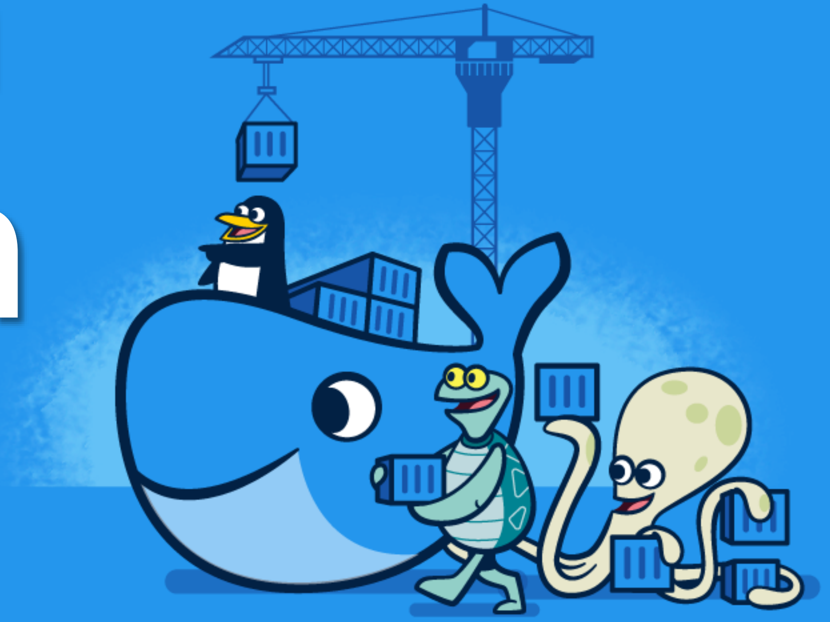
A Workshop On Containerization



~~Why should I care?~~

What are the benefits?

