

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)

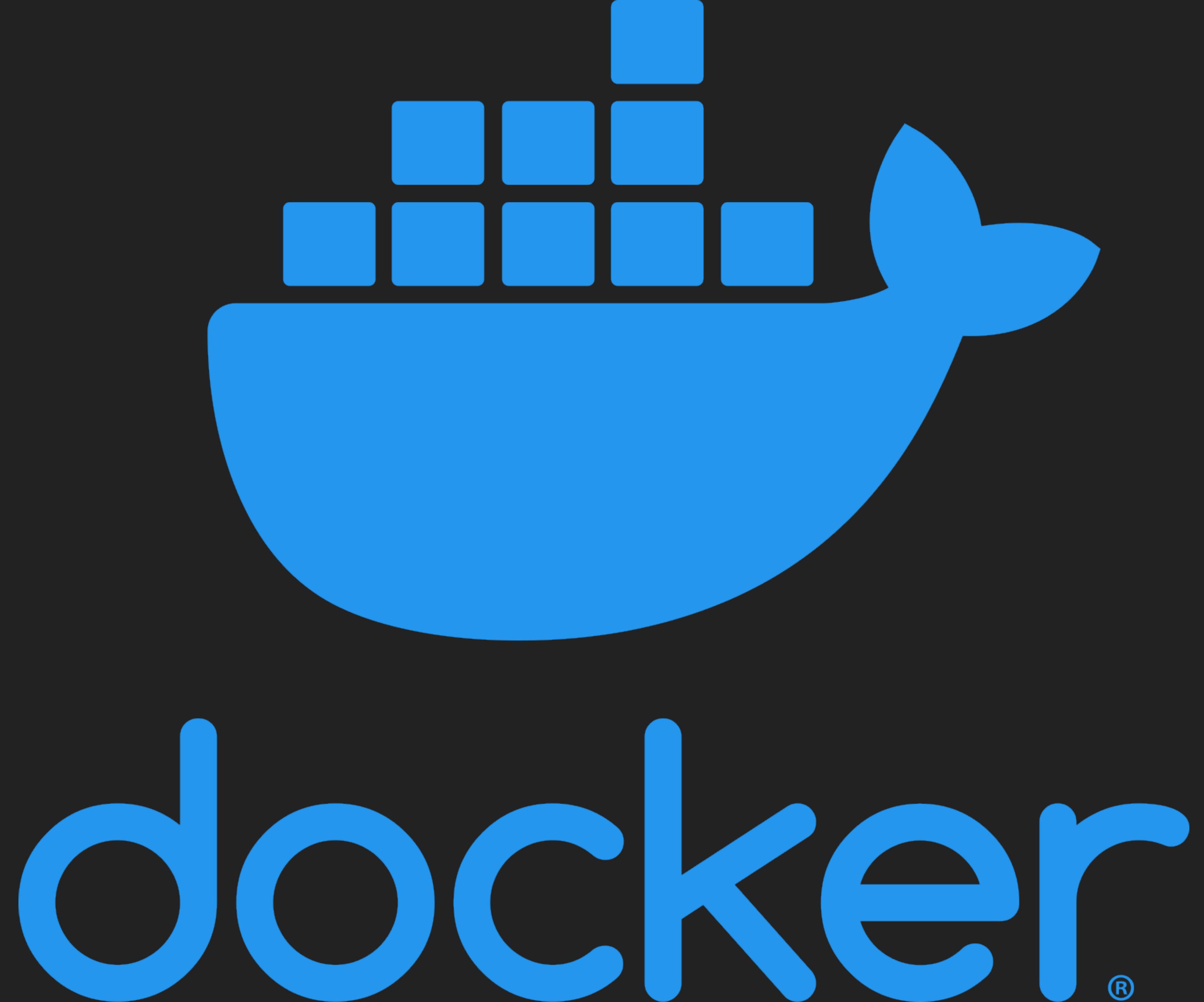
WHAT IS A CONTAINER?

