Docker Onboarding Presentation

Tanish and Peter

April 18, 2025

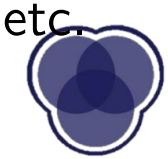
Table of Contents

What is Docker

- Docker is a platform that aims to ease the development, deployment, and management of applications across different environments
- It does this through **containers**, which are isolated environments that contain all dependencies to run an application
- Container design allows for the concurrent development of multiple pieces of software, and ensures ease of portability upon deployment

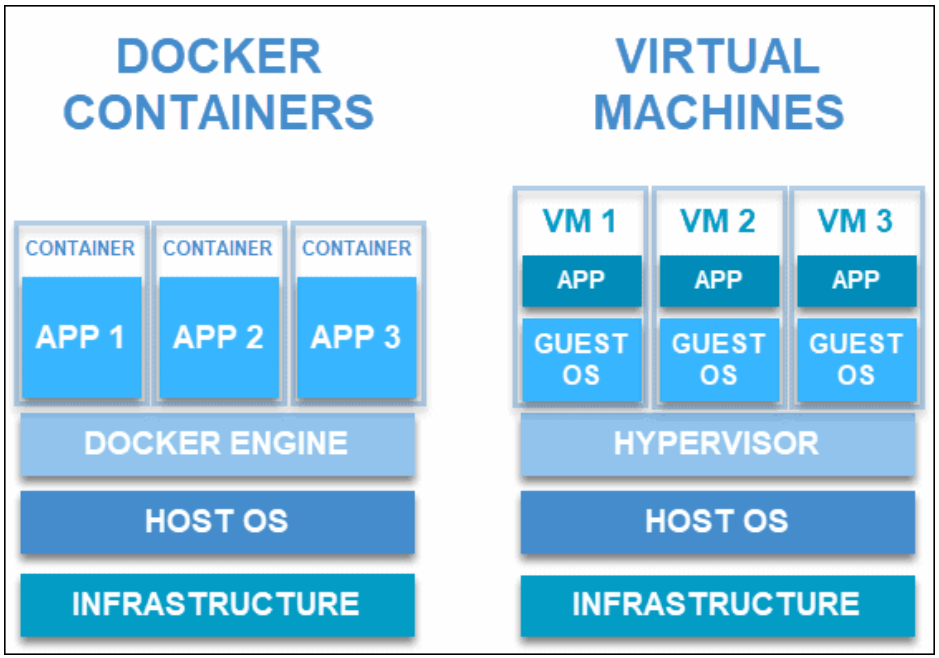
Why is Docker Important

- **Universal environment** - an application will run the same way regardless of client operating system
  - Containers encapsulate dependencies an application needs, reducing the risk of version conflicts and ensuring that applications run with the required libraries and tools
  - Beyond operating systems, this simplifies the migration of applications between different cloud providers, data centers, and more
- **Version control** - prior editions of Dockers are versioned, making production rollbacks seamless and increasing application stability
- **Isolation & security** - vulnerabilities and security breaches are contained within a singular container (even when multiple containers share the same infrastructure)
  - Extends to multiple processes, access to host system, file system, etc.



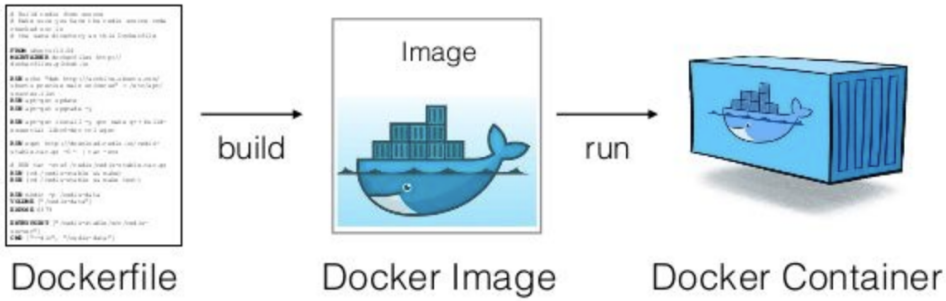
Benefits of Docker in ML

- **Scalability** - in conjunction w/ other DevOps platforms (e.g. Kubernetes), Docker has automated scaling & load-balancing services
- **Fast deployment** - the generalizability of Dockers make it very easy to serve models created w/ any ML framework
  - Allows for facilitation of CI/CD methodology to effectively train, test, and deploy ML apps

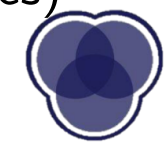


- **Lightweight** - containers share the host OS on a single server as opposed to virtual machines
  - This decreases their boot times and resource consumption, saving $$$ on cloud services
- **Open-source** - many ML projects have a core set of dependencies

Docker Architecture



- **Image** - Read-only "template" that a container runs
- **Dockerfile** - set of script commands for building an image
- In an image, you add dependencies you want to include in the container via the Dockerfile
- Once you the image is created, you can run it on any machine that has Docker installed
  - Instance of container is created from image; **Daemon** processes all requests within image (i.e. spinning up containers or storing images)
- **Docker Hub** - public database for users to share images



Basic Syntax - Creating Dockerfiles

- First - install Docker on either Mac, Windows, & Linux (`docker run hello-world` to check installation is working)
  - Creating Dockerfiles (save in root directory)
    - **Specify base image** - `FROM python:3.11.6`
    - **Set working directory & copy files into image** - `WORKDIR /src/app`, `COPY . .`
    - **Install dependencies** - `pip install --no-cache-dir -r requirements.txt`
    - **Specify port** - `EXPOSE 8888`
    - **Set running command** - `CMD ["python", "./app.py"]`
- **Build image** - `docker build [username/dir]`

```
$ docker build -t yourusername/catnip .
Sending build context to Docker daemon 8.704 kB
Step 1 : FROM python:3.8
# Executing 3 build triggers...
Step 1 : COPY requirements.txt /usr/src/app/
 ---> Using cache
Step 1 : RUN pip install --no-cache-dir -r requirements.txt
 ---> Using cache
Step 1 : COPY . /usr/src/app
 ---> 1d61f639ef9e
Removing intermediate container 4de6ddf5528c
Step 2 : EXPOSE 5000
 ---> Running in 12cfcf6d67ee
 ---> f423c2f179d1
Removing intermediate container 12cfcf6d67ee
Step 3 : CMD python ./app.py
 ---> Running in f01401a5ace9
 ---> 13e87ed1fbc2
Removing intermediate container f01401a5ace9
Successfully built 13e87ed1fbc2
```

Basic Synax - Running Images & Monitoring

- **Fetching from hub** - `docker pull [image name:version]`
- **Running Image** - `docker run -p 8888:5555 [username/dir]`
- Monitoring Dockers
  - **Memory management** - `docker stats [image name]`
  - **Active containers** - `docker -a`
  - **Running processes** - `docker top`
  - **Locally available images** - `docker images`
- **Stopping containers** - `docker stop [container id]`

Real World Examples

- Primary use of docker in real-world & CDS is for **containerization of micro-services**
- Spotify - use ML for **recommendation services**, extraction of high-quality audio signals, **serving ads**, etc.
- Shopify - use ML for **inventory forecasting**, **fraud detection** in transactions, orchestrating email marketing, etc.
- These uses of ML can be packaged into individual micro-service containers that can be developed, tested, & deployed in isolation
- Last year, **HuggingFace** (#1 open-source of ML models) partnered with Docker to allow users to easily integrate HuggingFace models in Docker images