Containerization Dominates

zomato

SWIGGY

OpenAI

Razorpay

JUSPAY

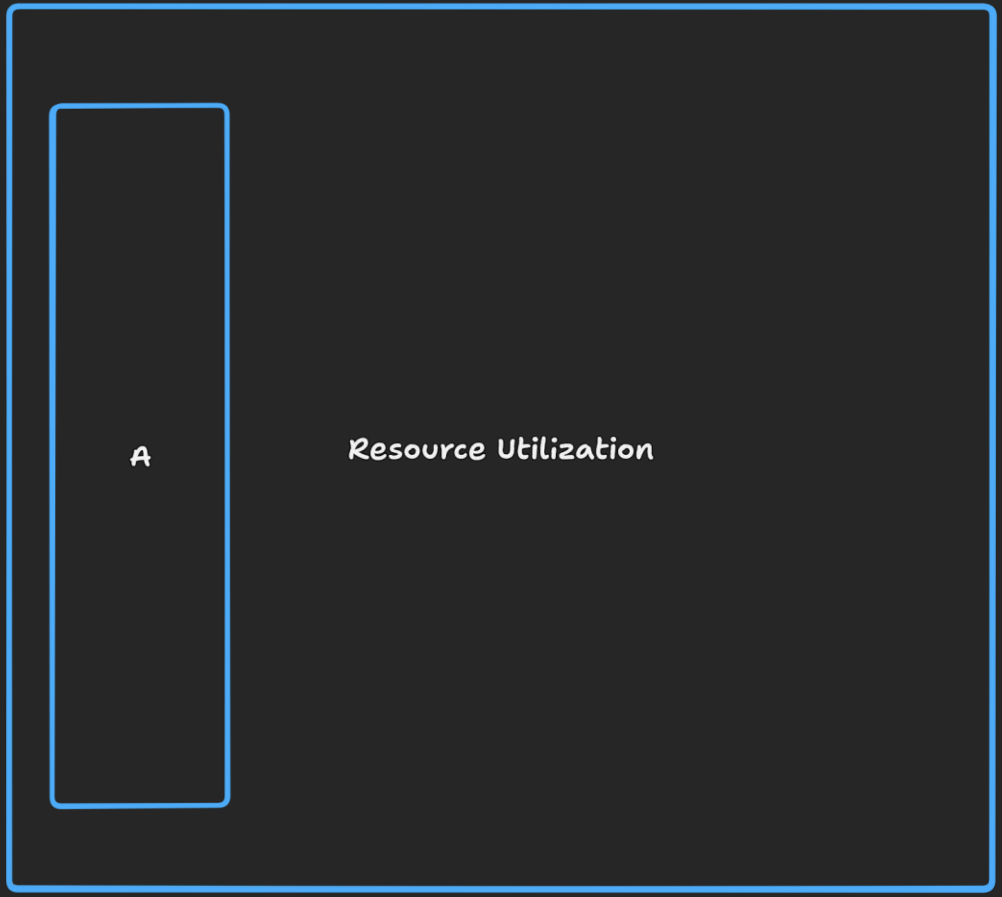
Meta

amazon

Google

Uber

An example

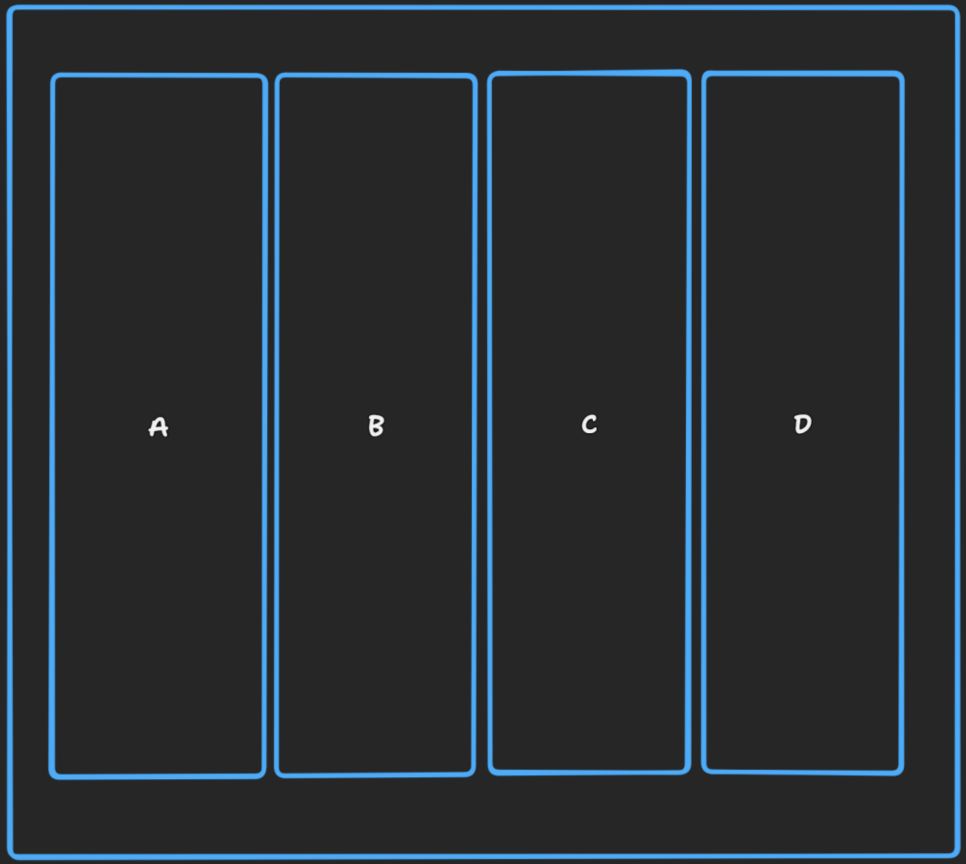


CPU
16
Cores

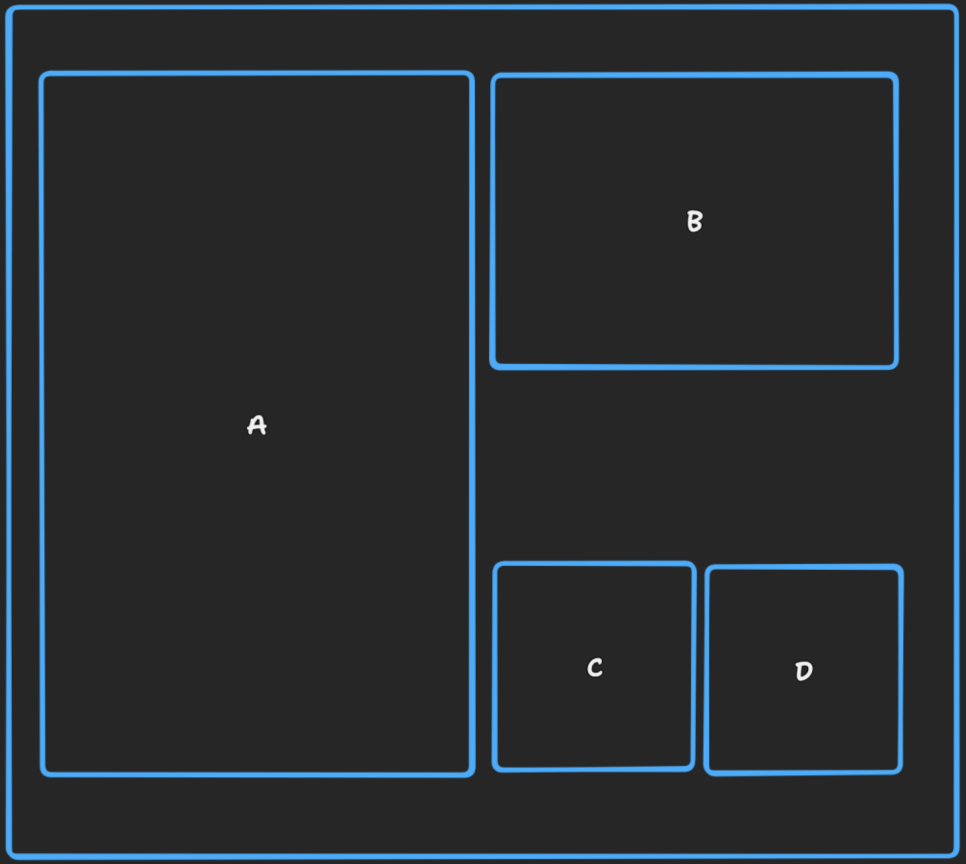
Memory
64 GB

$$$$$

Sharing Model

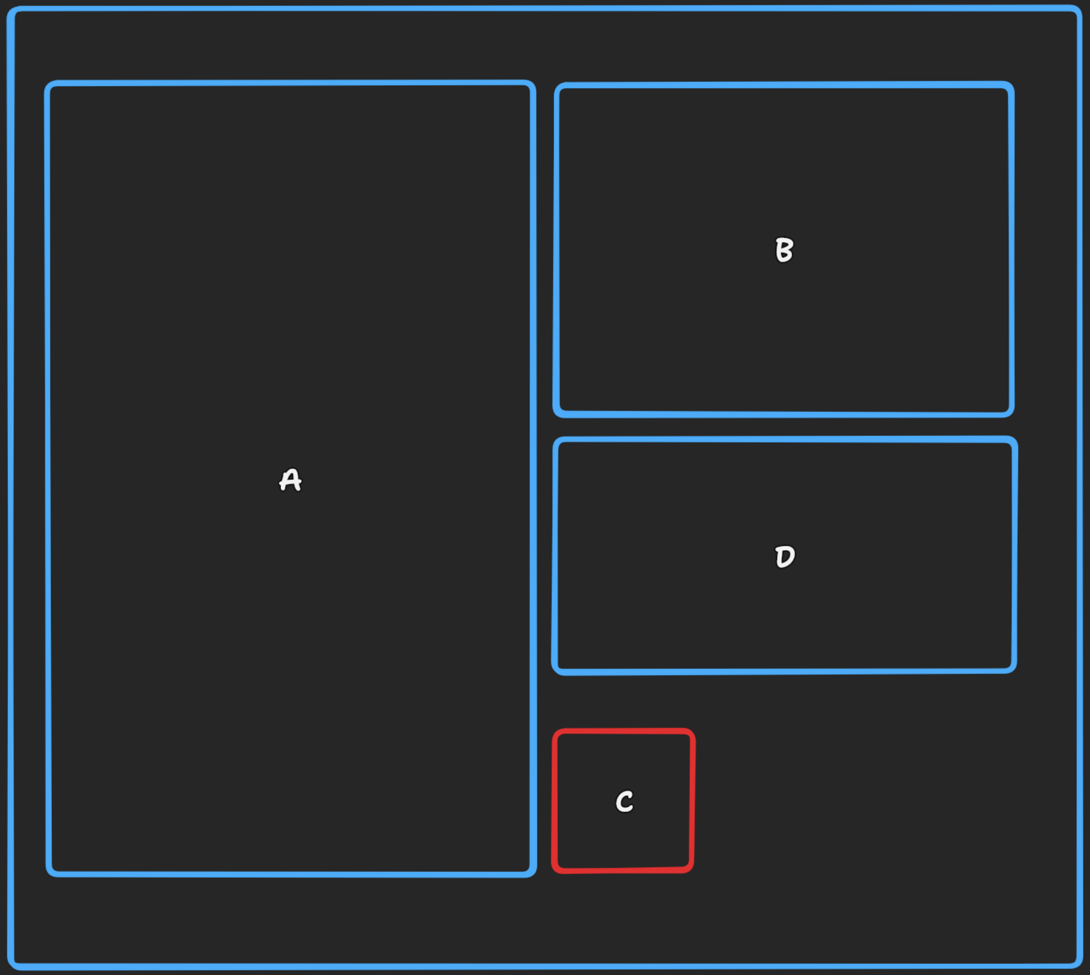


Expectation



Reality

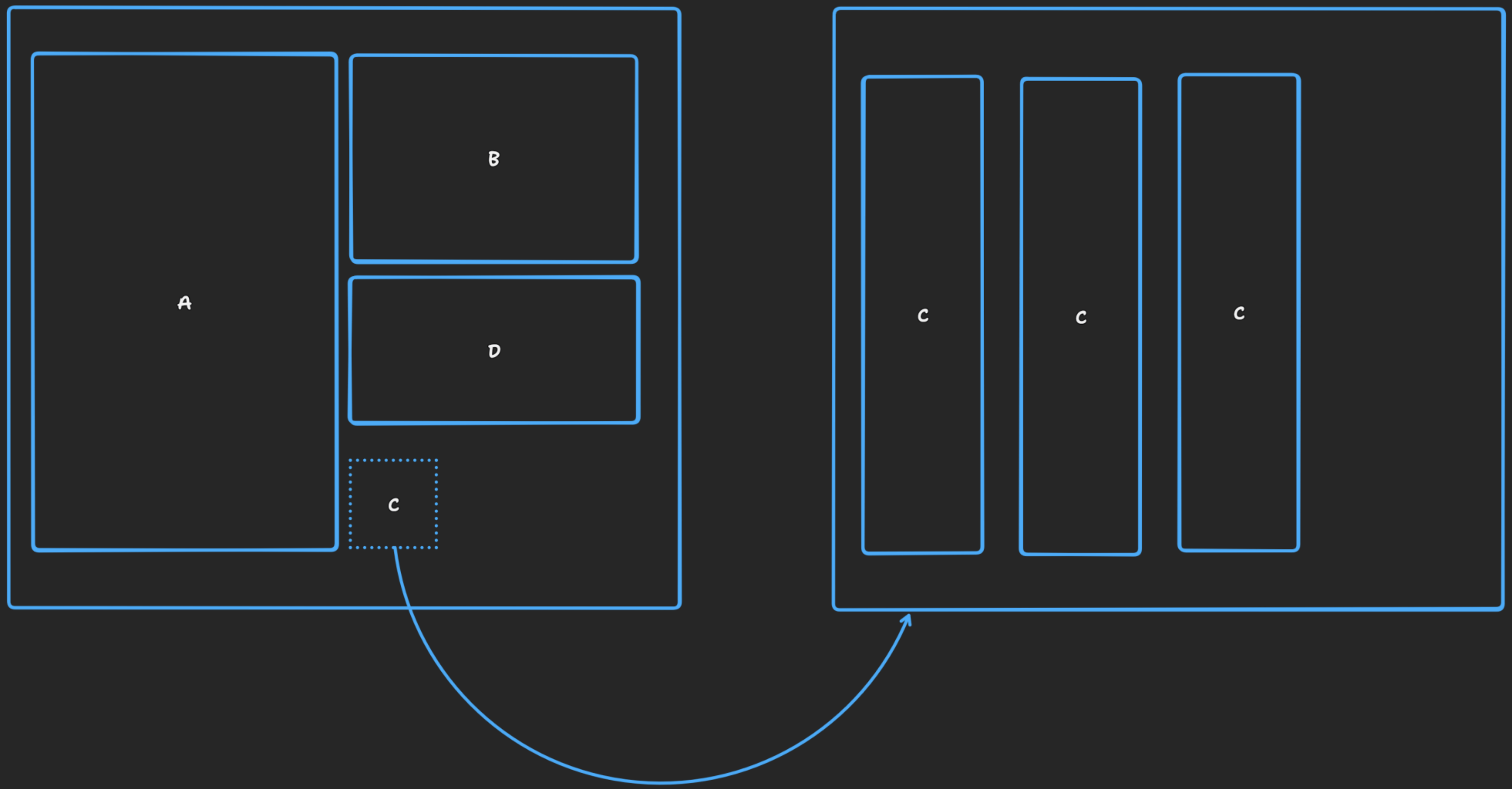
Failures



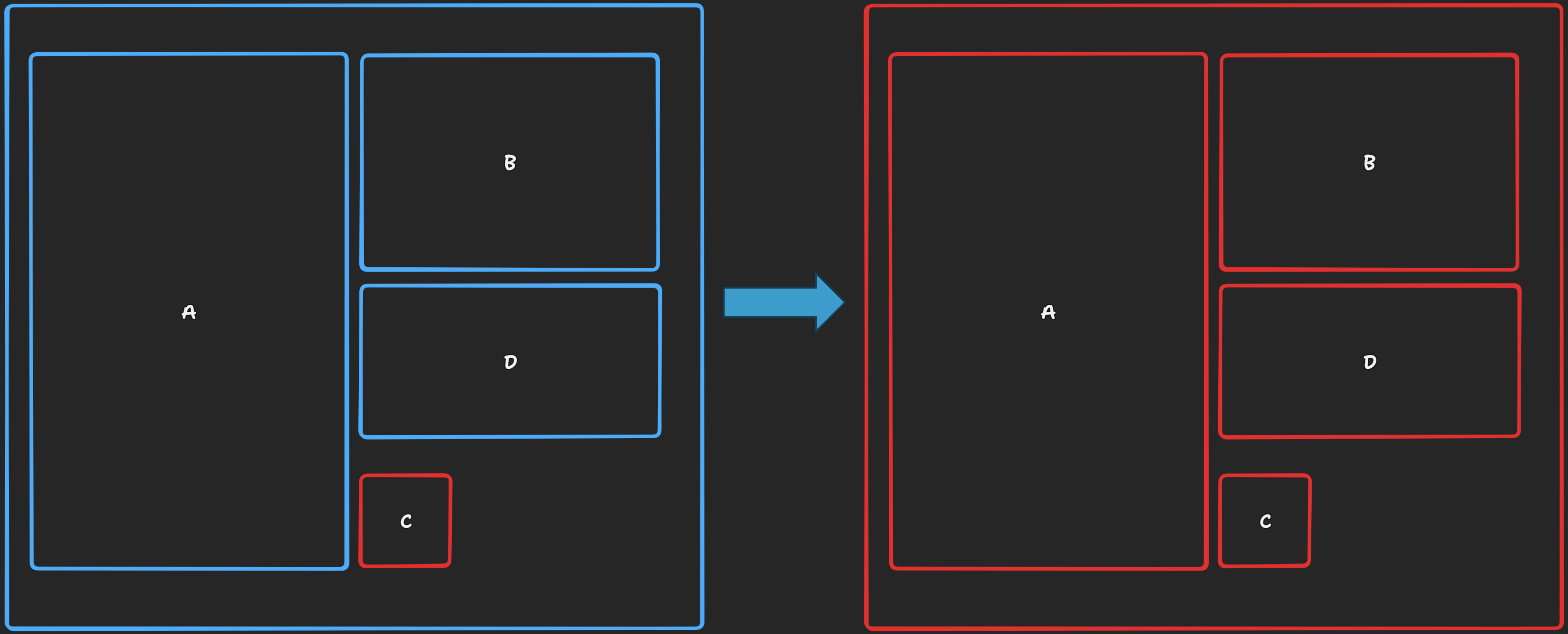
- Dependencies were missing
- Wrong version of dependencies
- Wrong kernel API call
- Clashing ENVs

