Actionable docker

Using docker to run packages locally.

Docker let's you do a lot of things, here's my tutorial on the same -

This tutorial is on `actionable docker` to start `packages` locally.

Installing Docker

`Docker GUI` is the easiest way to get off the ground.

You can find instructions to install docker on https://docs.docker.com/engine/install/

At the end of the installation, you need to make sure you're able to run the following command -

```
➜  ~ docker run hello-world
^[Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
478afc919002: Pull complete
Digest: sha256:d000bc569937abbe195e20322a0bde6b2922d805332fd6d8a68b19f524b7d21d
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

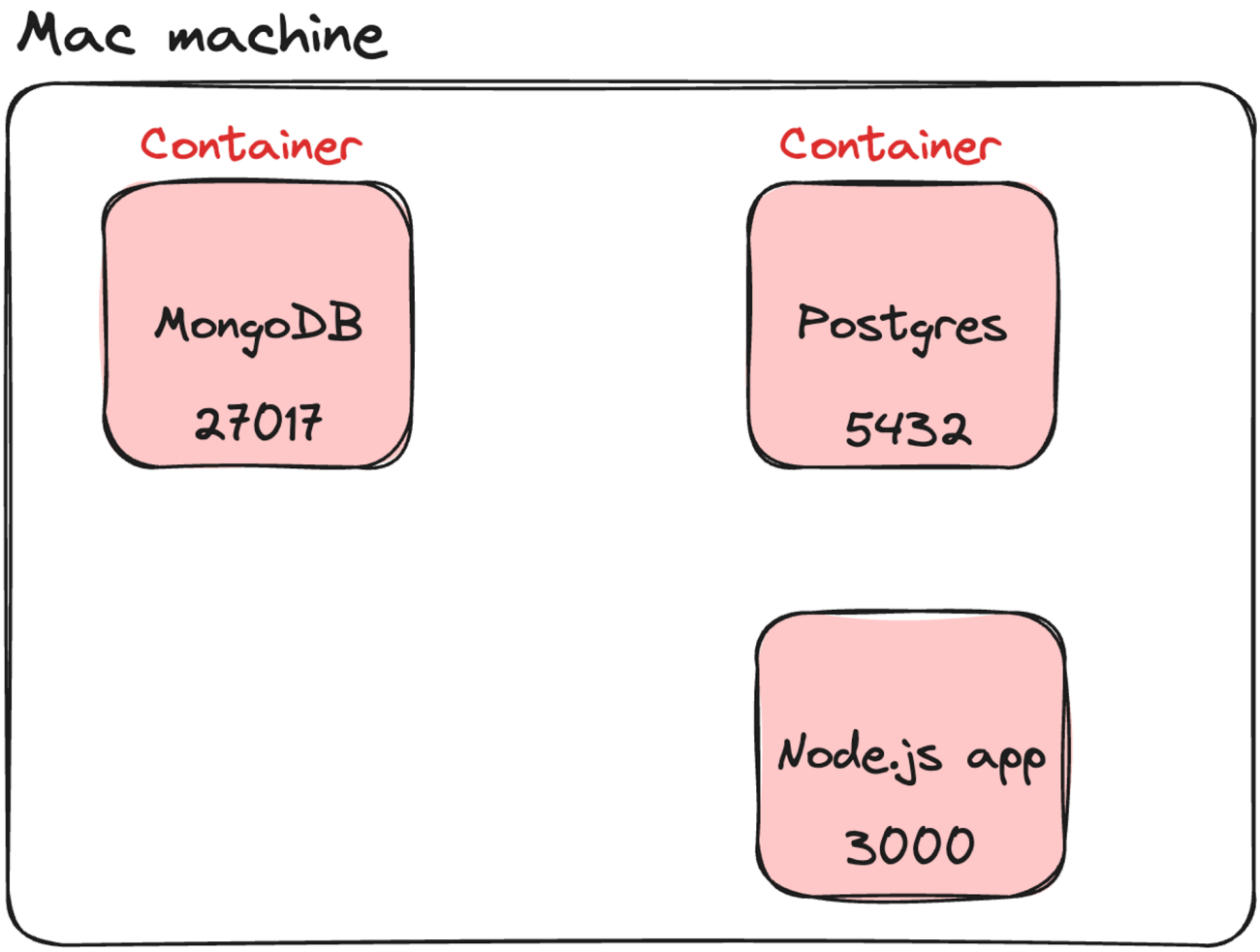