- ▶ 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



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

A black and white photograph of a massive concrete dam wall, likely the Hoover Dam. The wall is thick and curved, with a walkway running along its top edge. Two small figures of people are visible on the walkway, emphasizing the enormous scale of the structure.

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

A Dockerfile is shown in a purple box:

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

Annotations with arrows point from the text to the right:

- 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

A black and white photograph of a massive concrete dam wall, likely the Hoover Dam. The wall is thick and curved, with a walkway running along its top edge. Two small figures of people are visible on the walkway, emphasizing the enormous scale of the structure.

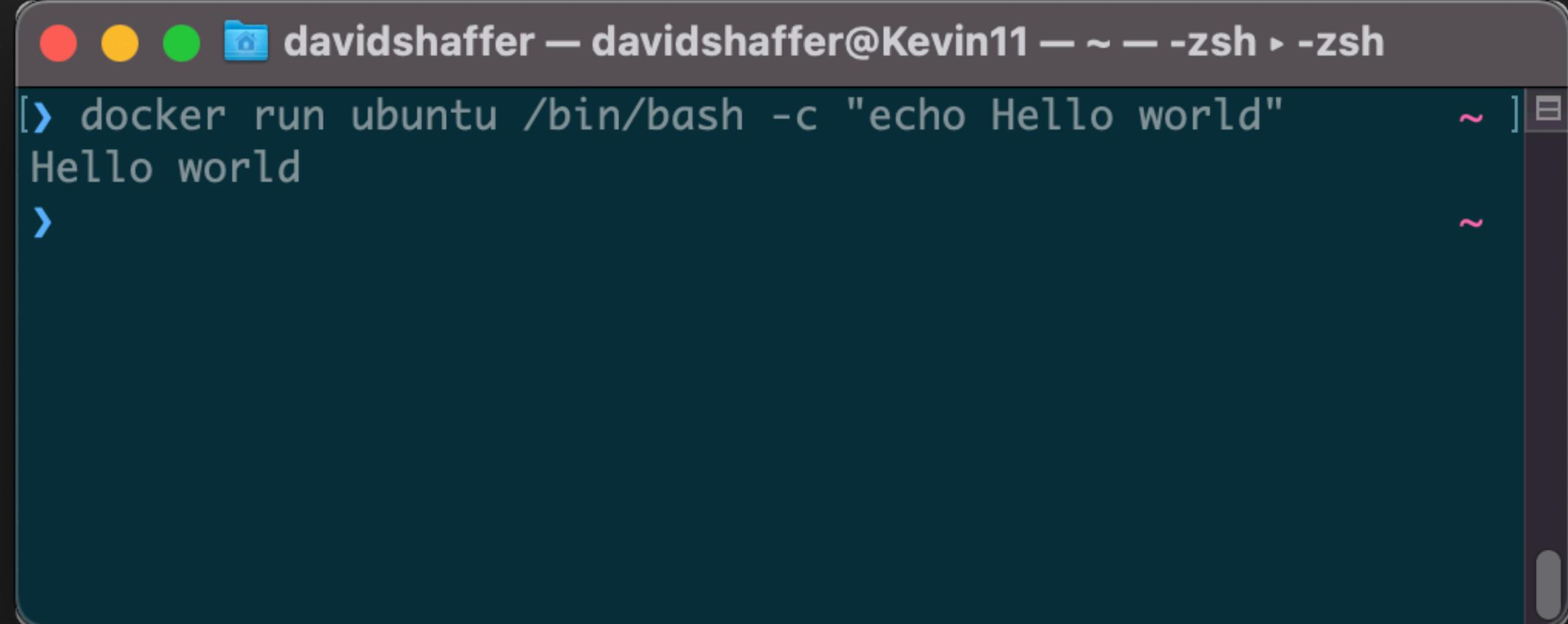
TELL ME HOW.

