# Intro To Containers Lab Parth Patel

# Docker hello-world

## Obtaining hello-world image

Text

Description automatically generated

Since we are unable to find the image locally, we must pull it from a container registry online (DockerHub). One thing I noticed is that it pulls the *latest* image when you do not have it locally.

## Docker Image List – Local

Text

Description automatically generated

I have only one container image and thus it’s the only one displayed.

## Docker Run hello-world

Text

Description automatically generated

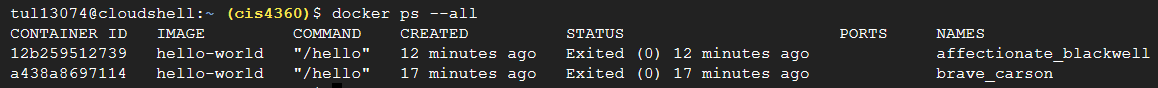
There is more text below but it is the same as text above so I didn’t wish to flood the document with redundant screenshots. I ran hello-world and it showed my container image was successfully installed and it ran correctly.

## Docker ps



This shows that we have no *active* containers because hello-world was only a print-out image: it was not a process that was required to persist after showing us the container was correctly installed.

## Docker ps -a



Shows all of our containers despite their *active* status.

# Docker build

## Defining dockerfile

Text

Description automatically generated

We first create a test container folder at our home directory and navigate it in. Now for writing to this file.

We use cat > dockerfile to write to the file.

I was happy to learn about the <<EOF. Essentially it reads the input until the char sequence after the << is read. If it’s read, then it stops writing to file.

## Making JS app

Text

Description automatically generated

I wrote the code on the VSCode-like IDE on GCP. Tried my best to understand the code since I only know Basic JS and frontend JS libs like react. In other words, I have only written browser-based code so server-side code (node) was new to me.

## Using Docker build

Text

Description automatically generated

My understanding of this is that we called docker build on our current directory, as specified by the `.` as the last argument. Then we tagged our app using a name and the version we are working on (0.1). From the dockerfile docker attempted to fetch all of our dependencies, this why we have all the pull complete messages. Now this container **can** run.

## Docker images

Text

Description automatically generated

The images we have installed on our GCP.

## Docker run

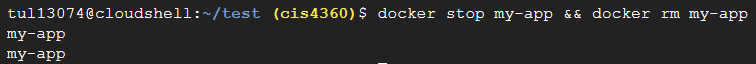


We run our image on the specified http port.

## 

Entered new tab in GCP Cloud Console and requested a server response.

## Docker stop



Closed my container instance and then removed my built image.

## Docker run detached mode



The id of our container is posted and it is now a background process.

A screenshot of a computer

Description automatically generatedI used curl again to test the response and then listed the active containers. Our node is the only one active.

## Docker logs



Retrieved the current logs for 199cefa649426ee17abe1fb779a13750aff50947b223724d1d010b627f854aaa

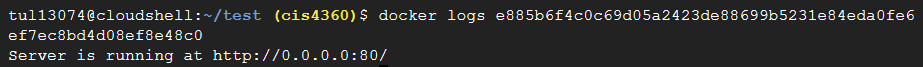
## Docker 2nd Instance of my-app

Text

Description automatically generated

Made another version of my-app by changing the response text. Ran it as detached and called it to display its message. We note that since there already exists already the node dependency it does not have to retrieve it from the registry.

# Debugging



Logging container v2.

Graphical user interface, text

Description automatically generated

Created an interactive bash shell for our container. Used Linux commands to view it’s internal file system. As specified in the dockerfile we are in the /app directory.

## Docker inspect

Text

Description automatically generated

Inspected container: node-app:0.2. The output of this is very long

# Publish

A screenshot of a computer

Description automatically generated with medium confidence

We added our image to gcr.io which is the Google Cloud Registry.

## Docker push

Text

Description automatically generated

I had to first enable containerregistry.googleapis.com as it wasn’t active on my account. Then I pushed the image to the google container registry.

## Container exists in GCR

Graphical user interface, text, application

Description automatically generated

## Docker stop and remove for images

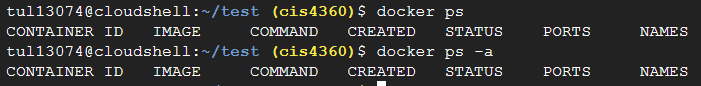
A picture containing text

Description automatically generated

We first stop all docker containers and then delete the containers.

Text

Description automatically generated



Deleted all docker images from local storage. I then ran ps to ensure that no images persisted. We feed the rmi(remove image) command the names of all images so it could delete them all.

## Docker pull

Text

Description automatically generated

I pulled the image we put on the GCR and then ran it as detached. Then I ran the docker image that we pulled. I looked at docker ps to ensure it was running properly. Then we requested the server and got the plain text back.