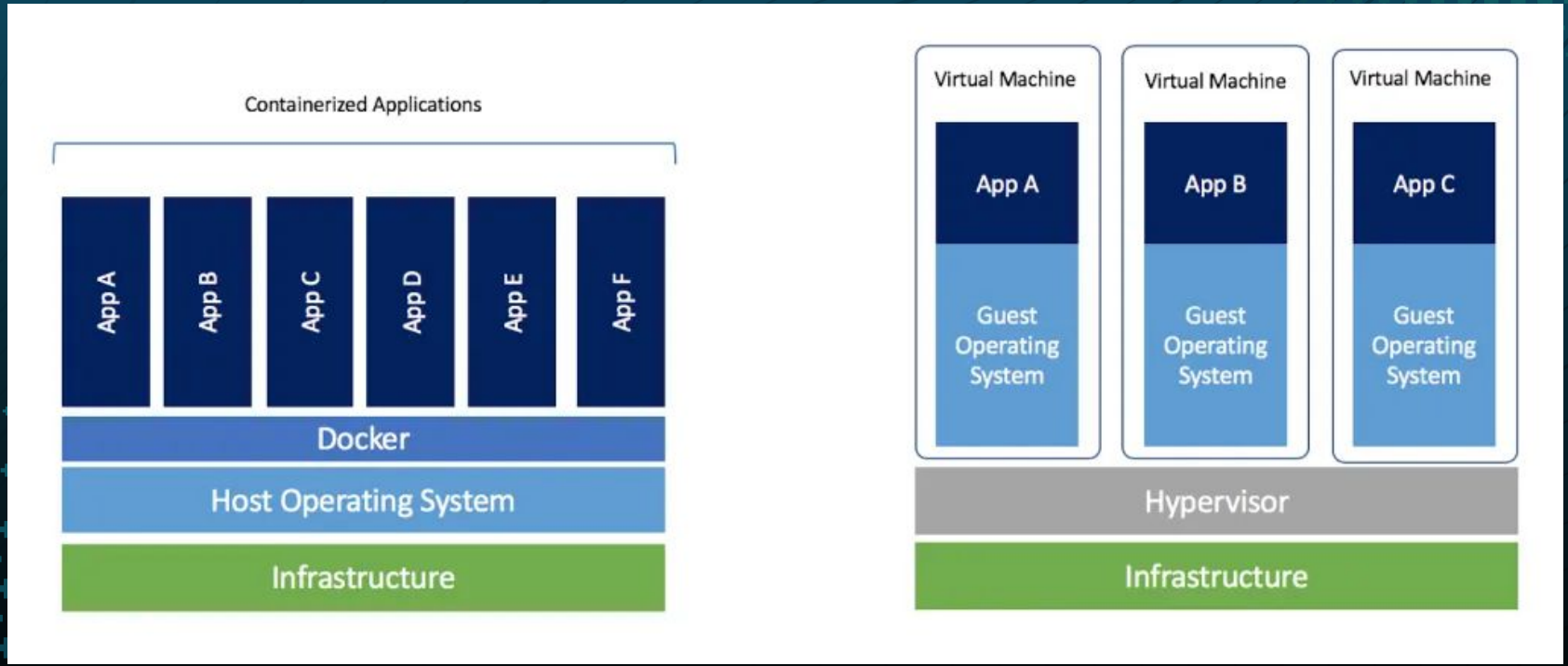
Docker 101

What a pretty container! =)