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



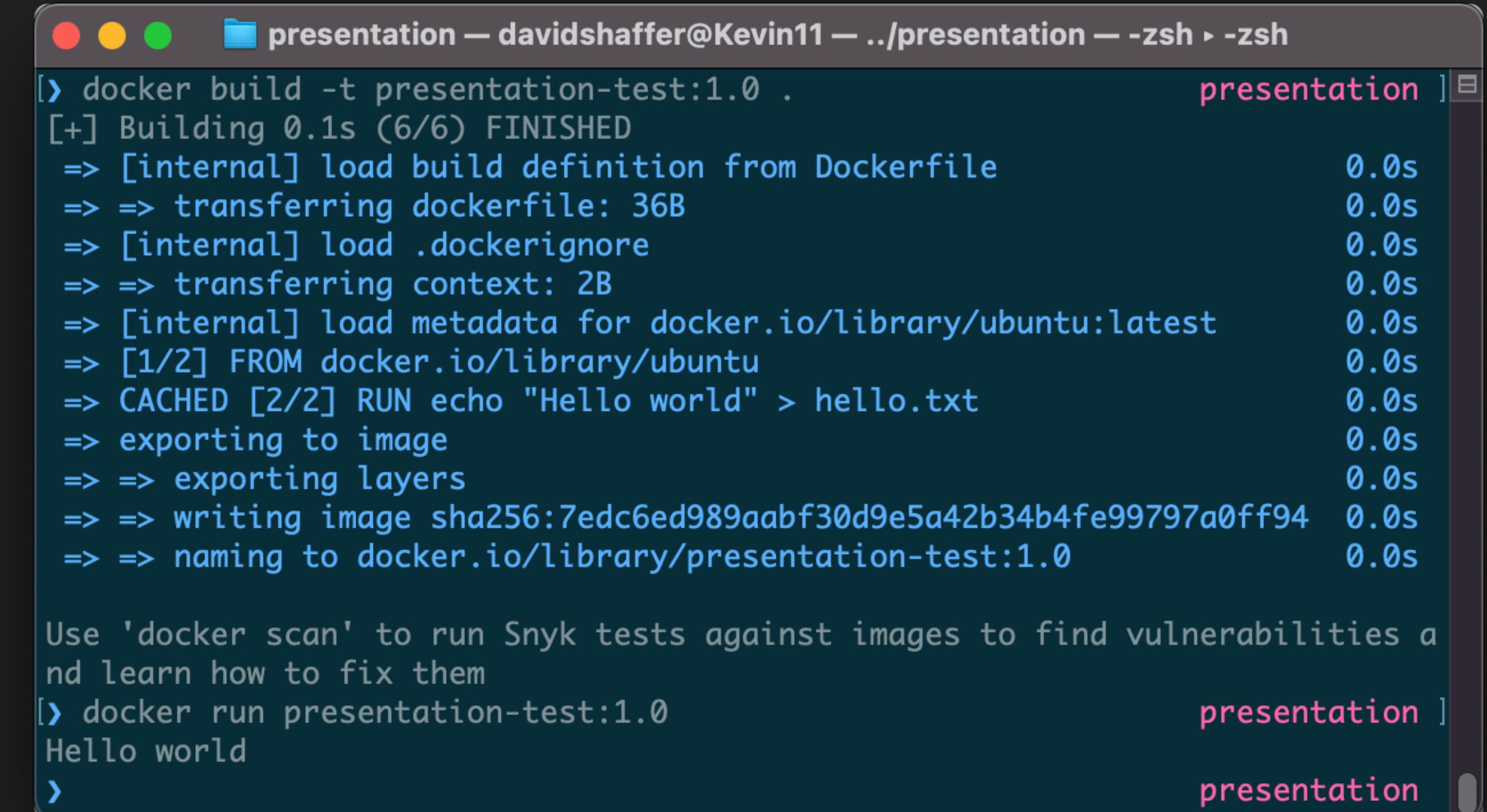
A screenshot of a terminal window titled "davidshaffer — davidshaffer@Kevin11 ~ - zsh". The window shows the command `docker run ubuntu /bin/bash -c "echo Hello world"` being run. The output of the command, "Hello world", is displayed below the command line.

```
davidshaffer — davidshaffer@Kevin11 ~ - zsh
[docker run ubuntu /bin/bash -c "echo Hello world"
Hello world
>
```

