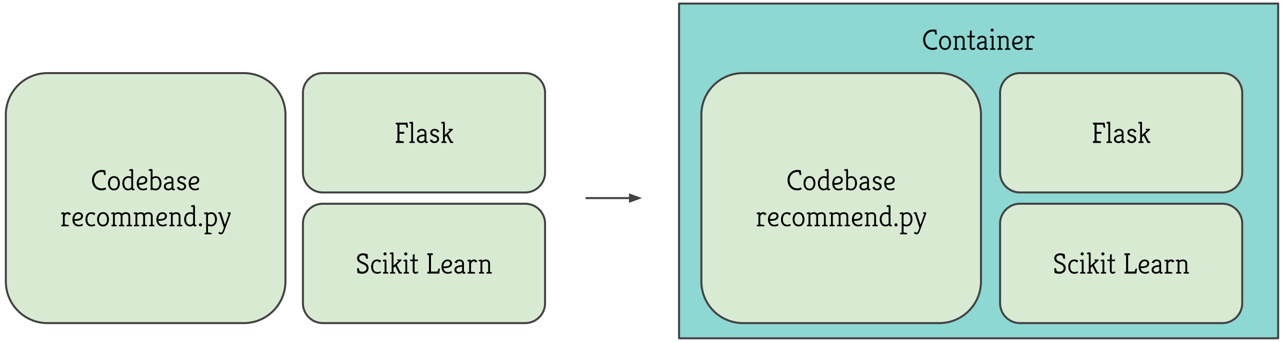
Recitation 9

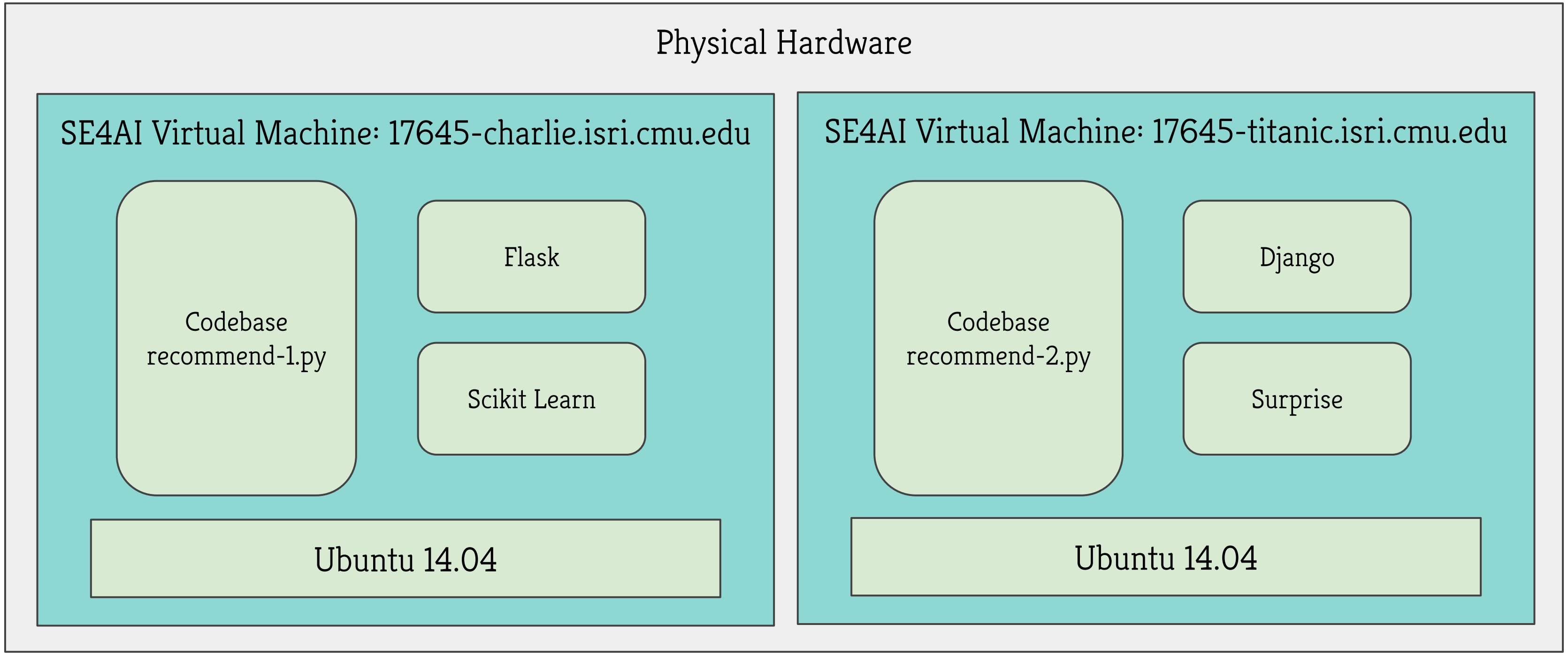
Introduction to Docker

Containerization

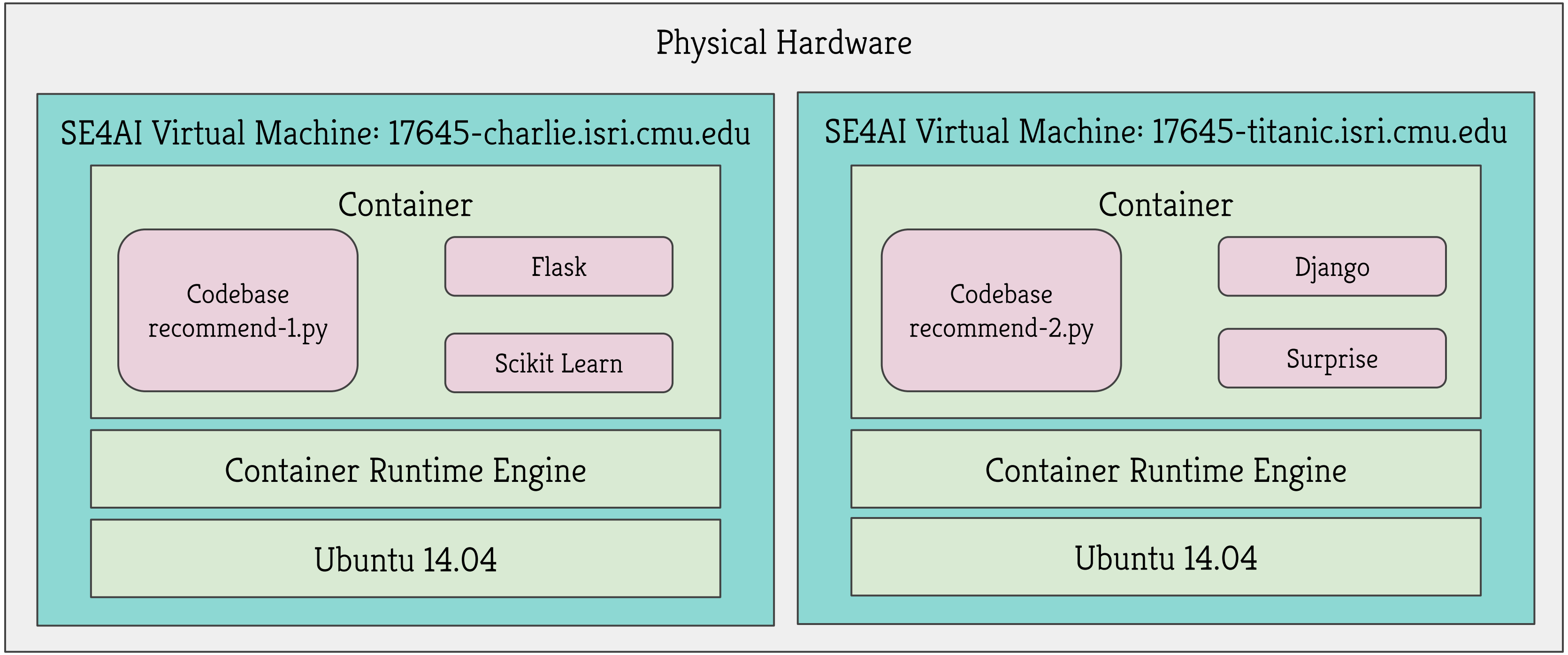
- A way to encapsulate (or) package an executable such that
    - It is isolated from other executables
    - It is easy to move along with its dependencies
    - It is a standalone executable that can be deployed and run independently
    - It is lightweight in terms of loading and transporting



