Docker & Container Orchestration

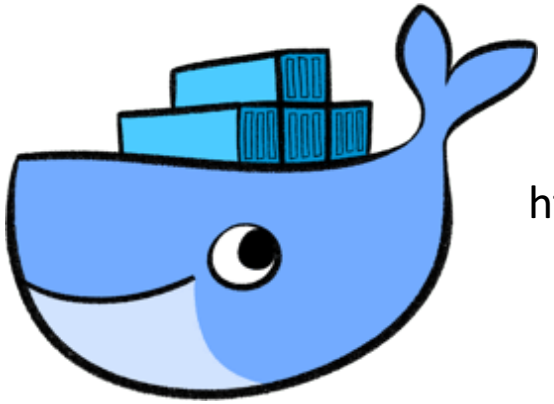
Nga Quach

Team:
Frank Greguska
Thomas Huang
Joseph Jacob
Brian Wilson

Jet Propulsion Laboratory
California Institute of Technology

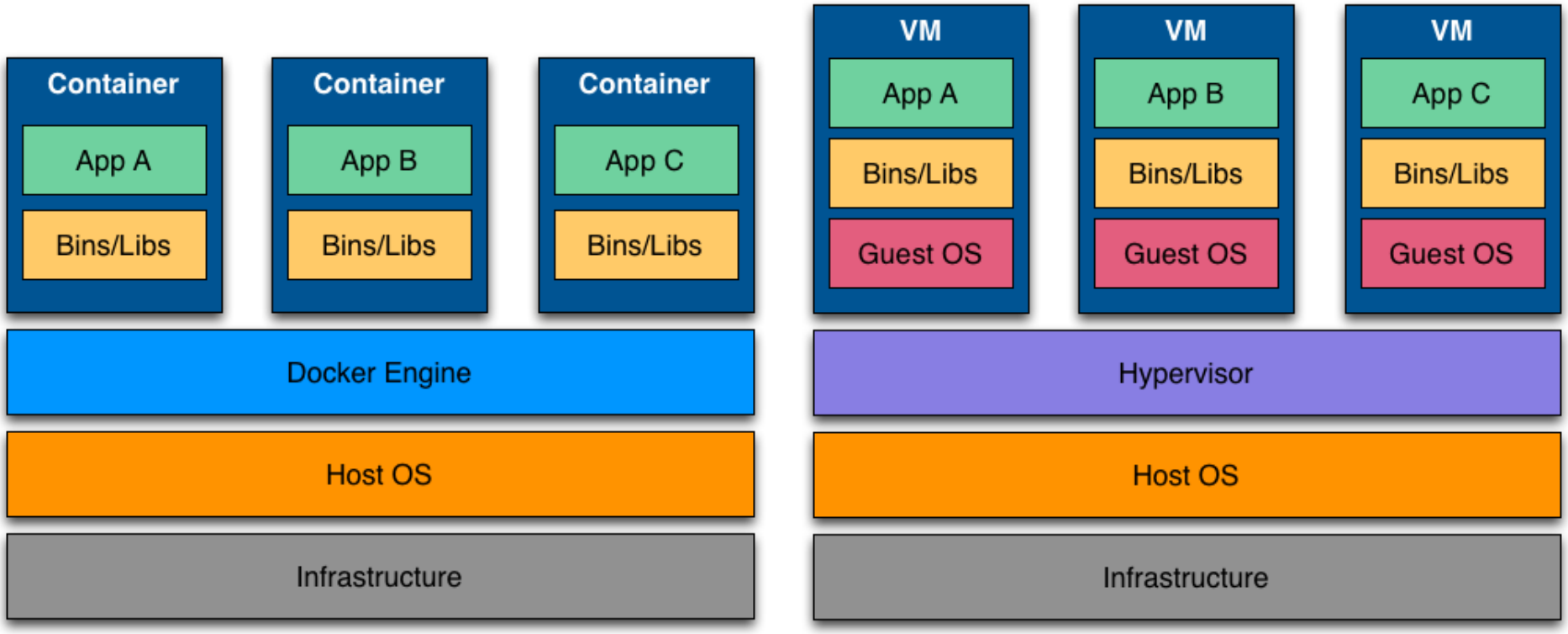
What is Docker?

