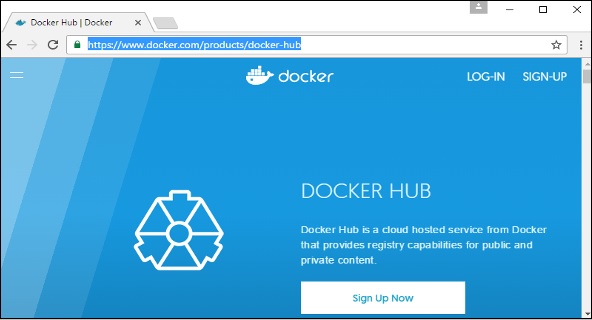
Managing Ports

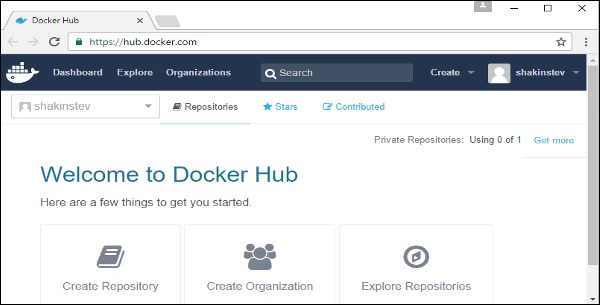
In Docker, the containers themselves can have applications running on ports. When you run a container, if you want to access the application in the container via a port number, you need to map the port number of the container to the port number of the Docker host. Let’s look at an example of how this can be achieved.

In our example, we are going to download the Jenkins container from Docker Hub. We are then going to map the Jenkins port number to the port number on the Docker host.

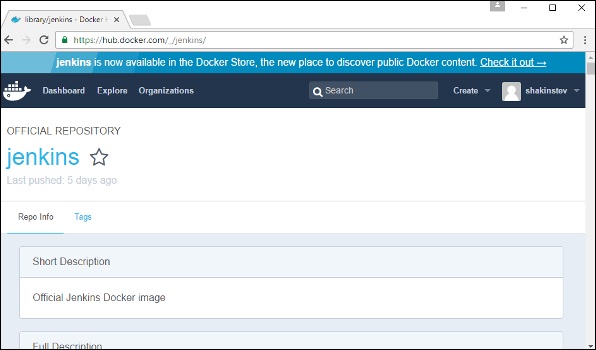
**Step 1** − First, you need to do a simple sign-up on Docker Hub.



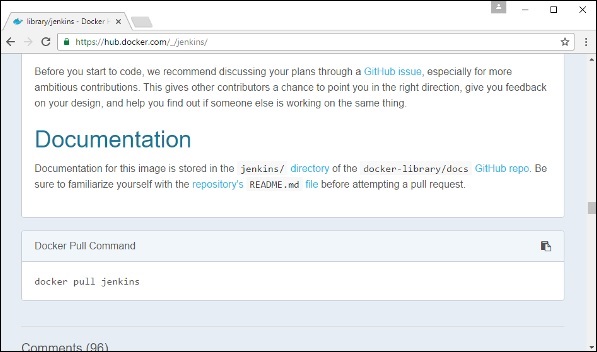
**Step 2** − Once you have signed up, you will be logged into Docker Hub.



**Step 3** − Next, let’s browse and find the Jenkins image.

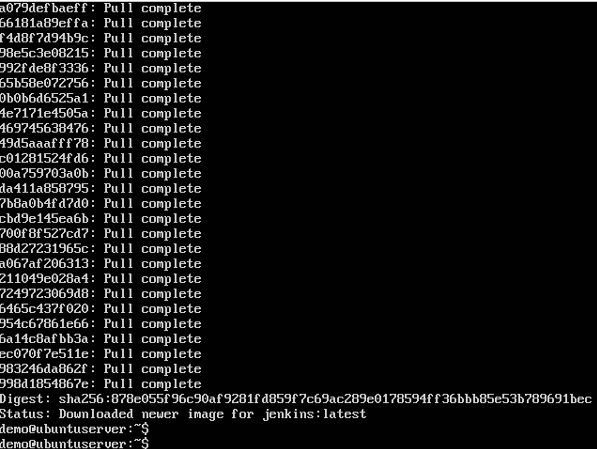


**Step 4** − If you scroll down on the same page, you can see the Docker **pull**command. This will be used to download the Jenkins Image onto the local Ubuntu server.