IT WORKS ON MY MACHINE
THEN WE'LL SHIP YOUR MACHINE
AND THAT IS HOW DOCKER WAS BORN

Scale up



Crashes

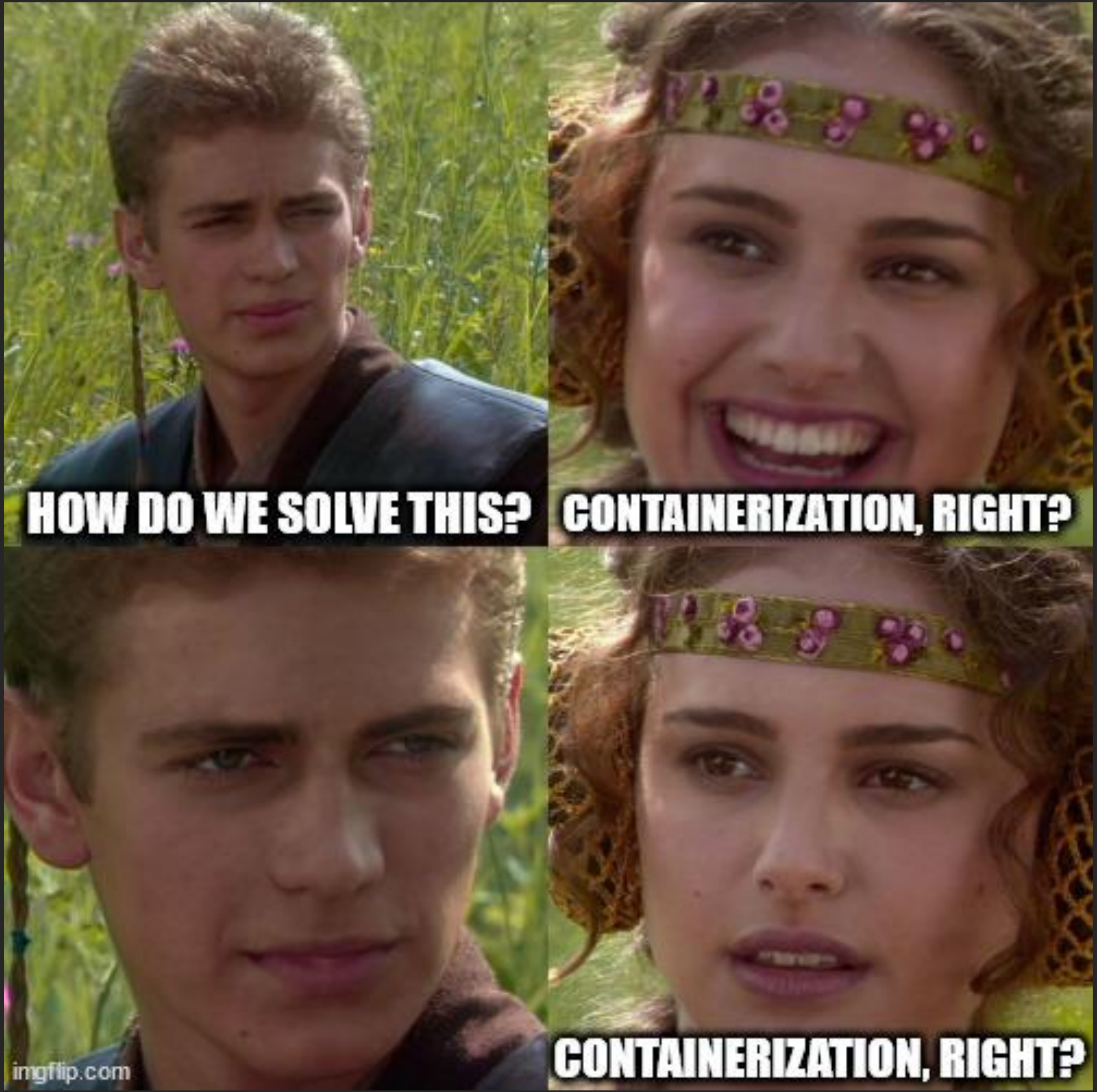


Isolation

Set Resource Bounds

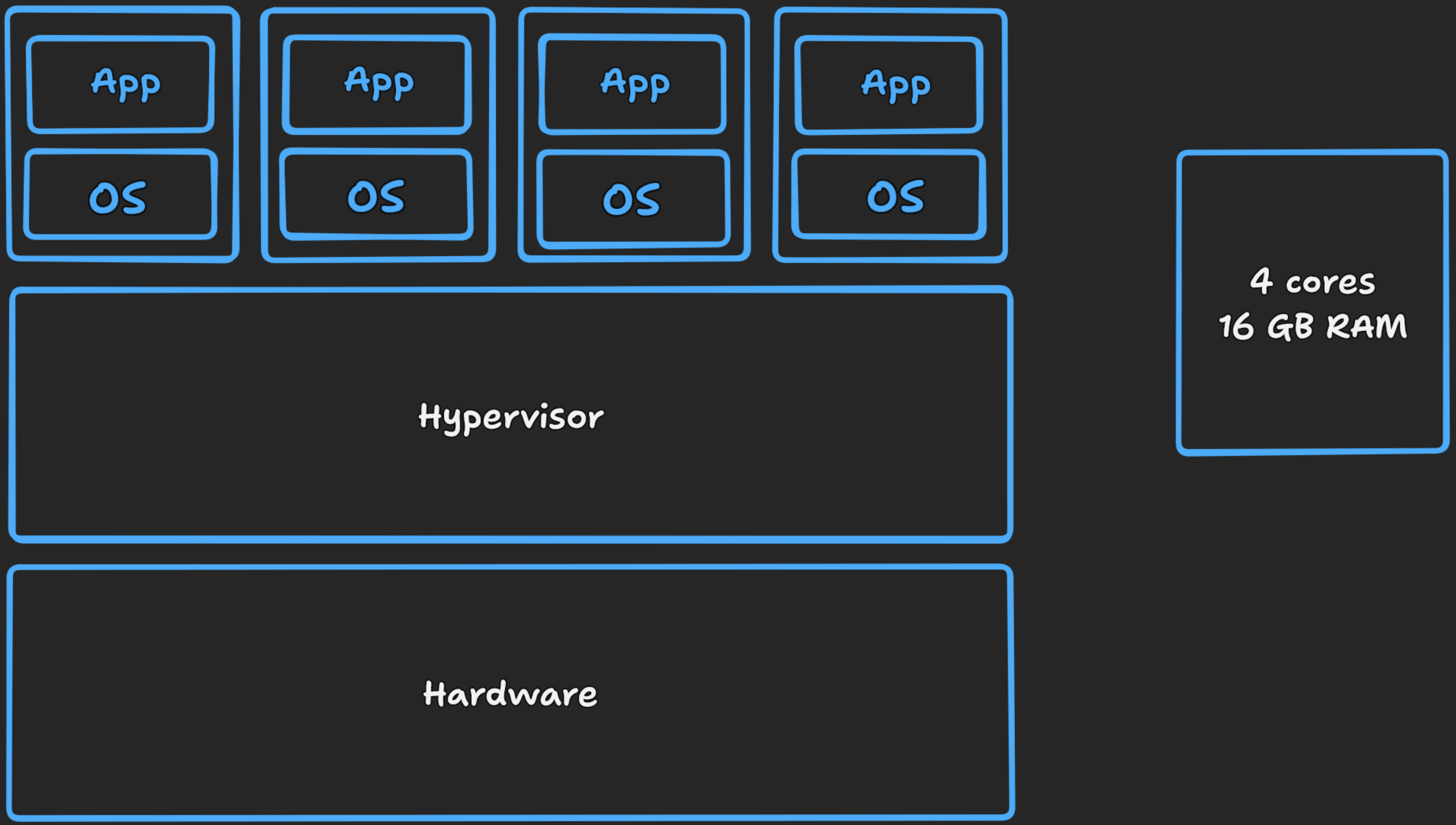
Portability

Reproducible
aka
Write once, run anywhere



HOW DO WE SOLVE THIS?
CONTAINERIZATION, RIGHT?
CONTAINERIZATION, RIGHT?
imgflip.com

Virtualization



App App App App

OS OS OS OS

Hypervisor

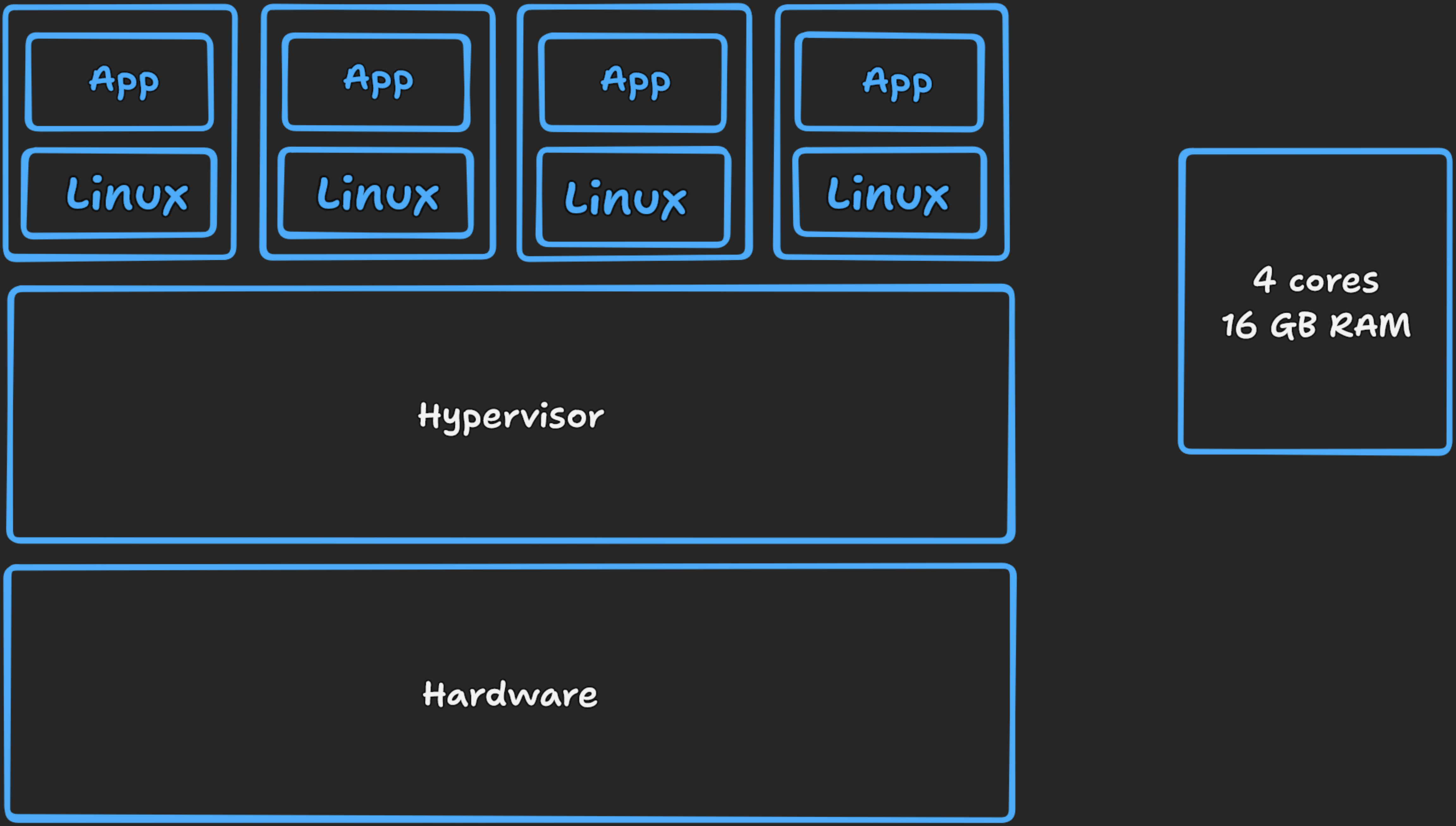
Hardware

4 cores
16 GB RAM

aws
vmware®
DigitalOcean

Pros

- Strong Isolation
- Strong Security
- Scalable
- Flexible
- Portability
- Resource Optimization
- Cost savings
- Do more with less



