PART 1: CONCEPTS

PRACTICAL DOCKER

OBJECTIVES

- 1.1 Understand containerization
- 1.2 Understand how Docker implements containerization
- 1.3 Learn basic Docker commands with hands-on examples
- 1.4 Prepare for next part in series

CONTAINERIZATION

OBJECTIVES

- What is a computer?
- What is a virtual machine?
- What is a container?
- What problems do containers solve?

WHAT IS A COMPUTER?

Aristotle, probably

WHAT IS A COMPUTER?

- Components
  - Hardware (CPU, RAM, storage, GPU, I/O)
  - Software (operating system, programs)
- Operating system kernel
  - Controls all processes
  - Acts as broker between hardware and software

WE NEED TO GO DEEPER.

Leonardo DiCaprio – "Titanic" (1995)

WHAT IS A VIRTUAL MACHINE?

- Virtual machines (VMs) are emulations of computer hardware and software running inside a host machine
- "Full virtualization"
  - Simulates entire machine
  - VirtualBox, VMWare
- Each VM uses its own resources from the host
- Persistent

WE NEED TO GO DEEPER.

Nelson Mandela (2012)

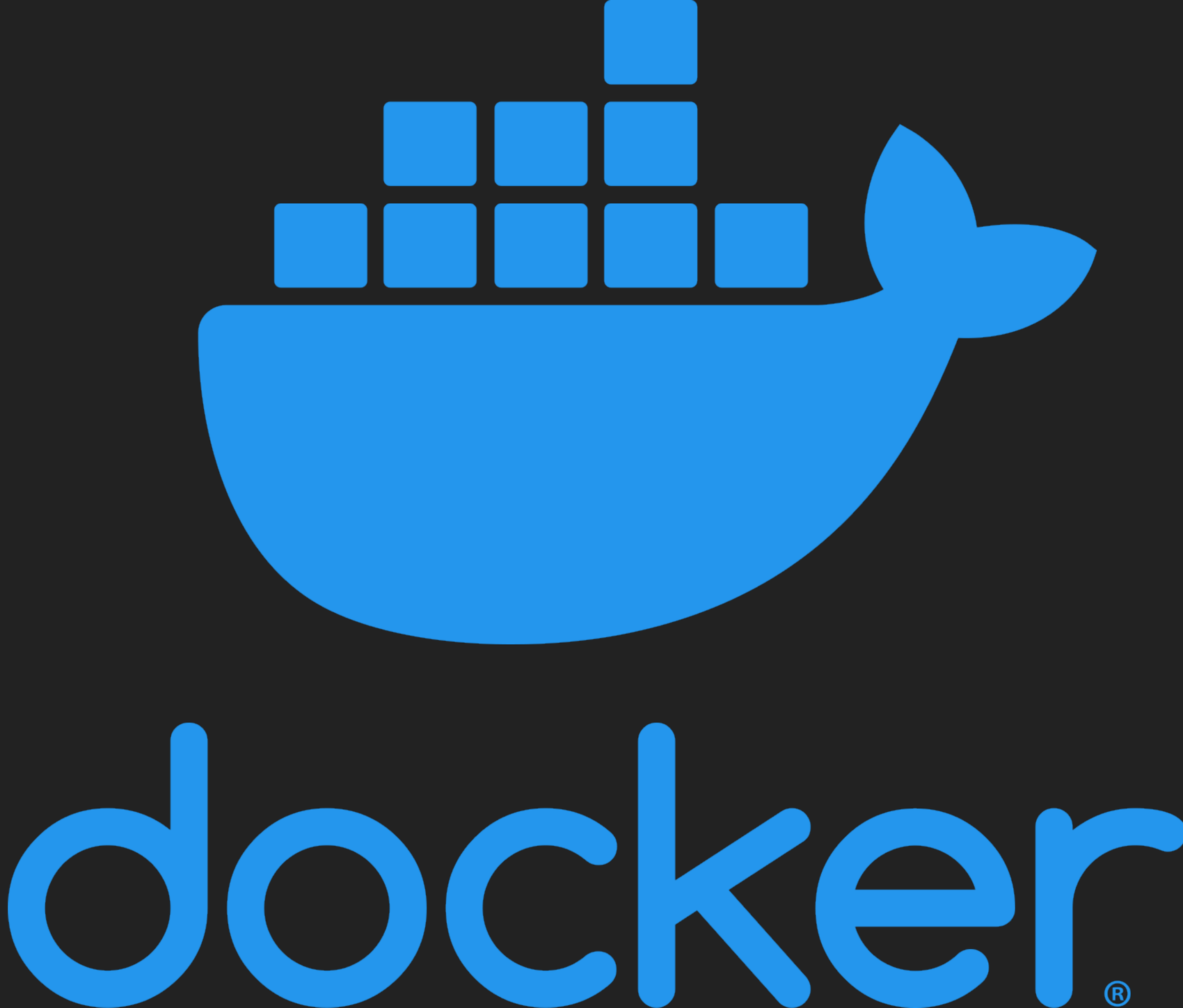
WHAT IS A CONTAINER?