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
```



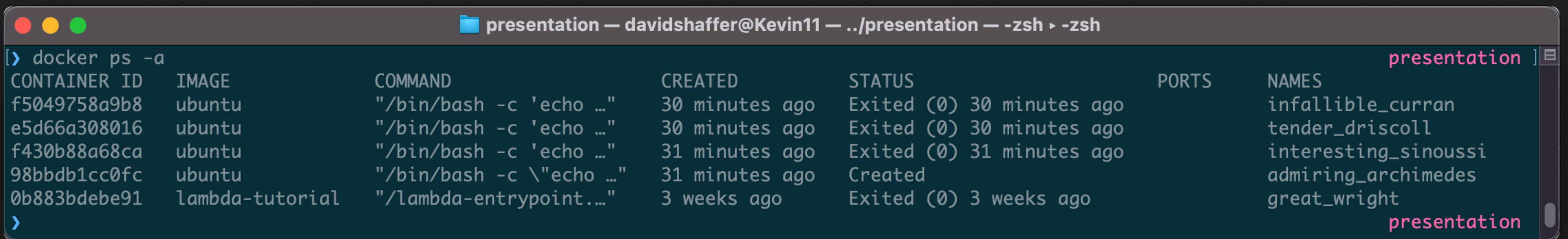
A screenshot of a terminal window titled "presentation — davidshaffer@Kevin11 — .. / presentation — -zsh ▶ -zsh". The window shows the command "docker build -t presentation-test:1.0 ." being run. The output details the build process, including loading the Dockerfile, transferring context, and creating a new image named "presentation-test:1.0". It also includes a note about running Snyk tests and a final command to run the image.

```
[> docker build -t presentation-test:1.0 .
[+] Building 0.1s (6/6) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 36B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [1/2] FROM docker.io/library/ubuntu
=> CACHED [2/2] RUN echo "Hello world" > hello.txt
=> exporting to image
=> => exporting layers
=> => writing image sha256:7edc6ed989aabf30d9e5a42b34b4fe99797a0ff94
=> => naming to docker.io/library/presentation-test:1.0