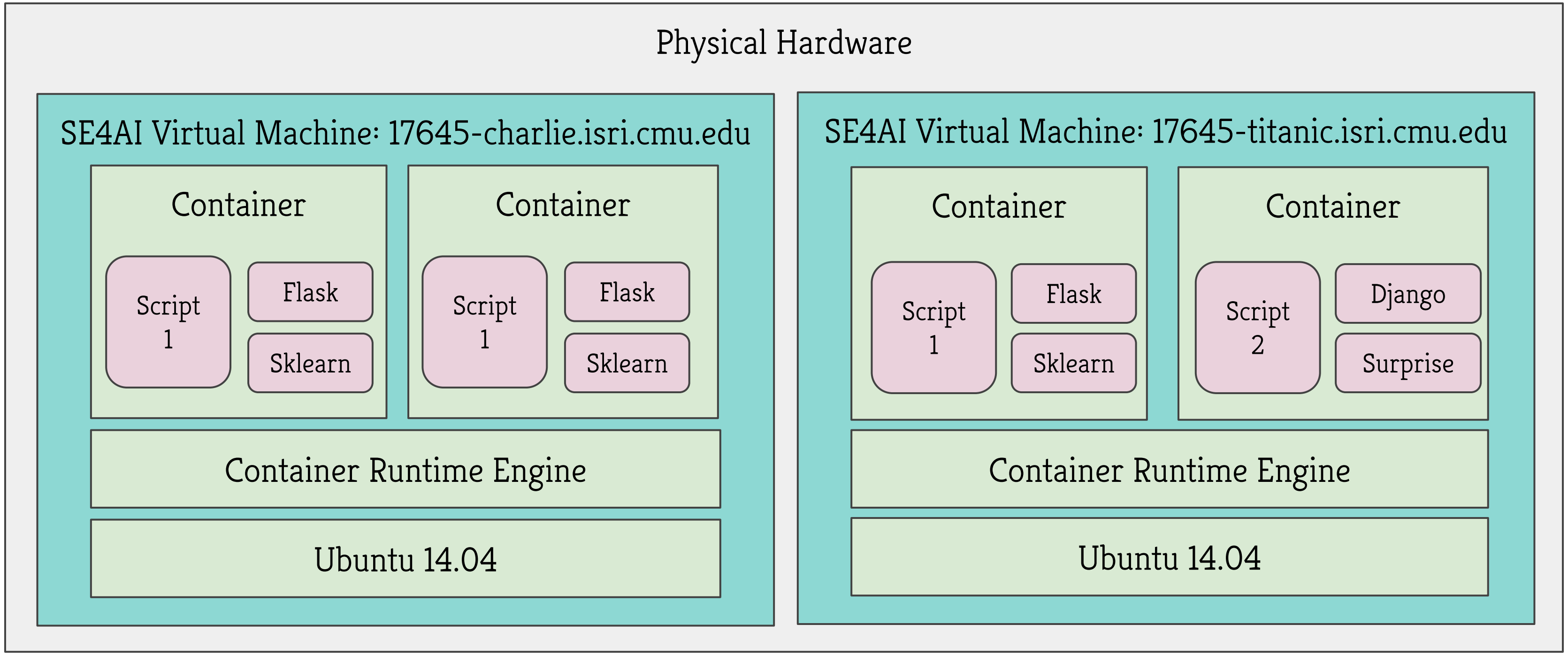
Containerization



Containerization



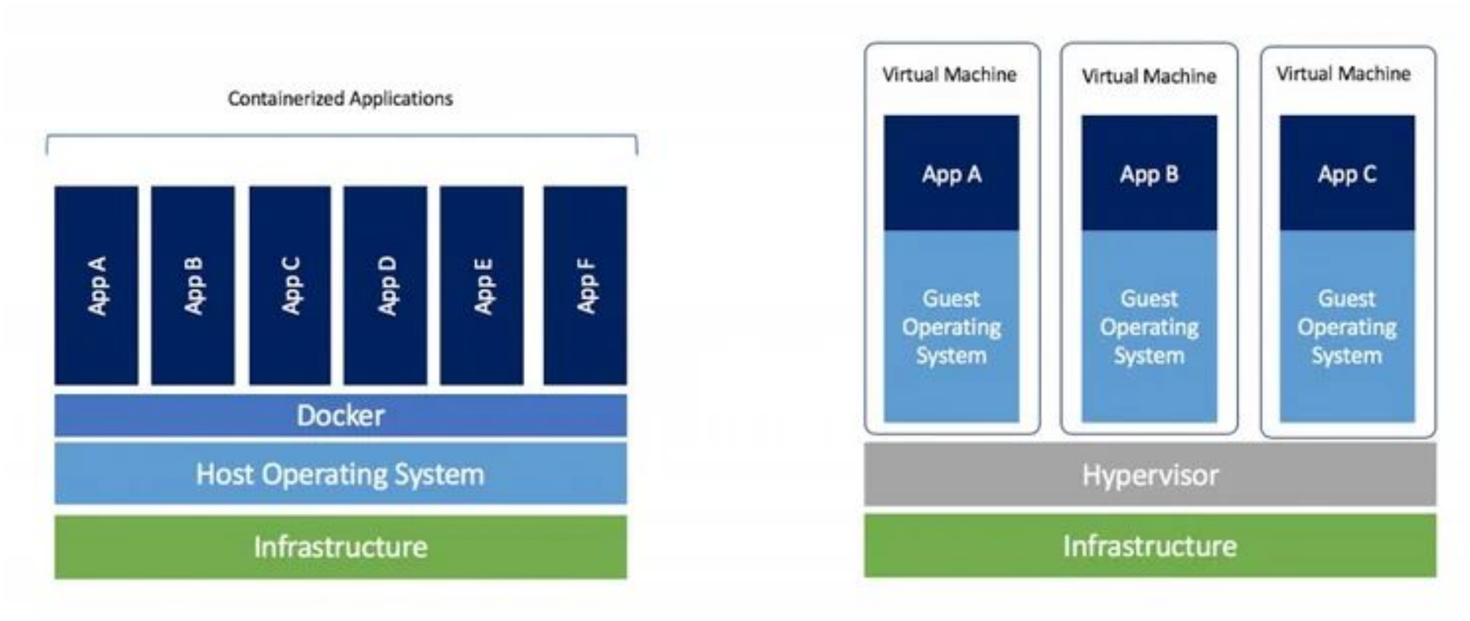
Containerization



Virtual Machine vs Container

- Similarities
  - Both encapsulate your application
- Differences
  - Size:
    - Containers are smaller in size because they do not contain the OS
    - Containers are easier to transport over the network (same reason as above)
  - Portability:
    - VMs are more portable (OS comes along with the VM)
    - Containers are portable as long as the container runtime supports the same format, ie. the same runtime engine has to be installed in the machine where it needs to run
  - OS:
    - Containers are constrained by the operating systems available; VMs are not
    - Each VM has its own operating system; Containers share OS of the host machine
  - State
    - Containers are stateless by default (can be made stateful, although not recommended)

Virtual Machine vs Container



Source: https://www.sdxcentral.com/containers/definitions/containers-vs-vms/

Docker

- Platform-as-a-service product
- Uses OS-level virtualization
- Used to deliver software in packages called “containers”
- Terminologies:

