer [=====] 5.754MB/5.754MB
3fad059e5b96: Loading layer [=====] 8.718MB/8.718MB
ac3d56834181: Loading layer [=====] 15.88MB/15.88MB
ac64291e7095: Loading layer [=====] 29.29MB/29.29MB
347c69d047c1: Loading layer [=====] 22.02kB/22.02kB
2bc68bdd74b4: Loading layer [=====] 15.88MB/15.88MB
Loaded image: goharbor/notary-server-photon:v2.6.1
a3f881ff8a8a: Loading layer [=====] 5.759MB/5.759MB
bf4fe2665116: Loading layer [=====] 90.88MB/90.88MB
1bbf13d3b736: Loading layer [=====] 3.072kB/3.072kB
6864945044da: Loading layer [=====] 4.096kB/4.096kB
e74206fce300: Loading layer [=====] 91.67MB/91.67MB
Loaded image: goharbor/chartmuseum-photon:v2.6.1
d1cca5e33760: Loading layer [=====] 126.9MB/126.9MB
f21ade3affb4: Loading layer [=====] 3.584kB/3.584kB
2b10bb22d396: Loading layer [=====] 3.072kB/3.072kB
cddb26029f4f: Loading layer [=====] 2.56kB/2.56kB
120e581fca06: Loading layer [=====] 3.072kB/3.072kB
b55ab4161be8: Loading layer [=====] 3.584kB/3.584kB
708b88dc9728: Loading layer [=====] 20.99kB/20.99kB
Loaded image: goharbor/harbor-log:v2.6.1
aa3c0eeab3fd: Loading layer [=====] 5.759MB/5.759MB
08acd59679e5: Loading layer [=====] 4.096kB/4.096kB
dbfa72b62e7c: Loading layer [=====] 17.1MB/17.1MB
3db46c922bff: Loading layer [=====] 3.072kB/3.072kB
db46f9ab20a1: Loading layer [=====] 29.15MB/29.15MB
c28b264c5c77: Loading layer [=====] 47.04MB/47.04MB
Loaded image: goharbor/harbor-registryctl:v2.6.1
46e1d8c22785: Loading layer [=====] 119.1MB/119.1MB
Loaded image: goharbor/nginx-photon:v2.6.1
ebe1f7ed9475: Loading layer [=====] 7.162MB/7.162MB
780db4ad3bef: Loading layer [=====] 4.096kB/4.096kB
dc07146a4e90: Loading layer [=====] 3.072kB/3.072kB
7cde8f5bc2a6: Loading layer [=====] 01.21MB/01.21MB
```

After that, the installer will be preparing the system environment for the harbor deployment and generate the necessary configurations. Then, the installation will begin.

```
[Step 3]: preparing environment ...
```

```
[Step 4]: preparing harbor configs ...
```

```
prepare base dir is set to /opt/harbor
```

```
Generated configuration file: /config/portal/nginx.conf
```

```
Generated configuration file: /config/log/logrotate.conf
```

```
Generated configuration file: /config/log/rsyslog_docker.conf
```

```
Generated configuration file: /config/nginx/nginx.conf
```

```
Generated configuration file: /config/core/env
```

```
Generated configuration file: /config/core/app.conf
```

```
Generated configuration file: /config/registry/config.yml
```

```
Generated configuration file: /config/registryctl/env
```

```
Generated configuration file: /config/registryctl/config.yml
```

```
Generated configuration file: /config/db/env
```

```
Generated configuration file: /config/jobservice/env
```

```
Generated configuration file: /config/jobservice/config.yml
```

```
Generated and saved secret to file: /data/secret/keys/secretkey
```

```
Successfully called func: create_root_cert
```

```
Generated configuration file: /compose_location/docker-compose.yml
```

```
Clean up the input dir
```

```
Note: stopping existing Harbor instance ...
```

When installation finished, you should see the output of harbor container services is running.