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[> docker run presentation-test:1.0
Hello world
[>
```

DOCKER COMMANDS

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

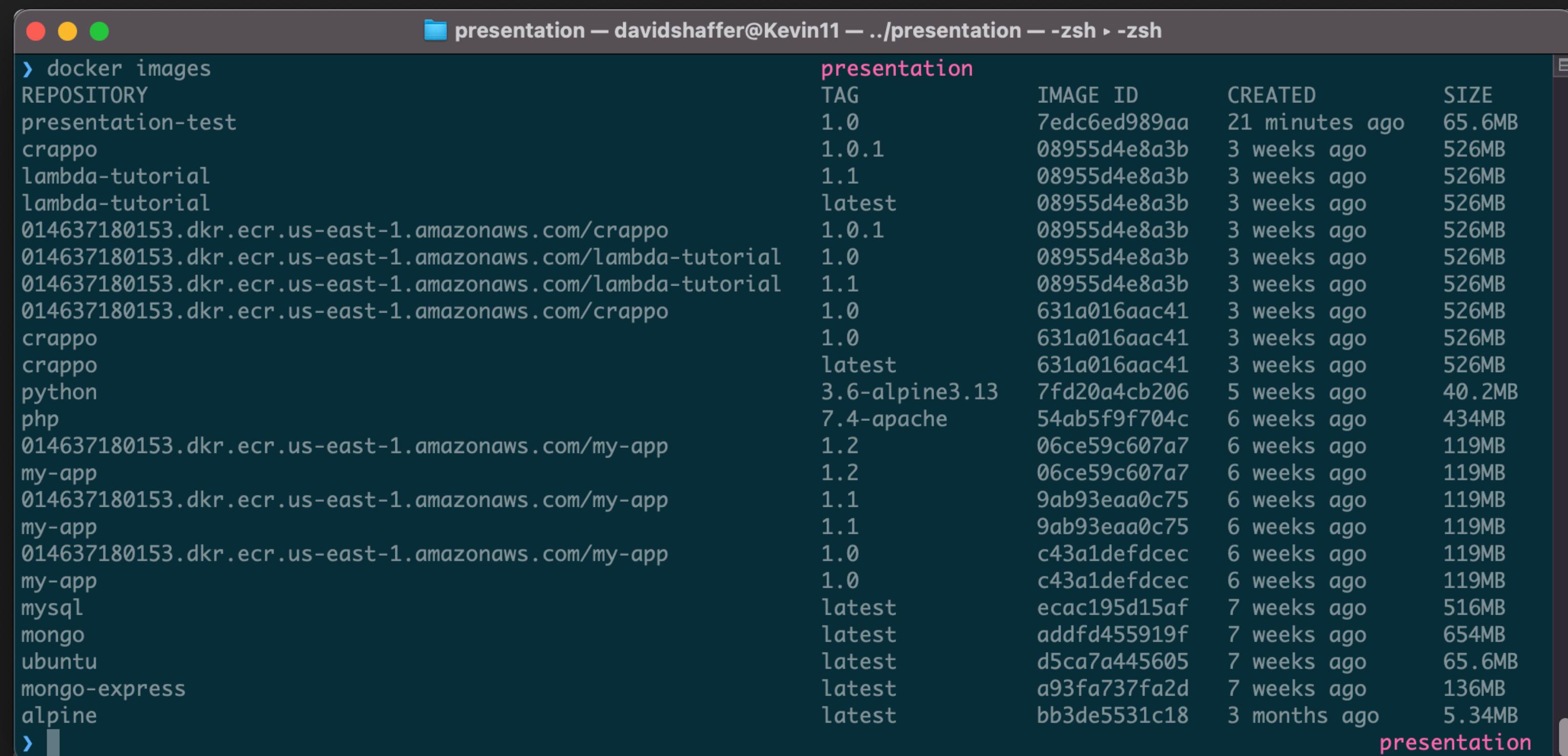


A screenshot of a terminal window titled "presentation — davidshaffer@Kevin11 — .. / presentation — -zsh > -zsh". The command "docker ps -a" is run, displaying a table of container information. The table has columns: CONTAINER ID, IMAGE, COMMAND, CREATED, STATUS, PORTS, and NAMES. There are six rows, each representing a container. The last row is for a container named "presentation" which has exited.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f5049758a9b8	ubuntu	"/bin/bash -c 'echo ..."	30 minutes ago	Exited (0) 30 minutes ago		infallible_curran
e5d66a308016	ubuntu	"/bin/bash -c 'echo ..."	30 minutes ago	Exited (0) 30 minutes ago		tender_driscoll
f430b88a68ca	ubuntu	"/bin/bash -c 'echo ..."	31 minutes ago	Exited (0) 31 minutes ago		interesting_sinoussi
98bbdb1cc0fc	ubuntu	"/bin/bash -c \"echo ..."	31 minutes ago	Created		admiring_archimedes
0b883bdebe91	lambda-tutorial	"/lambda-entrypoint..."	3 weeks ago	Exited (0) 3 weeks ago		great_wright
						presentation

DOCKER COMMANDS

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



