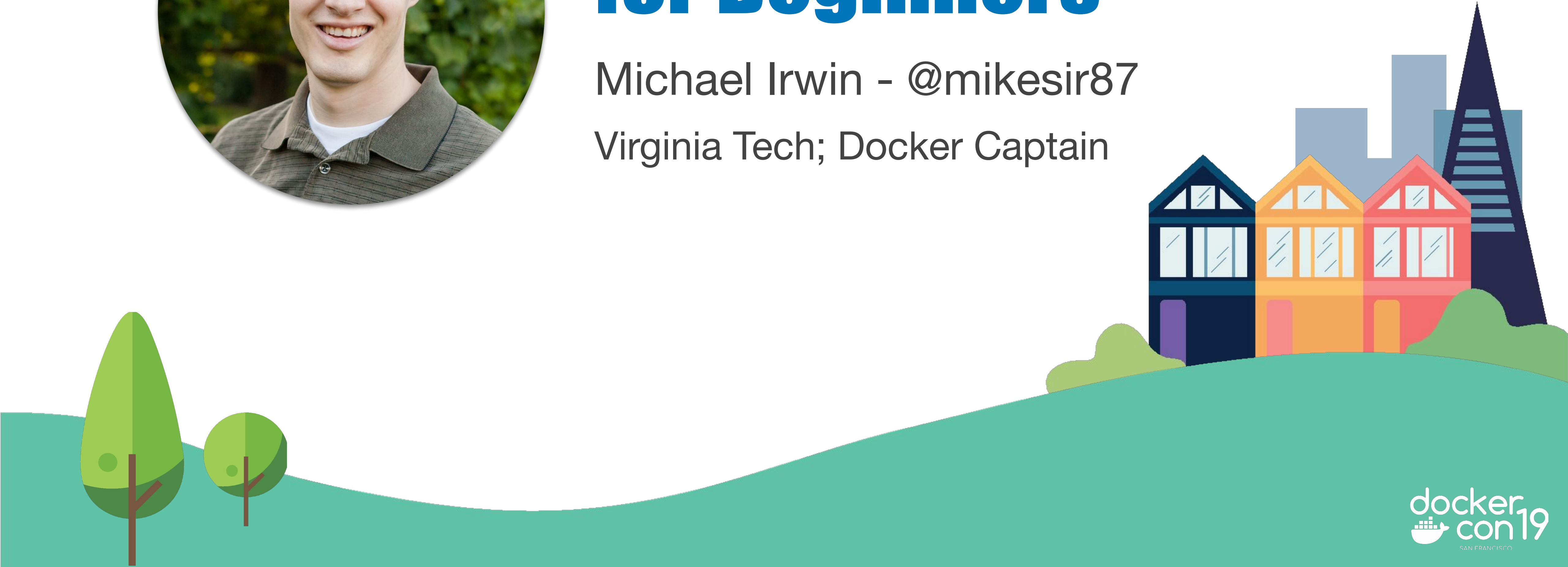
Containers for Beginners

Michael Irwin - @mikesir87

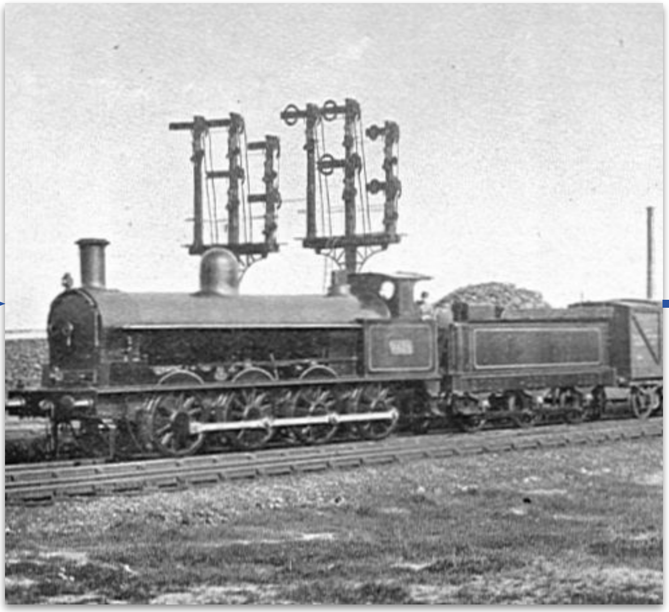
Virginia Tech; Docker Captain



Disclaimer: I cannot explain sprankle pods either!