```
[Step 5]: starting harbor ...
1) Running 2/22
2 Network harbor_harbor Created 0.1s
3 Container harbor_log Started 0.0s
4 Container redis Started 1.0s
5 Container harbor_db Started 1.0s
6 Container harbor_portal Started 1.0s
7 Container registryctl Started 2.0s
8 Container registry Started 1.0s
9 Container harbor_core Started 2.4s
10 Container nginx Started 3.0s
11 Container harbor_jobservice Started 3.0s
-----Harbor has been installed and started successfully.-----
root@harbour-server:/opt/harbor#
```

Now that you have installed harbor via the installer script, you will next verify harbor by checking the container services on your system and accessing the harbor image registry via the web browser.

Run the following 'docker compose' command to verify that the harbor container services are running.

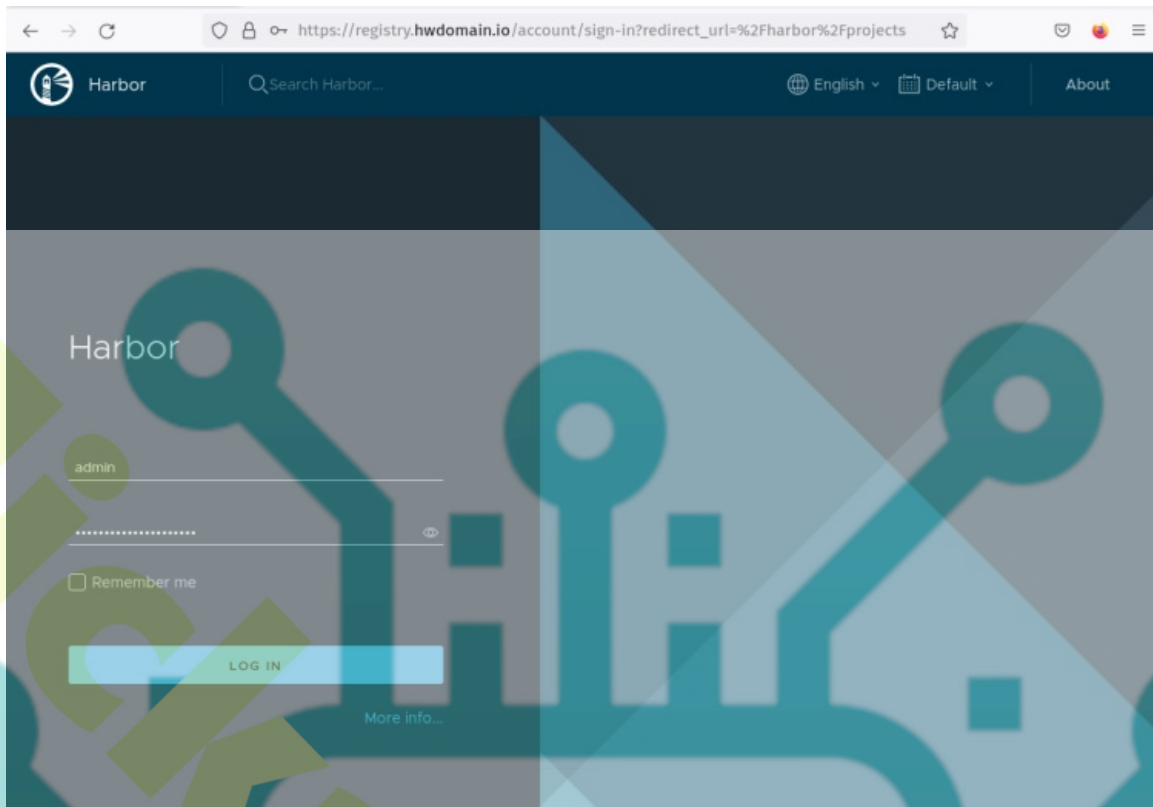
```
ls
docker compose ps
```

In the below output - the harbor container services are running and healthy.

