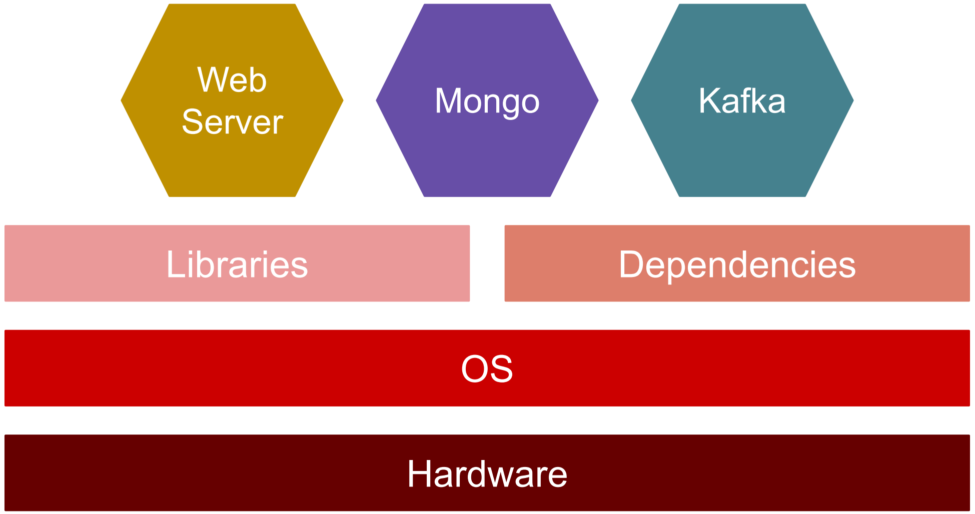
Introduction to Docker

Problem at hand

- App components
  - Web server - NodeJs
  - Database - Mongo
  - Messaging - Apache Kafka
- OS compatibility
  - Components must be compatible with OS
  - Component’s compatibility with OS libraries
- Matrix from Hell
- Developer experience
- Deployment environment

