

1/24/2023

Mahyar Nazari

Dotin

Harbor

In this document we describe how to push and pull a docker image or helm chart to/from an open-source private registry.



Harbor is an open-source registry that secures artifacts with policies and role-based access control, ensures images are scanned and free from vulnerabilities, and signs images as trusted. Harbor, a CNCF Graduated project, delivers compliance, performance, and interoperability to help you consistently and securely manage artifacts across cloud native compute platforms like Kubernetes and Docker.

## Create Projects in Harbor

A project in Harbor contains all repositories of an application. Images cannot be pushed to Harbor before a project is created. Role-Based Access Control (RBAC) is applied to projects, so that only users with the appropriate roles can perform certain operations.

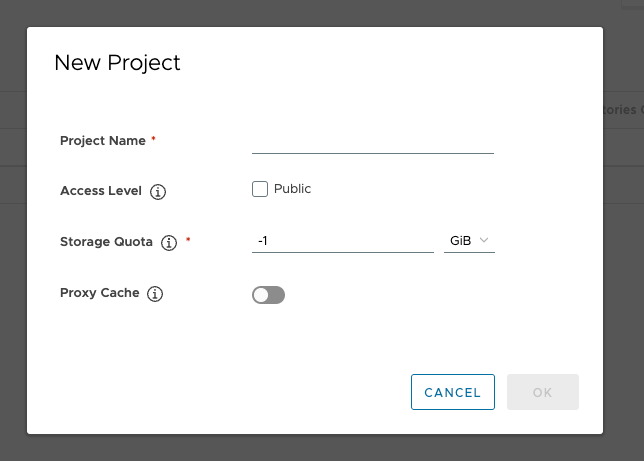
There are two types of project in Harbor:

* Public: Any user can pull images from this project. This is a convenient way for you to share repositories with others.
* Private: Only users who are members of the project can pull images

### **Procedure**

1. Go to **Projects** and click **New Project**.
2. Provide a name for the project.
3. (Optional) Check the **Public** check box to make the project public.

If you set the project to **Public**, any user can pull images from this project. If you leave the project set to **Private**, only users who are members of the project can pull images. You can toggle projects from public to private, or the reverse, at any moment after you create the project.



1. Click **OK**.

# **Docker Images**

## Pushing Images

Before you can push an image to Harbor, you must create a corresponding project in the Harbor First, log in from Docker client:

docker login <harbor\_address>

Tag the image:

docker tag ubuntu:14.04 <harbor\_address>/<harbor\_project>/ubuntu:14.04

Push the image:

docker push <harbor\_address>/<harbor\_project>/ubuntu:14.04

## Pulling Images

If the project that the image belongs to is private, you must sign in first:

docker login <harbor\_address>

You can now pull an image:

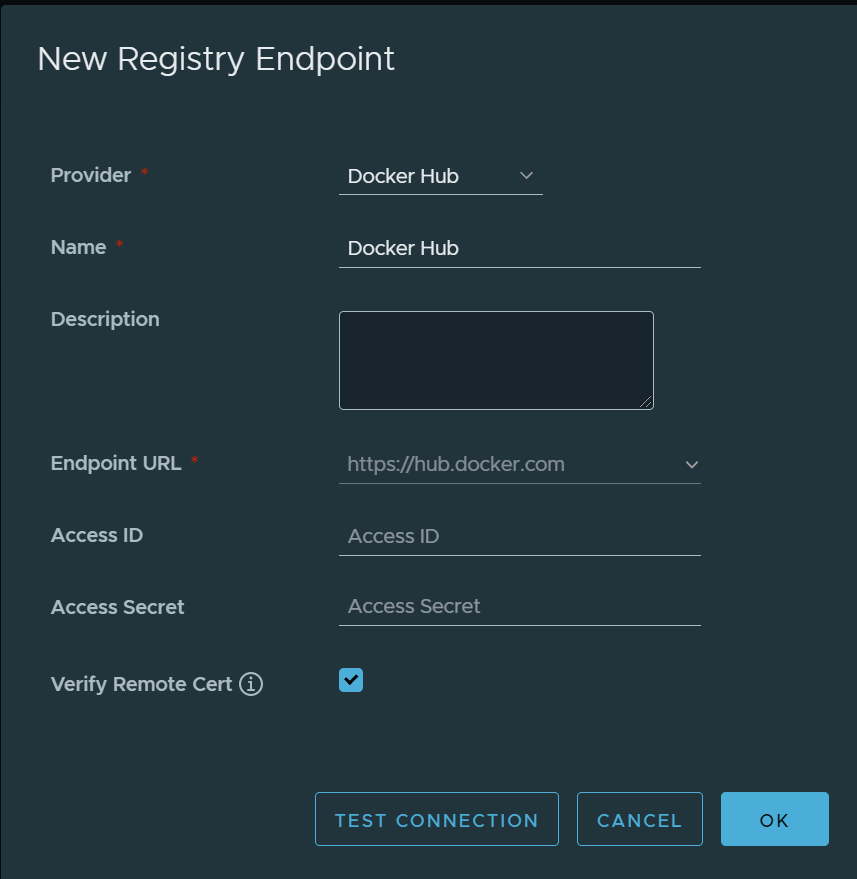
docker pull <harbor\_address>/<harbor\_project>/ubuntu:14.04

## Docker Hub Proxy:

Proxy lets you keep a local copy of images on an as-requested basis. Using proxying can be really useful to cut down on internet traffic as you pull images.

### **Set up a Registry Endpoint**

1. Go to Administration -> Registries and click the + New Endpoint button.
2. Set the Provider and Name both to Docker Hub.
3. You can leave the rest of the settings as default, unless you want access to private images, in which case add in your Access ID and Access Secret.

