Containerization

Docker - Podman - Compose - k8s

Abstracting a "computer" environment

A computer running on your current computer

The concept of containerization sometimes has to "click"

- It works on my computer
- Yes, but we are not going to give your computer to the client
docker

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

How to abstract an environment

VirtualBox vs Docker

Kernel space vs user space

Kernel = hardware abstraction / file system / process management

User space = GNU stuff - bash and what not

Docker virtualizes the user space

Docker uses the host kernel!

Docker is somewhat "light-weight"

cgroups - namespaces - filesystem isolation

Bocker: Docker in 100 lines of bash

Docker in 100 lines of Go

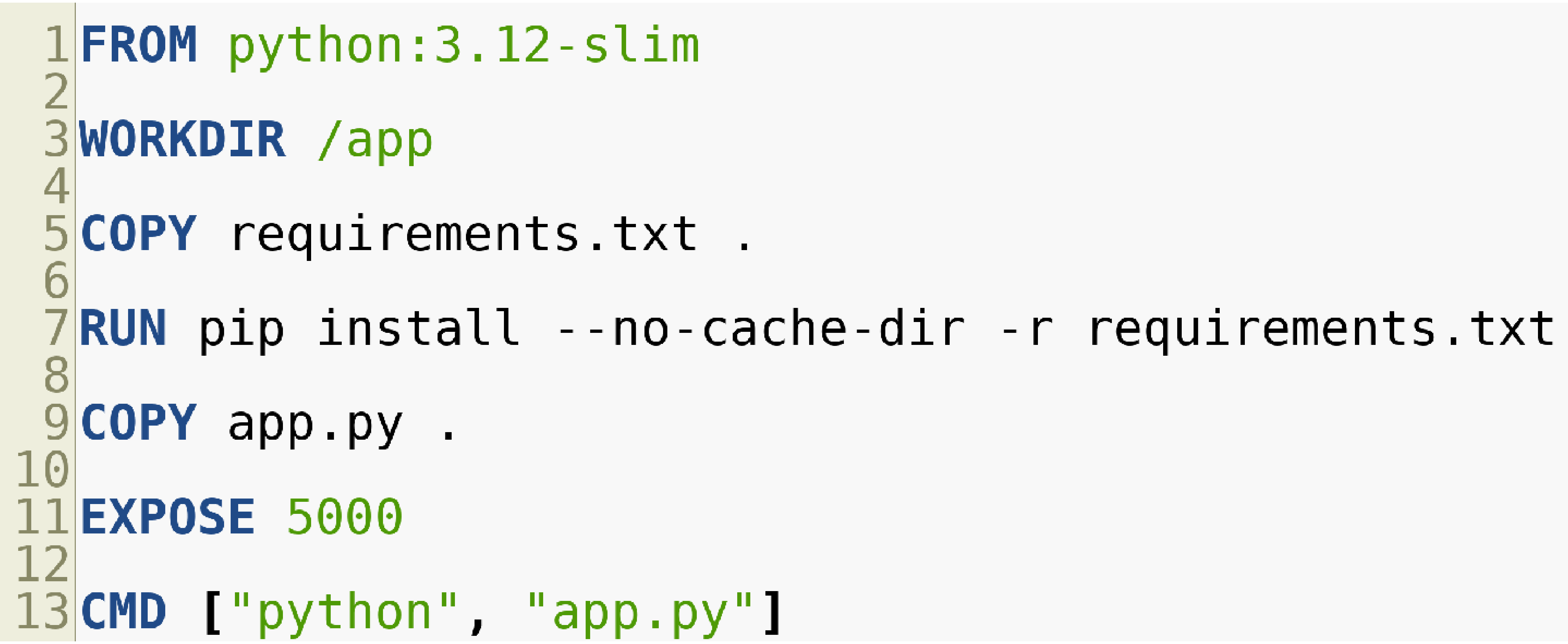
Image vs Container

Image: Container template

Container: A runnable instance

Dockerfile -> docker build -> Image -> docker run -> container

from Dockerfile to container - simple example



```
FROM python:3.12-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

COPY app.py .

EXPOSE 5000

CMD ["python", "app.py"]
```

```
# Build image
docker build -t mypythonapp .  # search for file Dockerfile

# Run container
docker run -d -p 5000:5000 mypythonapp
docker ps  # list all running containers

# CONTAINER ID   IMAGE           COMMAND             CREATED        STATUS                     NAMES
# abc123def456   mypythonapp     "python app.py"     2 hours ago    Exited (0) 30 mins ago     myapp
# def789ghi012   ubuntu          "/bin/bash"         1 day ago      Up 5 hours                 loving_goldberg

# Compare with docker images that lists the downloaded images.

# ====

docker exec -it <container_id_or_name> bash

# docker attach is the alternative, but it attached to the running command (python app.py) usually not what you want
```

The docker philosphy

New images are build layered, building on top of existing images

A container is responsible for running a single process.