Contents

01. WTF? Containers? -- Intro

Docker vs Virtual Machine



https://www.docker.com/blog/containers-replacing-virtual-machines/

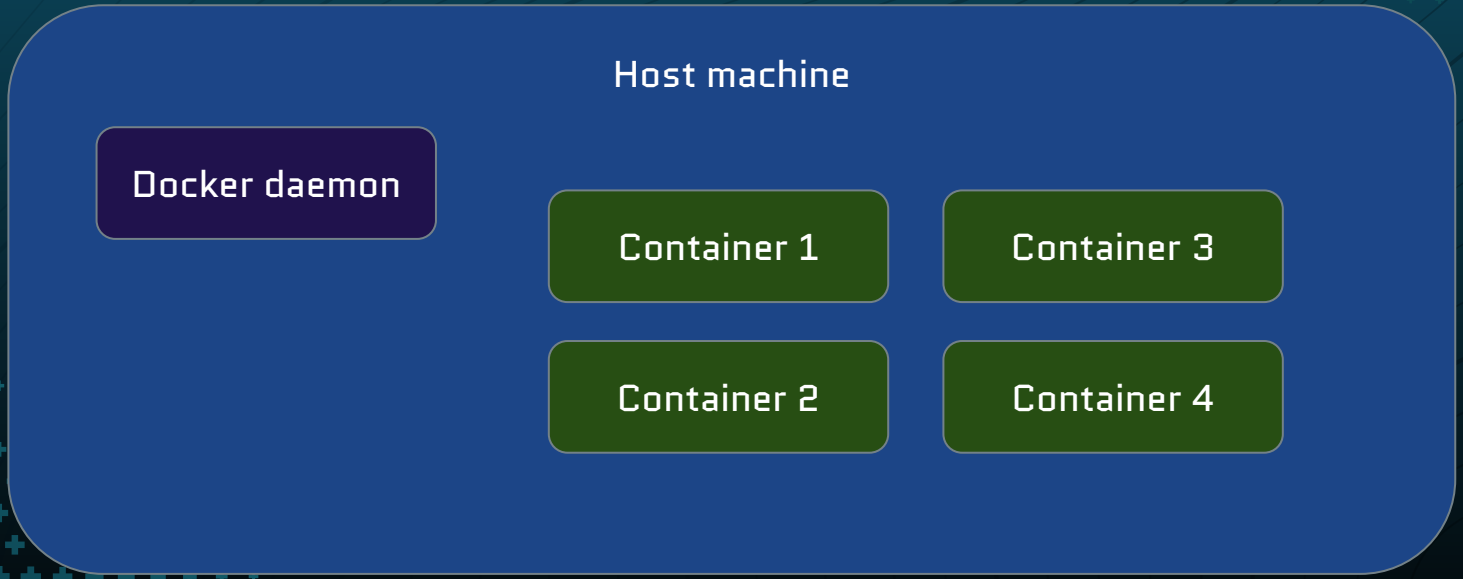
Image vs container

Like “class” vs “object” - one is the instance of another.

Images are like *snapshots* - immutable states of the system. We create *container* based on/from the *image*.

Purpose of containers: isolate application, make prepared environment and avoid “dependency hell”.