- Containers are abstractions of file systems
- "OS-level virtualization"
  - Simulates operating system
  - Docker, Solaris Containers
- Share same set of host resources
- Ephemeral

WHAT PROBLEMS DO CONTAINERS SOLVE?

- Packaged, ready-to-run dependencies
- Allows parity between development and deployment environments
- Reduces need for individual environment customizations
  - Anyone running an M1 MacBook Pro? 🙃

INTRODUCING DOCKER


docker

OBJECTIVES

- What is Docker?
- Docker components
  - Docker Engine
  - Docker CLI
  - Docker Desktop
  - Docker Hub

WHAT IS DOCKER?

– All of you reading this (right now)

WHAT IS DOCKER?

- Docker is an open-source, cross-platform containerization solution by Docker, Inc.
- Composed of several pieces of software
  - Docker Engine, Desktop, and API
    - Docker daemon (`dockerd`)
  - Docker CLI (`docker`)
    - Interacts with the daemon through the API
  - Docker Hub
  - Image repository

PHOTO: HEINER (PEXELS)

DOCKER BASICS

OBJECTIVES

- Images
- Containers
- Docker commands
- Docker Compose


IMAGES?

IMAGES

- "An Image is an ordered collection of root filesystem changes and the corresponding execution parameters for use within a container runtime."
  - Very cool. In English now?
- A root filesystem change is a modification
  - Start with `/`
  - Add a file at `/hello.txt`
  - Congratulations, you have made a root filesystem change
- An image can be thought of as a copy of a file system including its directory structure and folders
- Created from *Dockerfiles*

IMAGES, CONTINUED

- Dockerfiles are text files that contain imperative instructions to build images

```
FROM ubuntu
RUN echo "Hello world" > hello.txt
CMD cat hello.txt
```

FROM directive specifies base image

RUN directive creates a "layer"

CMD directive defines container execution

- Images are the basis of containers


COOL, SO CONTAINERS?

CONTAINERS ARE ABSTRACTIONS OF FILE SYSTEMS.

Some guy (a few minutes ago)

CONTAINERS, FINALLY

- Containers are *running instances* of images
  - In Docker parlance, "images" are actually the abstractions of file systems
- Can be short or long-lived
  - Short-lived: `echo "Hello, world!"`
  - Long-lived: `bundle exec rails s`
- Containers are ephemeral

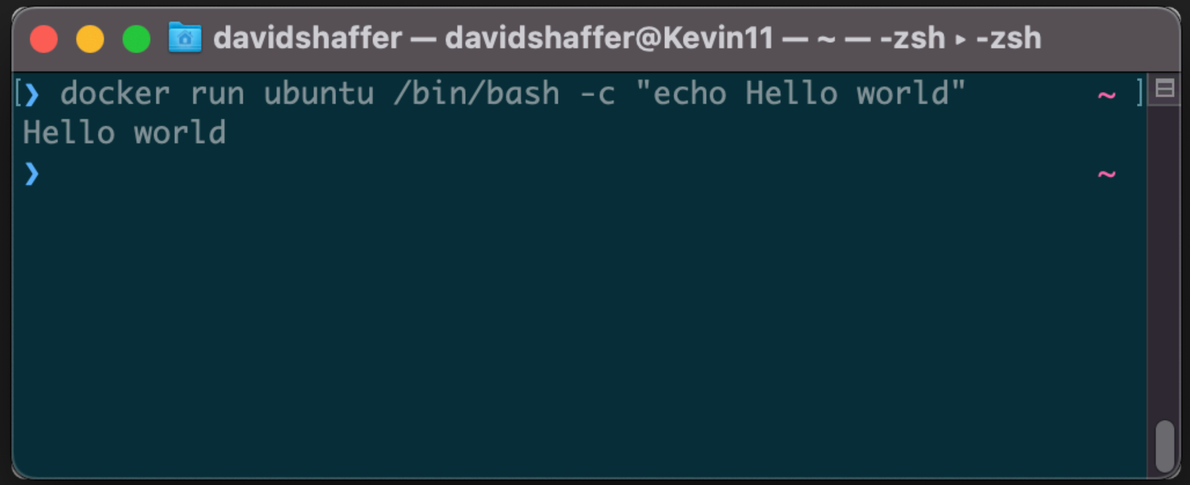

TELL ME HOW.