```
root@harbour-server:/opt/harbor#
root@harbour-server:/opt/harbor# ls
common common.sh docker-compose.yml harbor.v2.6.1.tar.gz harbor.yml harbor.yml.tpl install.sh LICENSE prepare
root@harbour-server:/opt/harbor#
root@harbour-server:/opt/harbor#
root@harbour-server:/opt/harbor# docker compose ps
NAME                COMMAND                                SERVICE    STATUS    PORTS
harbor-core         "/harbor/entrypoint..."             core      running (healthy)
harbor-db           "/docker-entrypoint..."             postgresql running (healthy)
harbor-jobservice   "/harbor/entrypoint..."             jobservice running (healthy)
harbor-log          "/bin/sh -c /usr/loc..."            log       running (healthy)    127.0.0.1:1514->16514/tcp
harbor-portal       "nginx -g 'daemon of..."            portal    running (healthy)
nginx               "nginx -g 'daemon of..."            proxy     running (healthy)    0.0.0.0:80->8080/tcp, 0.0.0.0:443->8443/tcp
redis               "redis-server /etc/r..."             redis     running (healthy)
registry            "/home/harbor/entryp..."            registry  running (healthy)
registryctl         "/home/harbor/start..."             registryctl running (healthy)
root@harbour-server:/opt/harbor#
```

Next, open your web browser and visit the domain name for your harbor installation (i.e: <https://registry.hwdomain.io/>). And you should now get the login page of your harbor image registry.

Login with the default user admin and the password that you have configured via the configuration file 'harbor.yml'.



