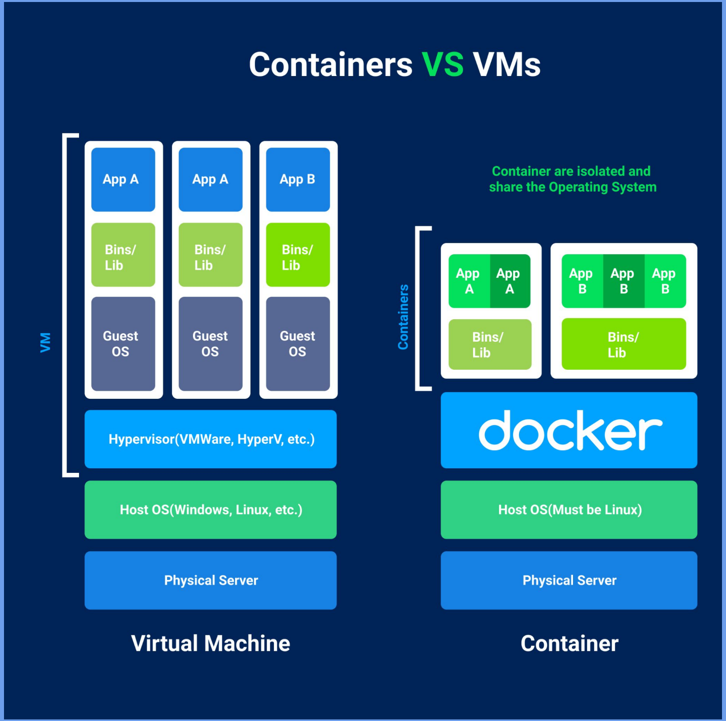
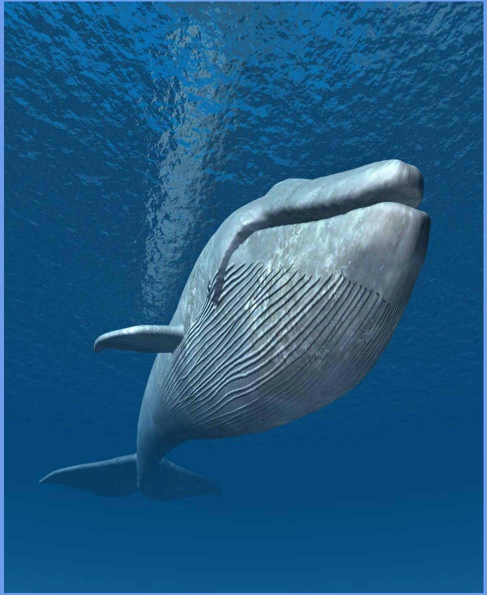