Layering is important!

Layers are cached

1. Large layers that infrequently change first

2. Reduce layers for speed

```dockerfile
FROM python:3.12-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

COPY app.py .

EXPOSE 5000

CMD ["python", "app.py"]
```

Even more slimming your container

Multi container build

Build dependencies in a different container

One container - one process

As an application consists of multiple process

Compose was born

(docker-)Compose

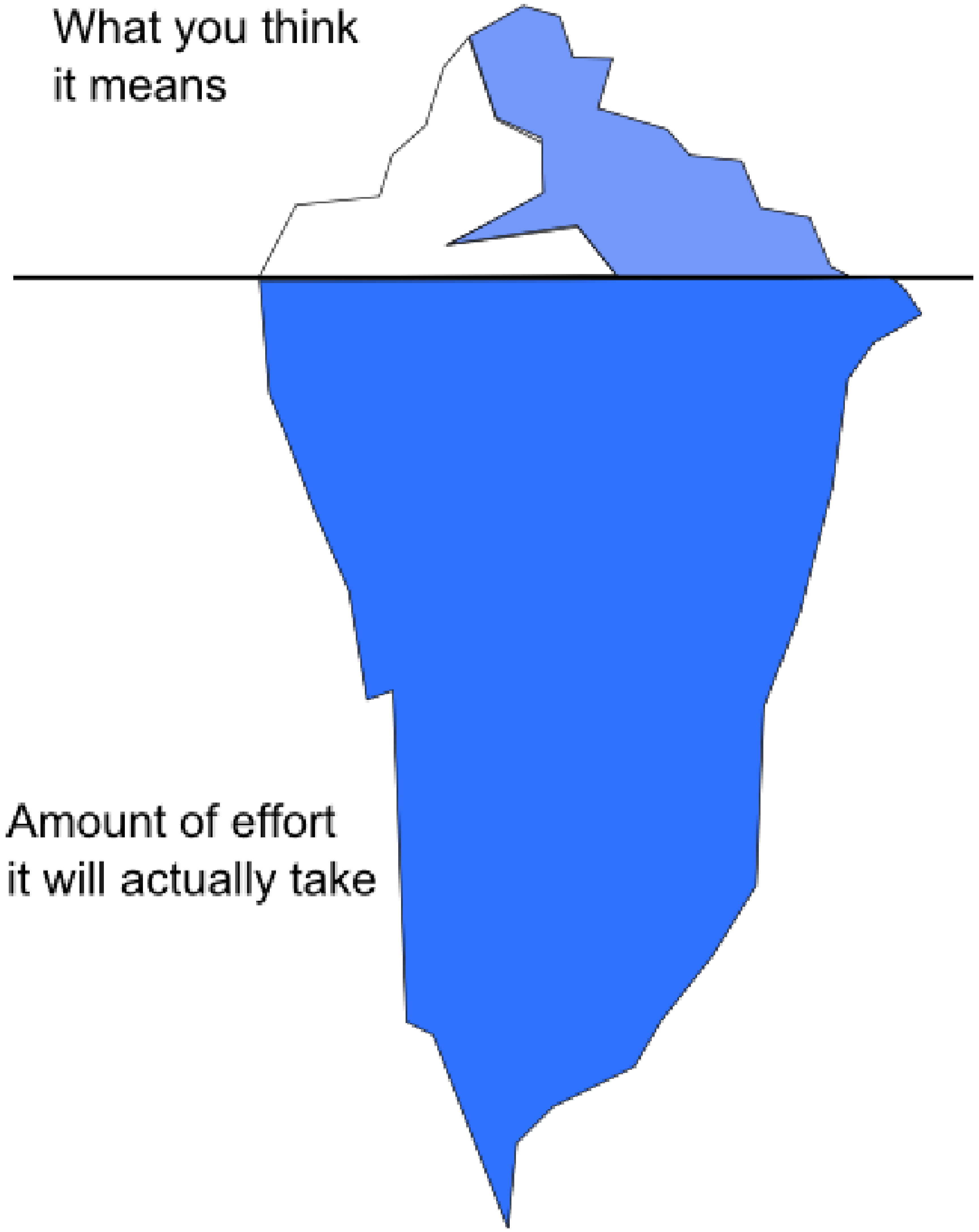
From a single yaml file creates all "low-level" commands

Includes an internal network

Which also can share volumes accross those internal networks.

```yaml
# docker-compose.yml
version: "3.9"  # deprecated

services:
  web:
    build: .
    container_name: mywebapp
    ports:
      - "5000:5000"
    volumes:
      - .:/app
    # from the .env file or env vars
    environment:
      - FLASK_ENV=${FLASK_ENV}
    depends_on:
      - db

  db:
    image: postgres:15
    container_name: mydb
    environment:
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_DB: ${POSTGRES_DB}
    volumes:
      - db_data:/var/lib/postgresql/data
    restart: unless-stopped

  # The database is now automatically available on db:5432 on the *internal compose network*

volumes:
  db_data:
```