- Open-source lightweight software container platform consisting of Docker Engine and Docker Registry
- Pack, ship and run any application as a lightweight container that can run anywhere
- Container bundles only application and libraries/binaries required by application



https://www.docker.com/

Container vs Virtual Machine



Containers

Virtual Machines

Pros and Cons

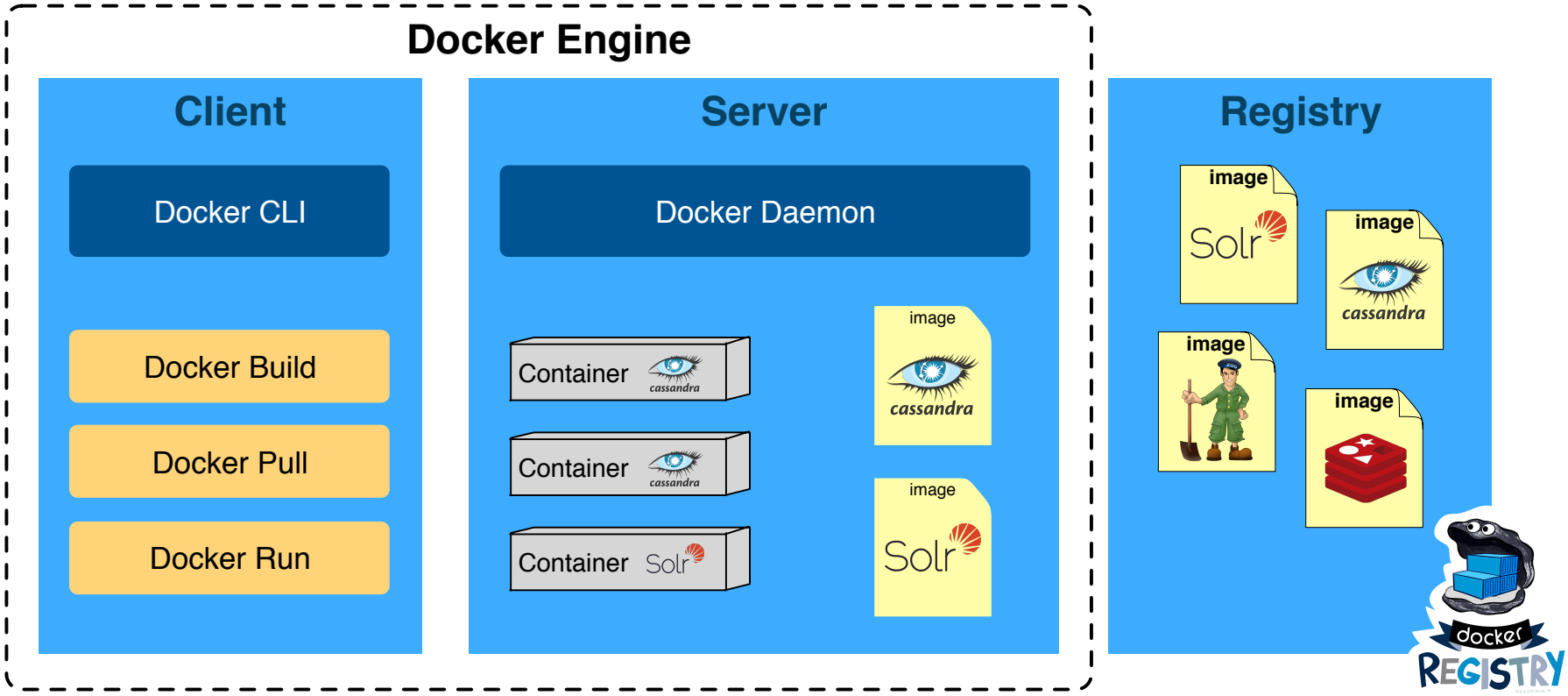
PROS

- Rapid application deployment
- Portability across machines
- Continuous Integration/Continuous Deployment
- Application-centric vs machine/server-centric
- Version control and component reuse
- Lightweight footprint
- Minimal overhead
- Simplified maintenance
- Quick to start up
- Better utilization of computing hardware

CONS

- Less isolation since containers share the kernel
- Containers have root access
- Networking can be tricky
- Persistent data storage is not trivial

Docker Architecture



Dockerfile

- Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image

- Dockerfile for creating customized Apache httpd image

```
FROM httpd:2.4
COPY index.html /usr/local/apache2/htdocs/index.html
```

- Use `docker build` to build an image from a Dockerfile

```
$ docker build -t ntquach/my-apache-image .
```