App

Linux

App

Linux

App

Linux

App

Linux

Hypervisor

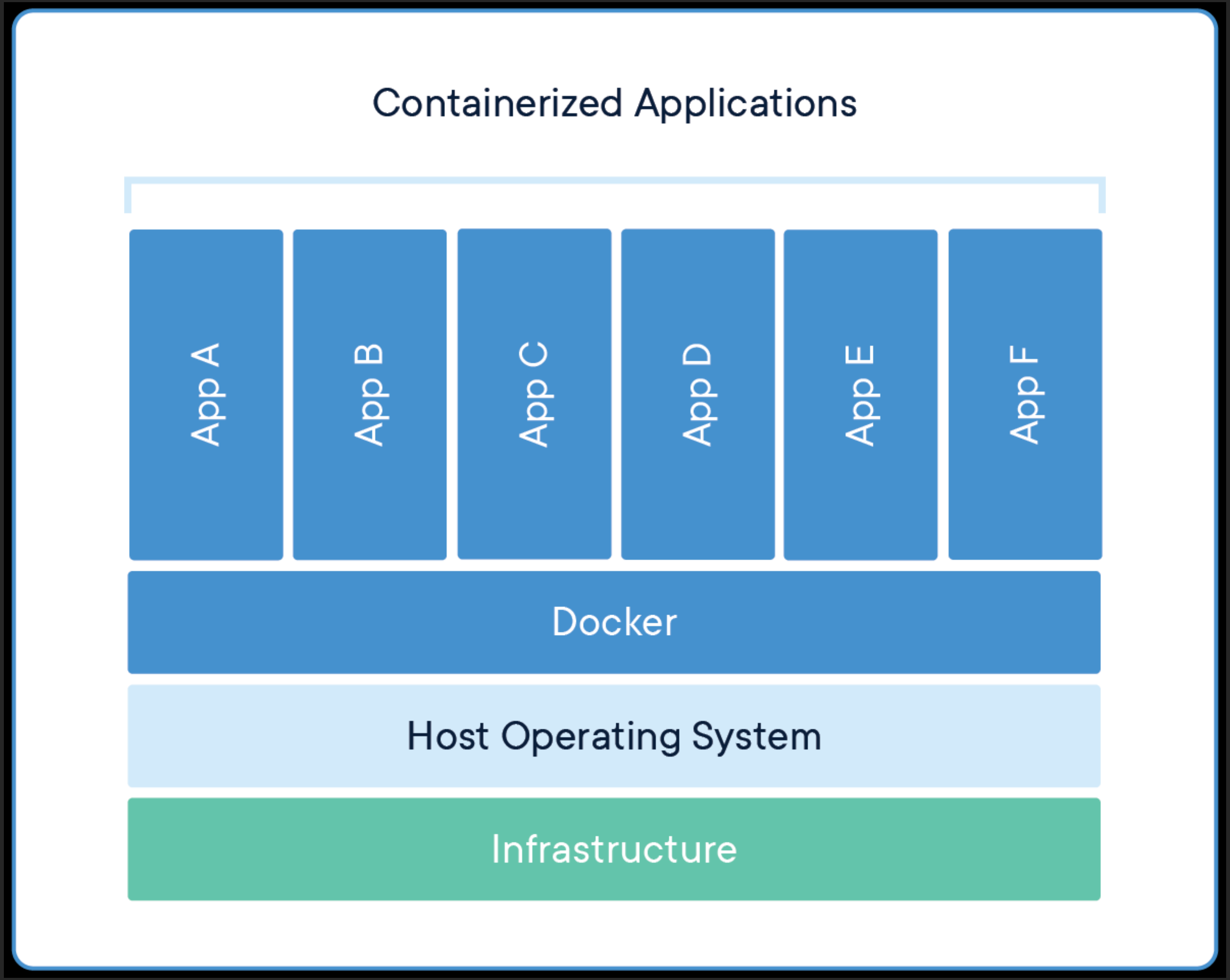
Hardware

4 cores
16 GB RAM

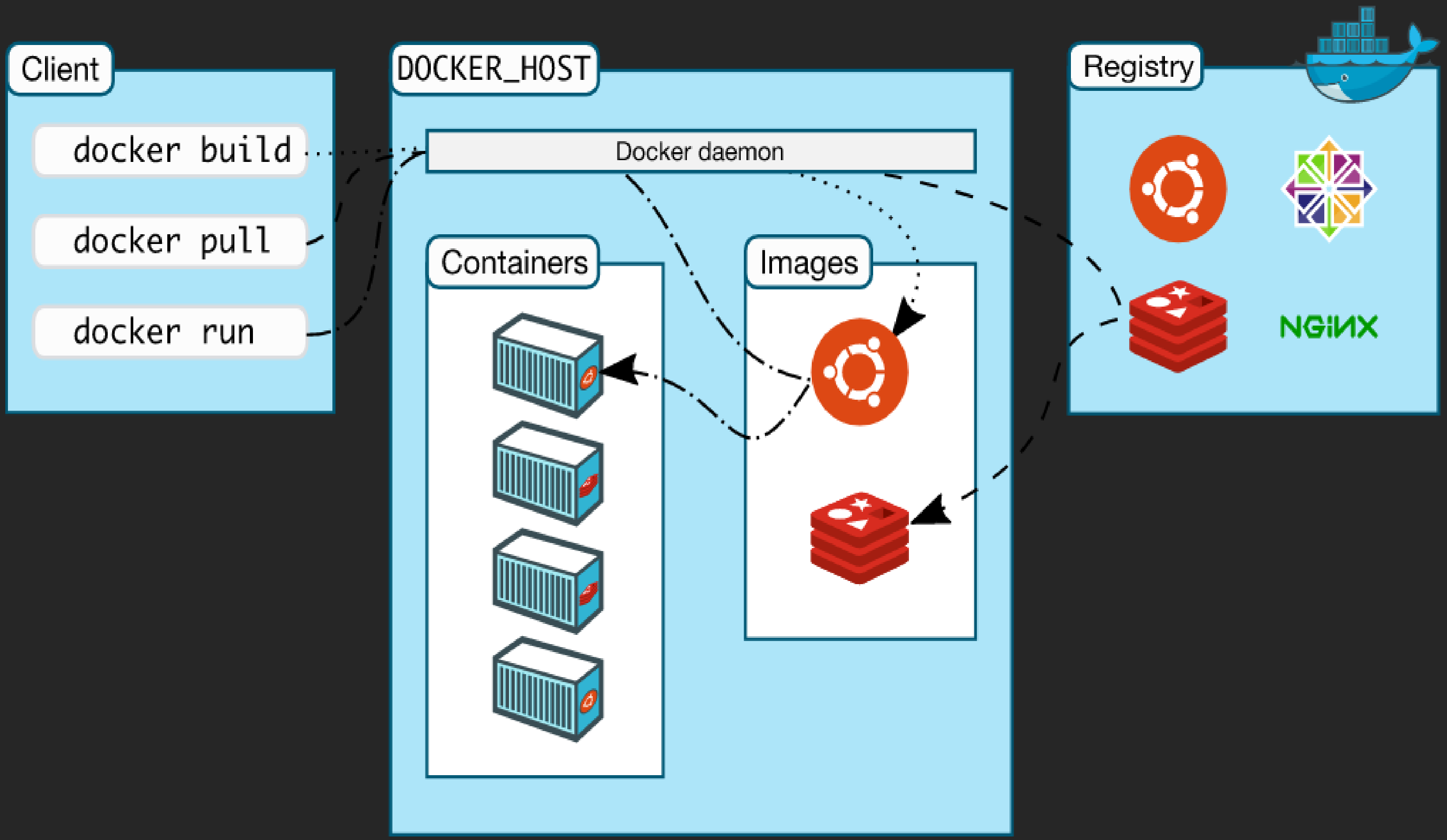
Cons

- High Resource Overhead
- Worse Performance compared to native
- Slow to boot
- Limited Portability
- Cannot Rapidly Scale

Docker


Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure

Key Components



Daemon

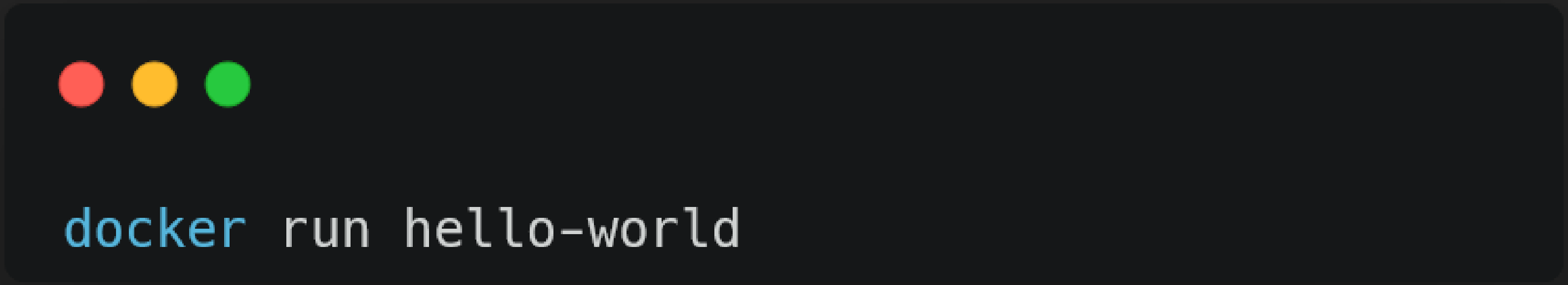
Image

Container

Registry

Docker run

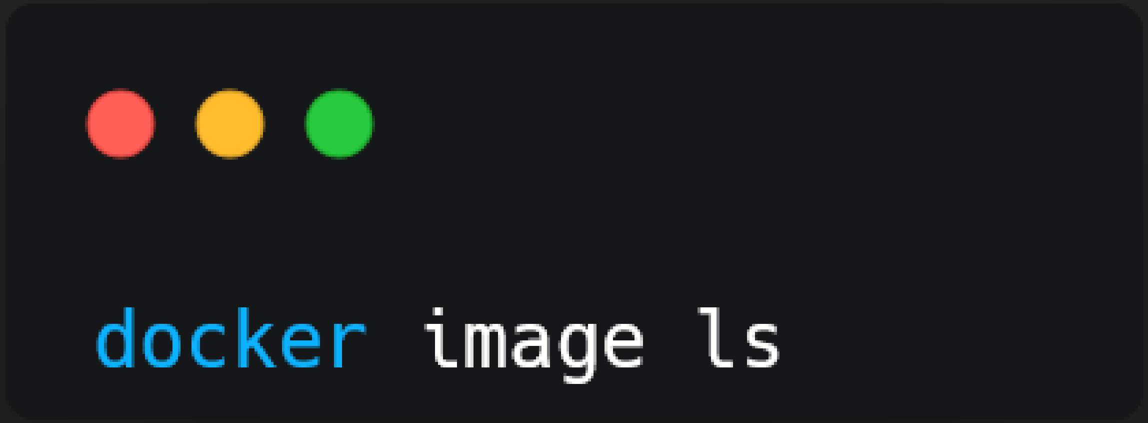
Creates a container from an image and starts it



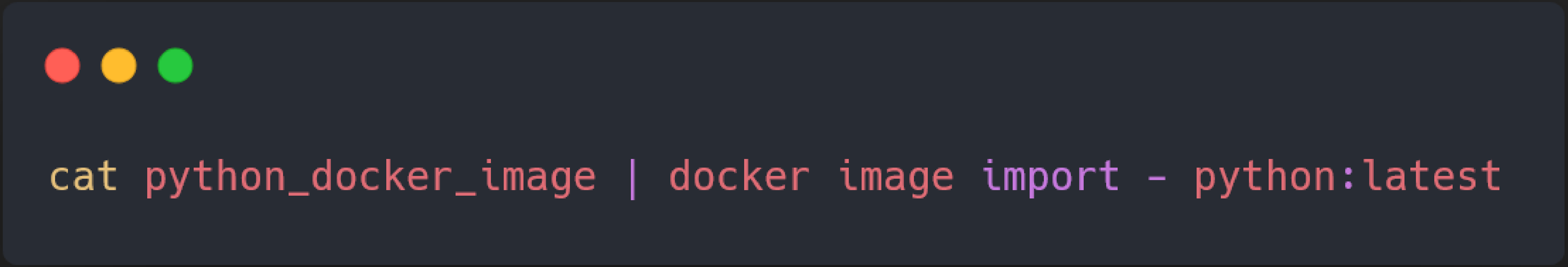
```
docker run hello-world
```

Docker image

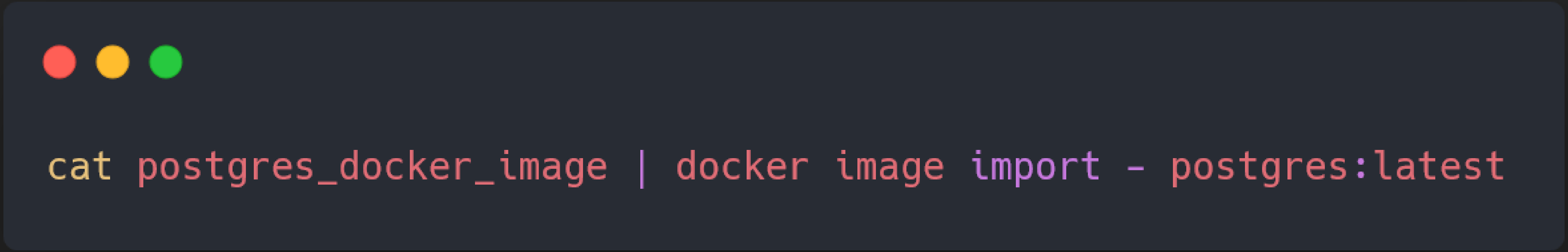
Get image information



```
docker image ls
```

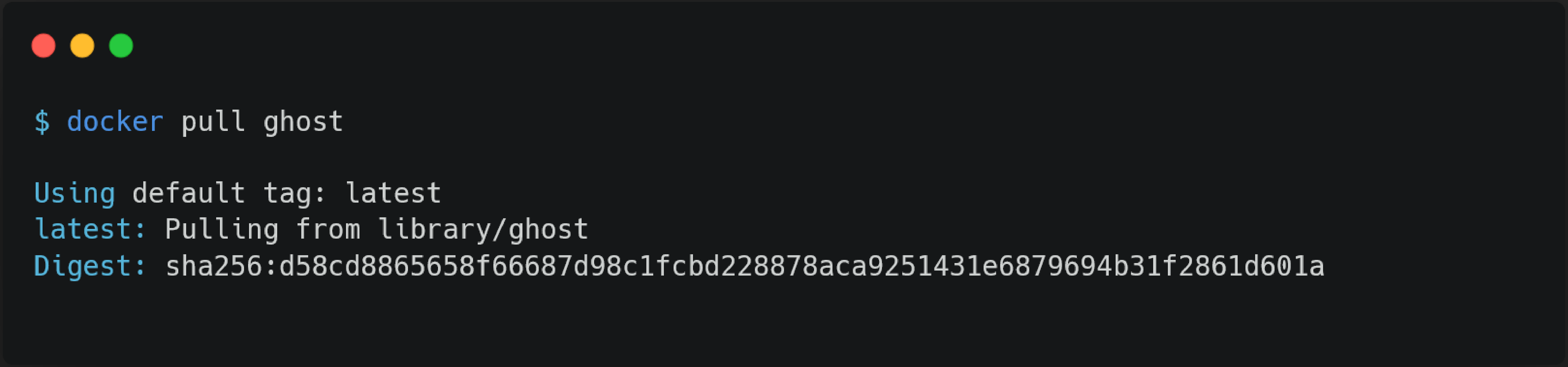


```
cat python_docker_image | docker image import - python:latest
```



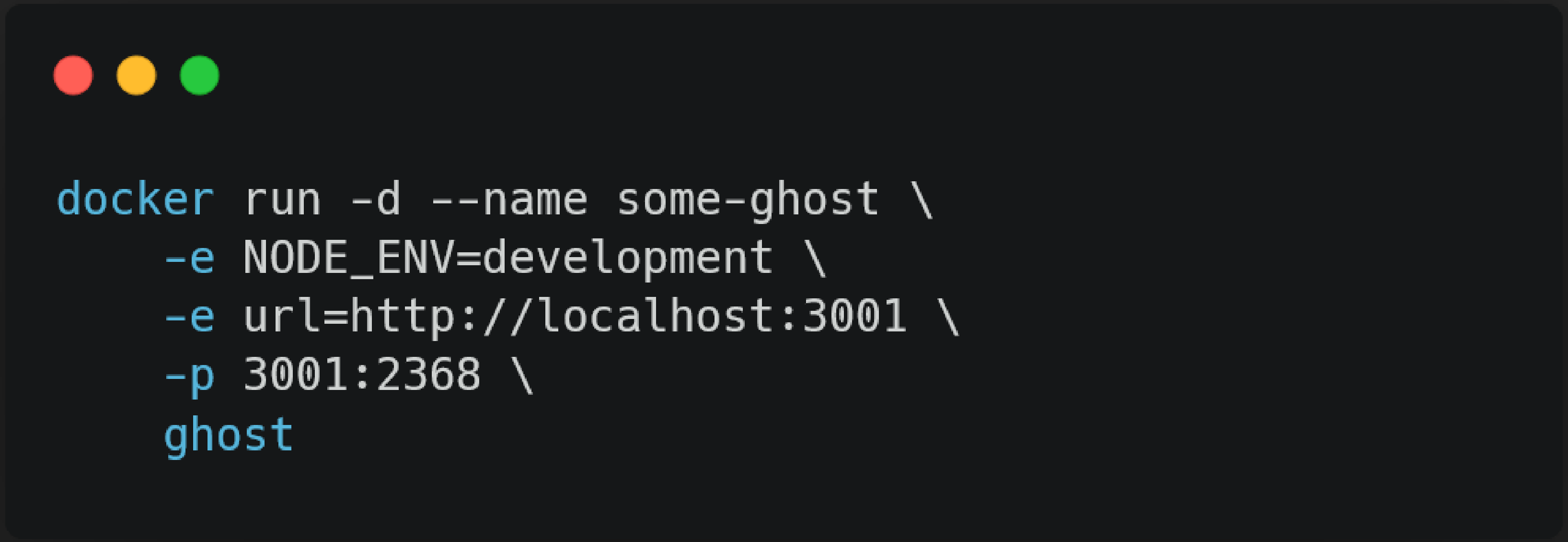
```
cat postgres_docker_image | docker image import - postgres:latest
```