You should now get the harbor dashboard when you have the correct password. Also, you can see on your screen the default project in the harbor with the name 'library'.



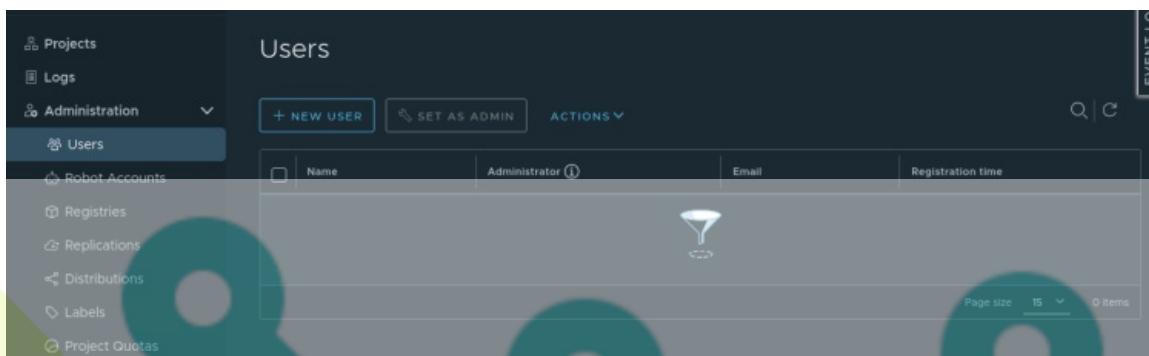
At this point, you have finished the baro deployment and verified that the harbor is running via the container orchestration Docker engine and Docker Compose.

In the next step, you will learn the basic administration of the harbor image registry. You will learn how to add a user, set up the project, log in via the docker CLI, and lastly upload images to the harbor.

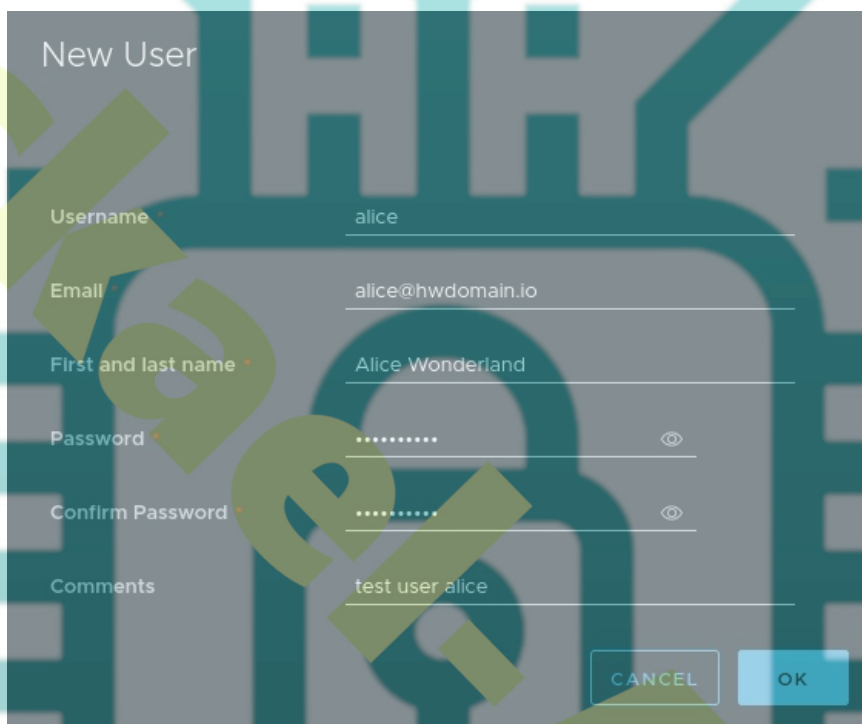
Creating Harbor User

After the harbor image registry is running, you should now learn how to set up and create a new user via the harbor administration dashboard.

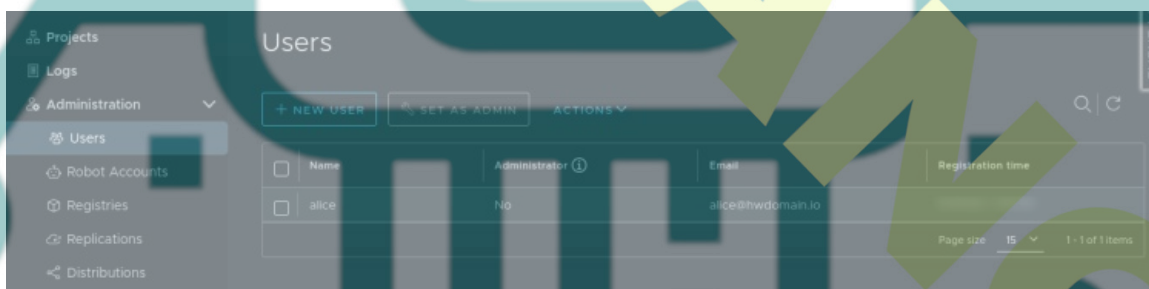
In the '**Administration**' section, click the menu '**Users**', then click the '**NEW USER**' button.



Input details about the new user and confirm by pressing the '**OK**' button. In this example, you will create a new harbor user '**alice**' that will be used to log in to the harbor via the docker cli.

A screenshot of the 'New User' form in the Harbor Administration interface. The form has the following fields: 'Username' with the value 'alice', 'Email' with the value 'alice@hwdomain.io', 'First and last name' with the value 'Alice Wonderland', 'Password' and 'Confirm Password' fields with masked characters (dots), and 'Comments' with the value 'test user alice'. At the bottom right, there are 'CANCEL' and 'OK' buttons.

Once the user is created, you should see your user on the 'User's menu list.



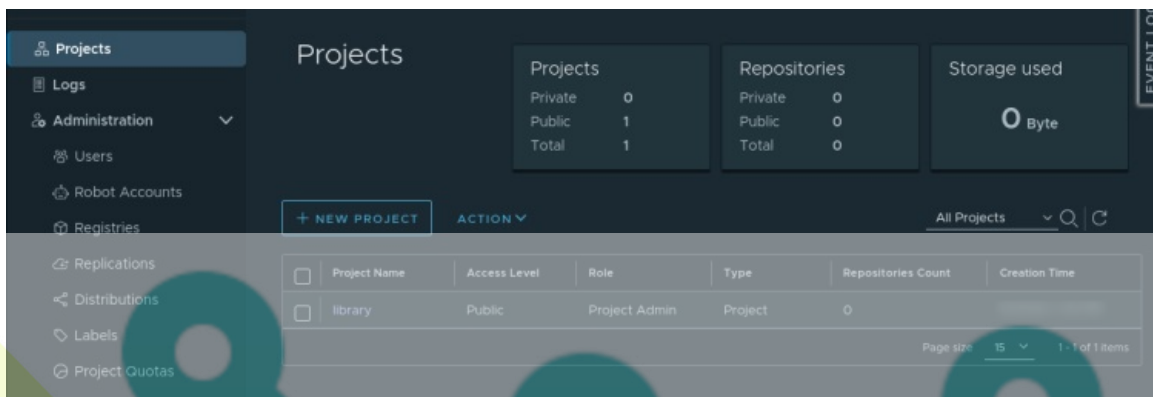
