docker run hello-world

docker images

Text

Description automatically generated

look at the running containers

Docker ps

see all containers, including ones that have finished executing

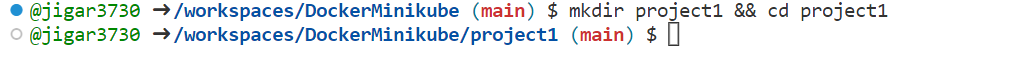
Docker ps -a

Graphical user interface, text, application

Description automatically generated with medium confidence

Create directory for docker image project

mkdir project1 && cd project1



Create Docker file

Touch Dockerfile

Text

Description automatically generated

Update Dockerfile with details to run node and expose port 80

This file instructs the Docker daemon on how to build your image.

* The initial line specifies the base parent image, which in this case is the official Docker image for node version long term support (lts).
* In the second, you set the working (current) directory of the container.
* In the third, you add the current directory's contents (indicated by the "." ) into the container.
* Then expose the container's port so it can accept connections on that port and finally run the node command to start the application.

# Use an official Node runtime as the parent image

FROM node:lts

# Set the working directory in the container to /app

WORKDIR /app

# Copy the current directory contents into the container at /app

ADD . /app

# Make the container's port 80 available to the outside world

EXPOSE 80

# Run app.js using node when the container launches

CMD ["node", "app.js"]

Text

Description automatically generated

Create a node application

Touch app.js



Update file with app data

const http = require('http');

const hostname = '0.0.0.0';

const port = 80;

const server = http.createServer((req, res) => {

    res.statusCode = 200;

    res.setHeader('Content-Type', 'text/plain');

    res.end('Hello World\n');

});

server.listen(port, hostname, () => {

    console.log('Server running at http://%s:%s/', hostname, port);

});

process.on('SIGINT', function() {

    console.log('Caught interrupt signal and will exit');

    process.exit();

});

Graphical user interface, text

Description automatically generated

Build a docker image

docker build -t project1:01 .

verify image file with

docker images

Text

Description automatically generated

Run image as app-run1 and map port to 4000

docker run -p 4000:80 --name app-run1 project1:01

Graphical user interface, text, application

Description automatically generated

Split terminal and Verify with

* Docker ps
* Curl
* And webpage

Graphical user interface, text, application

Description automatically generated

Stop the container with stop and container id

Verify with ps -a and curl

Text

Description automatically generated

Verify logs with

Docker logs container id

Docker logs 359

A picture containing chart

Description automatically generated

Update app.js text and create new image and verify images

Text

Description automatically generated

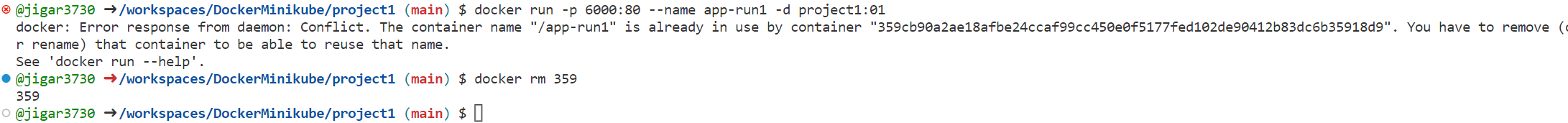
docker build -t project1:02 .

Text

Description automatically generated

Delete container created by earlier run

docker rm 359



Run both the images with -d option

* docker run -p 6000:80 --name app-run1 -d project1:01
* docker run -p 8080:80 --name app-run2 -d project1:02

Graphical user interface, text, application

Description automatically generated

Verify the running containers.

Graphical user interface, text, application, email

Description automatically generated

Start minikube

* minikube status
* minikube start

Graphical user interface, text, application

Description automatically generated

Rlseuse the Docker daemon from Minikube with eval $(minikube docker-env).

So to use an image without uploading it, you can follow these steps:

1. Set the environment variables with eval $(minikube docker-env)
2. Build the image with the Docker daemon of Minikube (eg docker build -t my-image .)
3. Set the image in the pod spec like the build tag (eg my-image)
4. Set the [imagePullPolicy](https://kubernetes.io/docs/concepts/containers/images/" \l "updating-images) to Never, otherwise Kubernetes will try to download the image.

Build docker image in minikube

* docker build -t project1:02 .
* docker images

Text

Description automatically generated

Update sources to load kubectl completion on tab

* source <(kubectl completion bash)

A picture containing text

Description automatically generated

Launch minikube dashboard

* minikube dashboard –url

Graphical user interface, text, application, email

Description automatically generated

Create a yaml file for deployment and verify

**kubectl create deploy firstapp --image=project1:01 --replicas=3 --port=80 --dry-run=client -o yaml >firstapp.yaml**

Text

Description automatically generatedku

Apply deployment and expose port

**kubectl apply -f firstapp.yaml**

**kubectl expose deployment firstapp --type=LoadBalancer --port=80**

**A picture containing diagram

Description automatically generated**

Verify that you are able to access the app via service url

minikube service firstapp –url

curl <http://192.168.49.2:31567>

A picture containing timeline

Description automatically generated