Docker pull



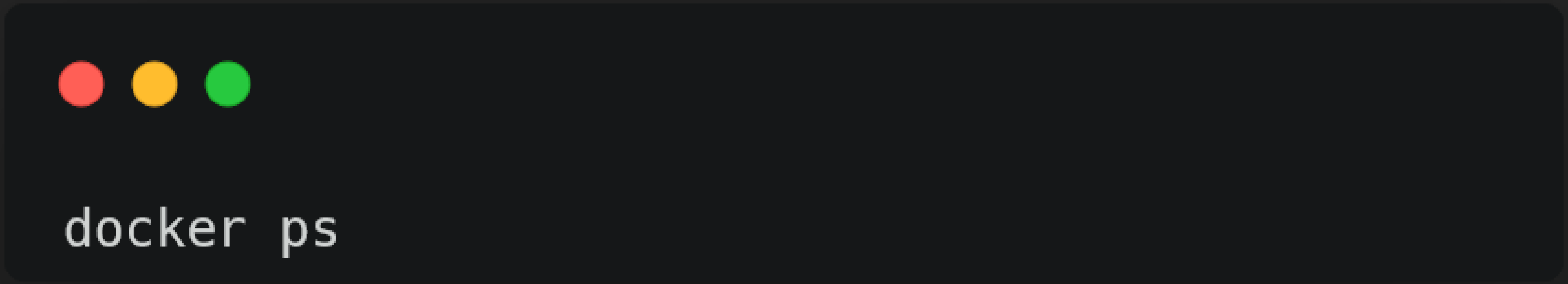
```
$ docker pull ghost

Using default tag: latest
latest: Pulling from library/ghost
Digest: sha256:d58cd8865658f66687d98c1fcbd228878aca9251431e6879694b31f2861d601a
```

Example



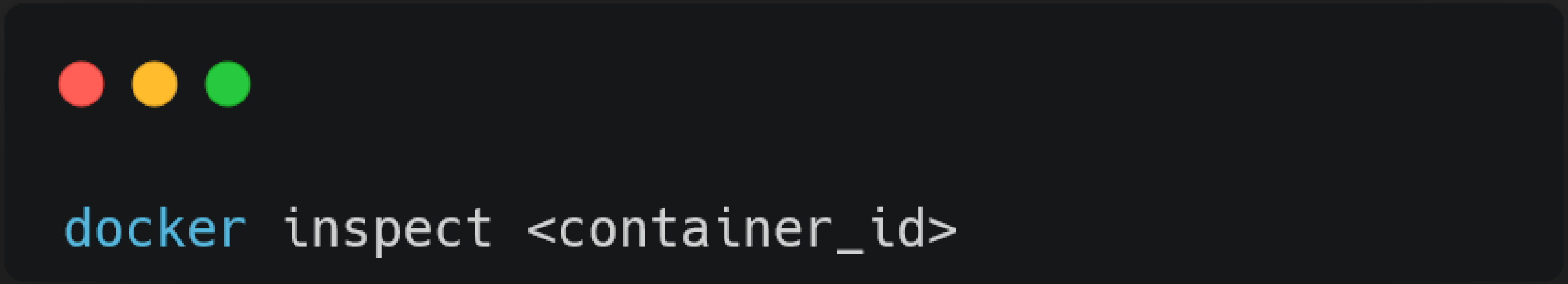
```
docker run -d --name some-ghost \
    -e NODE_ENV=development \
    -e url=http://localhost:3001 \
    -p 3001:2368 \
    ghost
```

Docker container list



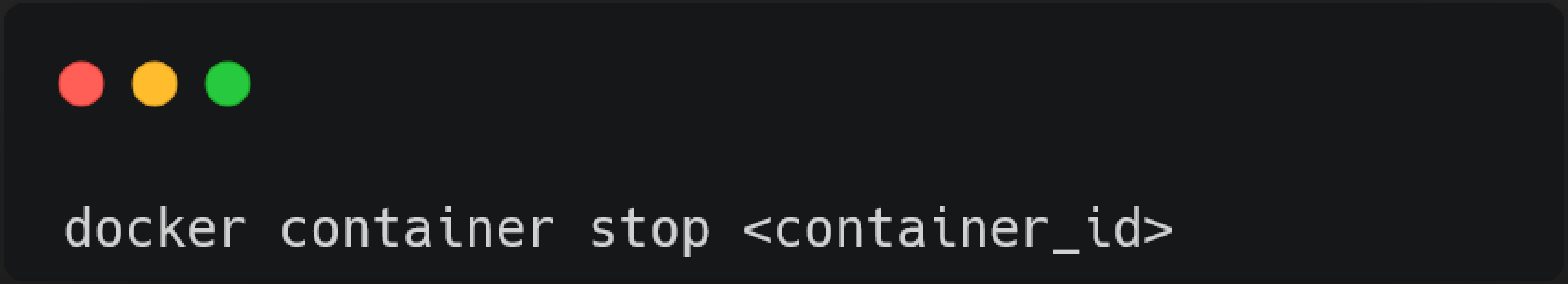
```
docker ps
```

Docker inspect

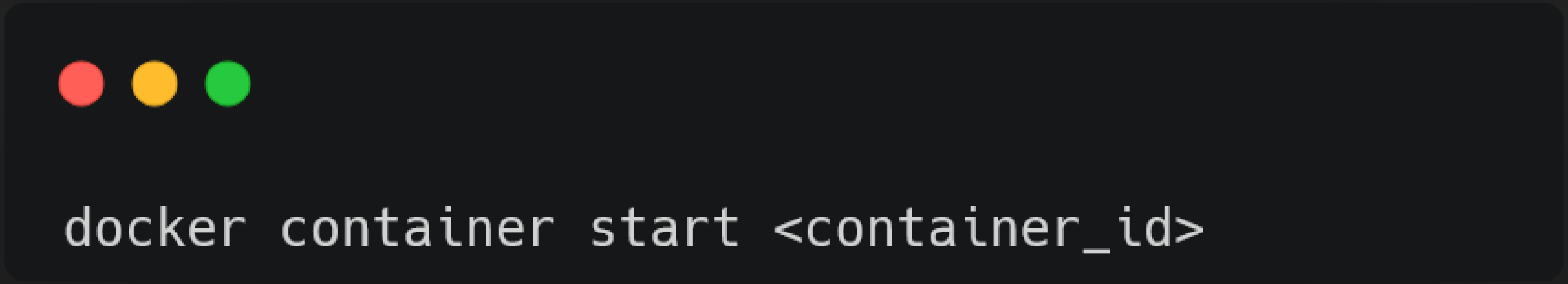


```
docker inspect <container_id>
```

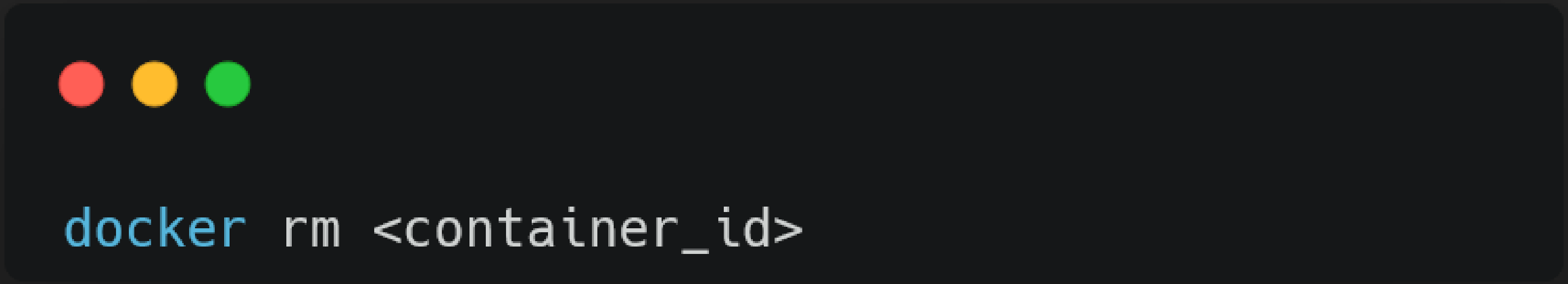
Docker container



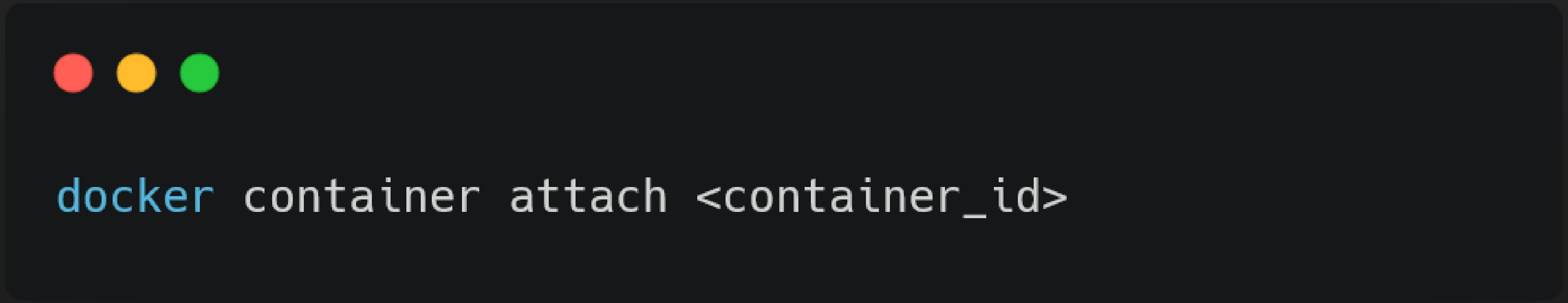
```
docker container stop <container_id>
```



```
docker container start <container_id>
```

