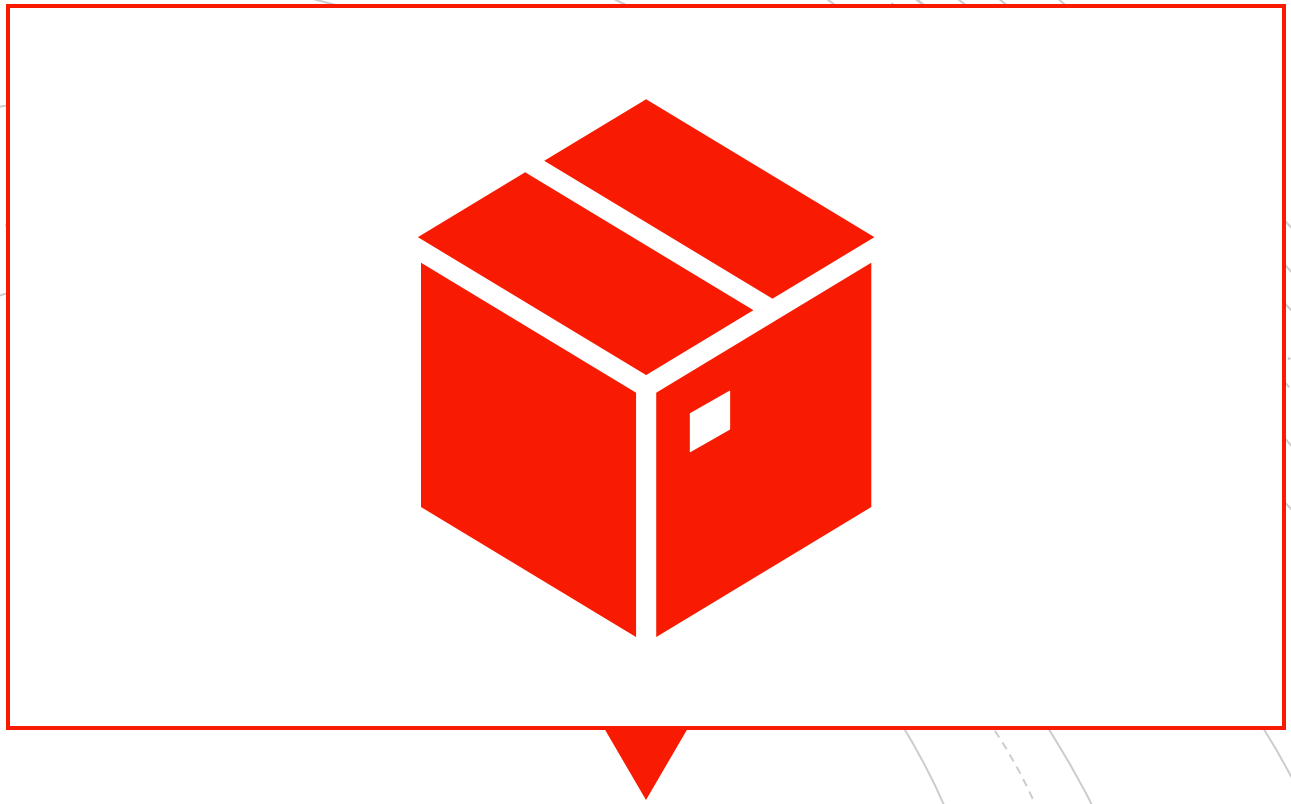
Docker

Session Two

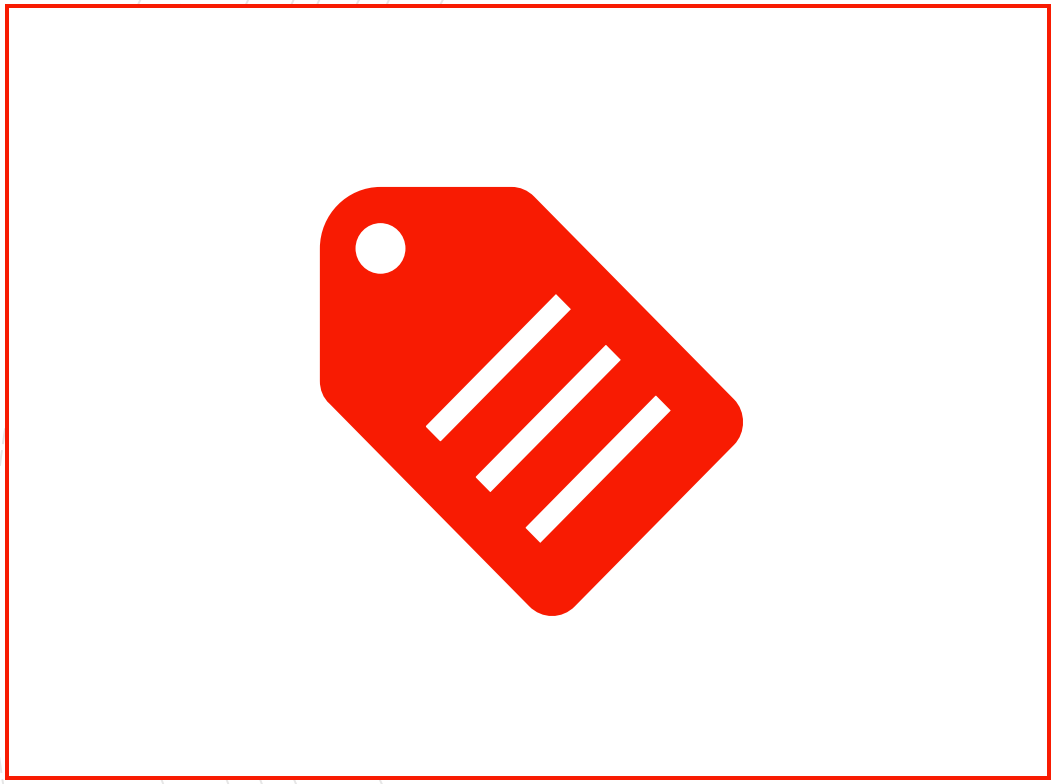
Agenda:

- Before Docker.
- containerization vs virtualization.
- What is Docker and Why?
- Docker Architecture.
- Container Lifecycle.
- Docker Volumes
- Docker Image.
- Docker Networks.
- Docker Compose.



Docker Images

Custom Images



- To create our own image, we can do one of the following:
  - Create a container from your desired image, add and edit
    - whatever you want then commit all these changes to an image.
    - docker commit CONTAINER_ID new_image_name
  - Use Dockerfile

Dockerfile

- Dockerfiles are instructions. They contains all commands used to build an image.
- Each represents a Dockerfile instruction.
- Layers are stacked.
- Each layer is a result of the changes from the previous layer.

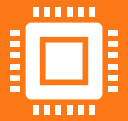
Dockerfile instructions

- FROM: Initializes a new build stage and sets the Base Image
- RUN: Will execute any OS commands in a new layer to build the image
- CMD:
  - Provides a default for an executing container.
  - There can only be one CMD instruction in a Dockerfile.
  - Only the last CMD will have an effect.
- ENTRYPOINT:
  - Allows for configuring a container that will run as an executable
  - There can only be one CMD instruction in a Dockerfile.
  - Only the last CMD will have an effect.
- EXPOSE: Informs Docker that the container listens on the specified network ports at runtime
- ENV: Sets the environment variable <key> to the value <value>
- ADD: Copies new files, directories or remote file URLs from <src> and adds them to the filesystem of the image at the path <dest>.
- COPY: Copies new files or directories from <src> and adds them to the filesystem of the container at the path <dest>.