DOCKER COMMANDS

- `docker run`
- `docker build`
  - Tagging
- `docker ps`
- `docker images`
- `docker rm & rmi`

DOCKER COMMANDS

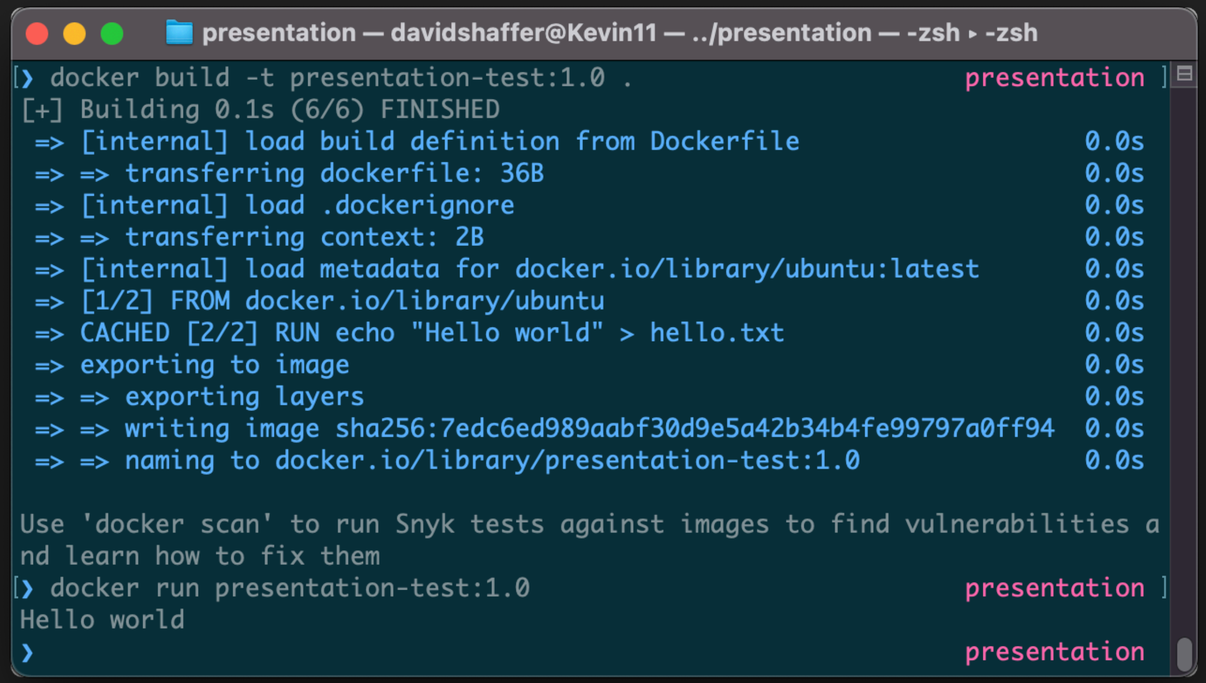
- `docker run`
  - Runs an image in a container
  - Requires an image name
  - Can optionally specify command
  - Downloads image automatically if not present