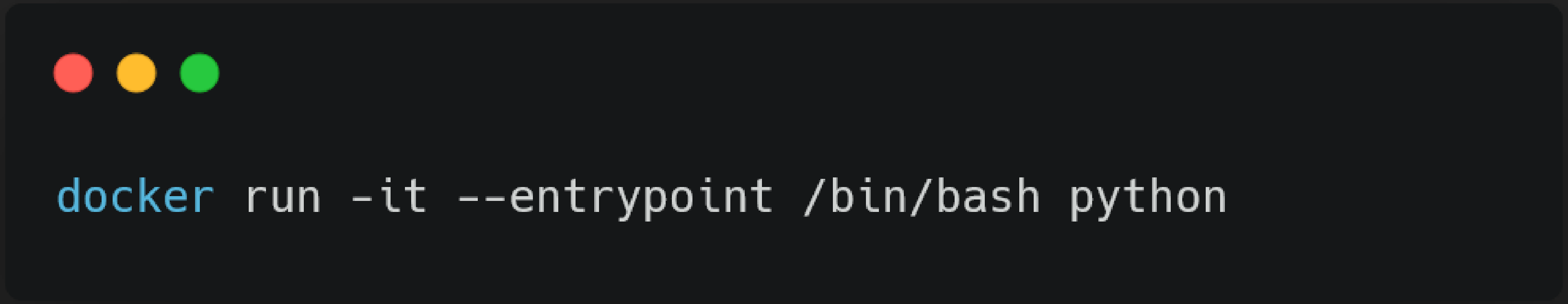


```
docker rm <container_id>
```



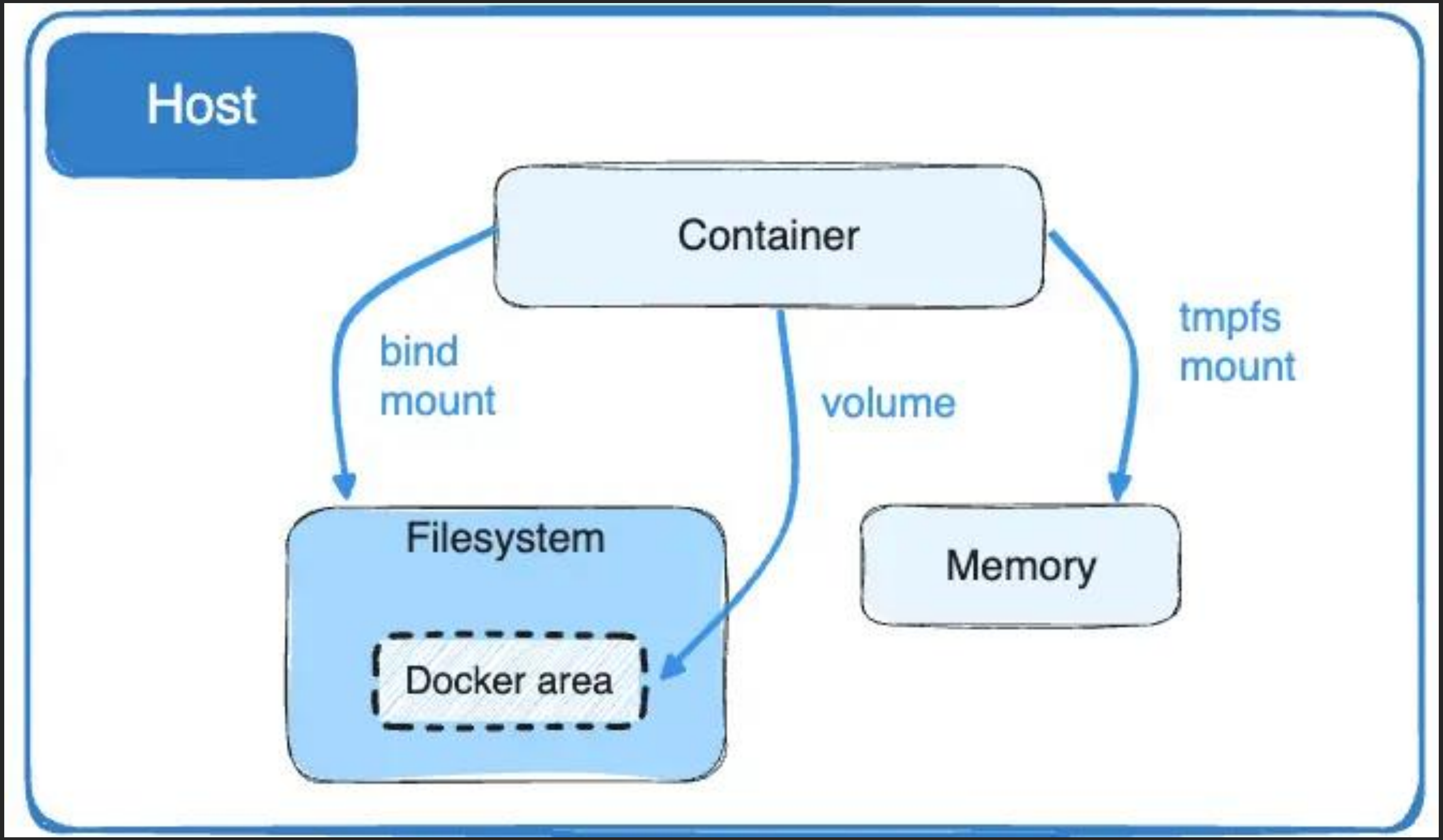
```
docker container attach <container_id>
```

Connect to a container

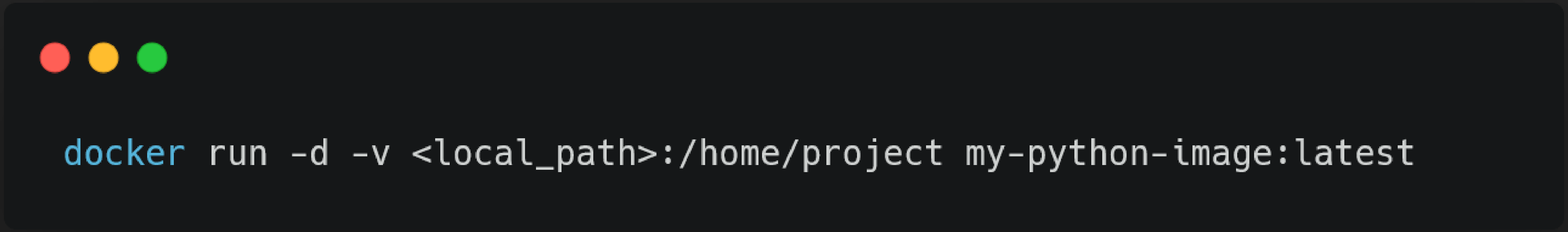


```
docker run -it --entrypoint /bin/bash python
```

Persist Data

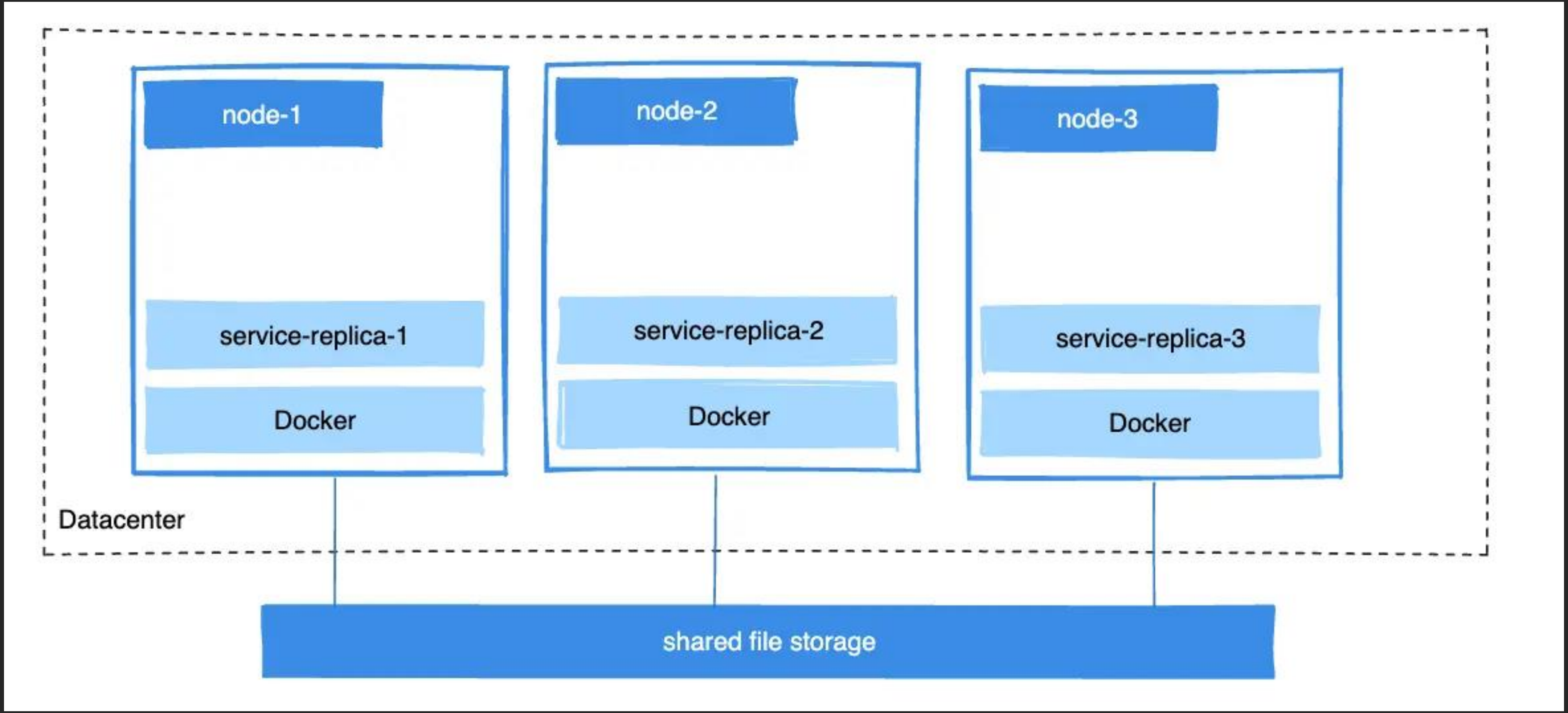


bind mount



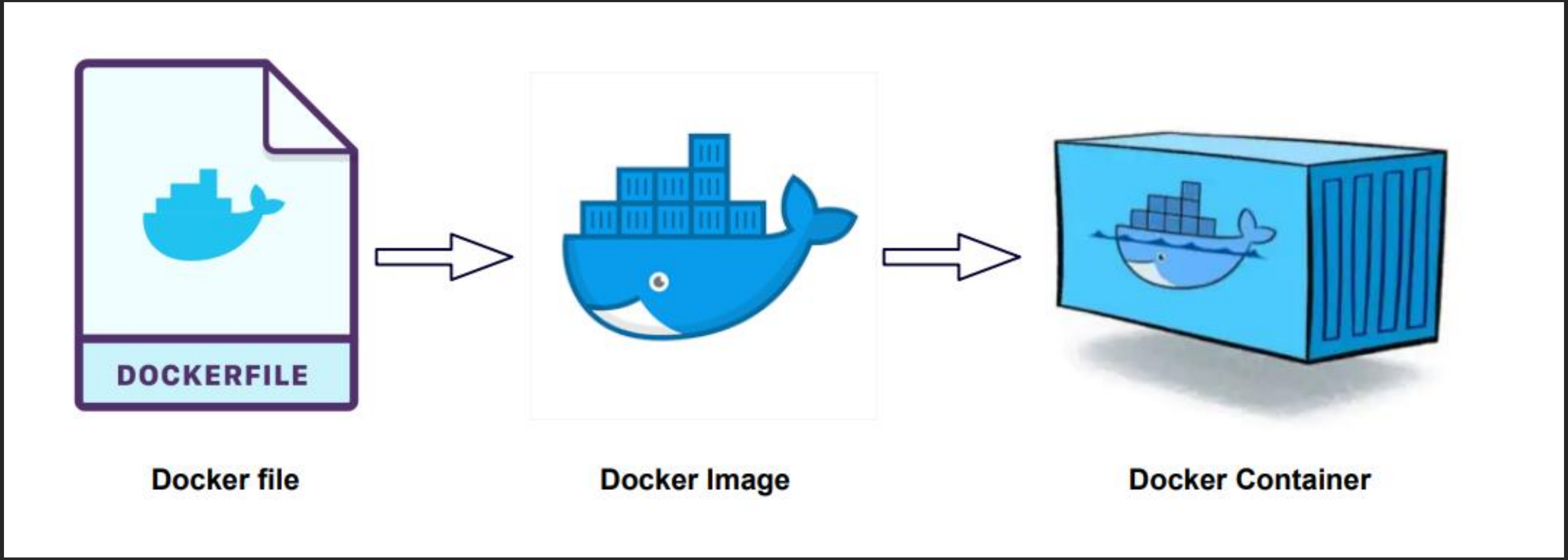
```
docker run -d -v <local_path>:/home/project my-python-image:latest
```

Docker volume

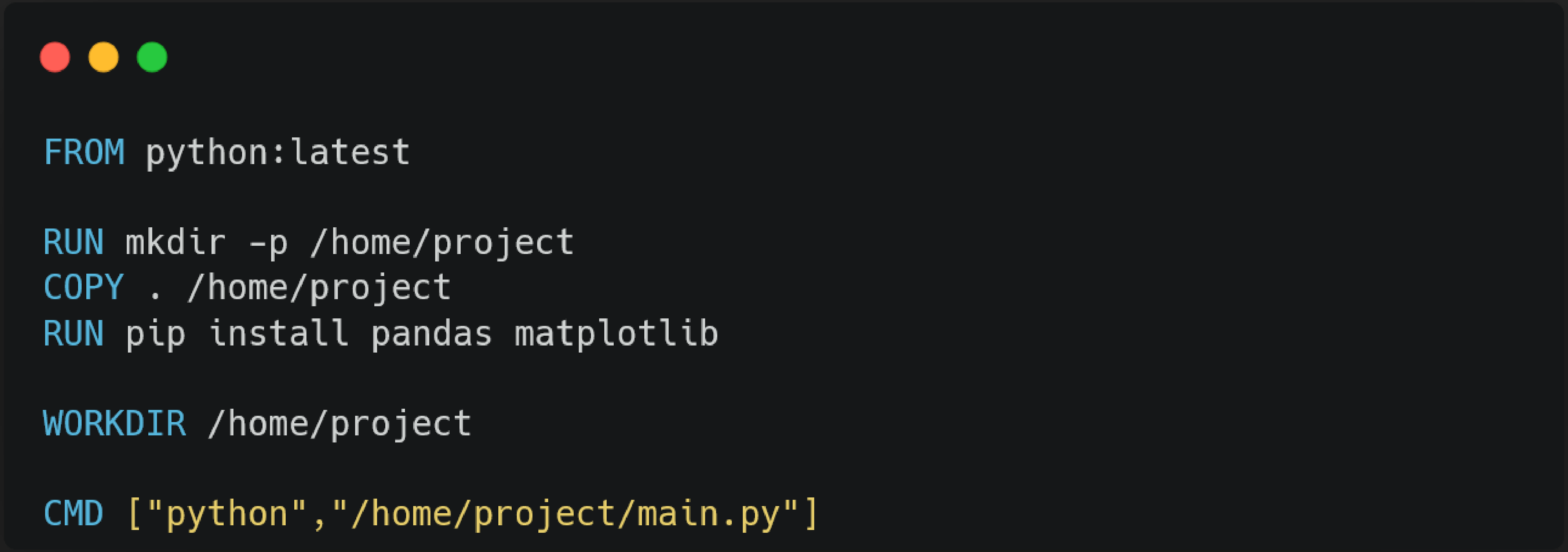


Bind Mount	Docker volume
Dependent on the host OS file system	Completely managed by docker
Harder to backup	Easier to backup
Worse performance on windows and MacOS	Higher performance on windows and macOS
Cannot be stored on a remote host	store volumes on remote hosts or cloud providers, encrypt the contents of volumes, or add other functionality
Cannot be pre-populated by the container	New volumes can have their content pre-populated by a container