Quick History of Shipping

Source: https://www.publicdomainpictures.net/en/view-image.php?image=275355



Source: https://en.wikipedia.org/wiki/Rail_freight_in_Great_Britain

Source: https://pxhere.com/en/photo/553345

Software = Shipping?

Shipping in Software

Source: https://www.usafe.af.mil/News/Photos/igphoto/2000887438/

docker

Either of these two scenarios sound familiar to you?

Welcome!
Glad to have you on the team!

Clone the repo, use the wiki for setup instructions, and update the docs as needed.
Good luck!

WORKED FINE IN DEV
OPS PROBLEM NOW
memegenerator.net

Imagine if…

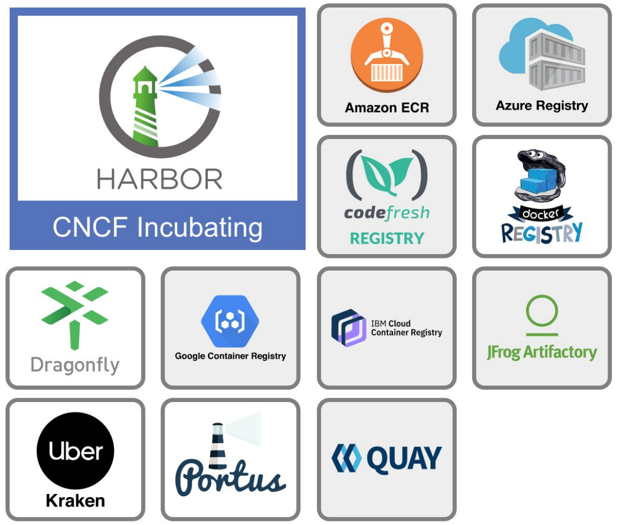
Creating Images

- Best practice is to use a `Dockerfile`
  - A text file that serves as a script to build an image
- Build using the `docker build` command

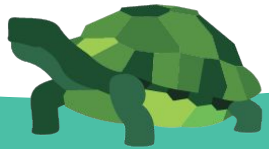
```
FROM node
WORKDIR /app
COPY package.json yarn.lock .
RUN yarn install
COPY src ./src
CMD ["node", "src/index.js"]
```

Sharing Images

- Once built, the image is only available locally
- To share, push it to a registry using `docker push`
  - Docker Hub is the default registry
  - Docker EE includes the Docker Trusted Registry
  - Many other third-party offerings available too
- Once shared, others can pull the image



Source: https://landscape.cncf.io



Let’s build an image!

What's a container then?

- While a container looks like a VM, it isn't!
  - A container is **just another process** on the machine
- It uses namespaces and control groups (cgroups) to provide isolation
  - Namespaces include network, process, user, IPC, mount, and others
- To run a container, use the `docker container run` command