A screenshot of a terminal window titled "presentation — davidshaffer@Kevin11 — .. / presentation — -zsh ▶ -zsh". The window displays the output of the "docker images" command. The output shows a table of Docker images with columns: REPOSITORY, TAG, IMAGE ID, CREATED, and SIZE. The table includes images from the "presentation-test" repository and various public repositories like "crappo", "lambda-tutorial", "my-app", "mysql", "mongo", "ubuntu", "mongo-express", and "alpine". Some images have multiple tags (e.g., 1.0, 1.0.1, 1.1, latest). The "SIZE" column shows values such as 65.6MB, 526MB, 526MB, 526MB, 40.2MB, 434MB, 119MB, 119MB, 119MB, 119MB, 516MB, 654MB, 65.6MB, 136MB, and 5.34MB.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
presentation-test	1.0	7edc6ed989aa	21 minutes ago	65.6MB
crappo	1.0.1	08955d4e8a3b	3 weeks ago	526MB
lambda-tutorial	1.1	08955d4e8a3b	3 weeks ago	526MB
lambda-tutorial	latest	08955d4e8a3b	3 weeks ago	526MB
014637180153.dkr.ecr.us-east-1.amazonaws.com/crappo	1.0.1	08955d4e8a3b	3 weeks ago	526MB
014637180153.dkr.ecr.us-east-1.amazonaws.com/lambda-tutorial	1.0	08955d4e8a3b	3 weeks ago	526MB
014637180153.dkr.ecr.us-east-1.amazonaws.com/lambda-tutorial	1.1	08955d4e8a3b	3 weeks ago	526MB
014637180153.dkr.ecr.us-east-1.amazonaws.com/crappo	1.0	631a016aac41	3 weeks ago	526MB
crappo	1.0	631a016aac41	3 weeks ago	526MB
crappo	latest	631a016aac41	3 weeks ago	526MB
python	3.6-alpine3.13	7fd20a4cb206	5 weeks ago	40.2MB
php	7.4-apache	54ab5f9f704c	6 weeks ago	434MB