Creating our own containers

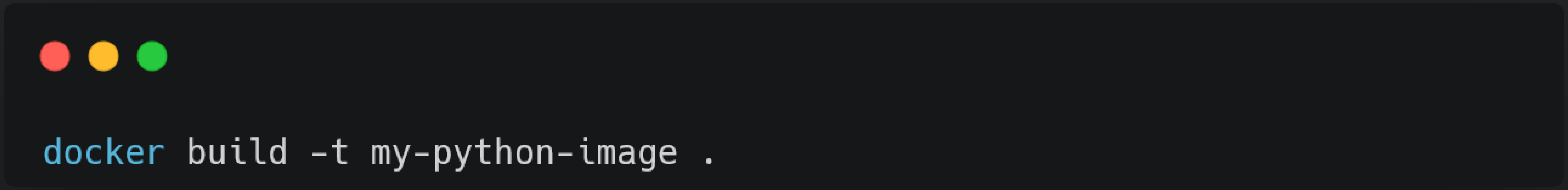


Your first Dockerfile



```
FROM python:latest

RUN mkdir -p /home/project
COPY . /home/project
RUN pip install pandas matplotlib

WORKDIR /home/project

CMD ["python","/home/project/main.py"]
```

Docker build



```
docker build -t my-python-image .
```

Dockerfile syntax

Command	Description
ADD	Add local or remote files and directories.
ARG	Use build-time variables.
CMD	Specify default commands.
COPY	Copy files and directories.
ENTRYPOINT	Specify default executable.
ENV	Set environment variables.
EXPOSE	Describe which ports your application is listening on.
FROM	Create a new build stage from a base image.

Command	Description
HEALTHCHECK	Check a container's health on startup.
LABEL	Add metadata to an image.
MAINTAINER	Specify the author of an image.
ONBUILD	Specify instructions for when the image is used in a build.
RUN	Execute build commands.
SHELL	Set the default shell of an image.
STOPSIGNAL	Specify the system call signal for exiting a container.
USER	Set user and group ID.
VOLUME	Create volume mounts.
WORKDIR	Change working directory.

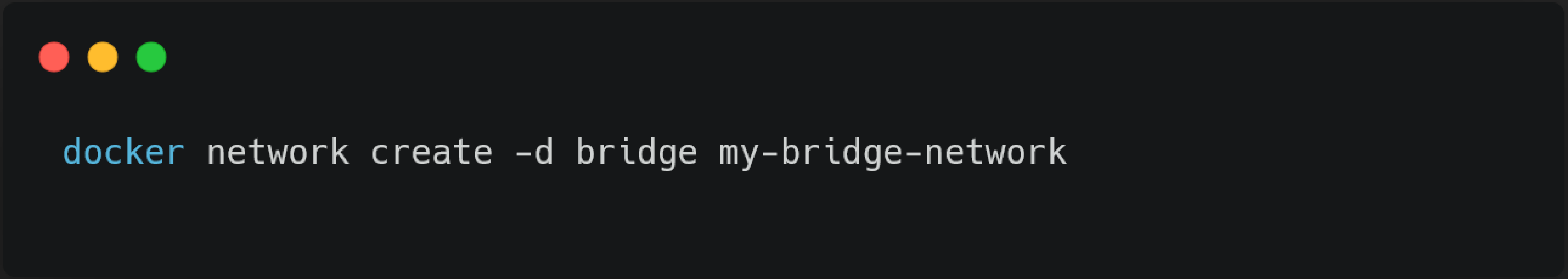
Docker network

Communicate with Containers

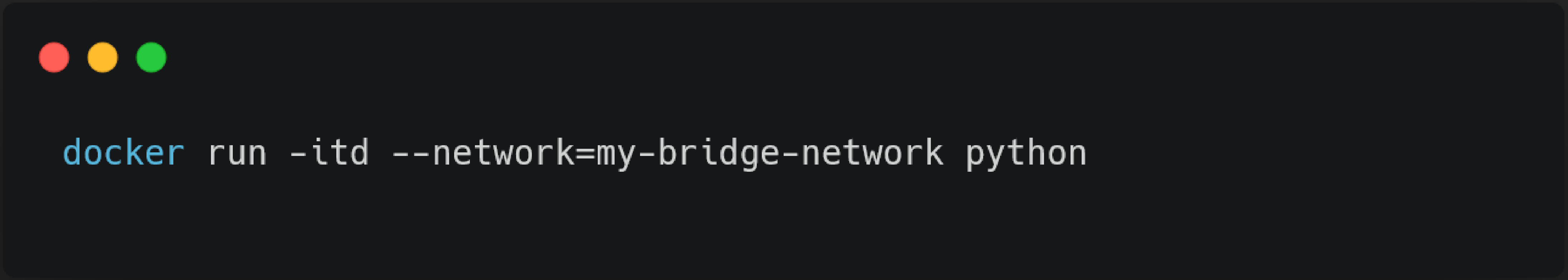
Communicate outside

Types

Type	Description
Bridge	The default network driver. Bridge networks let containers talk to each other on the same host
Host	Removes network isolation from between container and the host machine
None	Completely isolate the container

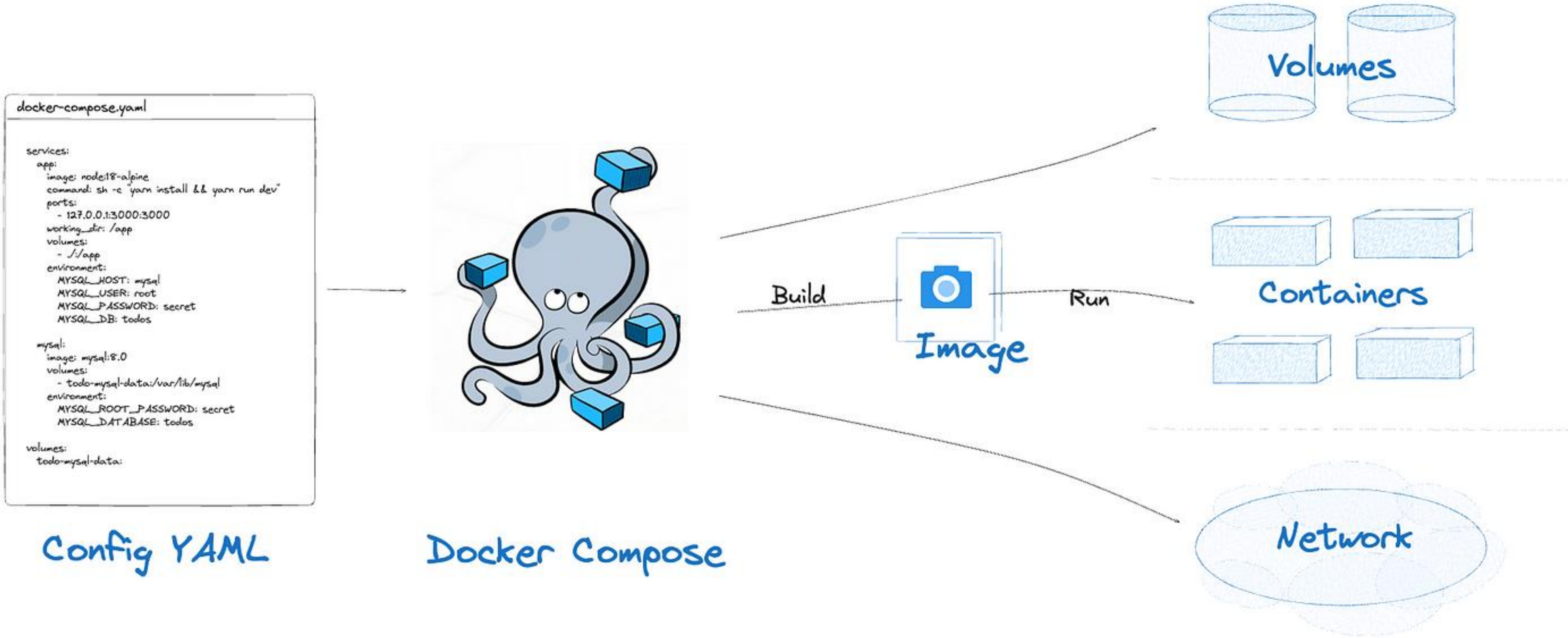


```
docker network create -d bridge my-bridge-network
```



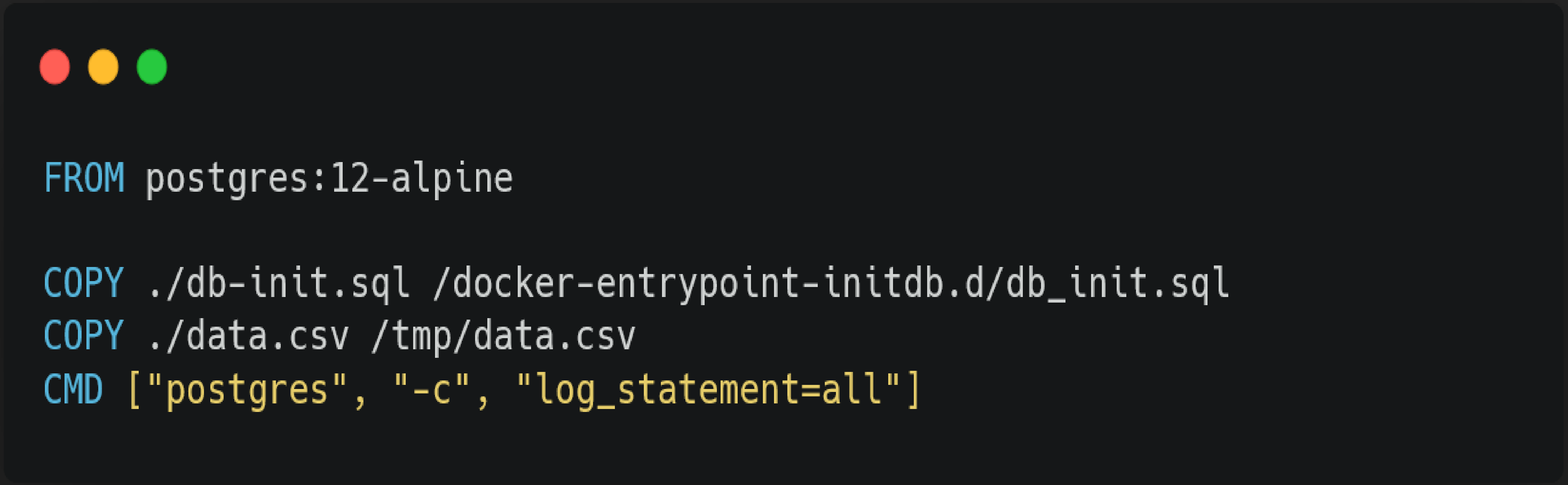
```
docker run -itd --network=my-bridge-network python
```