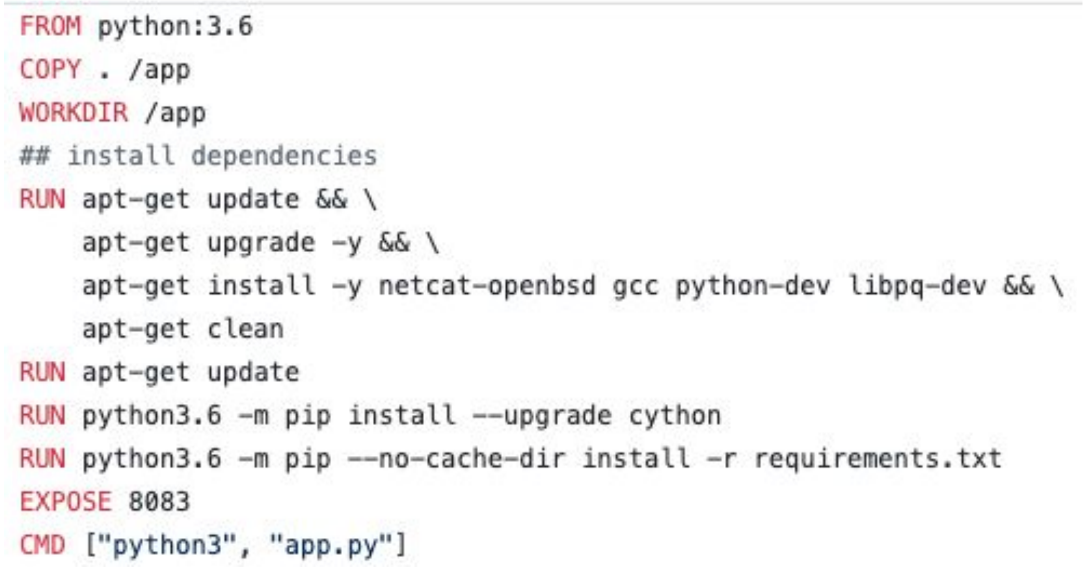
Image	- Set of bits (static executable file)
Container	- A running executable
Dockerfile	- Define how to create an image
Docker Compose	- Configuration for running multiple containers together
Docker Swarm	- Container orchestration tool (manage containers across multiple hosts)
Docker Service	- Similar to Docker Container (but in swarm mode)
Docker Stack	- Group of services (swarm equivalent of docker compose)

- Commands: https://docs.docker.com/reference/
- Installation: https://docs.docker.com/engine/install/ubuntu/

Demo

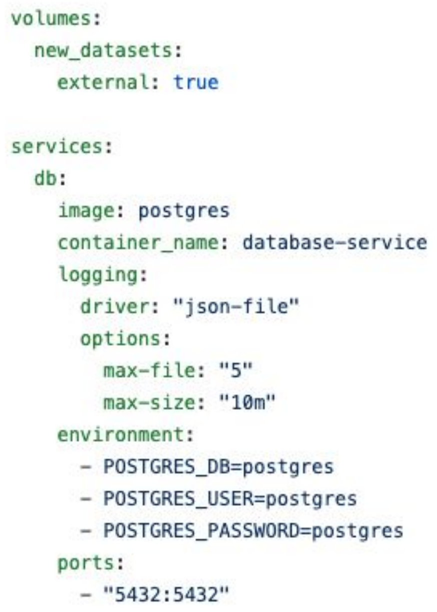
- Contents of Dockerfile
- Contents of docker-compose.yml
- Create a new docker image
- Run a docker image
- Check the logs of a running container
- Log in to a running database container, and query the database
- List docker images in local cache, and running containers
- Run docker compose to build and run a new container

Dockerfile



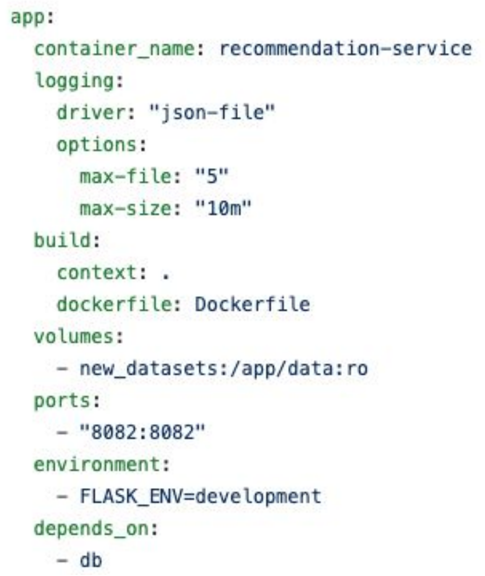
```
FROM python:3.6
COPY . /app
WORKDIR /app
## install dependencies
RUN apt-get update && \
    apt-get upgrade -y && \
    apt-get install -y netcat-openbsd gcc python-dev libpq-dev && \
    apt-get clean
RUN apt-get update
RUN python3.6 -m pip install --upgrade cython
RUN python3.6 -m pip --no-cache-dir install -r requirements.txt
EXPOSE 8083
CMD ["python3", "app.py"]
```