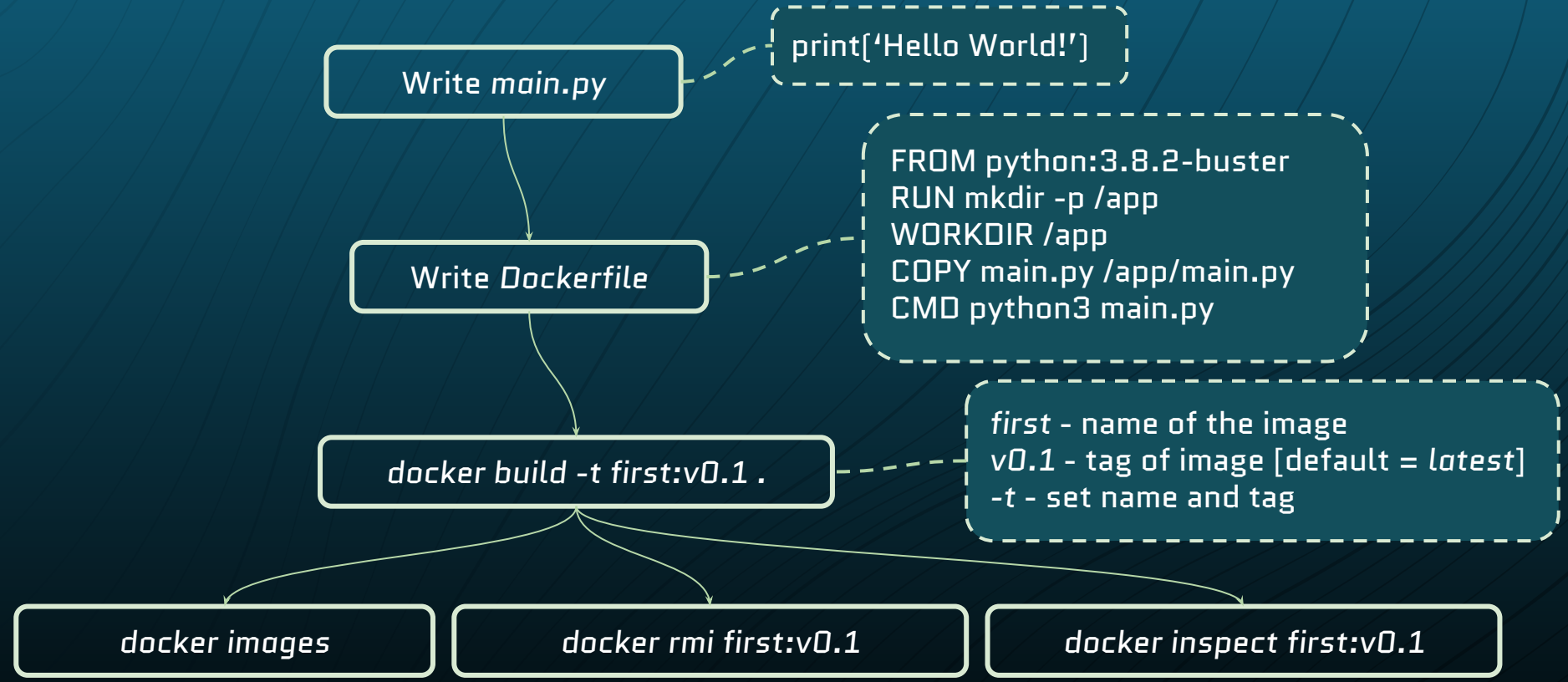
Host system vs container



Information sources

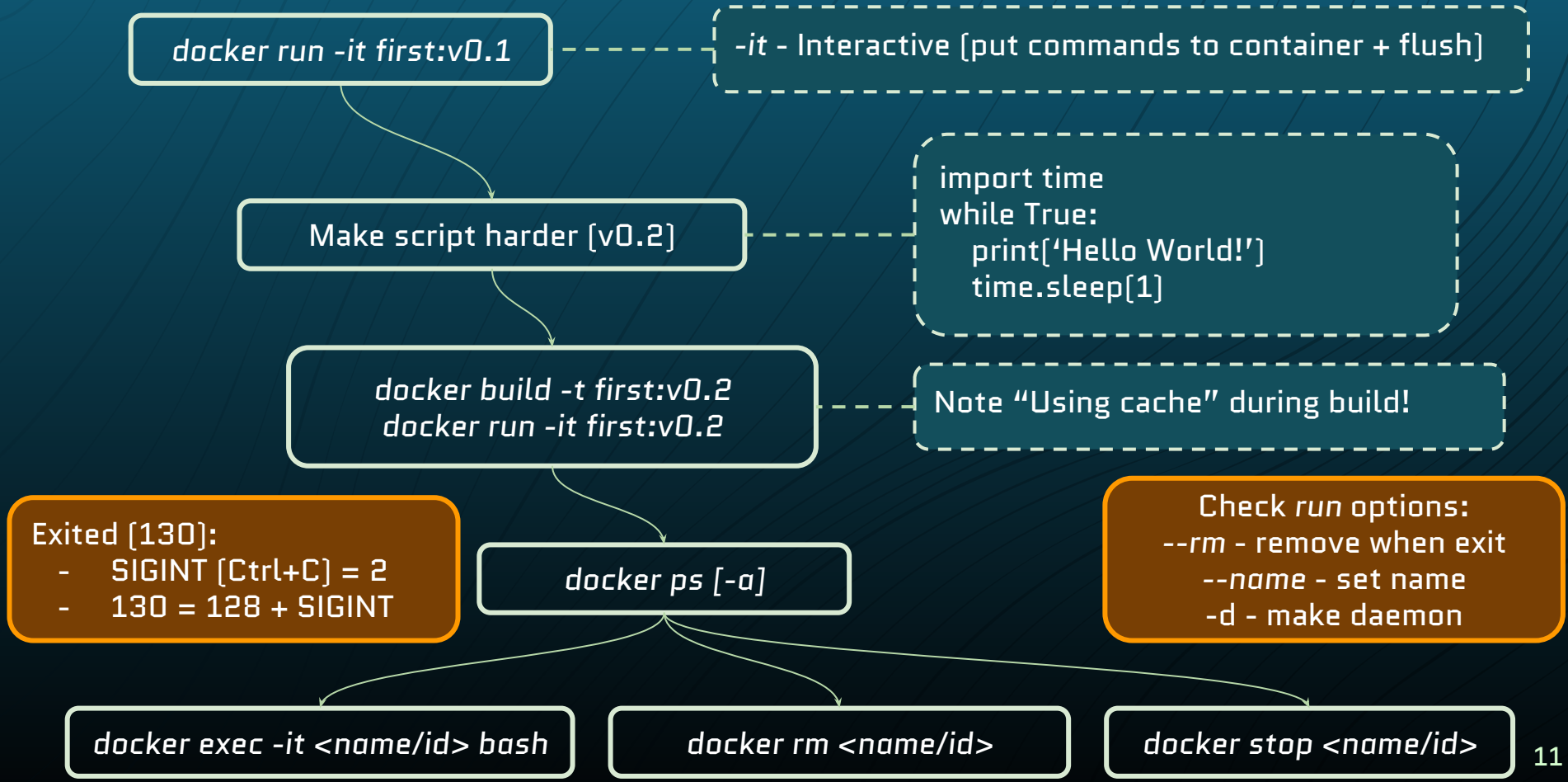
- --help
- <Tab>
- man
- Dockerfile: https://docs.docker.com/engine/reference/builder/
- Docker-compose reference: https://docs.docker.com/compose/compose-file/