WHEN YOU FIND OUT
CONTAINERS AREN'T JUST FOR STORING LEFTOVERS

Containers vs VMs

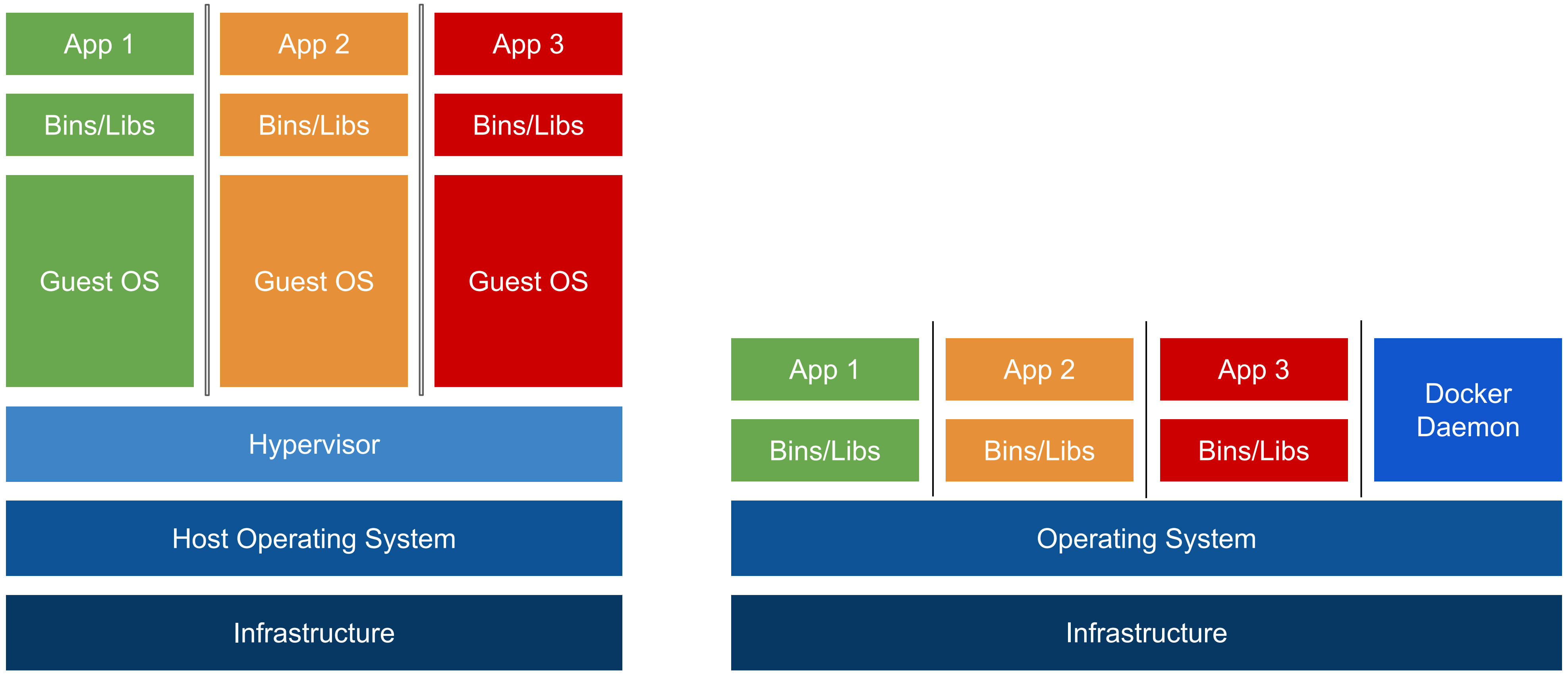
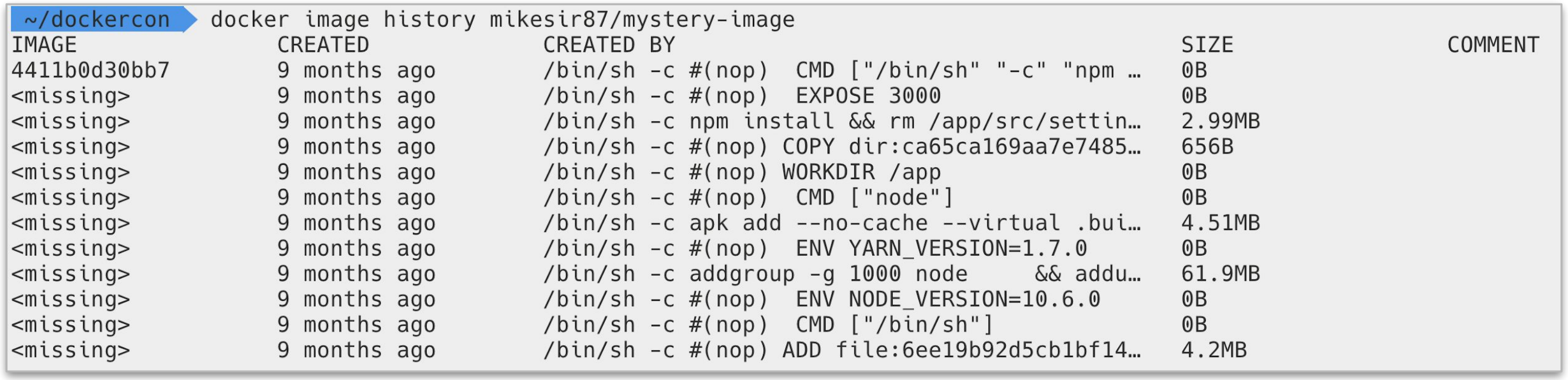