- Use `docker run` to start the container

```
$ docker run -dit -p 80:80 --name my-apache-app ntquach/my-apache-image
```

Docker Volume

- **Data volumes**
  - Specially-designated directory within container that persist data independent of container's lifecycle and does not automatically get deleted
  - Mount a host directory as a data volume
  - Mount a shared-storage volume as a data volume by using Docker volume plugins
- **Data volume containers**
  - Share persistent data between containers
  - Use data from non-persistent containers

Docker Networking

- List Docker networks using `docker network ls`

```
$ docker network ls

NETWORK ID          NAME                DRIVER
7fca4eb8c647        bridge              bridge
9f904ee27bf5        none                null
cf03ee007fb4        host                host
```

- Bridge network is default Docker network, containers are connected to bridge network by default
- None network contains no network interface
- Host network adds container on host's network stack
- User-defined networks
  - bridge network
  - overlay network
  - MACVLAN network

Docker Compose

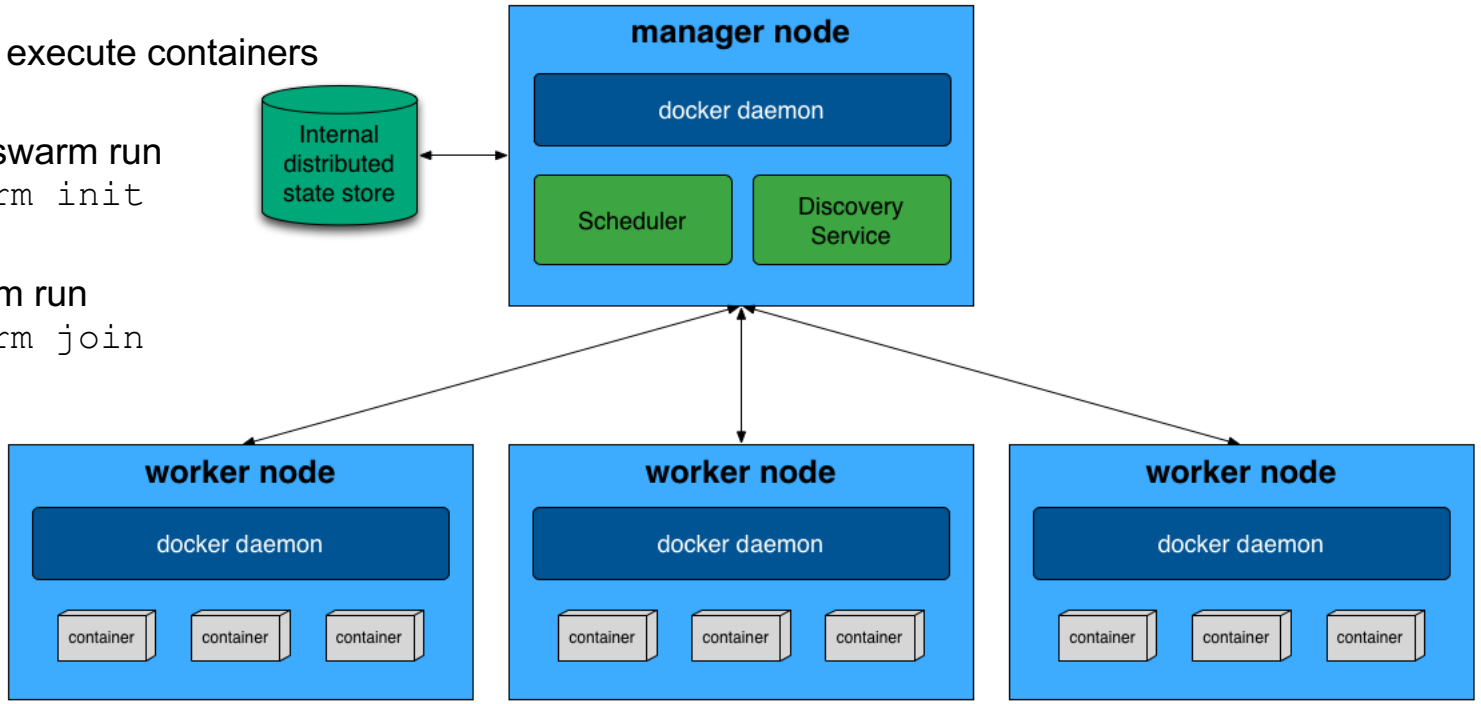
- Tool for defining and running multi-container applications
- Single command to start multiple containers
- Applications are defined in YAML file where options passed to `docker run` can be specified
- Compose file for starting httpd and redis

```
version: '2'
services:
   web:
      image: httpd:2.4
      ports:
         - "80:80"
      volumes:
         - .:/usr/local/apache2/htdocs/
   redis:
      image: redis:alpine
```