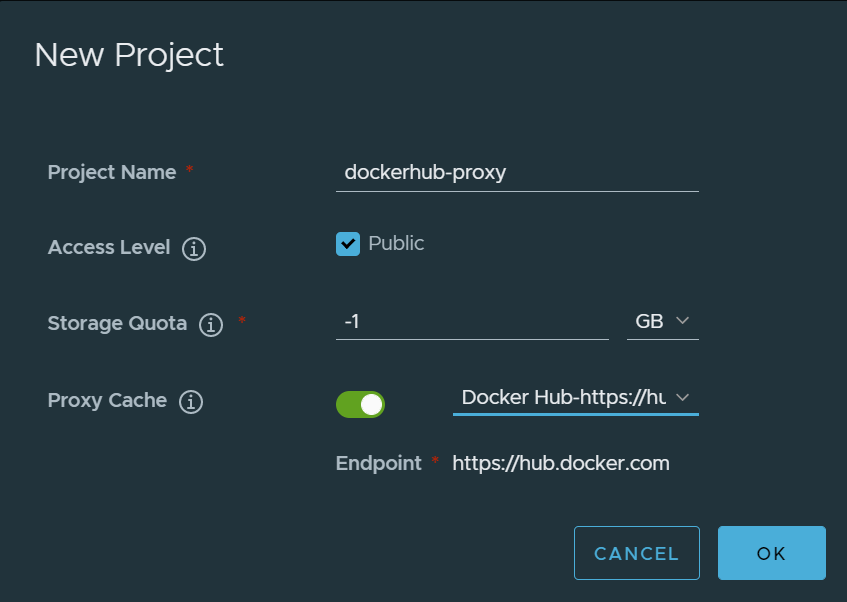


1. Press the Test Connection button and after a successful test, hit OK to save.

### **Create a Docker hub Proxy**

For more information about how Proxy Projects work, see the [official documentation](https://goharbor.io/docs/2.1.0/administration/configure-proxy-cache/).

1. Go to **Projects** and click the **+ New Project** button.
2. Set **Project Name** to dockerhub-proxy.
3. Set **Access Level** to Public (unless you intend to make it private and require login).
4. Leave **Storage Quota** at the default -1 GB.
5. Set **Proxy Cache** to Docker Hub (the Endpoint we created earlier).



### **Test the proxy is working with docker pull:**

$ docker pull < harbor\_address >/dockerhub-proxy/library/ubuntu:20.04

$ docker pull < harbor\_address >/dockerhub-proxy/soffid/linotp:latest

# **Helm Charts**

## Pushing Helm Charts

First package a chart:

helm package <helm\_chart\_directory>

Next, login to Harbor from Master node:

helm registry login <harbor\_address>

You can now push the helm chart package:

helm push <chart\_packagefile.tgz> oci://<harbor\_address>/<harbor\_project>

## Pulling Helm Charts

If the project that the helm chart belongs to is private, you must sign in first:

helm package <helm\_chart\_directory>

Now you can pull the helm chart into your directory:

helm pull oci://<harbor\_address>/<harbor\_project>/<helm\_chart\_name> --version <chart\_version>

# **Kubernetes**

## Pull an Image from a Private Registry

This part shows how to create a Pod that uses a [Secret](https://kubernetes.io/docs/concepts/configuration/secret/) to pull an image from Harbor.

### **Create a Secret by providing credentials on the command line**

Create this Secret, naming it harborcred:

kubectl create secret docker-registry harborcred --docker-server=<harbor\_address> --docker-username=<harbor\_username> --docker-password=<harbor\_password> --docker-email=<your-email>

You have successfully set your credentials in the cluster as a Secret called **harborcred**.

**Note:** Typing secrets on the command line may store them in your shell history unprotected, and those secrets might also be visible to other users on your PC during the time that kubectl is running.

### **Inspecting the Secret** *harborcred*

To understand the contents of the *harborcred* Secret you created, start by viewing the Secret in YAML format:

kubectl get secret harborcred --output=yaml

The output is similar to this:

apiVersion: v1

kind: Secret

metadata:

...

name: regcred

...

data:

.dockerconfigjson: eyJodHRwczovL2luZGV4L ... J0QUl6RTIifX0=

type: kubernetes.io/dockerconfigjson

The value of the .dockerconfigjson field is a base64 representation of your Harbor credentials.

To understand what is in the .dockerconfigjson field, convert the secret data to a readable format:

kubectl get secret harborcred --output="jsonpath={.data.\.dockerconfigjson}" | base64 --decode

The output is similar to this:

{"auths":{"harbor\_address":{"username":"janedoe","password":"xxxxxxxxxxx","email":"jdoe@example.com","auth":"c3R...zE2"}}}

To understand what is in the auth field, convert the base64-encoded data to a readable format:

echo "c3R...zE2" | base64 --decode

The output, username and password concatenated with a **:** , is similar to this:

janedoe:xxxxxxxxxxx

Notice that the Secret data contains the authorization token similar to your local ~/.docker/config.json file.

You have successfully set your Harbor credentials as a Secret called *harborcred* in the cluster

### **Create a Pod that uses your Secret**

Here is a manifest for an example Pod that needs access to your Docker credentials in *harbercred*:

apiVersion: v1

kind: Pod

metadata:

name: database

spec:

containers:

- name: mysql

image: <harbor\_address>/<harbor\_project>/mysql:5.6

imagePullSecrets:

- name: harborcred

Written by: Mahyar Nazari

Member of R&D team

IT Department of Dotin

Jan. 2023