Adding User to Harbor Project

In harbor, the user space that the user will be working on is the project. On the default installation, harbor provides the default project name '**library**'.

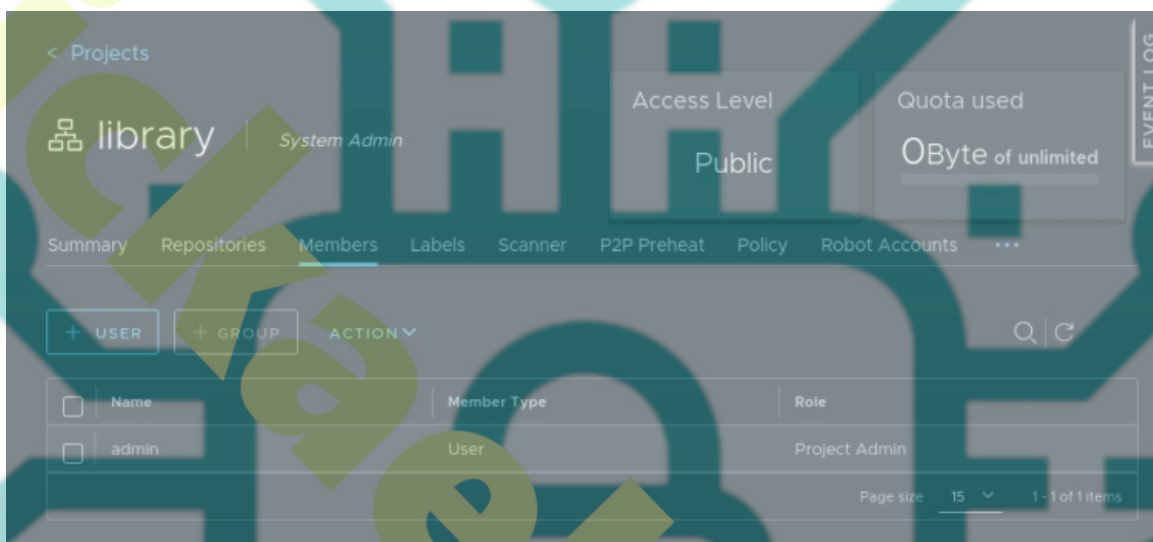
In this step, you will learn how to add the new user '**alice**' to the default project '**library**' so that the new user will be able to manage the '**library**' project. Also, you will need to assign the role for the new user on the target project.

Click on the '**Project**'; menu and you should see the default project '**library**' available on your harbor.

Click the project '**library**' to get the details settings of that project.

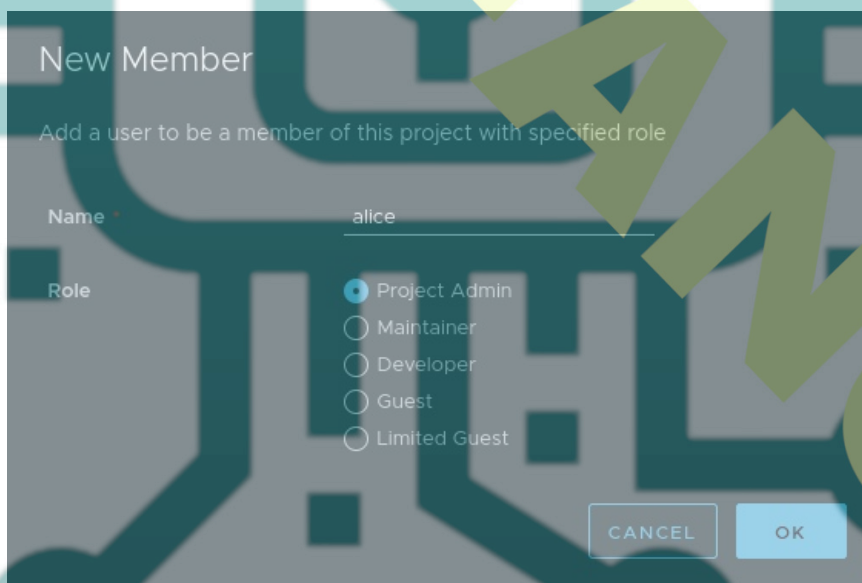


Click the menu 'Members' and you should see there is only one member admin on that project. Click the 'USER' button to add the harbor user to the project 'library'.

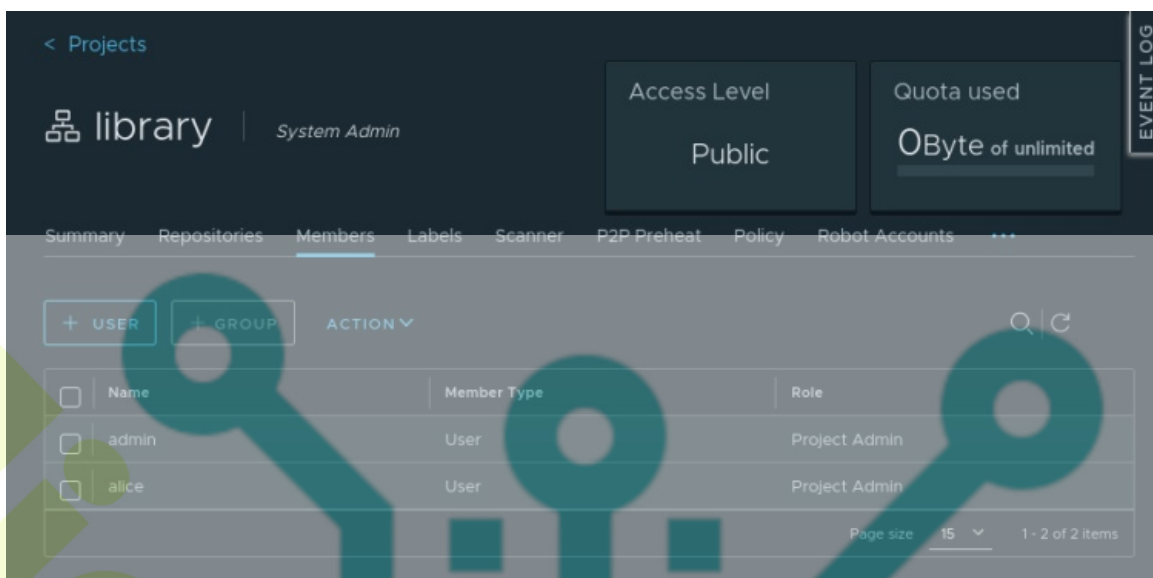


Input the user that you want to add to the project and assign the role to that user. This example will add the user 'alice' to the project 'library' with the role 'Project Admin'.

Click 'OK' to confirm.



You should now see the user 'alice' is available on the member list of the 'library' project.



Logging in to Harbor via Docker Client

In this step, you will learn how to set up the harbor image registry with the Docker CLI. You will be logging in to the harbor image registry via the Docker CLI with the user that you have created - this example uses the user 'alice'.

Back to your terminal server and run the following docker command to log in to the harbor image registry.