DOCKER COMMANDS

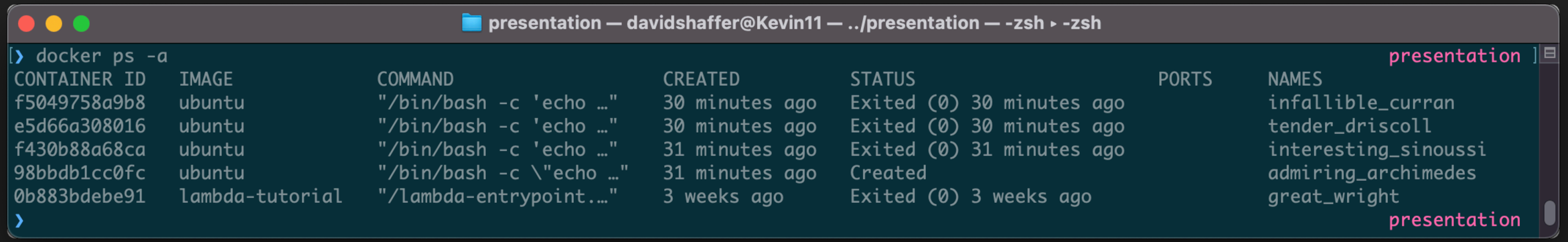
- `docker build`
  - Creates an image
  - Requires a Dockerfile location
  - Can optionally specify name and tag
  - Images are run with `docker run` or `docker start`

```
FROM ubuntu
RUN echo "Hello world" > hello.txt
CMD cat hello.txt
```



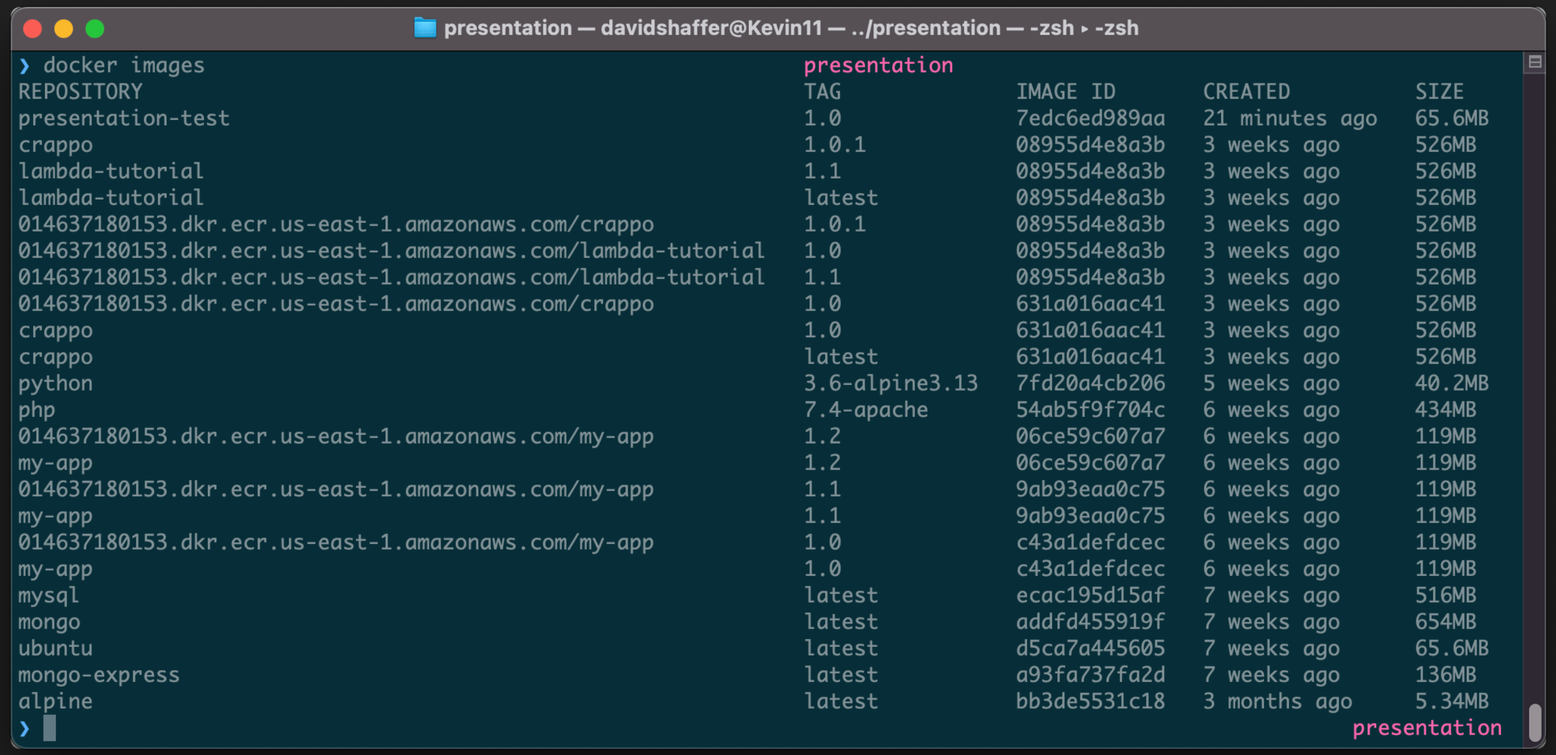
DOCKER COMMANDS

- `docker ps`
  - Lists containers
    - By default, lists only running containers
    - `-a` flag lists all containers