Dockerfile instructions

- LABEL: Adds metadata to an image
- VOLUME: Creates a mount point with the specified name and marks it as holding externally mounted volumes from native host or other containers
- USER: Sets the username (or UID) and optionally the user group (or GID) to use when running the image and for any RUN, CMD,
- WORKDIR:
  - Sets the working directory for any RUN, CMD, ENTRYPOINT, COPY, and ADD instructions that follow it in the Dockerfile
  - If the WORKDIR doesn't exist, it will be created even if it's not used in any subsequent Dockerfile instruction.
- ARG:
  - An ARG instruction goes out of scope at the end of the build stage where it was defined
  - Defines a variable that users can pass at build-time to the builder with the docker build command, using the --build-arg <varname>=<value> flag
  - is the only instruction that may precede FROM in the Dockerfile.
- ONBUILD: Adds a trigger instruction to the image that will be executed at a later time, when the image is used as the base for another build
- SHELL: Allows the default shell used for the shell form of commands to be overridden

RUN VS CMDVS ENTERYPOINT



RUN. Mainly used to build images and install applications and packages. Builds a new layer over an existing image by committing the results.

CMD. Sets default parameters that can be overridden from the Docker Command Line Interface (CLI) when a container is running.

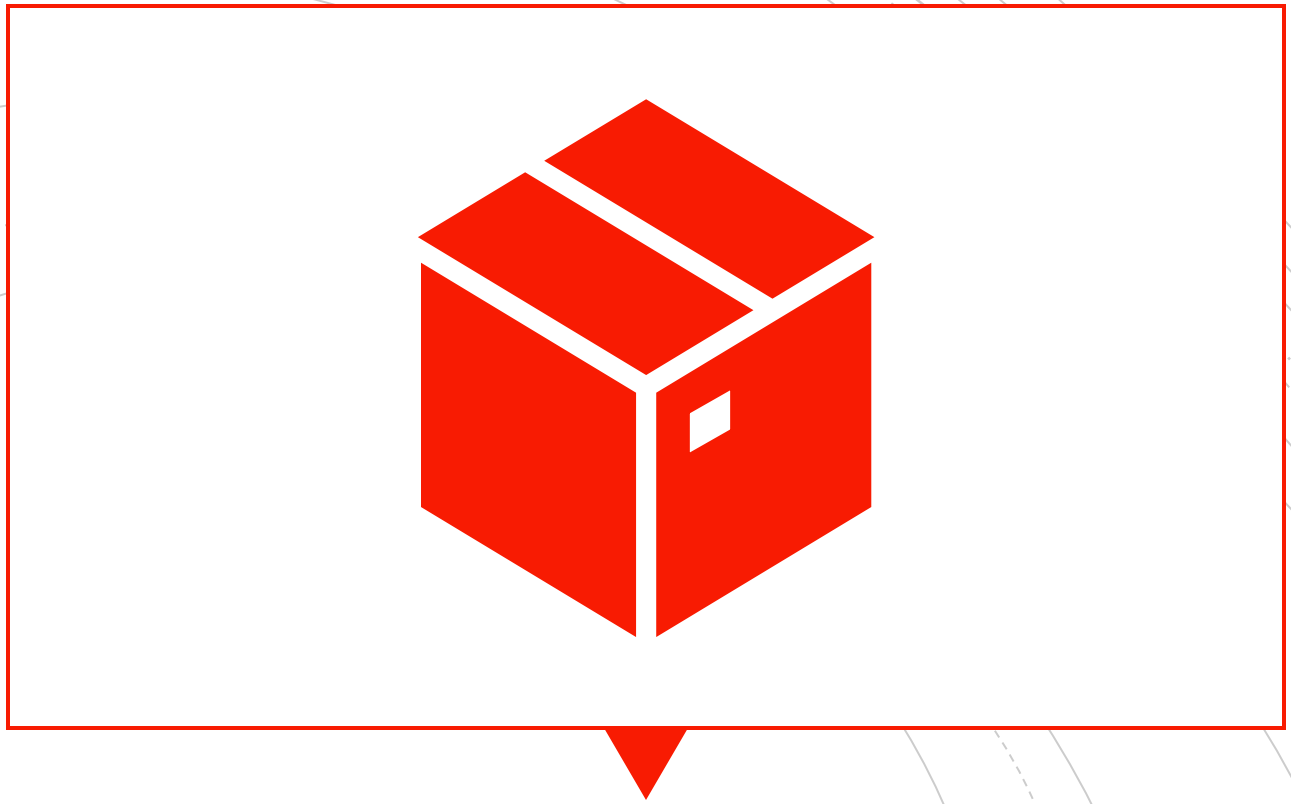
ENTRYPOINT. Default parameters that cannot be overridden when Docker Containers run with CLI parameters.