02. My first image



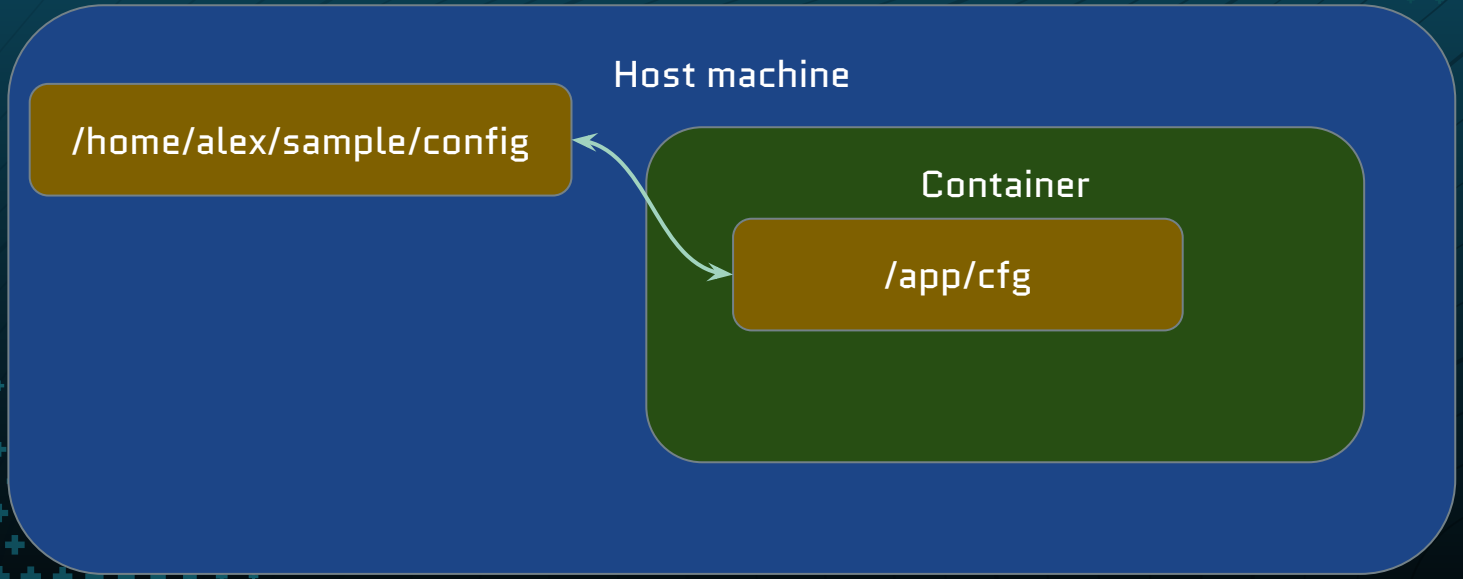
Test 'buster' vs 'slim': https://hub.docker.com/_/python?tab=tags

03. It`s alive!


docker run -it first:v0.1
-it - Interactive (put commands to container + flush)
Make script harder (v0.2)
import time
while True:
print('Hello World!')
time.sleep(1)
docker build -t first:v0.2
docker run -it first:v0.2
Note "Using cache" during build!
Exited (130):
- SIGINT (Ctrl+C) = 2
- 130 = 128 + SIGINT
docker ps [-a]
Check run options:
--rm - remove when exit
--name - set name
-d - make daemon
docker exec -it <name/id> bash
docker rm <name/id>
docker stop <name/id>

04. Mounting, forwarding and other smart words

Folder mounting



Let`s try config sample (v0.3)

Folder mounting (options for *run*):