**Step 5** − Now go to the Ubuntu server and run the command −

sudo docker pull jenkins



**Step 6** − To understand what ports are exposed by the container, you should use the Docker **inspect command** to inspect the image.

Let’s now learn more about this **inspect** command.

## docker inspect

This method allows one to return low-level information on the container or image.

### Syntax

docker inspect Container/Image

### Options

* **Container/Image** − The container or image to inspect

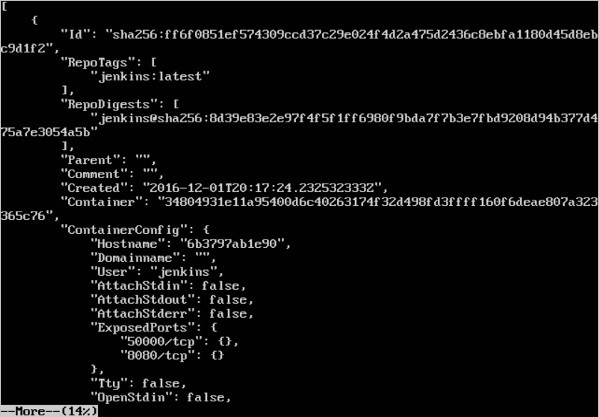
### Return Value

The low-level information of the image or container in JSON format.

### Example

sudo docker inspect jenkins

### Output



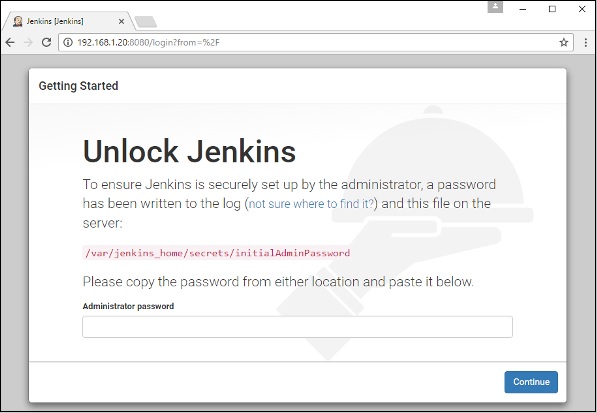
The output of the **inspect** command gives a JSON output. If we observe the output, we can see that there is a section of "ExposedPorts" and see that there are two ports mentioned. One is the **data port** of 8080 and the other is the **control port** of 50000.

To run Jenkins and map the ports, you need to change the Docker **run**command and add the ‘p’ option which specifies the port mapping. So, you need to run the following command −

sudo docker run -p 8080:8080 -p 50000:50000 jenkins

The left-hand side of the port number mapping is the Docker host port to map to and the right-hand side is the Docker container port number.

When you open the browser and navigate to the Docker host on port 8080, you will see Jenkins up and running.



# Private Registries

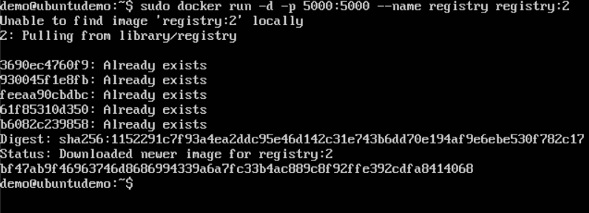
You might have the need to have your own private repositories. You may not want to host the repositories on Docker Hub. For this, there is a repository container itself from Docker. Let’s see how we can download and use the container for registry.

**Step 1** − Use the Docker **run** command to download the private registry. This can be done using the following command.