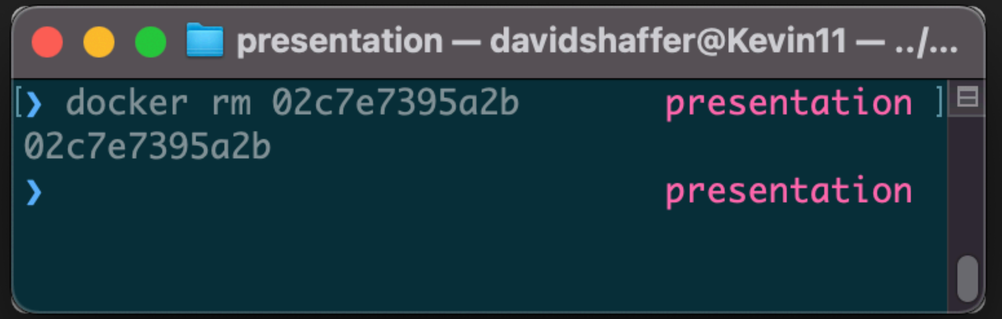
DOCKER COMMANDS

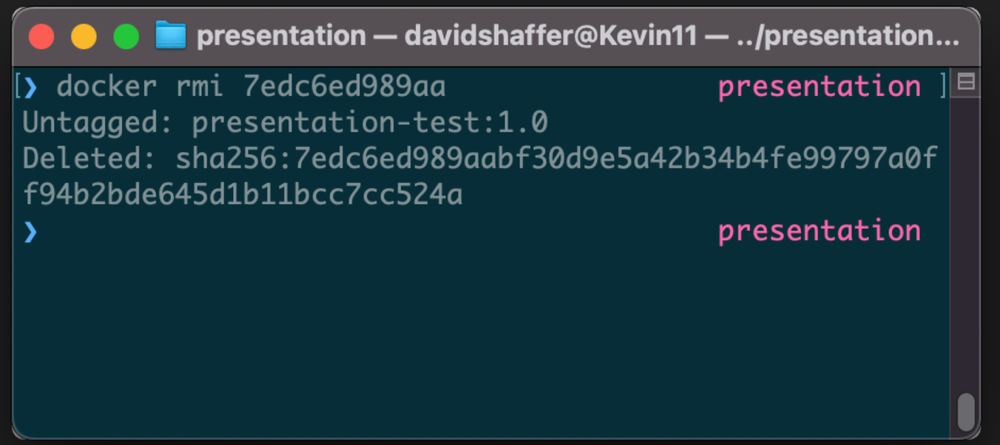
- `docker images`
  - (also `docker image ls`)
  - Lists images
  - Many filtering options available