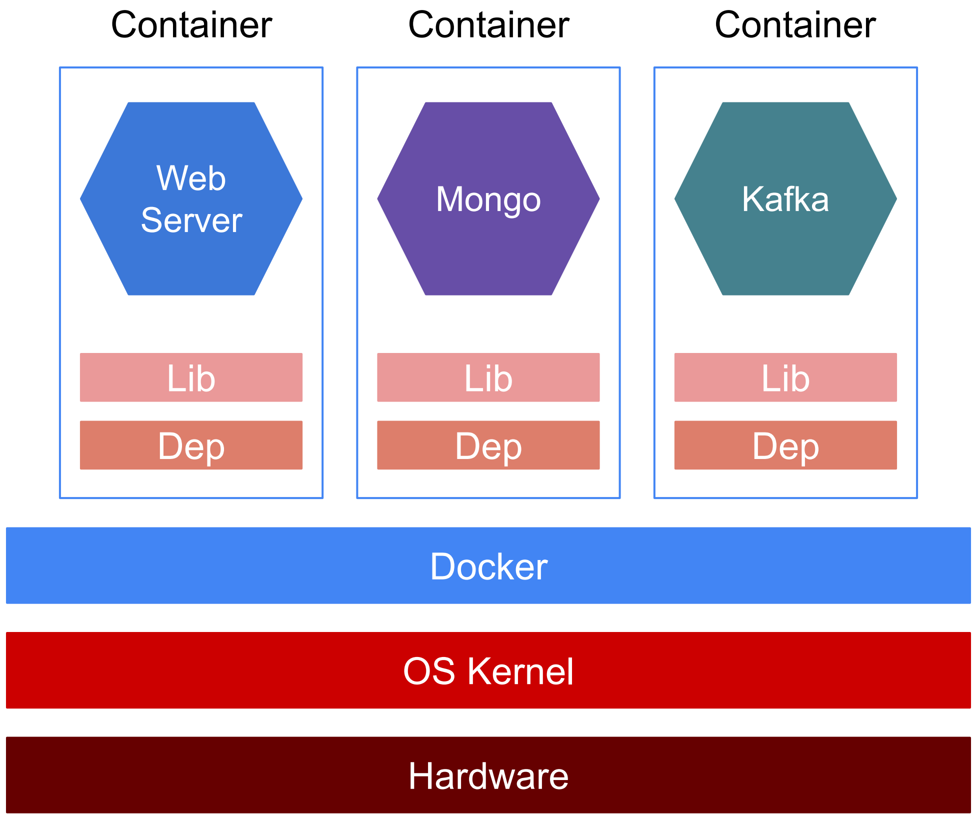
Problem at hand



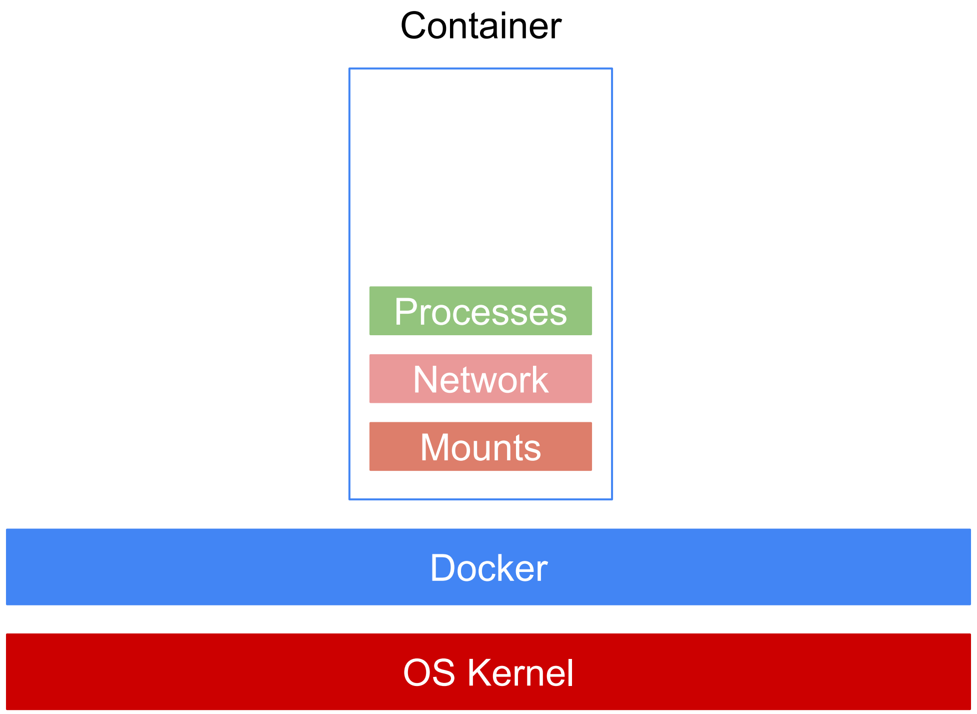
Solution - Docker

- Component as container
  - Own dependencies
  - Own libraries
- Containers run on host machine
- Developer experience
  - Docker installed
- Deployment environments
  - Docker installed

Solution - Docker



Solution - Docker



Docker

- Open platform
- Developing, Running and Shipping apps
- Separates app from infrastructure

Docker

- Container
    - Environments to run apps
    - Lightweight, contain everything required for the app
    - Sharing contains ensure same environment across
- Multi containers on an host

Docker - Container Lifecycle

- Develop app using containers
- Distribute app using containers
- Deploy app using containers
- The app runs same way everywhere
    - Local
    - Testing
    - Deployment