```
docker login https://registry.hwdomain.io/
```

When prompted, input the username and password for your user. This example will be using the user 'alice'.

When login success, you should see a message such as 'Login Succeeded'.

```
root@harbor-server:~#
root@harbor-server:~# docker login https://registry.hwdomain.io
Username: alice
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@harbor-server:~#
```

After logging in to harbor via the docker cli, you can now pull and push images to the harbor via the docker cli.

Pushing Images to Harbor Registry

After being authenticated to the harbor image registry, you will now learn how to push/upload an image to your harbor image registry via the Docker CLI.

Before you start, run the following docker command to download the latest version of image 'nginx:alpine'.

```
docker pull nginx:alpine
```

Now verify the list of images via the docker command below. You should see the image 'nginx:alpine' downloaded.

```
docker images
```

```
root@harbor-server:~#
root@harbor-server:~# docker images
REPOSITORY          TAG          IMAGE ID        CREATED         SIZE
nginx                alpine      b997307a58ab   11 days ago    23.6MB
goharbor/harbor-exporter  v2.6.1     ee5cb7b491d2   3 weeks ago    94.4MB
goharbor/chartmuseum-photon v2.6.1     1ae20308ca7a   3 weeks ago    225MB
goharbor/redis-photon   v2.6.1     934d02f2e8d4   3 weeks ago    154MB
goharbor/trivy-adapter-photon v2.6.1     efdae5e79d17   3 weeks ago    253MB
goharbor/notary-server-photon v2.6.1     d38566e39b93   3 weeks ago    112MB
goharbor/notary-signer-photon v2.6.1     a5d9988521e2   3 weeks ago    109MB
goharbor/harbor-registryctl v2.6.1     5cbe7d1a442c   3 weeks ago    136MB
goharbor/registry-photon v2.6.1     aa52e06997ab   3 weeks ago    77.5MB
goharbor/nginx-photon   v2.6.1     a4c36adb555e   3 weeks ago    153MB
goharbor/harbor-log     v2.6.1     d3fd13498bdc   3 weeks ago    161MB
goharbor/harbor-jobservice v2.6.1     98f077f5b101   3 weeks ago    241MB
goharbor/harbor-core    v2.6.1     a34774b55989   3 weeks ago    207MB
goharbor/harbor-portal  v2.6.1     9bbdf6f14337   3 weeks ago    162MB
goharbor/harbor-db      v2.6.1     3b95e61dedfe   3 weeks ago    225MB
goharbor/prepare        v2.6.1     6f69bcbe07f1   3 weeks ago    164MB
postgres                14-alpine  aac01494762a   3 weeks ago    216MB
clickhouse/clickhouse-server 22.6-alpine 2d52aa63cc1f   3 weeks ago    804MB
plausible/analytics      latest     f2bfe5a49b44   9 months ago   274MB
bytemark/smtplib         latest     cd5c77c3bcd8   4 years ago    130MB
root@harbor-server:~#
root@harbor-server:~#
```

To push the custom images to the harbor image registry, you can change the tag of the current image with the format such as **'harbor-domain.com/project/image:version'**.

So, run the following command to change the default tag **'nginx:alpine'** with the **'registry.hwdomain.io/library/nginx:alpine'**.