Container Orchestration

Docker Swarm

- Integrated with Docker Engine to provide cluster management and orchestration of containers across multiple hosts
- A swarm is a cluster of Docker engines, or nodes, where you deploy services
- Service defines tasks to execute on worker nodes
- Manager node dispatches tasks to worker nodes and performs orchestration and management functions
- Worker nodes execute containers
- To initialize a swarm run
  `docker swarm init`
- To join a swarm run
  `docker swarm join`



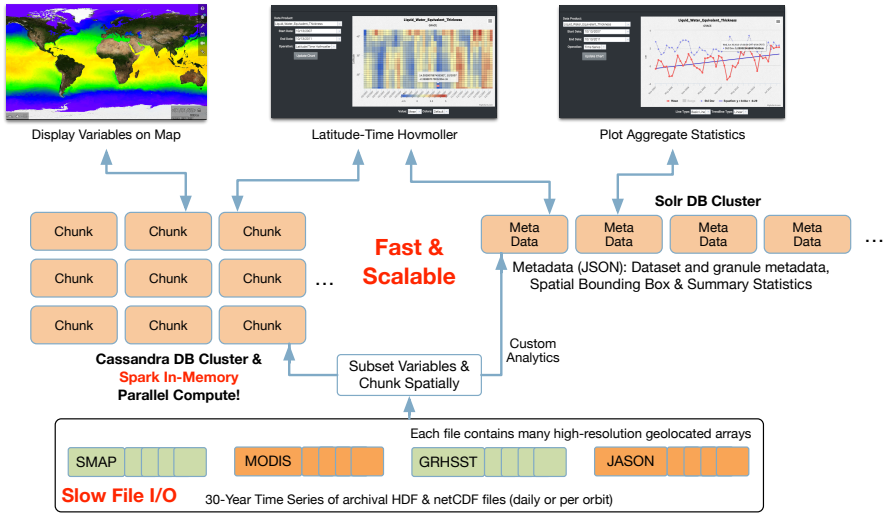
Docker Stack Deploy

- Run `docker stack deploy` to deploy complete application stack to the swarm
- Application stack can be defined in a Compose version 3 file

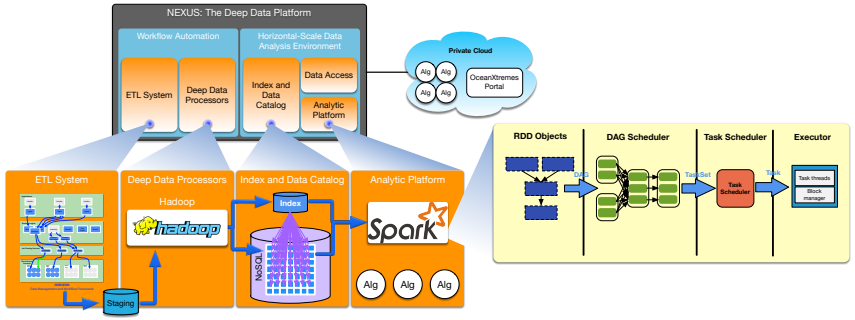
```
version: '3'
services:
   nginx:
      image: nginx:latest
      ports:
         - 8080:80
      deploy:
         mode: replicated
         replicas: 3
         update_config:
            parallelism: 2
            delay: 10s
         restart_policy:
            condition: on-failure
```

Docker Application Example
NEXUS: Deep Data Platform

- A data-intensive analysis solution using a new approach for handling science data to enable large-scale data analysis
- Streaming architecture for horizontal scale data ingestion
- Scales horizontally to handle massive amount of data in parallel
- Provides high-performance geospatial and indexed search solution
- Provides tiled data storage architecture to eliminate file I/O overhead
- A growing collection of science analysis webservices using Apache Spark: parallel compute, in-memory map-reduce framework
- Pre-Chunk and Summarize Key Variables
  - Easy statistics instantly (milliseconds)
  - Harder statistics on-demand using Spark (in seconds)
  - Visualize original data (layers) on a map quickly (Cassandra store)
- **Algorithms** – Time Series | Latitude/Time Hovmöller| Longitude/Time Hovmöller| Latitude/Longitude Time Average | Area Averaged Time Series | Time Averaged Map | Climatological Map | Correlation Map | Daily Difference Average