Docker in Production



What you think
it means

Amount of effort
it will actually take

Exploring the iceberg

The init process

Host kernel problems

Docker vs Podman

Kubernetes (k8s)

OCI spec

Named volume vs bind mount

Using a named volume with podman rootless on NFS

The init process (pid1)

The first process is responsible for

* Signal handling

* Child reaping

Start script should have a exec

Some processes (Java) require --init (init: true) that sets tini as pid1

Host kernel problems

Kernel feature required

Kernel module (NVIDIA) required

Security - file system permissions are ""not"" abstracted

Docker vs Podman

Very similar in functional parity

Docker has a root background process

Podman treats containers as user processes -> rootless mode

Docker has the hub (but with pull limits)

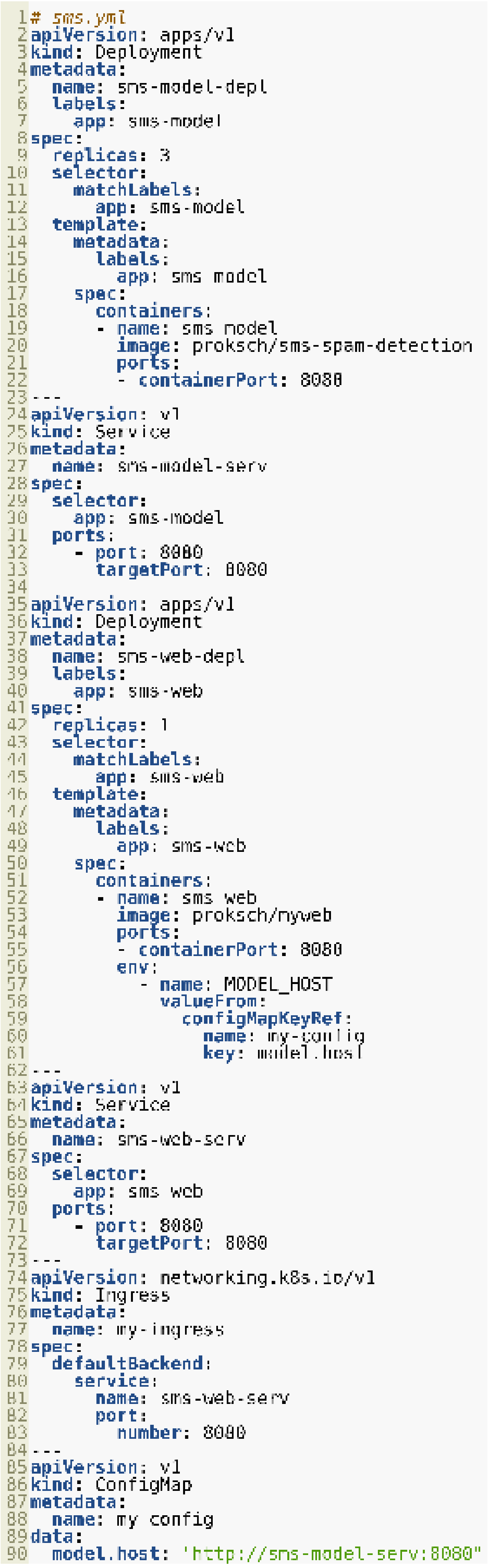
Podman is a RedHat initiative

Kubernetes (k8s)