Docker

-Byte Club-

Week

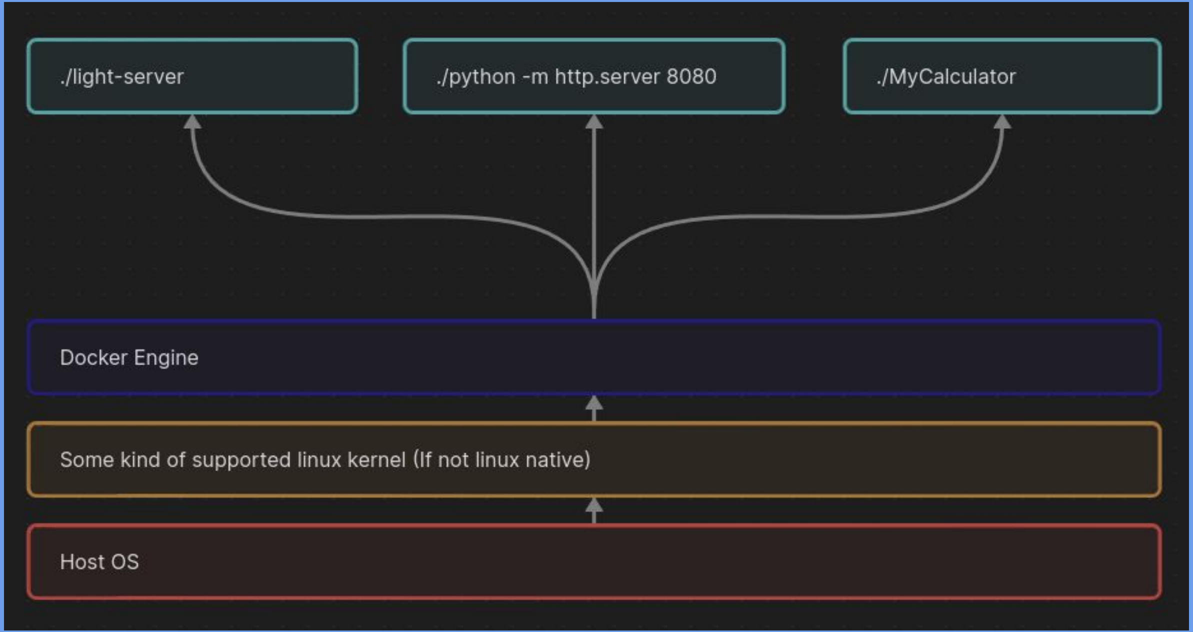
2



Overview

The docker daemon provides an api to the docker cli tool via the docker engine.

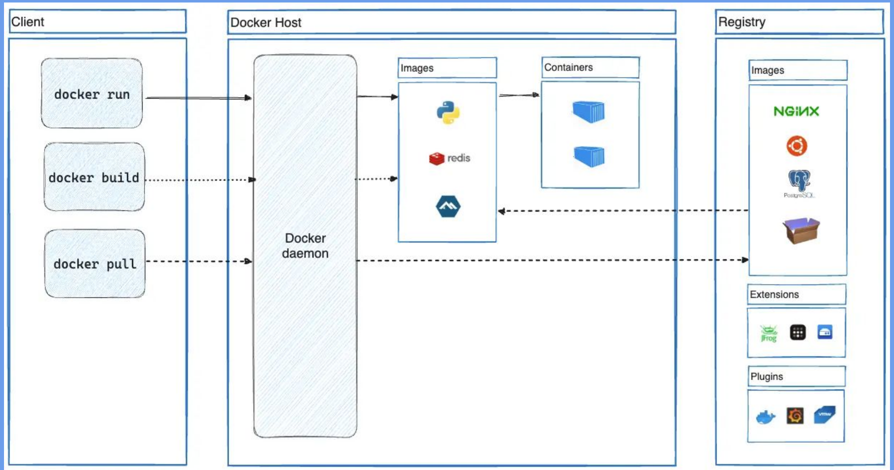
This allows us to manipulate active containers, and archived images.