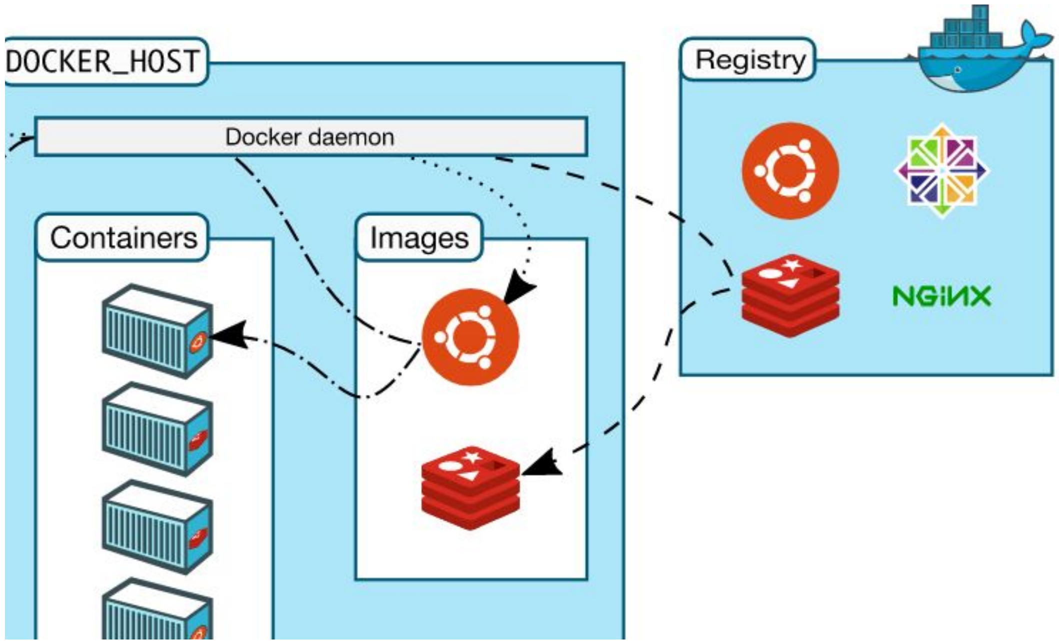
Docker - Architecture

- Client - Server
- Docker
  - CLI Client
- Docker daemon
  - Client interacts with the daemon
  - Daemon executes the commands
  - Client and daemon communicate using REST over UNIX Sockets
- Docker Compose
  - CLI Client
  - Orchestrate multiple containers

Docker - Architecture

- Docker desktop
  - GUI
  - Window / Linux environments
  - Installs docker daemon, docker, docker compose, etc.
- Docker registry
  - Docker image stores
  - Docker Hub
    - Default public store
  - Private registry supported
  - `pull, run`, etc. interact with registry

Docker - Architecture



Docker - Objects

- Images
  - Template with instructions
  - Used for creating container
  - Dockerfile
  - Each instruction is a layer
    - Build only what’s changed
  - Images based on another
  - e.g. install apache web server using ubuntu

Docker - Objects

- Containers
  - Runnable instance
  - Create, start, stop, move and delete containers
  - Connect, execute commands within container
  - Isolated from other containers and host machine
    - Default
    - Configurable
  - When deleted state is lost if not persisted

Self Learning

- Hypervisor vs. Docker
- Containers vs.Virtual Machines

Self Learning

- All content will be removed by next week
- Submit assignments after the test

Docker installation

https://docs.docker.com/get-docker/

Docker - Commands

- Run container
  - `docker run <image>`
  - Pulls image from docker hub first time
  - Subsequent calls uses same image
  - `docker run nginx`

Docker - Commands

- List running containers
    - `docker ps`

```
CONTAINER ID   IMAGE   COMMAND   CREATED   STATUS   PORTS    NAMES
As65147        nginx             7m ago    Up 6m    80/tcp   silly_beast
```

- List all containers
    - `docker ps -a`

Docker - Commands

- Stop container
    - `docker stop <container_name>`
- Remove container
    - `docker rm <container_name>`

Docker - Commands

- List images
  - `docker images`

```
REPOSITORY    TAG       IMAGE ID    CREATED    SIZE
nginx         latest    a894s99     7m ago     18MB
```

Docker - Commands

- Remove image
  - `docker rmi <image_name>`
  - No containers should be running on the image
  - Stop and delete containers

Docker - Commands

- Pull image
  - `docker pull <image>`

Docker - Commands

- Execute command
  - `docker exec <container_name> <command>`
  - `docker exec silly_beast cat /etc/hosts`

Docker - Commands

- Detached mode
    - `docker run <image> -d`
- Attach back
    - `docker attach <container_id>`
    - `docker attach as19s1`

Docker - Run

- Run container using tag
  - `docker run <image>:<version>`
  - Runs a specific image version
  - Latest by default

Docker - Run

- Run container interactively
  - `docker run -it <image>`
  - `-i` - interactive
    - Attaches stdin of host to container
  - `-t` - terminal
    - Attaches the terminal of container to host

Docker - Run

- Run with port mapping
  - `docker run -p <host-port>:<container:port> <image>`

Docker - Run

- Run with volume mapping
- Container has its file system
- Deleting container deletes data
- Persist data by volume mapping
    - `docker run -v <host/directory>:<container/directory> <image>`

Docker - Inspect

- Inspect container
  - `docker inspect <container-name>`

Docker - Container Logs

- Container logs
    - `docker logs <container-name>`

Docker - Building image

- Dockerfile
  - Commands to build an image
  - Usually built on a base image
- Format
  - `INSTRUCTION <arguments>`