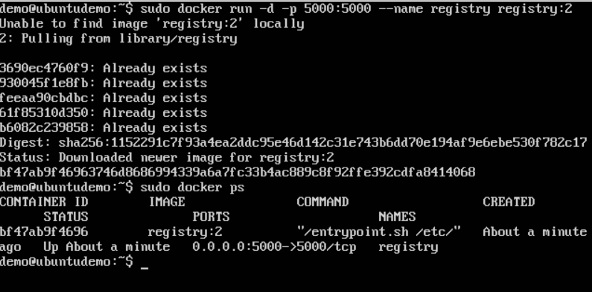
sudo docker run –d –p 5000:5000 –-name registry registry:2

The following points need to be noted about the above command −

* **Registry** is the container managed by Docker which can be used to host private repositories.
* The port number exposed by the container is 5000. Hence with the **–p command**, we are mapping the same port number to the 5000 port number on our localhost.
* We are just tagging the registry container as “2”, to differentiate it on the Docker host.
* The **–d** option is used to run the container in detached mode. This is so that the container can run in the background



**Step 2** − Let’s do a **docker ps** to see that the registry container is indeed running.



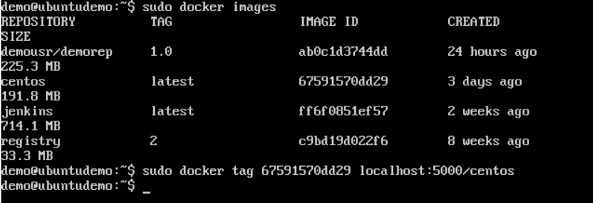
We have now confirmed that the registry container is indeed running.

**Step 3** − Now let’s tag one of our existing images so that we can push it to our local repository. In our example, since we have the **centos** image available locally, we are going to tag it to our private repository and add a tag name of **centos**.

sudo docker tag 67591570dd29 localhost:5000/centos

The following points need to be noted about the above command −

* **67591570dd29** refers to the Image ID for the **centos** image.
* **localhost:5000** is the location of our private repository.
* We are tagging the repository name as **centos** in our private repository.



**Step 4** − Now let’s use the Docker **push** command to push the repository to our private repository.

sudo docker push localhost:5000/centos

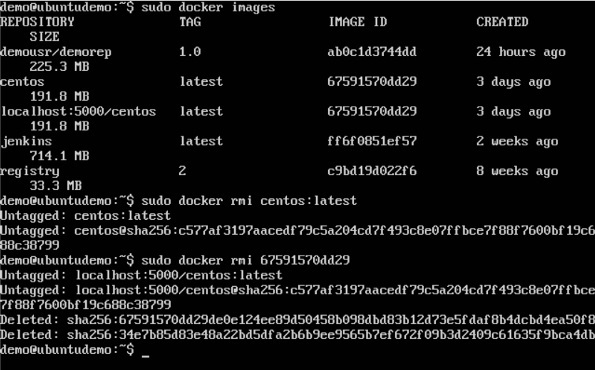
Here, we are pushing the **centos** image to the private repository hosted at **localhost:5000**.



**Step 5** − Now let’s delete the local images we have for **centos** using the **docker rmi** commands. We can then download the required **centos** image from our private repository.

sudo docker rmi centos:latest

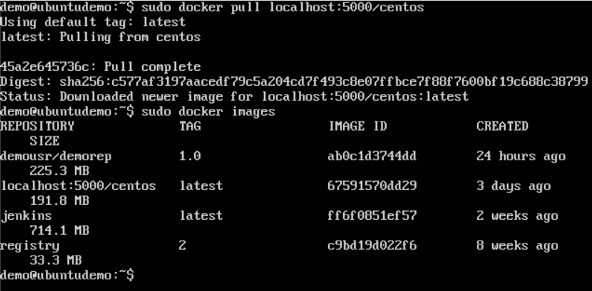
sudo docker rmi 67591570dd29



**Step 6** − Now that we don’t have any **centos** images on our local machine, we can now use the following Docker **pull** command to pull the **centos** image from our private repository.

sudo docker pull localhost:5000/centos

Here, we are pulling the **centos** image to the private repository hosted at **localhost:5000**.



If you now see the images on your system, you will see the **centos** image as well.