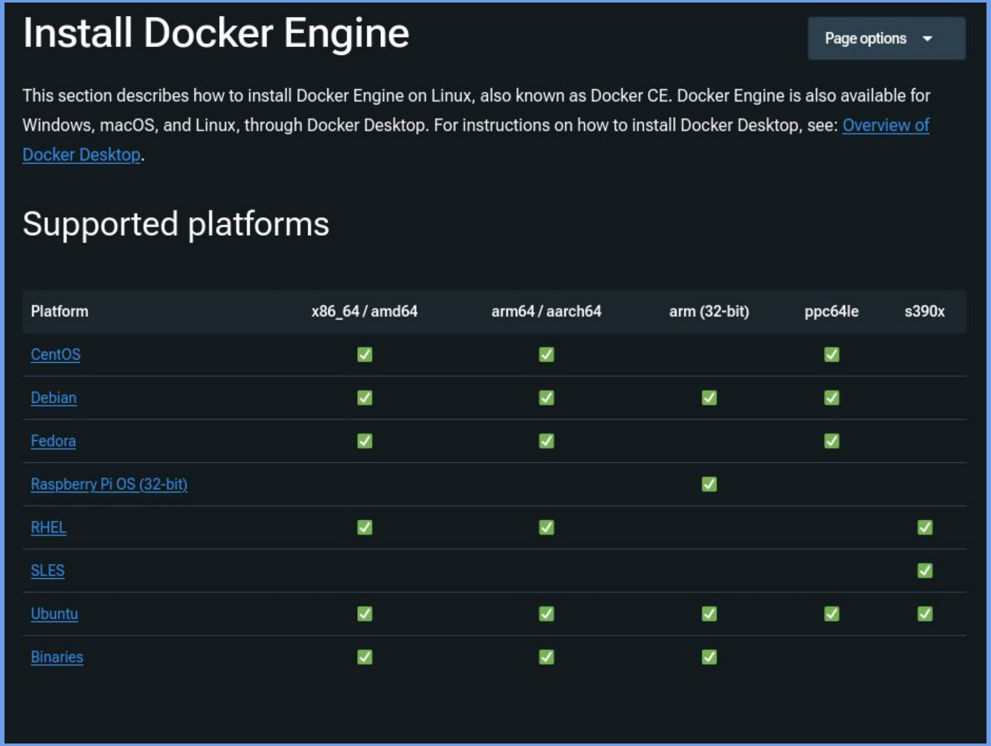

./light-server
./python -m http.server 8080
./MyCalculator
Docker Engine
Some kind of supported linux kernel (If not linux native)
Host OS

Docker Engine

The docker engine allows us to run containers.

It also let’s us call out remotely to the container registry to pull images not archived locally.





Install Docker Engine

This section describes how to install Docker Engine on Linux, also known as Docker CE. Docker Engine is also available for Windows, macOS, and Linux, through Docker Desktop. For instructions on how to install Docker Desktop, see: Overview of Docker Desktop.

Supported platforms

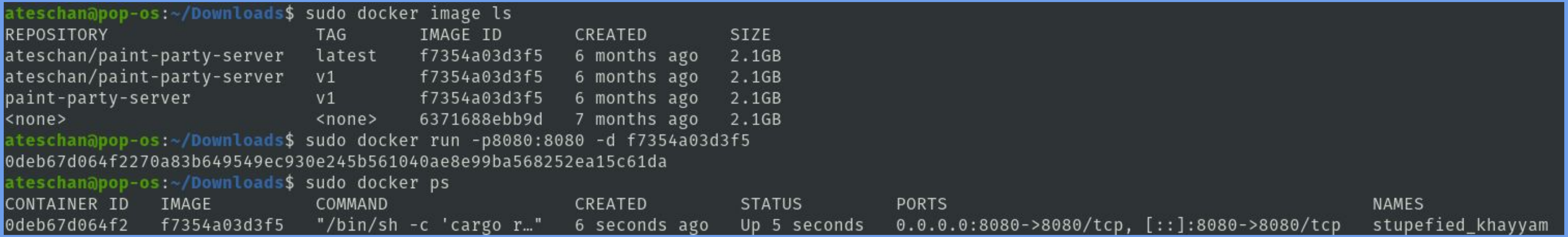
Platform	x86_64 / amd64	arm64 / aarch64	arm (32-bit)	ppc64le	s390x
CentOS	✓	✓		✓	
Debian	✓	✓	✓	✓	
Fedora	✓	✓		✓	
Raspberry Pi OS (32-bit)			✓		
RHEL	✓	✓			✓
SLES					✓
Ubuntu	✓	✓	✓	✓	✓
Binaries	✓	✓	✓		

Images && Containers

An “image” is a single template of a stripped down linux os with your software setup and ready to go!

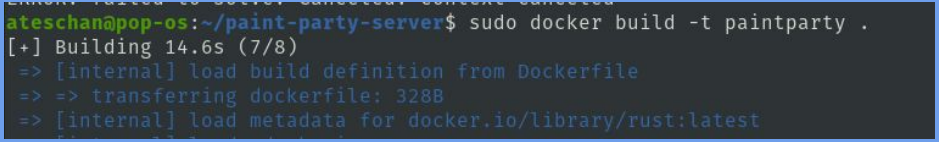
It is very easy to create, destroy, log, and distribute working versions of your software without the need to setup libraries / jvm / whatever interpreter.