What are we using docker for?

Docker let's you do a lot of things.

It let's you `containerise` your applications.

It let's you run other people's `code + packages` in your machine.

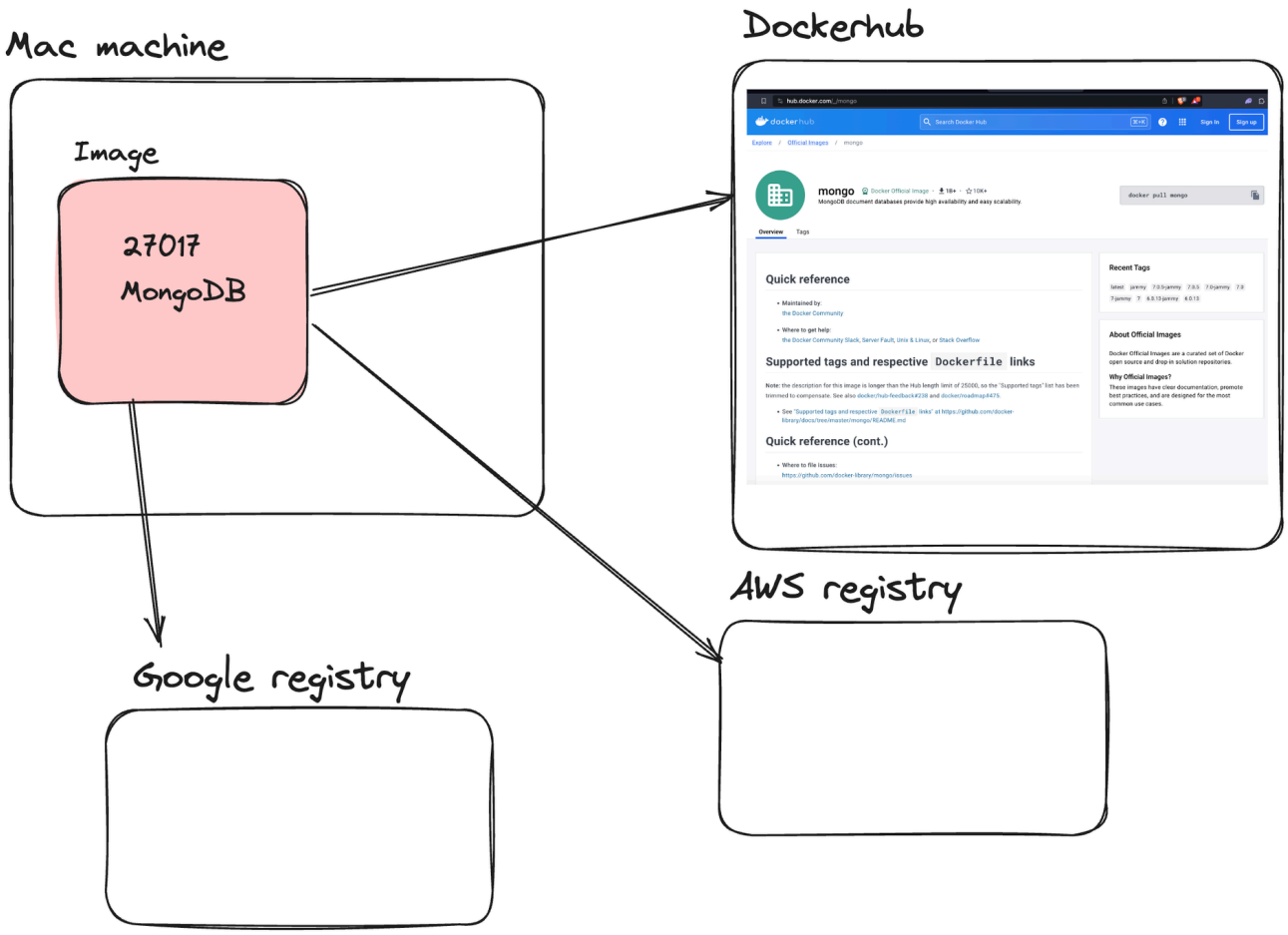
It let's you run common software packages inside a `container` (For eg - Mongo, Postgres etc)



Where can we get packages from?

Just like you can push your `code` to Github/Gitlab.

You can push `images` to `docker registries`



Common commands to know

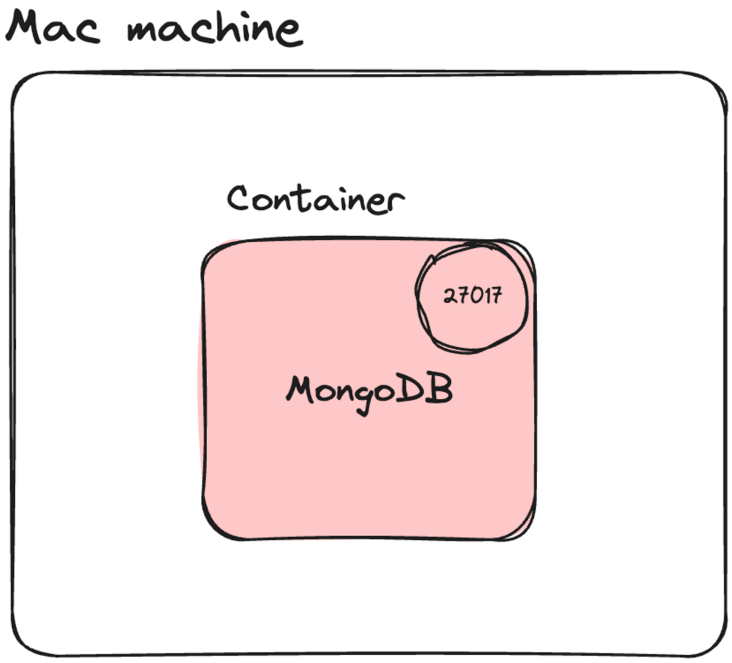
1. docker run
2. docker ps
3. docker kill

Running an image

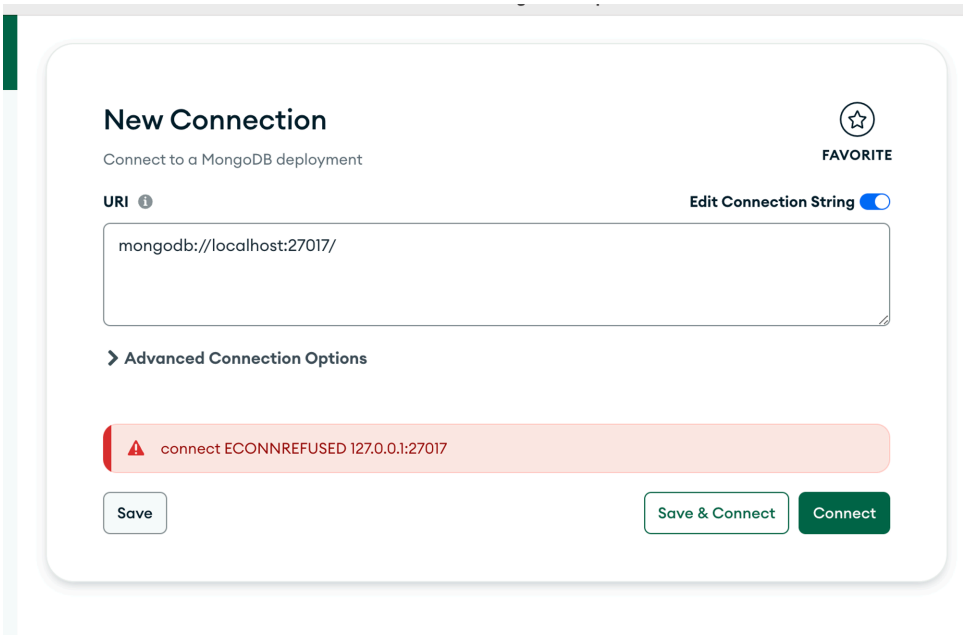
1. Running a simple image

Let's say you wan't to run MongoDB locally https://hub.docker.com/_/mongo