ADD VS COPY

COPY takes in a src and destination. It only lets you copy in a local or directory from your host (the machine-building the Docker image) into the Docker image itself.

ADD lets you do that too, but it also supports 2 other sources. First, you can use a URL instead of a local file/directory. Secondly, you can extract tar from the source directory into the destination.

Break



Docker Networks

Docker Networks

- Bridge:
  - Private internal network created by docker on the host.
  - Default network the container attach to.
  - Each created container get internal IP address usually in range 172.17.x.x.
  - Containers can access each. Others by this IP
  - To Access any of these containers "PORT MAPPING"
    - By adding –p port-outside:port-inside when running the container

- Host:
  - No type of isolation between docker container and docker host
  - No need to "PORT MAPPING"
  - Only one docker container can be access

Docker Networks

- None:
  - Docker container not attached to any network
  - It will not be accessible outside even from other containers

- overlay:
  - Create new internal private network that allow all containers to access each others.
  - Docker Swarm can create it.
  - Docker network create --driver overlay --subnet 10.0.9.0/24 name
  - Then attach the container or services to this network during creation

Lab

Lab 2

- P1: Create your own nginx docker image based on ubuntu “NEVER USE FROM nginx”
  - Install nginx
  - Two index.html one as file and another as .tar "/var/www/html"
  - Expose
  - Start
  - Port mapping
- P2: Create react app docker container "using single stage, Multi-Stage Dockerfile"
- P3: What is the rest of Docker Networks ? “Name and Definition”
- P4: Create your bridge network, two containers from ubuntu image with different names and try to ping each other using NAME.

Thank You

Docker

Session Three

Agenda:

- Before Docker.
- containerization vs virtualization.
- What is Docker and Why?
- Docker Architecture.
- Container Lifecycle.
- Docker Volumes
- Docker Image.
- Docker Networks.
- Docker Compose.

Docker Compose

Docker Compose

- Docker Compose is used to run multiple containers as a single service.
- Container named Service
- All services are to be defined in a YAML format.
- compose file name MUST be “docker-compose.yml”

Docker Compose Versions

- Version 1
  - Compose files that do not declare a version are considered "version 1"
  - Do not support named volumes, user-defined networks or build arguments
  - Every container is placed on the default bridge network and is reachable from every other container at its IP address. You need to use links to enable discovery between containers
  - No DNS resolution using container names
- Version 2
  - Links are deprecated. DNS resolution through container names
  - All services must be declared under the 'services' key
  - Named volumes can be declared under the volumes key, and networks can be declared under the networks key
  - New bridge network to connect all containers
- Version 3
  - Support for docker swarm

Docker Compose File Syntax

```
version: '3' # if no version is specified then v1 is assumed.
services: # containers. same as docker run
  service_name1: # service name. this is also DNS name inside network
    container_name: ##container name
    image: # name of the image
    command: # Optional, replace the default CMD specified by the image
    environment: # same as -e in docker run
    ports: # same as –p in docker run
    volumes: # same as -v in docker run
  service_name2:
volumes: # Optional, same as docker volume create
networks: # Optional, same as docker network create
```

Docker Compose Commands

- docker-compose up => build services
- docker-compose kill => Kill the containers
- docker-compose logs => Show the logs of the containers
- docker-compose down => Stop and remove containers and networks
- docker-compose rm => Remove stopped containers

Lab

Lab 3

- P1: Convert the created react app multi-stage docker image into compose format.

- P2: Create flask app to count number of visits to browser:
  - Create new directory called flask then add app.py and requirements.txt files
  - Create Dockerfile for the python app
  - Create docker-compose for the app and use Redis as temp DB.

Thank You