A “container” is a single instance of an image.
Like an instance of a class in java.

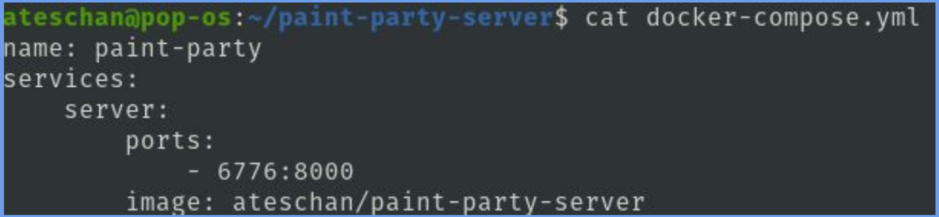


```
ateschan@pop-os:~/Downloads$ sudo docker image ls
REPOSITORY                    TAG       IMAGE ID       CREATED        SIZE
ateschan/paint-party-server   latest    f7354a03d3f5   6 months ago   2.1GB
ateschan/paint-party-server   v1        f7354a03d3f5   6 months ago   2.1GB
paint-party-server            v1        f7354a03d3f5   6 months ago   2.1GB
<none>                        <none>    6371688ebb9d   7 months ago   2.1GB
ateschan@pop-os:~/Downloads$ sudo docker run -p8080:8080 -d f7354a03d3f5
0deb67d064f2270a83b649549ec930e245b561040ae8e99ba568252ea15c61da
ateschan@pop-os:~/Downloads$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                                         NAMES
0deb67d064f2   f7354a03d3f5   "/bin/sh -c 'cargo r…"   6 seconds ago   Up 5 seconds   0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp   stupefied_khayyam
```

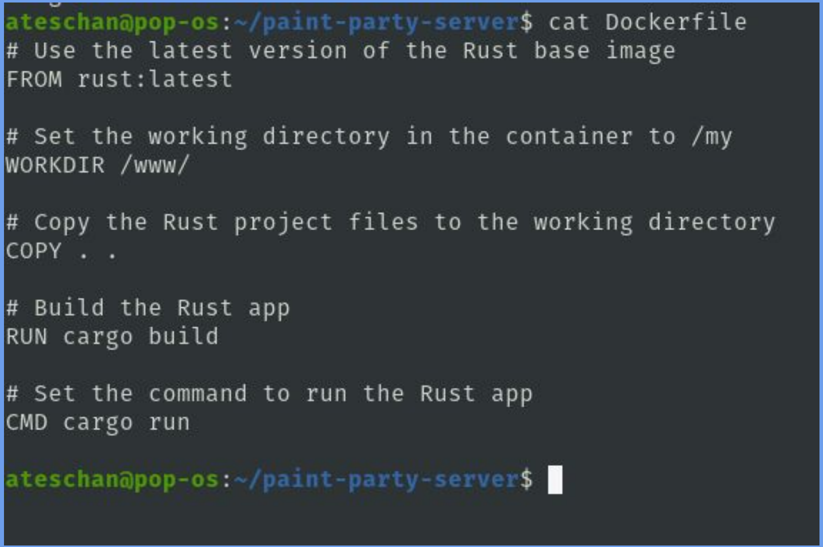
Dockerfile && Docker Compose



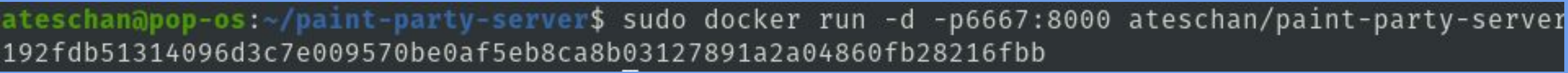
```
ateschan@pop-os:~/paint-party-server$ sudo docker build -t paintparty .
[+] Building 14.6s (7/8)
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 328B
 => [internal] load metadata for docker.io/library/rust:latest
```



