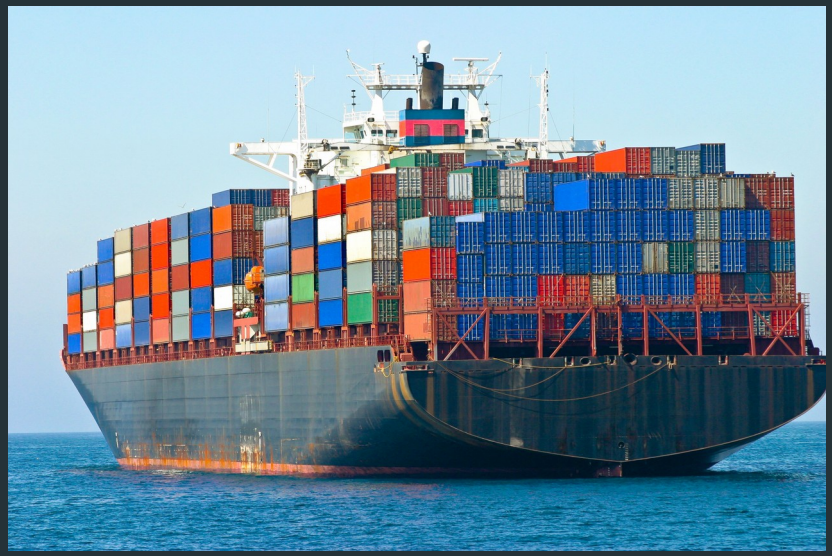
Docker – Containerizing Apps

Summary

- What are containers ?
  - From shipping logistics to virtualization
- Problems docker solves:
  - “It works on my machine”
  - Keeping your versions and sanity in check
  - Acheiving interplanetary scale
- How docker is used locally
- How docker is used in production
- Demo

What are containers?



Container Virtualization

- Power of OSS
  - Originiate from Linux Containers (LXC 2005)
  - Docker released open source in 2012
  - Docker released Docker Enterprise in 2015
  - Today 3,300 contibutors, 43,000 stars on repo
- Layered like an onion
- Distributable

Problem 1: “It works on my machine”

- Programs like web servers, databases, programming languages setup differently in other environments
  - Location of files
  - Different configs by default
- Any changes that make the environments slightly different could cause unexpected affects from system to system.
  - Modified a config file for 1 project?
  - Making all systems in an org being exactly that same is difficult.

Problem 2: Keeping your versions and sanity check

- Python 2.6 vs 3, PHP 5 vs 7, Go 1.8 vs 1.11, Mysql 5 vs Mysql 5.6...
  - Code built in 1 version might not be backward compatible
  - Managing several installs of a language is not easy
  - Constantly have to change system to use whatever version you’re developing with
  - Some tools not even available for your system?

Problem 3: Achieving interplanetary scale

- Distributing entire systems with a single image file
  - Minimal setup
  - Minimal startup
  - Transportable
- Deploying apps across data centers is now possible
  - Docker handles networking, dependency installs, restarts
- Writing autoscaling infastructure is made possible with docker
  - Kubernetes orchestration

How is it used locally?

Dockerfile

```
FROM python:3
ENV PYTHONUNBUFFERED 1
RUN mkdir /code
WORKDIR /code
ADD requirements.txt /code/
RUN pip install -r requirements.txt
ADD . /code/
```

Docker Compose

```
version: '3'
services:
  web:
    build: .
    ports:
    - "5000:5000"
    volumes:
    - .:/code
    - logvolume01:/var/log
    links:
    - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

How is it used in production?

- Kubernetes
  - Uses docker images created from dockerfiles
  - Just run the video...
  - How does Kubernetes just “spin up” new instances ? Where does it get these resources from ?
    - Cloud controller managers act as the glue that allows Kubernetes to interact providers with different capabilities, features, and APIs while maintaining relatively generic constructs internally. This allows Kubernetes to update its state information according to information gathered from the cloud provider, adjust cloud resources as changes are needed in the system, and create and use additional cloud services to satisfy the work requirements submitted to the cluster.

Demo time

- Pull repo
  - Oh no, we don’t have Go or Redis installed... how do we run or test?
- Create docker-compose file
- Build image
- Run container
- Run tests inside container

Done