Image Layering

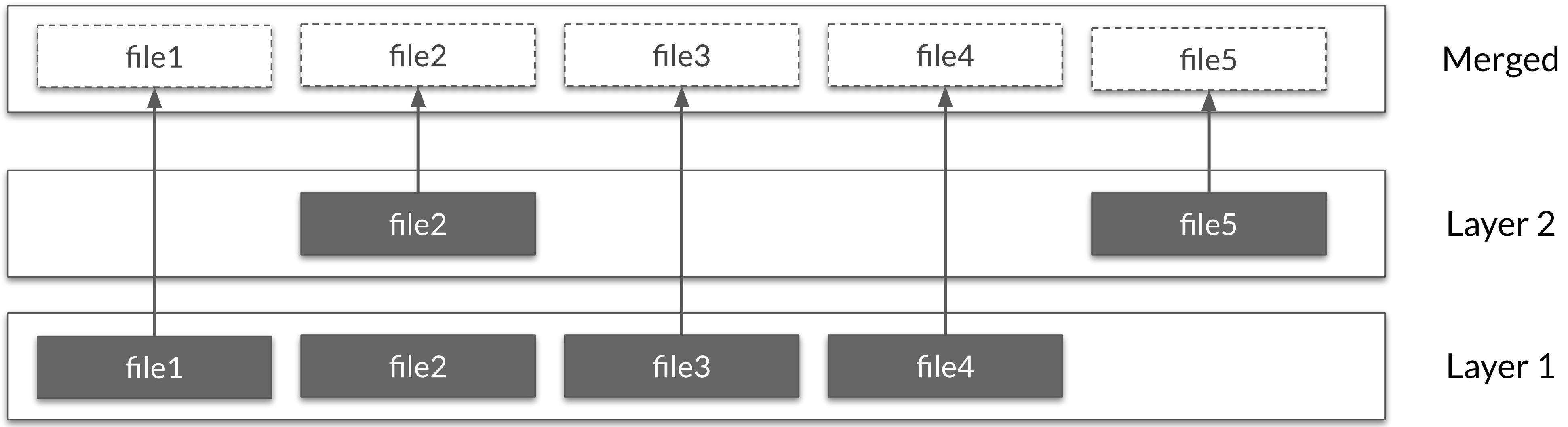
- Images are composed of layers of filesystem changes
  - Each layer can add or remove from the previous layer
  - Each layer's filesystem changes are stored as a single tar file
- Each command in a `Dockerfile` creates a new layer
- Use the `docker image history` command to see the layers and the command that was used to create each layer



```
~/dockercon  docker image history mikesir87/mystery-image
IMAGE               CREATED             CREATED BY                                      SIZE                COMMENT
4411b0d30bb7        9 months ago        /bin/sh -c #(nop)  CMD ["/bin/sh" "-c" "npm …   0B
<missing>           9 months ago        /bin/sh -c #(nop)  EXPOSE 3000                  0B
<missing>           9 months ago        /bin/sh -c npm install && rm /app/src/settin…   2.99MB
<missing>           9 months ago        /bin/sh -c #(nop) COPY dir:ca65ca169aa7e7485…   656B
<missing>           9 months ago        /bin/sh -c #(nop) WORKDIR /app                  0B
<missing>           9 months ago        /bin/sh -c #(nop)  CMD ["node"]                 0B
<missing>           9 months ago        /bin/sh -c apk add --no-cache --virtual .bui…   4.51MB
<missing>           9 months ago        /bin/sh -c #(nop)  ENV YARN_VERSION=1.7.0       0B
<missing>           9 months ago        /bin/sh -c addgroup -g 1000 node     && addu…   61.9MB
<missing>           9 months ago        /bin/sh -c #(nop)  ENV NODE_VERSION=10.6.0      0B
<missing>           9 months ago        /bin/sh -c #(nop)  CMD ["/bin/sh"]              0B
<missing>           9 months ago        /bin/sh -c #(nop) ADD file:6ee19b92d5cb1bf14…   4.2MB
```