Docker Compose



```yaml
volumes:
  new_datasets:
    external: true

services:
  db:
    image: postgres
    container_name: database-service
    logging:
      driver: "json-file"
      options:
        max-file: "5"
        max-size: "10m"
    environment:
      - POSTGRES_DB=postgres
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
    ports:
      - "5432:5432"
```



```yaml
app:
  container_name: recommendation-service
  logging:
    driver: "json-file"
    options:
      max-file: "5"
      max-size: "10m"
  build:
    context: .
    dockerfile: Dockerfile
  volumes:
    - new_datasets:/app/data:ro
  ports:
    - "8082:8082"
  environment:
    - FLASK_ENV=development
  depends_on:
    - db
```

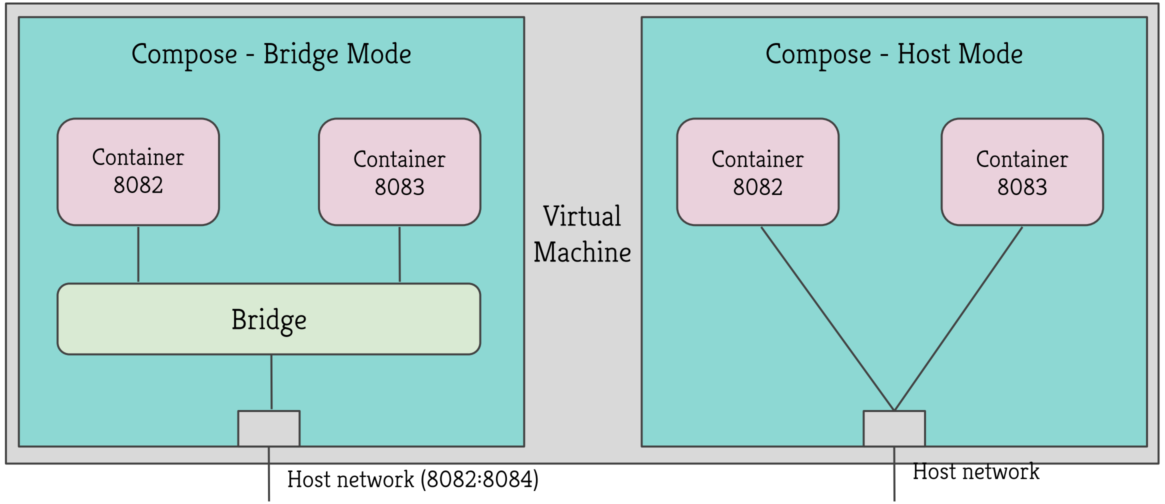
Tips:

Check out the “healthcheck” property in docker compose to ensure that the database is running before starting the recommendation service

Always limit logs to prevent “Out of memory” errors in your VM

Docker Networking (Single Host)

- Bridge network
  - Default, virtual private network between containers
  - Containers within the same bridge network can be accessed by each other using the service name
  - Use "-p" option to expose a port outside the bridge network (called port forwarding)
- Host network
  - Connect containers to host's network directly (same IP as the host)



Docker Commands

docker pull <image_name>	- Download an image from artifactory to the local cache
docker build -t <image_name> .	- Build a new image and store it in the local cache
docker run [options]	- Start a new container from an existing image
docker image ls	- List all images in the local cache
docker container ls	- List all containers in the local cache
docker container rm <container_name>	- Remove a running container
docker stop <container_name>	- Stop a running container
docker exec -it <container_id> /bin/bash	- Log in to a running container
docker cp	- Copy content to/from a running container
docker container logs <container_id> -f	- Display the application logs of a running container
docker-compose up -d --build	- Start all containers from docker-compose.yml
docker-compose down	- Remove all running containers started from compose

- ..
- More commands - https://docs.docker.com/reference/

More Resources

- Recitation - Fall 2019 - More on Docker Basics
- Recitation - Summer 2020 - Kubernetes
- Check - “docker networking”
- Check - “docker volume mounting”
- Check - “docker service” and “docker stack”
- Check - “rolling updates” with docker swarm to promote availability
  - https://blog.container-solutions.com/rolling-updates-with-docker-swarm
  - https://capstonec.com/2018/06/28/zero-downtime-deployment-with-docker-rolling-updates/
- Check - “docker layers” and “docker container life cycle”

Thank You!