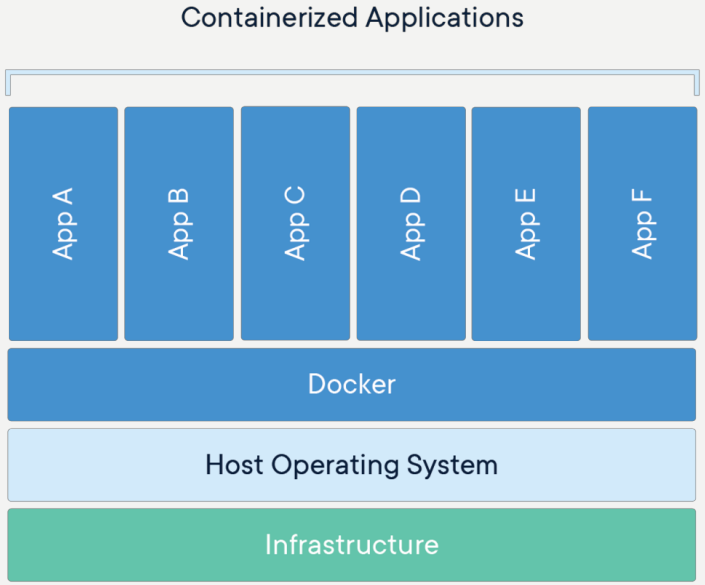
DOCKER

AGENDA

- What is Container?
- What Problem Container Addresses?
- Docker Setup
- Common Terminology

CONTAINER

- Standardized Unit of Software


Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure

TRADITIONAL DEPLOYMENT/DEVELOPMENT PROBLEM

- Runtime Dependencies are managed in different way than Code
  - Manage Platform Specific Dependencies
- Inconsistencies related to Application dependencies across environments
- Deployment Complexity (for system with complex interdependencies)
- Development Environment Lacks adequate environment to identify integration issues
- Difficulty in ensuring consistent version of binaries across every team member system
- Challenges in doing CI due to lack of environment

CONTAINER CHEAP AND EFFECTIVE OPTION

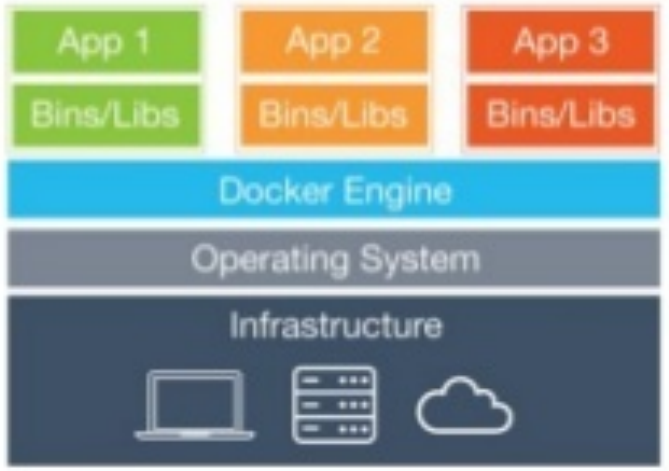
- Container bundles entire runtime dependencies along with code
  - Eliminates the inconsistency issues
  - Easier to manage deployment as all we need to worry now is single artefact which is container image
- Container lightweight option to provide process and runtime isolation than some of the other virtualized options
- Easier to create/destroy environment
- Environment creation is codified and managed as a regular source code

CONTAINER OPTIONS

- Docker
- Core Os rkt (rocket)
- Cloud Foundry Garden
- ECS
- Azure Container Service (ACS)
- LXD
- Linux Vserver

Containers vs. VMs

Virtual Machines



Containers

VIRTUAL MACHINE

Advantages

- Tools associated with VM are easier to access and simpler to work with
- Allows container to be run inside VM