```
docker run mongo
```
Copy



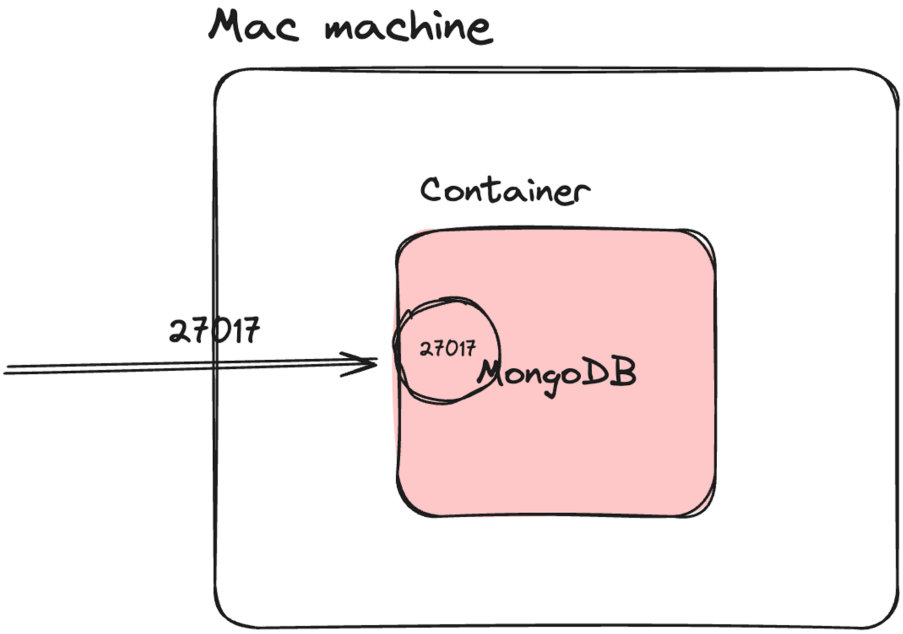
You will notice you can't open it in `MongoDB Compass` .



Adding a port mapping

The reason is that you haven't added a `port mapping`