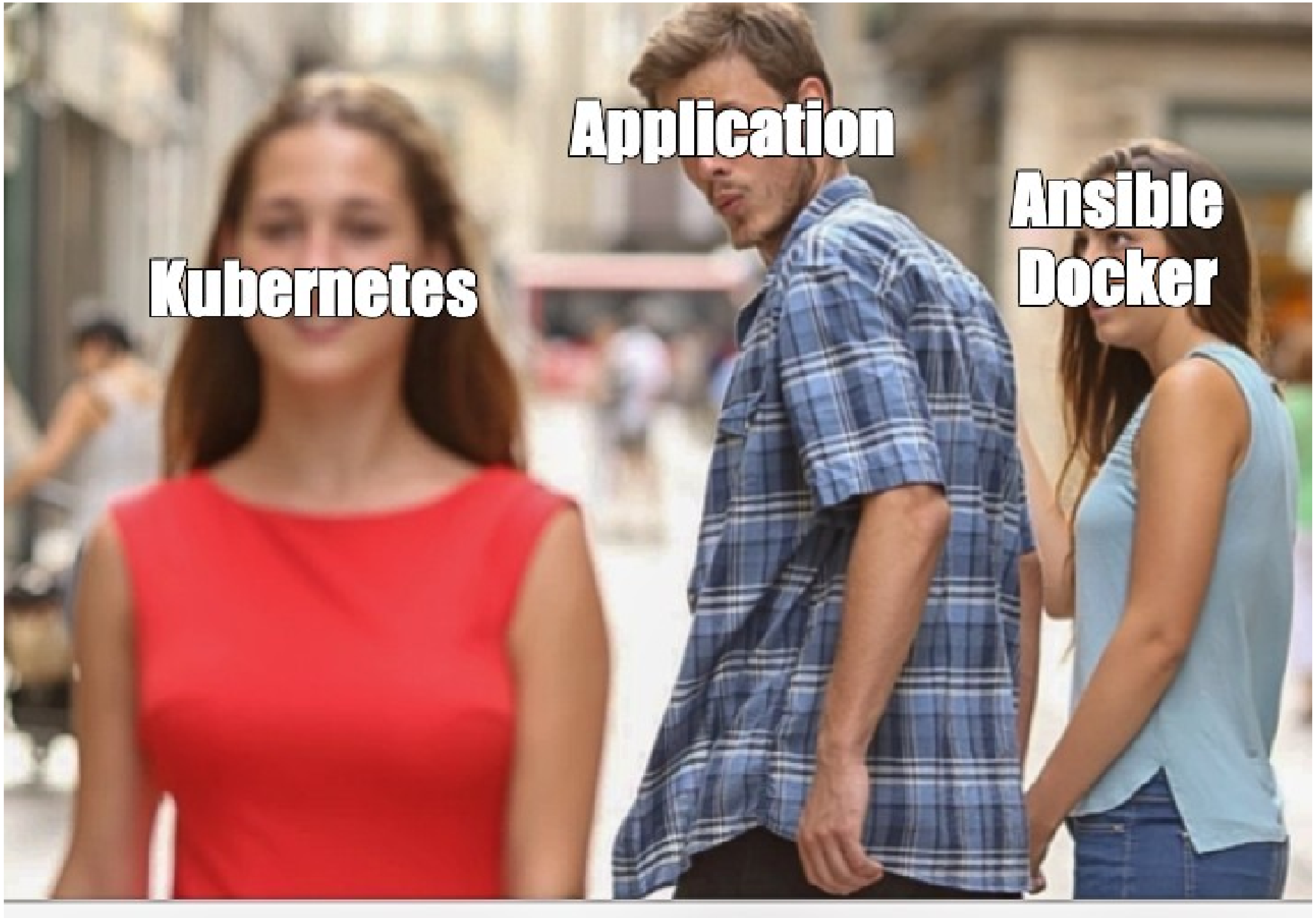
Every container runs on a seperate computer (node)

Can automatically add more computers when load increases

"You must be this big"



```yaml
# sms.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sms-model-depl
  labels:
    app: sms-model
spec:
  replicas: 3
  selector:
    matchLabels:
      app: sms-model
  template:
    metadata:
      labels:
        app: sms-model
    spec:
      containers:
      - name: sms-model
        image: proksch/sms-spam-detection
        ports:
        - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: sms-model-serv
spec:
  selector:
    app: sms-model
  ports:
    - port: 8080
      targetPort: 8080

apiVersion: apps/v1
kind: Deployment
metadata:
  name: sms-web-depl
  labels:
    app: sms-web
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sms-web
  template:
    metadata:
      labels:
        app: sms-web
    spec:
      containers:
      - name: sms-web
        image: proksch/myweb
        ports:
        - containerPort: 8080
        env:
          - name: MODEL_HOST
            valueFrom:
              configMapKeyRef:
                name: my-config
                key: model.host
---
apiVersion: v1
kind: Service
metadata:
  name: sms-web-serv
spec:
  selector:
    app: sms-web
  ports:
    - port: 8080
      targetPort: 8080
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
spec:
  defaultBackend:
    service:
      name: sms-web-serv
      port:
        number: 8080
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-config
data:
  model.host: 'http://sms-model-serv:8080"
```


Application
Ansible
Docker
Kubernetes

OCI spec

Images and containers follow a spec

Thus you can spin a podman container from a docker image

k8s can use podman or docker images

Bind mount vs named volume

Bind mount - Directly mount it, including permission

Named volume - Lives in its own directory, abstracted permissions

Using a named volume with podman rootles...

Main problem: Databases require specific file permission

On the HPC we need to use rootless

Named volumes can abstract the permissions

However, the abstractions lives in extended attributes (xattr)

xattr is not available on NFS (newer kernel version might)

Going even deeper - unshare

With unshare you can shift from host file permission to container file permission

This might allow setting up a bind mount with the correct permissions

Questions