Drawbacks

- Large Image Sizes
- High IO Overhead
- Maintenance Costs

DOCKER

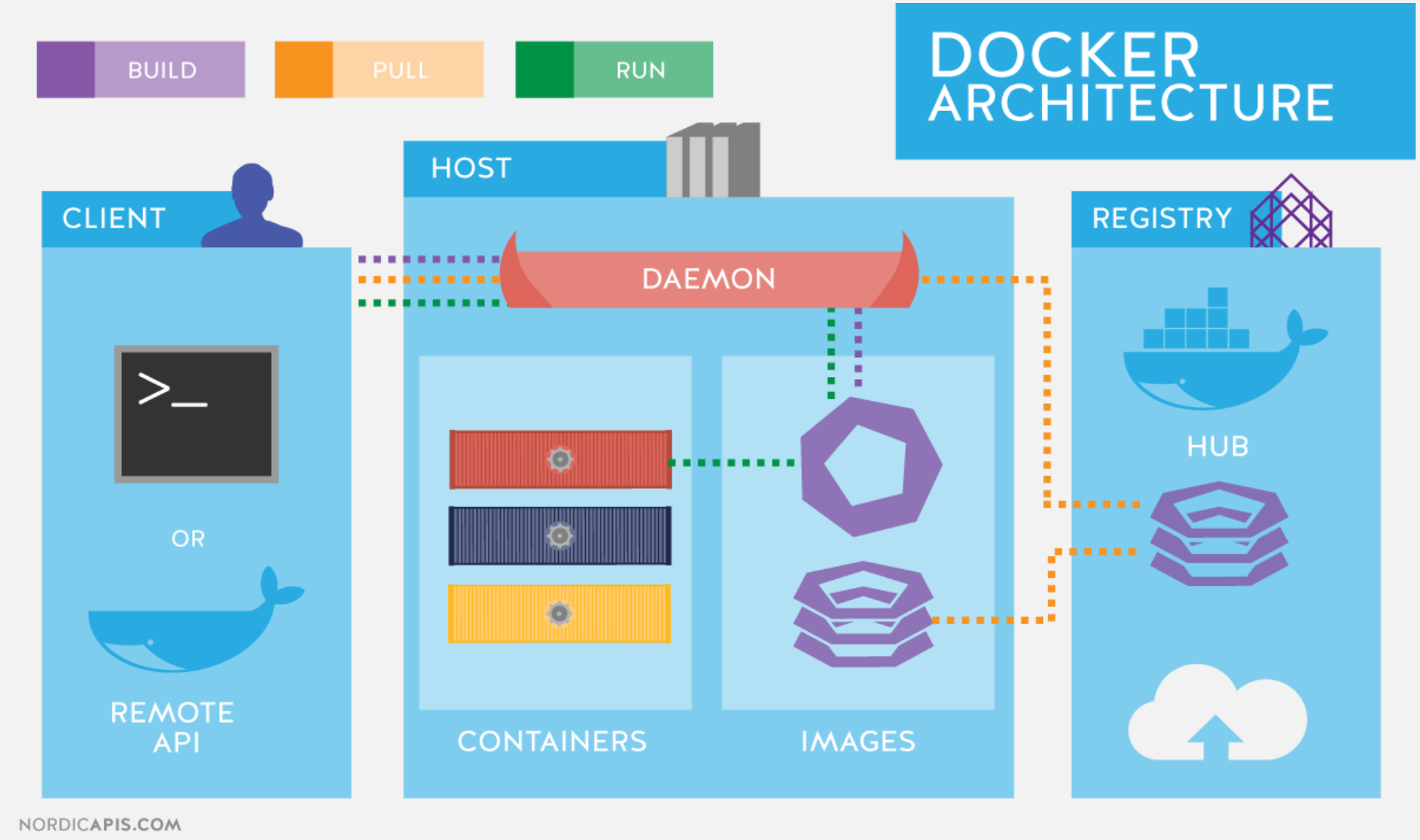
Advantages

- Don't require hypervisor
- Provides process isolation
- Container much smaller as compared to VM and so requires less resources
- Very fast in comparison to VMs
- Booting time ranges from milliseconds to seconds (VMs usually takes minute to boot)
- Containers are portable

Drawbacks

- Tooling ecosystem is quite complicated
- Challenges with security

DOCKER ARCHITECTURE



DOCKER TERMINOLOGY

- Image
- Container
- Registry/Repository
- Plugins

DOCKER SETUP

CONCEPT TO FOCUS ON

- Docker Basic Commands
- Docker Volumes
- Docker Networking
- Docker Security
- Monitoring and Logging
- Docker Compose
- Docker Swarm
- Kubernetes

DOCKER BASIC OPERATIONS

DOCKER IMAGE REPOSITORIES

- Docker Hub is a central Image Repository

Docker Hub is Docker's public registry instance

Docker Trusted Registry (commercially supported version)

Other Public Registries

- ECR (Amazon Elastic Container Registry)
- GCR (Google Container Registry)
- ACR (Azure Container Registry)
- CoreOS Quay
- Private Docker Registry