014637180153.dkr.ecr.us-east-1.amazonaws.com/my-app	1.2	06ce59c607a7	6 weeks ago	119MB
my-app	1.2	06ce59c607a7	6 weeks ago	119MB
014637180153.dkr.ecr.us-east-1.amazonaws.com/my-app	1.1	9ab93eaa0c75	6 weeks ago	119MB
my-app	1.1	9ab93eaa0c75	6 weeks ago	119MB
014637180153.dkr.ecr.us-east-1.amazonaws.com/my-app	1.0	c43a1defdcec	6 weeks ago	119MB
my-app	1.0	c43a1defdcec	6 weeks ago	119MB
mysql	latest	ecac195d15af	7 weeks ago	516MB
mongo	latest	addfd455919f	7 weeks ago	654MB
ubuntu	latest	d5ca7a445605	7 weeks ago	65.6MB
mongo-express	latest	a93fa737fa2d	7 weeks ago	136MB
alpine	latest	bb3de5531c18	3 months ago	5.34MB

DOCKER COMMANDS

- ▶ `docker rm`

- ▶ Removes containers

- ▶ Requires container id

- ▶ `docker rmi`

- ▶ Removes images

- ▶ Requires image id

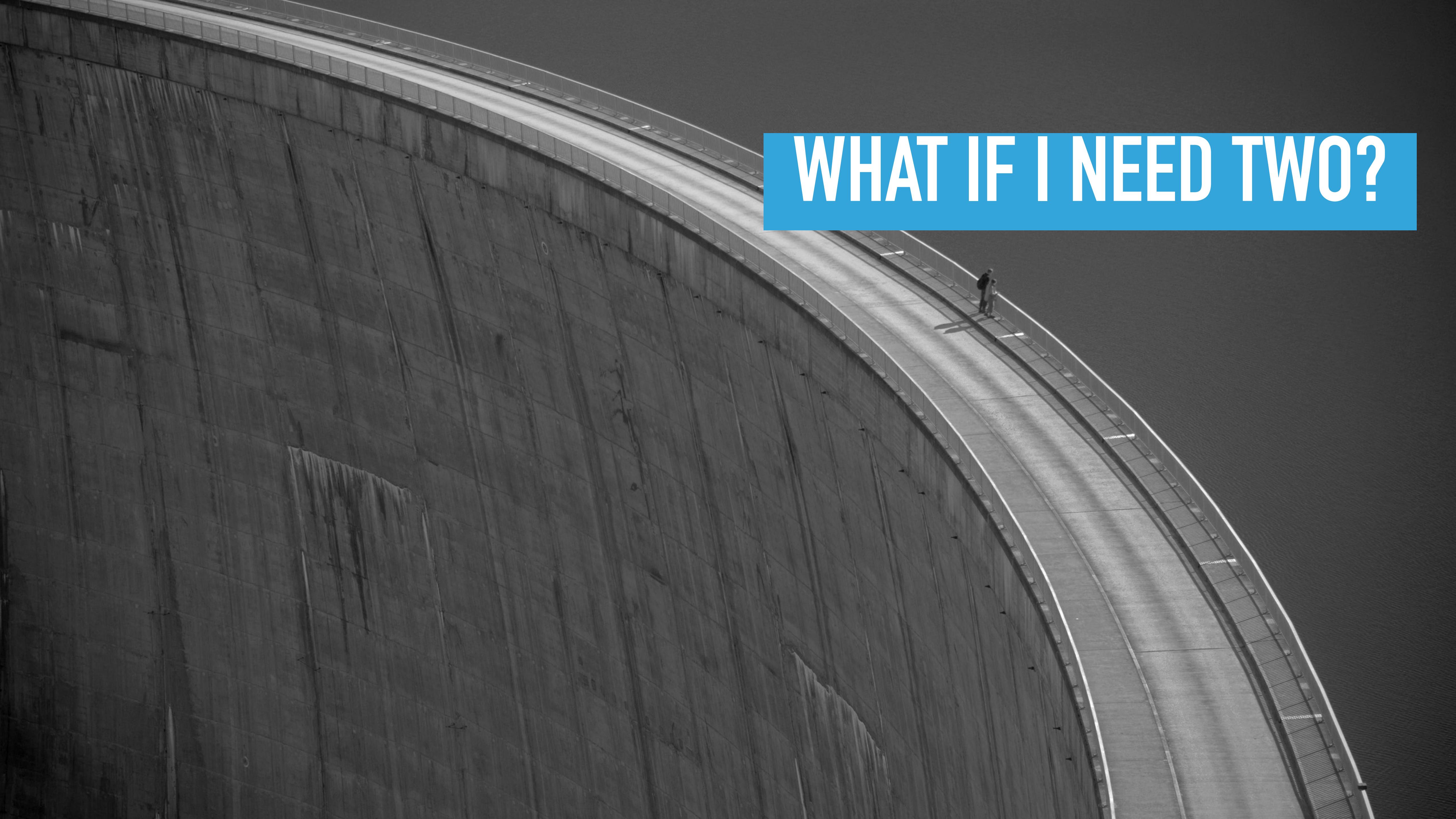
- ▶ Will fail if image is in use

```
presentation — davidshaffer@Kevin11 — .....
```

```
[> docker rm 02c7e7395a2b      presentation ]|⌘
02c7e7395a2b
>                                presentation
```

```
presentation — davidshaffer@Kevin11 — ..//presentation...
```

```
[> docker rmi 7edc6ed989aa      presentation ]|⌘
Untagged: presentation-test:1.0
Deleted: sha256:7edc6ed989aabf30d9e5a42b34b4fe99797a0f
f94b2bde645d1b11bcc7cc524a
>                                presentation
```



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>



THANK YOU

David Shaffer