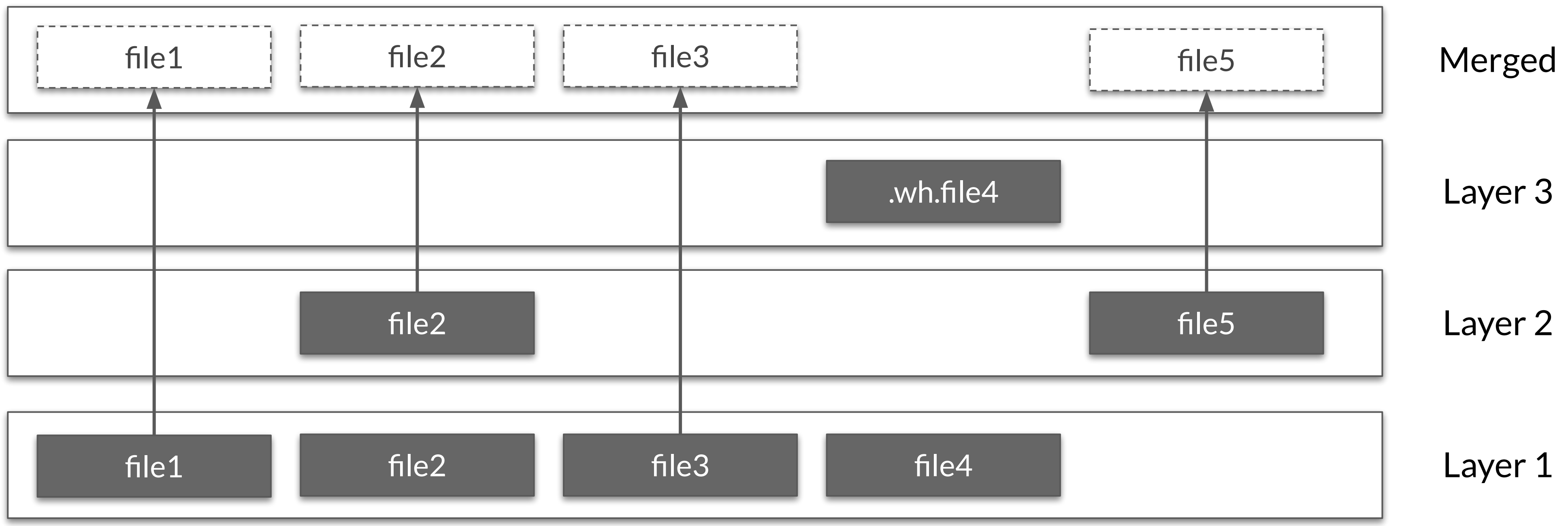
Layer contents

- Layers are unioned together to make a full filesystem
  - Each layer can add files as needed
  - Files in “higher” layers replace the same file in “lower” layers
- The container uses the “merged” view



What about deleted files?

- Deleted files are represented in a layer as a “whiteout” file
- Whiteout files are only used by the filesystem driver and not visible in the merged filesystem



WARNING!

Be careful what you put into images. Deleted files might not actually be gone!

Two Best Practices Incoming!

Clean up as you go!

- Don’t wait until the end of the Dockerfile to “clean” up
- Chain RUN commands together to clean things as you go

```
FROM ubuntu
RUN apt-get update
RUN apt-get install -y python python-pip
RUN pip install awscli
RUN apt-get autoremove --purge -y python-pip
```

Net change of image size from 512MB to 183MB (64% reduction)

```
FROM ubuntu
RUN apt-get update && \
    apt-get install -y python python-pip && \
    pip install awscli && \
    apt-get autoremove --purge -y python-pip && \
    rm -rf /var/lib/apt/lists/*
```

Keep images tight and focused

- Only install the deps/tools/packages that are necessary
- Use multi-stage builds to separate build-time and run-time dependencies

```
FROM node AS build
WORKDIR /usr/src/app
COPY package.json yarn.lock .
RUN yarn install
COPY public ./public
COPY src ./src
RUN yarn build

FROM nginx:alpine
COPY nginx.conf /etc/nginx/nginx.conf
COPY --from=build /usr/src/app/build /usr/share/nginx/html
```

Sample multi-stage build for a React app

How do you persist data?

Volumes

- Volumes provide the ability to persist/supply data
- Bind mount volumes
  - You choose where to persist the data
  - Example: `-v $HOME/mysql-data:/var/lib/mysql`
- Named volumes
  - Let Docker choose where to persist the data
  - Can use `docker volume inspect` to find actual location
  - Example: `-v mysql-data:/var/lib/mysql`

Show me these volumes!