CREATE OUR OWN REGISTRY

```
docker run -d -p 5001:5000 --restart always --name registry registry:2.6.2
```

DOCKER BASIC COMMAND

- docker pull
- docker run
- docker ps
- docker images
- docker stop
- docker rm
- docker rmi

BASICS OF RUN COMMAND

- docker pull redis
- docker images
- docker run redis
- docker ps

DOCKER STOP

- docker stop <containerid or containername>

DOCKER START

- docker start <containerid or containername>

DOCKER ATTACH

- docker attach <container_name or containerid>

DOCKER LOGS

- docker logs <container_name or containerid>

DOCKER TOP/STATS

- Inspecting the process inside the container
  - docker inspect <container_name>

- Showing Statistics of one or more running containers
  - docker top container_a container_b

DOCKER EXEC

- Execute a command inside a container
  - docker exec -it <container_name> /bin/bash

DOCKER INSPECT

- To gather more information about the running container
  - docker inspect container_name

DOCKER RM

- To remove a container
  - docker rm <container_name>

IMAGE RELATED COMMAND

TEKMentors

DOCKER IMAGES

- List all images in docker host
  - docker images

DOCKER PULL

- Pull an image from docker registry
  - docker pull ubuntu:12.04

DOCKER SEARCH

- Search an image from docker hub
  - docker search ubuntu

DOCKER COMMIT

- Commit an state of a container to an image, this method is not recommended, instead use Dockerfile
  - docker commit -m "message" -a ”arunstiwari" 8841e75b49a3 container_new_name:new

DOCKER BUILD

- Build a docker image from a Dockerfile
  - docker build -t="keraton/container_name:tag_v1 .

DOCKER HISTORY

- Show history of an image
  - docker history <image_id>

DOCKER PORT

- To get an information about container to host mapping.
  - $ docker port <containerid> 80

DOCKER MANAGEMENT COMMANDS

DOCKERFILE

TEKMentors

DOCKERFILE

- Specification file for building a docker image

DOCKERFILE SAMPLE

```
FROM node:8.11
MAINTAINER arunstiwari@gmail.com
ENV APP_HOME /var/www/client/
WORKDIR $APP_HOME
COPY ./package.json $APP_HOME
RUN npm install --production
COPY ./ $APP_HOME
RUN npm run build
```

COMMON DOCKER DIRECTIVES

- FROM
- MAINTAINER
- ADD
- ENV

MAINTAINER

- It is an informational instruction of a Dockerfile
- Enables author to set the details of an image
- Recommendation is to place it after FROM directive

```
MAINTAINER authors_detail
```

FROM

- Sets the base image for the build process
- Subsequent commands would use this base image and build on top of it

```
FROM imagename:tag
```

```
FROM node:8.1.2
```

ADD

- This directive is used to copy the file from the Docker host to the filesystem of the new image

```
ADD dist /var/www/html
```

ENV

- This directive is used to set an environment variable in the Docker image
- The environment variable can be accessed by script or instruction or application in the image

```
ENV  APP_HOME /var/www/html
```

USER

- This directive is used to set the start up user ID in the new Docker image

```
USER  UID |  username
```

```
USER  1001 |  arun
```

STORING DATA CONTAINERS

TEKMentors

PRACTICES TO SLIM IMAGES

TEKMentors

SINGLE STAGE BUILDS ISSUE

- One of the challenge in Docker Image building process is to keep the Docker image small
  - Small Images have following advantages
    - Performance is Fast
    - Lesser Security Vulnerability
- Multi Stage Builds helps to optimize the size of Docker image and yet keeping the Dockerfile readable and maintainable

BUILDER PATTERN

- Common Practice to have two Dockerfile one each for Production and Non-Prodction environment
  - Development Version contained everything needed to build your application
  - Production Version contained only your application and what was needed to run it

Builder Pattern

MANAGING SECRETS IN CONTAINERS

TEKMentors

MANAGING DOCKER CONTAINERS

TEKMentors

APPENDIX -A : PERFORMANCE IMPLICATIONS OF VOLUME MOUNTS

- Another key aspect of volume mounts is that the write speed to a volume mount is far greater than the write speed within the container's filesystem. The reason for this is that the default container filesystem uses features such as thin provisioning and copy-on-write. These features can introduce latency for write-heavy applications.
- By using a host-based volume mount, the containerized application can achieve the same write speeds as the host itself.