Docker compose

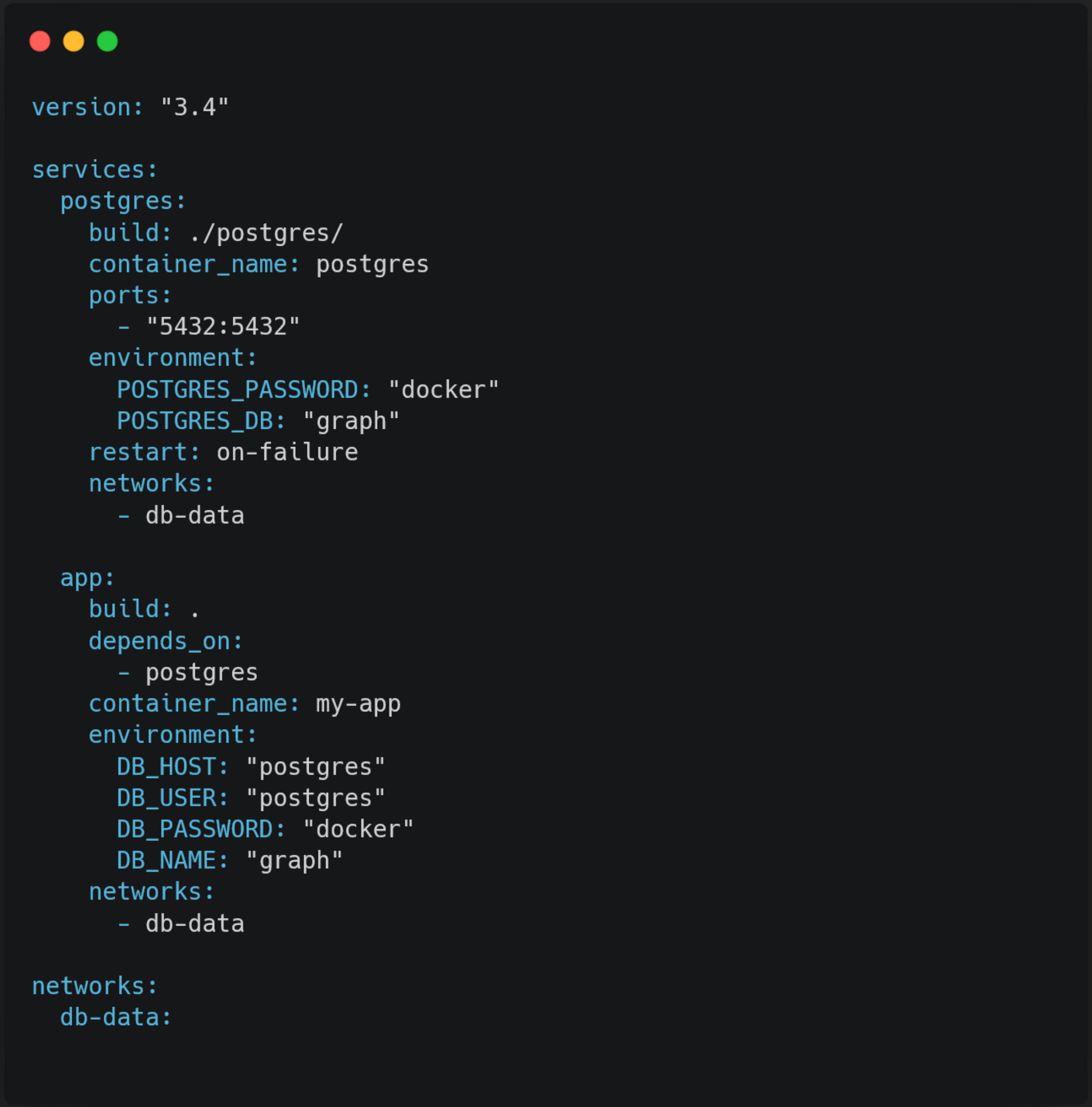


docker-compose.yaml

```yaml
services:
  app:
    image: node:18-alpine
    command: sh -c "yarn install && yarn run dev"
    ports:
      - 127.0.0.1:3000:3000
    working_dir: /app
    volumes:
      - ./:/app
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: root
      MYSQL_PASSWORD: secret
      MYSQL_DB: todos

  mysql:
    image: mysql:8.0
    volumes:
      - todo-mysql-data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: todos

volumes:
  todo-mysql-data:
```

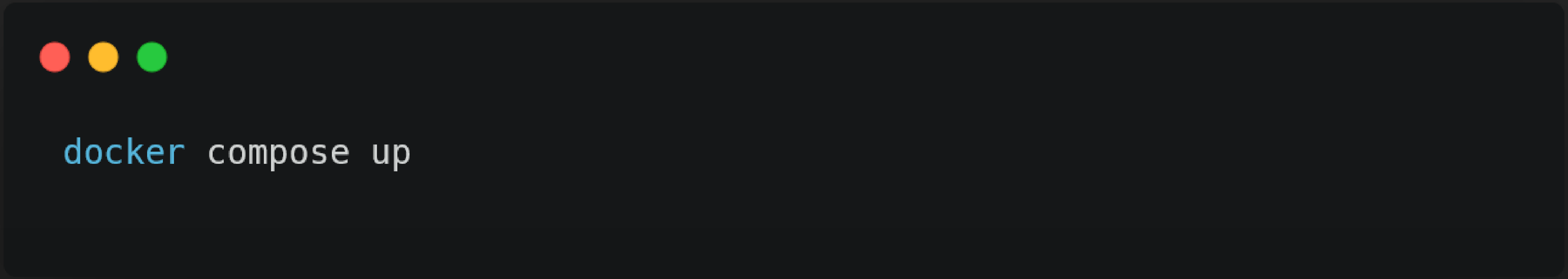


```
FROM postgres:12-alpine

COPY ./db-init.sql /docker-entrypoint-initdb.d/db_init.sql
COPY ./data.csv /tmp/data.csv
CMD ["postgres", "-c", "log_statement=all"]
```

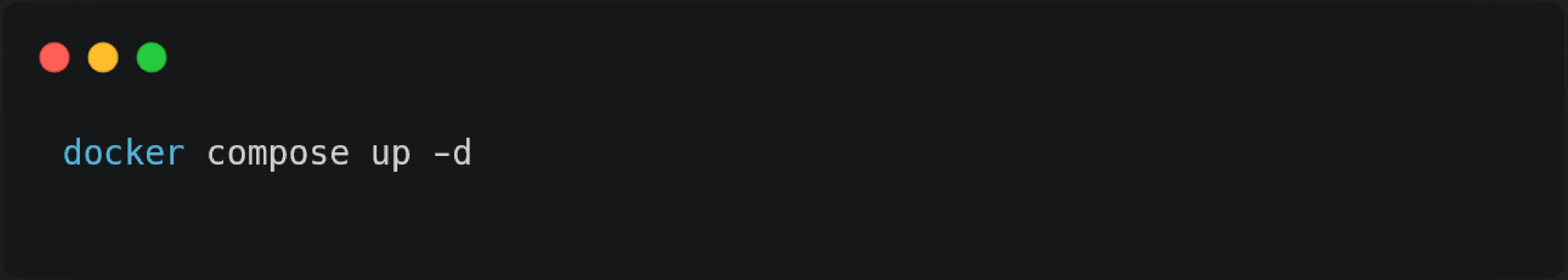


```
version: "3.4"

services:
  postgres:
    build: ./postgres/
    container_name: postgres
    ports:
      - "5432:5432"
    environment:
      POSTGRES_PASSWORD: "docker"
      POSTGRES_DB: "graph"
    restart: on-failure
    networks:
      - db-data

  app:
    build: .
    depends_on:
      - postgres
    container_name: my-app
    environment:
      DB_HOST: "postgres"
      DB_USER: "postgres"
      DB_PASSWORD: "docker"
      DB_NAME: "graph"
    networks:
      - db-data

networks:
  db-data:
```

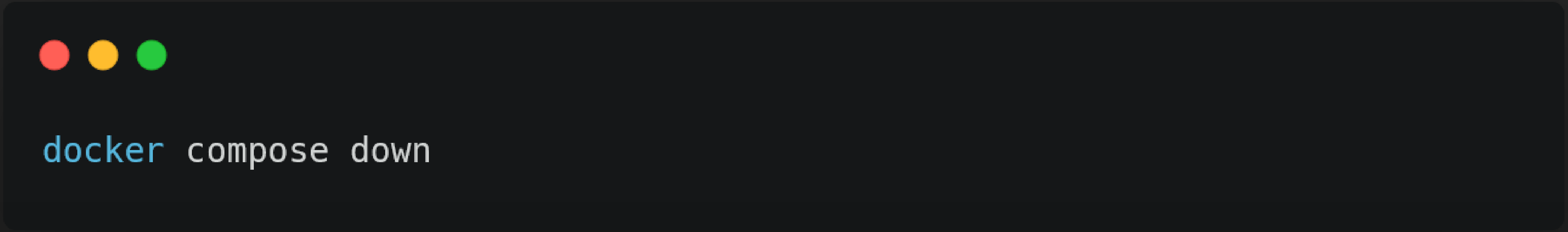
Command



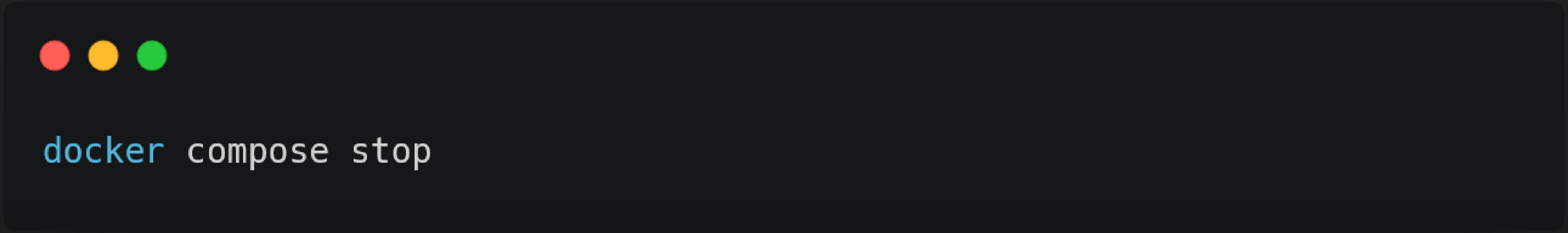
```
docker compose up
```



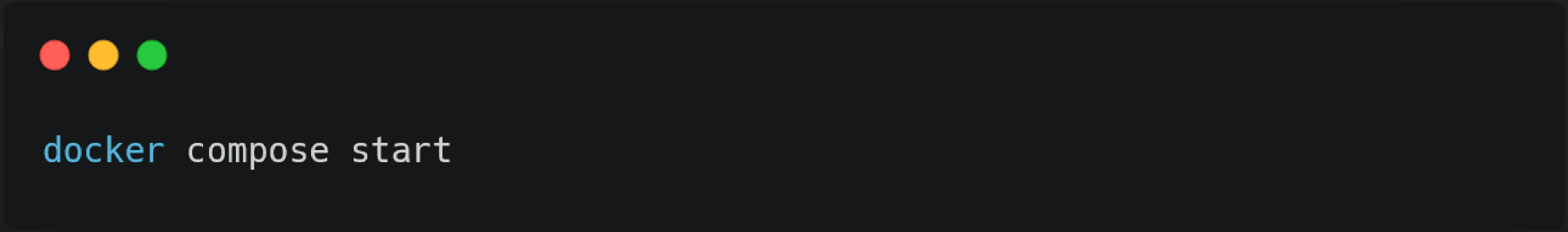
```
docker compose up -d
```



```
docker compose down
```



```
docker compose stop
```



```
docker compose start
```

What is a container?

A Standard Unit of Software

An Isolated Process

“Magic” behind containers

- Chroot
- Linux namespaces
- cgroups
- LXC, containerd

What is an image?

A collection of changes to a file system, affecting -

- Files
- Binaries
- Libraries
- configurations

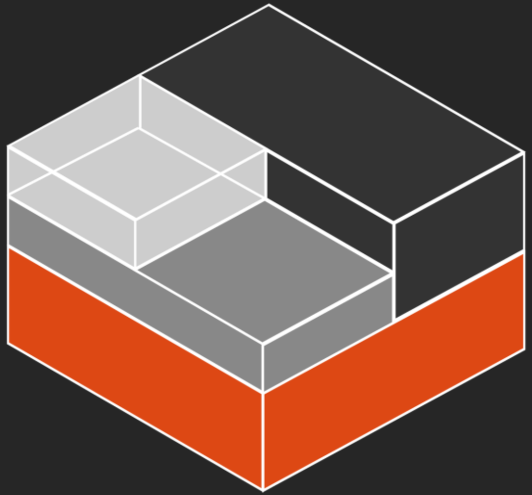
Principles of an image

- Images are immutable
- Images are composed of layers
- Each layer represents fs change

Virtualization vs Containerization

Aspect	Virtualization	Containerization
Resource Usage	High overhead due to multiple OS instances	Low overhead; shares host OS kernel
Startup Time	Slower (minutes)	Faster (milliseconds)
Isolation Level	Strong isolation between VMs	Weaker isolation; shares OS kernel
Portability	Limited portability across platforms	Highly portable across different systems
Use Cases	Legacy apps, multi-tenant environments	Microservices, cloud-native applications

Docker Alternatives



Learn More

https://medium.com/@saschagrunert/demystifying-containers-part-i-kernel-space-2c53d6979504

https://medium.com/@saschagrunert/demystifying-containers-part-ii-container-runtimes-e363aa378f25

https://medium.com/@saschagrunert/demystifying-containers-part-iii-container-images-244865de6fef