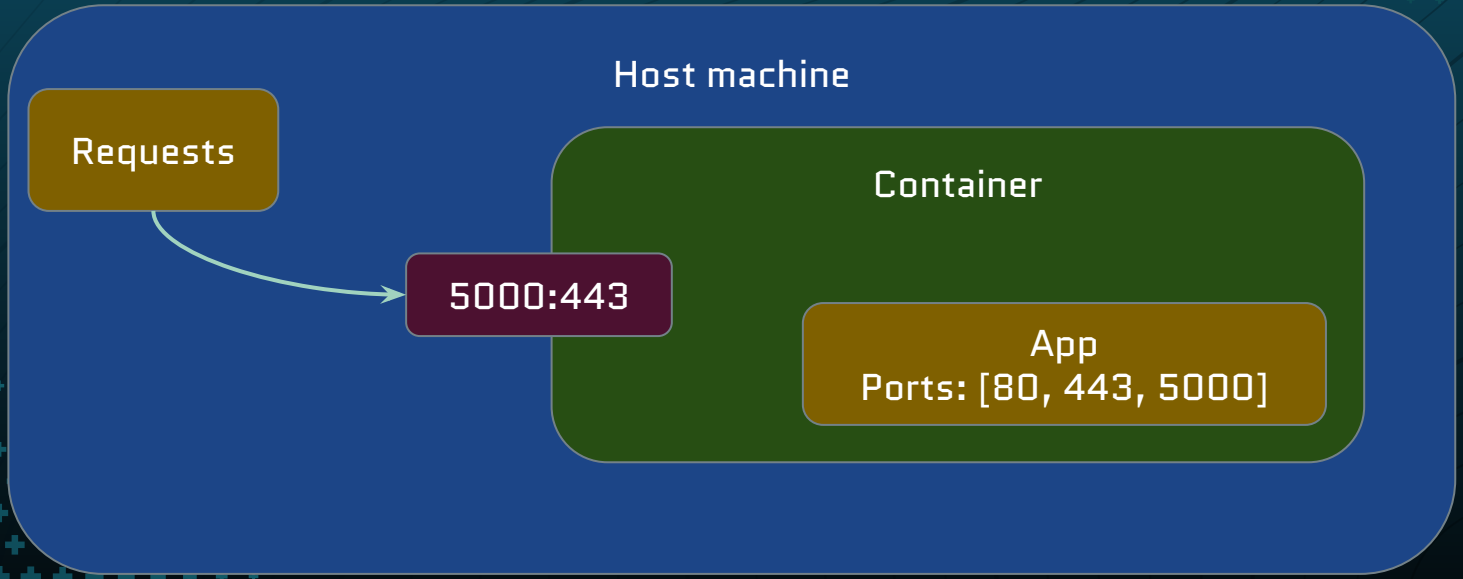
-v <host_dir>:<container_dir>

Test it:

-v $(pwd)/config:/app/cfg

-v $(pwd)/config/config.json:/app/cfg/config.json

Port forwarding



Let`s try network sample (v0.4)

Port forwarding (options for *run*):

-p <host_port>:<container_port>

Test it:

-p 8080:8080

-p <another_port>:8080

* **EXPOSE** in Dockerfile: only for container documentation

05. Docker registry

DockerHub

- Main site: https://hub.docker.com/
- Registry - remote storage for images (like GitHub for git)
- Local config: `$HOME/.docker/config.json`

- `docker login`
- `docker push <username>/<name>:<tag>`
- `docker pull <username>/<name>:<tag>`

Some practice

- Create image with name:

  <username>/first:registry

- Login to DockerHub
- Push image to registry
- Remove image locally
- Pull image and start it

06. docker-compose - the best to be organized!

Some info

Compose file organizes containers environment.

- Start compose: **docker-compose up [-d]**
- Stop containers: **docker-compose stop**
- Clean resources with environment: **docker-compose down**
- Show states in environment: **docker-compose ps**
- Pull images from registry (**image** tag): **docker-compose pull**
- Build containes (**build** tag): **docker-compose build**

07. Questions

Some good practices

- Each last container is tagged twice:

  docker build -t first:v0.1 .

  docker tag first:v0.1 first:latest

- Use git hash:

  TAG=v0.1-$(git log -1 --pretty=%h)

  docket build -t first:$TAG .

Some good practices

- Use scripts:

```
NAME=first
VERSION=v0.1
TAG=$VERSION-$(git log -1 --pretty=%h)
IMG=$NAME:$TAG
docker build -t $IMG .
docker tag $IMG $NAME:latest
```

Questions

Autocompletion:

"Enable Docker command-line auto completion in bash on Centos/Ubuntu" by ismail yenigül https://link.medium.com/RDALJOxqo5

Swarm vs compose:

Docker Swarm / k8s - production

docker-compose - development

Thanks to everyone!