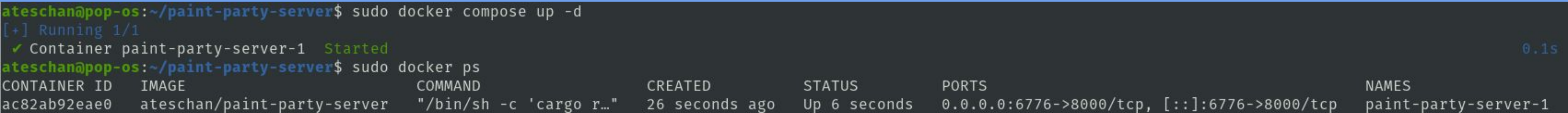
```
ateschan@pop-os:~/paint-party-server$ cat docker-compose.yml
name: paint-party
services:
    server:
        ports:
            - 6776:8000
        image: ateschan/paint-party-server
```



```
ateschan@pop-os:~/paint-party-server$ cat Dockerfile
# Use the latest version of the Rust base image
FROM rust:latest

# Set the working directory in the container to /my
WORKDIR /www/

# Copy the Rust project files to the working directory
COPY . .

# Build the Rust app
RUN cargo build

# Set the command to run the Rust app
CMD cargo run

ateschan@pop-os:~/paint-party-server$
```



```
ateschan@pop-os:~/paint-party-server$ sudo docker run -d -p6667:8000 ateschan/paint-party-server
192fdb51314096d3c7e009570be0af5eb8ca8b03127891a2a04860fb28216fbb
```

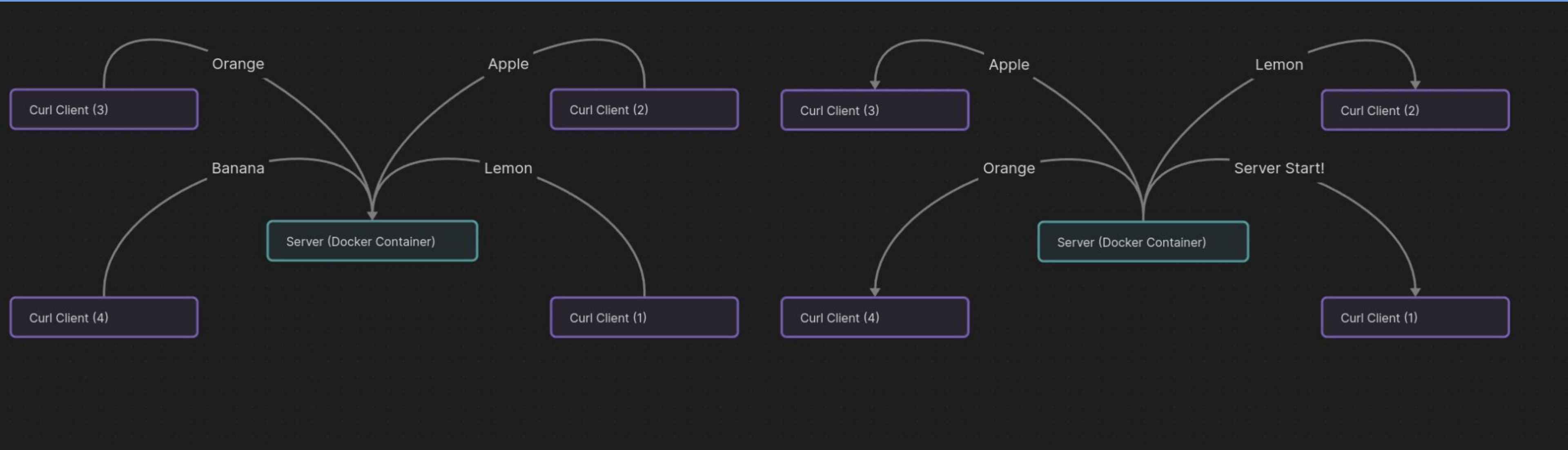


```
ateschan@pop-os:~/paint-party-server$ sudo docker compose up -d
[+] Running 1/1
 ✔ Container paint-party-server-1  Started                                                                    0.1s
ateschan@pop-os:~/paint-party-server$ sudo docker ps
CONTAINER ID   IMAGE                         COMMAND                  CREATED          STATUS         PORTS                                       NAMES
ac82ab92eae0   ateschan/paint-party-server   "/bin/sh -c 'cargo r…"   26 seconds ago   Up 6 seconds   0.0.0.0:6776->8000/tcp, [::]:6776->8000/tcp   paint-party-server-1
```

Application

Any software can be dockerized, for some it may be unnecessary, but it is very easy to produce working cross platform images!