```
docker run  -p 27017:27017 mongo
```
Copy



Starting in detached mode

Adding `-d` will ensure it starts in the background

```
docker run -d -p 27017:27017 mongo
```
Copy

Inspecting a container

```
docker ps
```
Copy

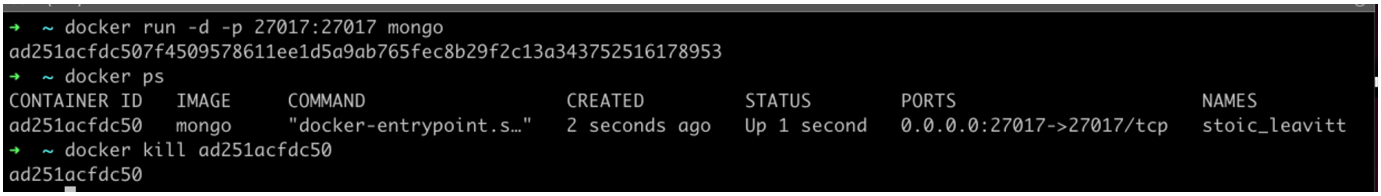
This will show you all the containers you are running.

Stopping a container

```
docker kill <container_id>
```
Copy

Will stop the container that you are running

In the end, this is the flow of commands -



Common packages

Mongo

```
docker run -d -p 27017:27017 mongo
```
Copy

Postgres

```
docker run -e POSTGRES_PASSWORD=mysecretpassword -d -p 5432:5432 postgres
```
Copy

The connection string for this postgres would be

```
postgresql://postgres:mysecretpassword@localhost:5432/posty
```

Copy

▼ Code to test it out

Copy

```
// Import the pg library
const { Client } = require('pg');

// Define your connection string (replace placeholders with your actuc
const connectionString = 'postgresql://postgres:mysecretpassword@local

// Create a new client instance with the connection string
const client = new Client({
  connectionString: connectionString
});

// Connect to the database
client.connect(err => {
  if (err) {
    console.error('connection error', err.stack);
  } else {
    console.log('connected to the database');
  }
});

// Run a simple query (Example: Fetching the current date and time frc
client.query('SELECT NOW()', (err, res) => {
  if (err) {
    console.error(err);
  } else {
    console.log(res.rows[0]);
  }

  // Close the connection
  client.end();
});
```