Docker - Instructions

- Must begin with `FROM`
    - Parent image to build on
    - `FROM nginx:alpine`

Docker - Instructions

- Set Environment variable with `ENV`
    - Set once, use throughout
    - `ENV <variable>=<value>`

Docker - Instructions

- Execute commands with `RUN`
    - `RUN <command>`
    - `RUN [“command”, “param1”, ...]`

Docker - Instructions

- Execute commands with `CMD`
  - `CMD [“command”, “param1”, ...]`
- Override command during run
  - `docker run ubuntu <command>`
  - `docker run ubuntu sleep 10`

Docker - Instructions

- Define `ENTRYPOINT`
- Appends value
  - `ENTRYPOINT [“sleep”]`
  - `docker run ubuntu 10`
- CMD and ENTRYPOINT can be used in conjunction

Docker - Instructions

- CMD and ENTRYPOINT can be used in conjunction
  - `ENTRYPOINT [“sleep”]`
  - `CMD [“5”]`
  - `docker run ubuntu 10`
- Override entrypoint
  - `docker run --entrypoint delay 10 ubuntu`

Docker - Instructions

- Add new files using `ADD`
    - `ADD <source> <dest>`
    - Source can be directory or URL

Docker - Instructions

- Copy files using `COPY`
  - `COPY <source> <dest>`

Docker - Building image

- Build
  - `docker build -t <image-name> <docker-file-path>`
  - `docker build -t simple-web-app .`
- Run
  - `docker run <image-name>`

Docker - Environment variable

- Pass variable
  - `docker run -e <ENV_VARIABLE>=<VALUE> <image-name>`
- Inspect existing env variable
  - `docker inspect <container-name>`
  - Config section

Docker - Web app

Docker

- `docker run` for single container
- Multiple container orchestration
    - `docker run --name=web-app web-app`
    - `docker run --name=db database`
    - `...`
- The problem

Docker

- Manual linking
    - `docker run --name=db database`
    - `docker run --name=web-app --link db:db web-app`
    - `...`
- Creates host entry for the container
- The value could be used in app code
- Linking is deprecated

Docker Compose

- `Yaml` format
- Sample

```
web-app:
     image: <image>
     ports:
          - 5001:80
     links:
          - db
db:
     image: <image>
...
```

Docker Compose - Version 2

- Services root

```
version: 2
services
     web-app:
          image: <image>
          ports:
               - 5001:80
          links:
               - db
     db:
          image: <image>
     …
```

Docker Compose - Version 2

- Depends on

```
version: 2
services
     web-app:
          image: <image>
          ports:
               - 5001:80
          links:
               - db
          depends_on:
               - db
     db:
          image: <image>
     …
```

Docker Compose - Network

- Network

```
version: 2
services
      web-app:
            image: <image>
            depends_on:
                  - db
            networks:
                  - front-end
      db:
            image: <image>
            networks:
                  - back-end
    …
networks:
      front-end:
      back-end:
```

Docker Engine

- Daemon
- REST API server
- Docker CLI

Docker Engine

- Docker CLI and daemon can be on different systems
    - `docker run -H=<remote-ip-address:with-port> <image>`

Docker Engine

- All container processes run on docker host
- Problem
  - Two processes cannot have same PID
- Namespacing
  - Docker host process PID to container process PID mapping
  - E.g., nginx PID on host is 10, but PID on container is 2

Docker Engine

- Container by default has access to all of host resource
- Restrict resource
  - `docker run --cpu=0.5 <image>`
  - `docker run --memory=100m <image>`

Docker Storage

- Images are layered
- Layers are reused
- Image layers are read-only
- Container creates a new read-write layer
    - Any changes in container are exclusive to the container

Docker Storage

- Create a volume
  - `docker volume create new-volume`
- Its created under volume dir
- Attach
  - `docker run -v new-volume:/container/directory <image>`
- Data is not lost when container is deleted

Docker Networking

- Types
  - Bridge (default)
  - None
    - `docker run --network=none`
  - Host
    - `docker run --network=host`

Docker Networking

- Bridge
  - Private network
  - On host
- Containers get internal IPs
- Containers can access each other using IP
- Port mapping

Docker Networking

- Host
    - Maps directly to host port
    - No port mapping
    - Cannot run multiple web containers like bridge network

Docker Networking

- None
  - Isolated network
  - No access

Docker Networking

- User defined
  - `docker network create --driver bridge --subnet <subnet> <name>`
- List networks
  - `docker network ls`