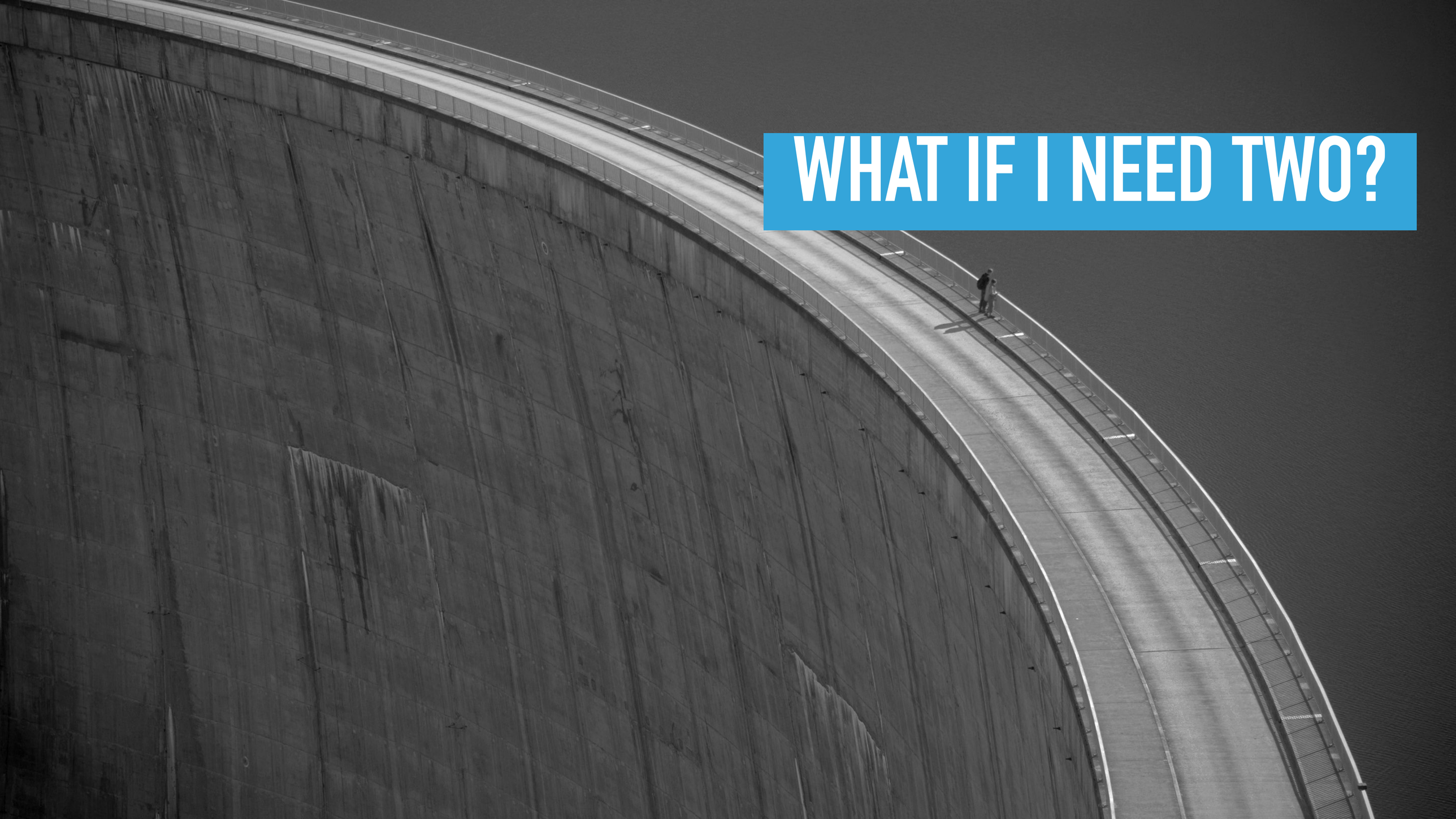


DOCKER COMMANDS

- `docker rm`
  - Removes containers
  - Requires container id
- `docker rmi`
  - Removes images
  - Requires image id
  - Will fail if image is in use






WHAT IF I NEED TWO?

DOCKER COMPOSE

- Docker Compose is a tool for defining and running multi-container Docker applications.
- Defined by YAML files
  - Create services
  - Networking
- Docker Compose usage is not directly required for this series, but will be revisited slightly more fully in the next talk
- https://github.com/pexels/cloudflare-logs/blob/master/docker-compose.yml

NEXT STEPS

STUFF I DIDN'T COVER

- File storage
  - What if I want my stuff to persist?
  - Docker volumes & bind mounts
- `docker exec`
- Networking
- Docker Compose
- Private image repositories
- Best practices

NEXT TALK

- Real-world usage
- Cloud 66 (our beloved PaaS)
  - BuildGrid (private container repository)
  - Docker + Docker Compose + Kubernetes
- Analysis of Pexels Docker usage
  - Problems
  - Recommendations

CONCLUSION

WHAT HAVE WE LEARNED?

- Containerization
- Docker introduction
- Docker basics
- Setup for next steps

PHOTO: KASIA PALITAVA (PEXELS)

REFERENCES & FURTHER LEARNING

- Docker: https://www.docker.com/
- Docker documentation: https://docs.docker.com/
- Docker Hub: https://hub.docker.com/
- TechWorld with Nana:
  - Not someone's grandmother teaching you DevOps
  - YouTube Channel: https://www.youtube.com/c/TechWorldwithNana
  - 3 hour Docker course: https://www.youtube.com/watch?v=3c-iBn73dD

PHOTO: EBERHARD GROSSGASTEIGER (PEXELS)

THANK YOU

David Shaffer