# **8.1** Build a Multi-Container Docker Application Locally

A green arrow pointing to a black text

Description automatically generated

**Objectives:**

* Build container image and create an application locally.
* Modify the application and rebuild the image.
* Push the local image to Docker Hub.
* Build an application locally with Docker Hub Image.
* Delete the images locally.

**Part 1. Build an Application Locally**

1. Start Docker Desktop
2. Download file **voting\_redis.zip** given online and unzip it.
3. Using command line, switch to directory **voting\_redis**.
4. Understand the code files in this directory.
5. The following command uses the **docker-compose.yaml** file to create the container image, download the Redis image, and start the application.

$ docker compose up -d

1. Use the docker images command to see the created images. One image is downloaded from Microsoft repository, and one image is created locally. The **vote\_front** image contains the front-end application. The **redis** image is the backend database service.

$ docker images

A black text on a white background

Description automatically generated

1. Run the docker ps command to see the running containers.

$ docker ps

A close up of a website

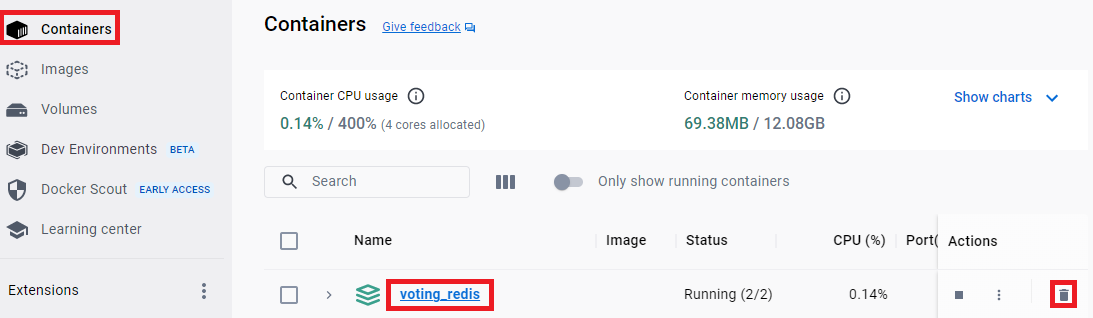
Description automatically generated

1. To test your application locally, navigate to http://localhost:8080 in a web browser.

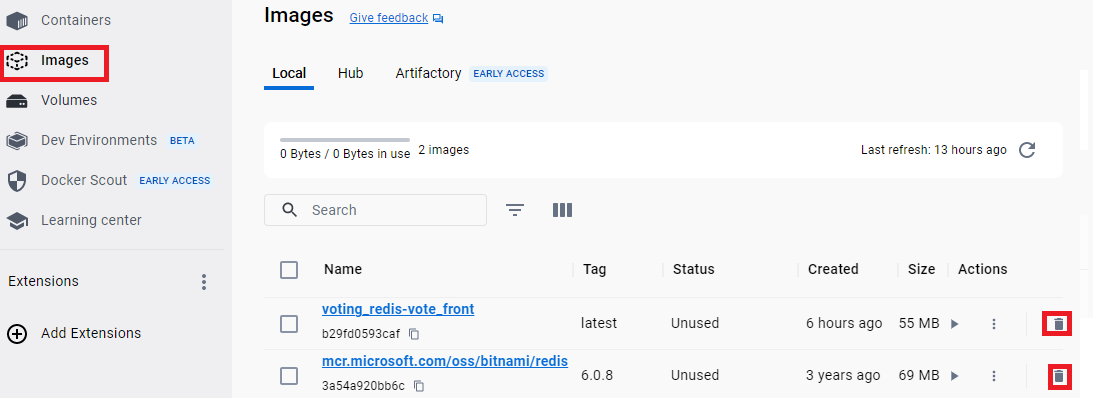
A green rectangular box with white text

Description automatically generated

1. From Docker Desktop, delete the container.



1. From Docker Desktop, delete the images.



**Part 2. Modify the Application**

1. Modify the configuration file at voting\_redis🡺vote🡺voting🡺 config\_file.cfg, change ‘Owls’ to “Robots”. Save the changes.
2. Create new images with the following command.

**$ docker compose up -d**

1. Navigate to <http://localhost:8080>. You will see the new web page.

A green rectangular object with text

Description automatically generated

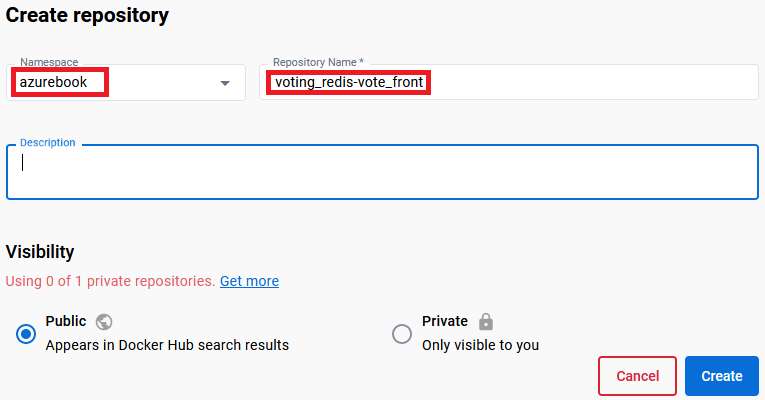
**Part 3. Push a local image to Docker Hub**

1. Sign into Docker hub from a web browser.
2. Click “Create repository”.

A blue rectangle with white text

Description automatically generated

1. Repository name is same as your local image.



1. Sign into Docker hub from Docker Desktop.
2. From command line, use the docker tag command to associate your local image with the remote image. Replace YOUR-DID with your Docker ID (Namespace). In my case, YOUR-DID is **azurebook**.

**$ docker tag voting\_redis-vote\_front YOUR-DID/voting\_redis-vote\_front**

1. From command line, push the local image to Docker Hub.

**$ docker push YOUR-DID/voting\_redis-vote\_front**

1. From Docker hub web page, you can see the image.

A screenshot of a computer

Description automatically generated

1. From Docker Desktop, delete the container and delete all local images.
2. Note here, your remote image at Docker Hub is not deleted.

**Part 4. Build an Application locally with Docker Hub image**

1. Start Command Line.
2. Create a new directory, say voting2, on your local drive.
3. Download file **docker-compose.yaml** given online.

A screenshot of a computer

Description automatically generated

1. Open docker-compose.yaml with a text editor. Compared with the file given in Part 1, the build command line is replaced with an image source on Docker hub. See the two files below.

**Compose file used in Part 1:**

A screenshot of a computer code

Description automatically generated

**Compose file used in Part 4:**

A screenshot of a computer code

Description automatically generated

1. Execute the following command, which uses the docker-compose.yaml file to create the container image, download the Redis image, and start the application.

**$ docker compose up -d**

1. Use the docker images command to see the images. Two images are downloaded. The **vote\_front** image contains the front-end application. The **redis** image provides the backend database service.

**$ docker images**

1. Here is the output.

A black text on a white background

Description automatically generated

1. Run the docker ps command to see the running containers.

**$ docker ps**

1. The output looks like the following.

A black text on a white background

Description automatically generated

1. To test your application, navigate to <http://localhost:8080> in a web browser.

A screenshot of a computer

Description automatically generated

1. From Docker Desktop, delete the container and delete all local images.