Ready? See https://github.com/nvc-infinite-loops/ascii-donation-server



Synopsis

Docker - A suite of software tools that allow you to containerize your software.

Dockerfile - Instructions in a file that build out your software into an image

Image - An os template of your software working in a linux distro

Container - A single instance of a running image

Docker Compose - A docker extension that allows you to spin up and down a complex docker run command within a single manageable file.

Docker Compose File - Instructions in YML for docker compose

i.e. “docker-compose.yml”

Resources

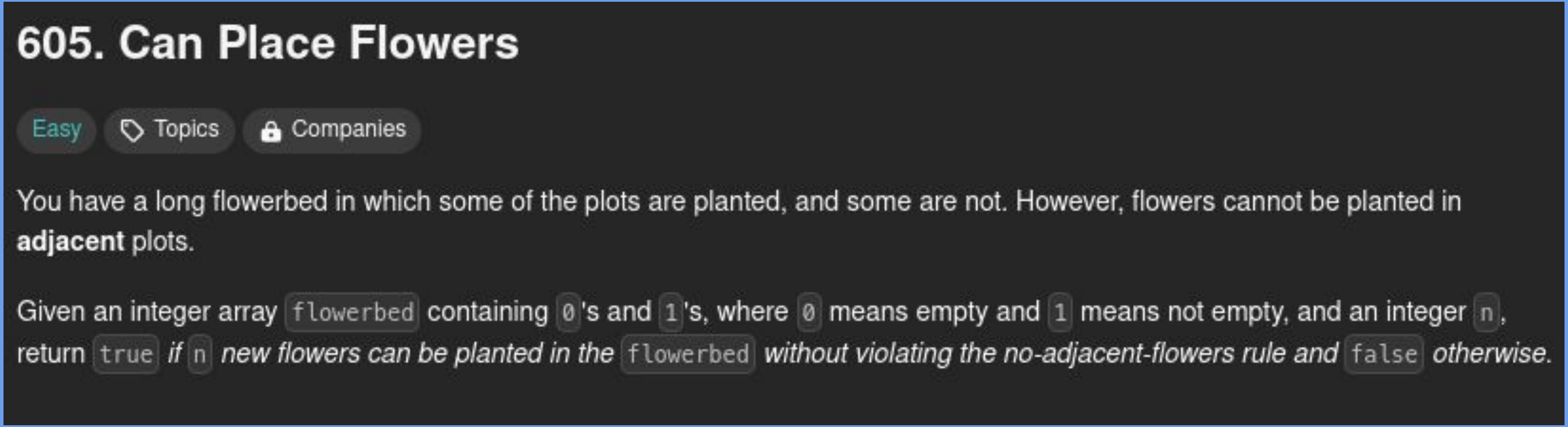
https://docs.docker.com/engine/install/

https://docs.docker.com/get-started/docker-concepts/building-images/writing-a-dockerfile/

https://docs.docker.com/compose/install/linux/

https://www.composerize.com/

LOTW - 605



605. Can Place Flowers

You have a long flowerbed in which some of the plots are planted, and some are not. However, flowers cannot be planted in **adjacent** plots.

Given an integer array `flowerbed` containing `0`'s and `1`'s, where `0` means empty and `1` means not empty, and an integer `n`, return `true` *if* `n` *new flowers can be planted in the* `flowerbed` *without violating the no-adjacent-flowers rule and* `false` *otherwise.*

When you are finished, please fork and contribute to the repo in the org
Watch out for the edge cases!