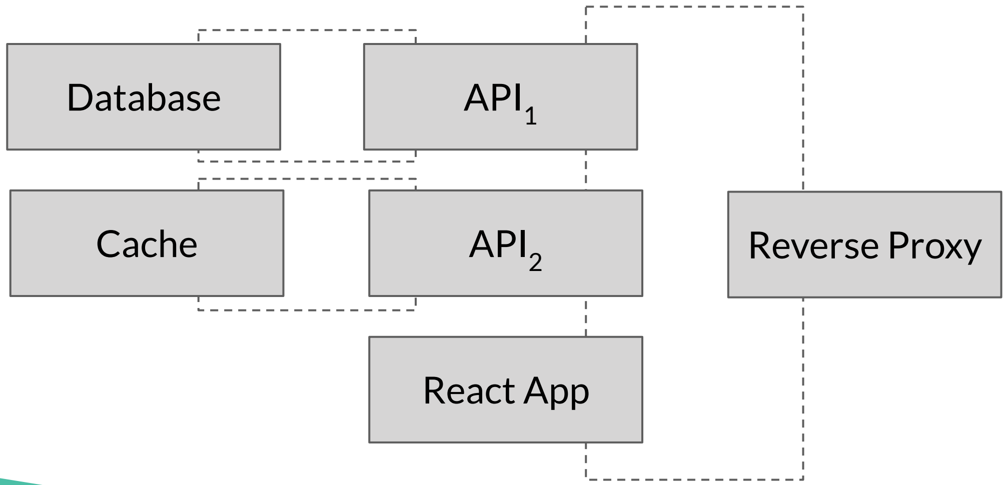
GOOD LUCK
REMEMBERING ALL
OF THE OPTIONS

Docker Compose

- Makes defining and running multi-container apps super easy
- Uses a YAML file for configuration (`docker-compose.yml`)
    - Often included in project source repo at the root of the project
- With a single command, start all containers/services for an app
- Tool is bundled with Docker Desktop

Docker Networking

- Think of networking in terms of communication boundaries/isolation
    - If two containers are on the same network, they can talk to each other
- Docker runs its own DNS resolver on each network
    - Allows it to resolve IP addresses of other containers using “aliases”



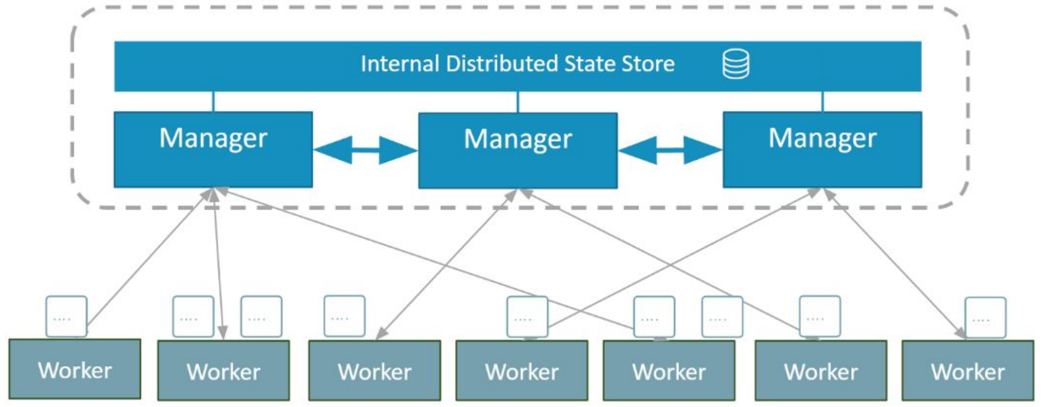
Quick compose demo!

Container Orchestration

- Orchestration provides the ability to manage the running of container workloads, often over a fleet of machines
- You define the **expected state** (the desired state)
- The system then tries to make **actual state** reflect expected state

Actors in Orchestration

- Every orchestrator has the concept of two types of nodes
- Managers
  - Serve as the brains of the cluster
  - Maintain state and schedule work
  - Sometimes called masters
- Worker nodes
  - Perform the actual work, as instructed by a manager
  - Sometimes called agents or nodes



Various Orchestrators

- Docker Swarm
  - Shipped with the Docker engine
  - Very user friendly and easy to get up and running
  - Satisfies most needs, though not all; built to be extensible, but takes some work
- Kubernetes
  - Spun out of work done within Google and contributed to CNCF
  - Think of it more as a toolkit - so not as easy to get up and running
  - Very configurable and extensible
- Amazon ECS
  - Made by Amazon Web Services and provided for free
  - Provides deep integration with AWS resources (IAM, ALBs, Auto-scaling, etc.)

Quick Swarm Demo!

Recap

- Containers/images are here to standardize application packaging
  - No longer require host configuration
  - Docker Compose builds on the abstraction to make multi-service apps easier
  - Container orchestration builds on this idea
- Be mindful of how you build your images and what you include
- Volumes allow data to be persisted longer than the container
- Networking serves provides communication paths/isolation

WARNING!

Containers are NOT a silver bullet that will fix your company culture

Thank you!
Rate the session!

Keep in touch!

@mikesir87; mikesir87@vt.edu