```
docker tag nginx:alpine registry.hwdomain.io/library/nginx:alpine
```

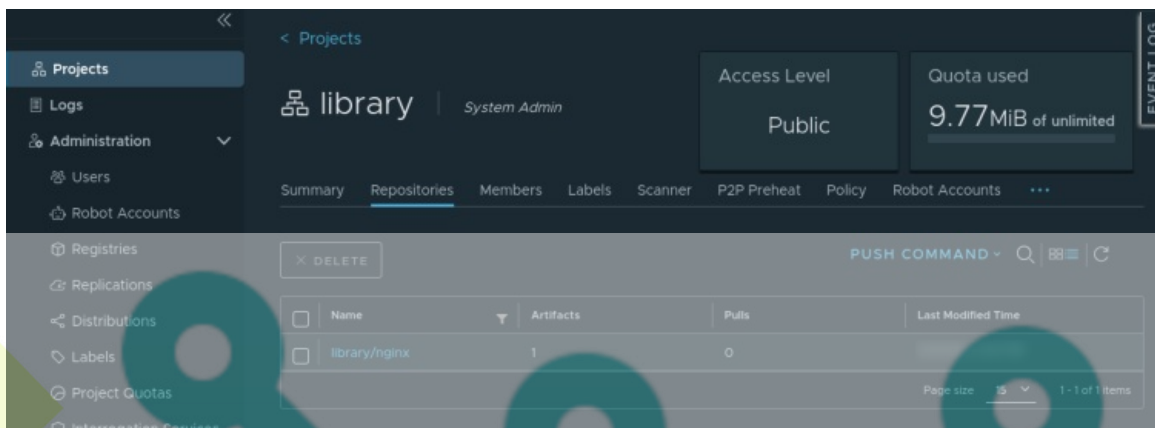
After that, upload and push the image to the harbor image registry with the following command. This command will upload the image to the image registry **'registry.hwdomain.io'** with the project name **'library'**.

```
docker push registry.hwdomain.io/library/nginx:alpine
```

Below is the output during the process of pushing the image to the harbor image registry.

```
root@harbor-server:~#
root@harbor-server:~# docker tag nginx:alpine registry.hwdomain.io/library/nginx:alpine
root@harbor-server:~#
root@harbor-server:~# docker push registry.hwdomain.io/library/nginx:alpine
The push refers to repository [registry.hwdomain.io/library/nginx]
0618d1e529fa: Pushed
6e96dd581d79: Pushed
acf5e0b2cf08: Pushed
d51445d70778: Pushed
b96b16a53835: Pushed
994393dc58e7: Pushed
alpine: digest: sha256:fcba10206c0e29bc2c6c5ede2d64817c113de5bfaecf908b3b7b158a89144162 size: 1568
root@harbor-server:~#
root@harbor-server:~#
```

Once the image is uploaded to the harbor, back to the harbor administration dashboard and click the **'Project'** menu and click **'Repositories'**. You should see the new image is uploaded to the harbor image registry via the docker command line.



Conclusion

You have installed the harbor image registry via Docker on an Ubuntu 22.04 server. You also have secured the harbor deployment with SSL certificates, so you can now access the harbor via an HTTPS secure connection.

For the harbor basic administration, you have learned how to set up users and assign a role in harbor. You also have learned the basic administration for managing projects in the harbor. And lastly, you have successfully added harbor to Docker as an image registry and uploaded an image to the harbor.

From here, you can begin adding a new user and set up a new project that will be used for your development team.