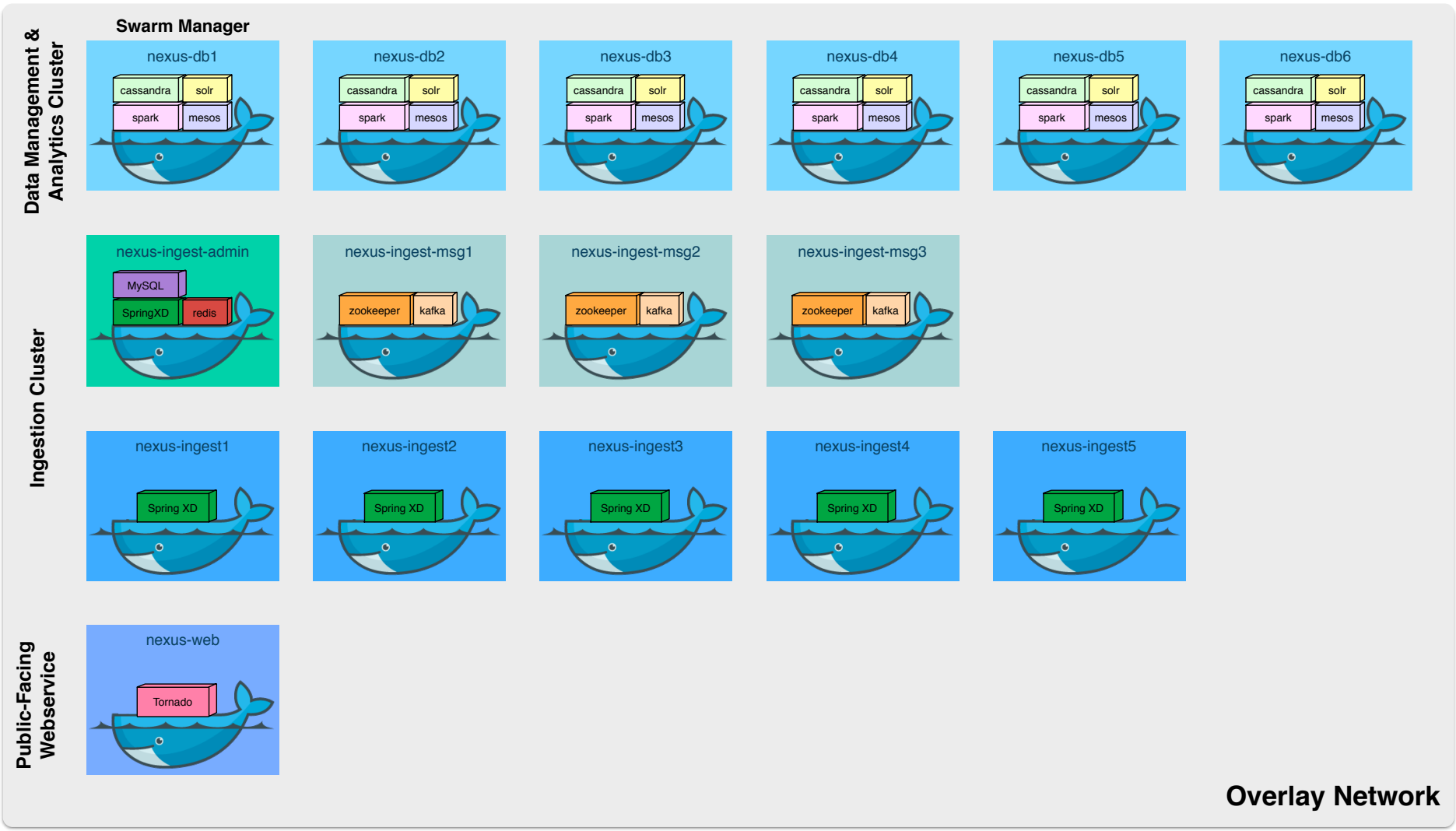
Two-Database Architecture



Deep Data Computing Environment (DDCE)

Open Source: Apache License 2
https://github.com/dataplumber/nexus

Docker Swarm
NEXUS

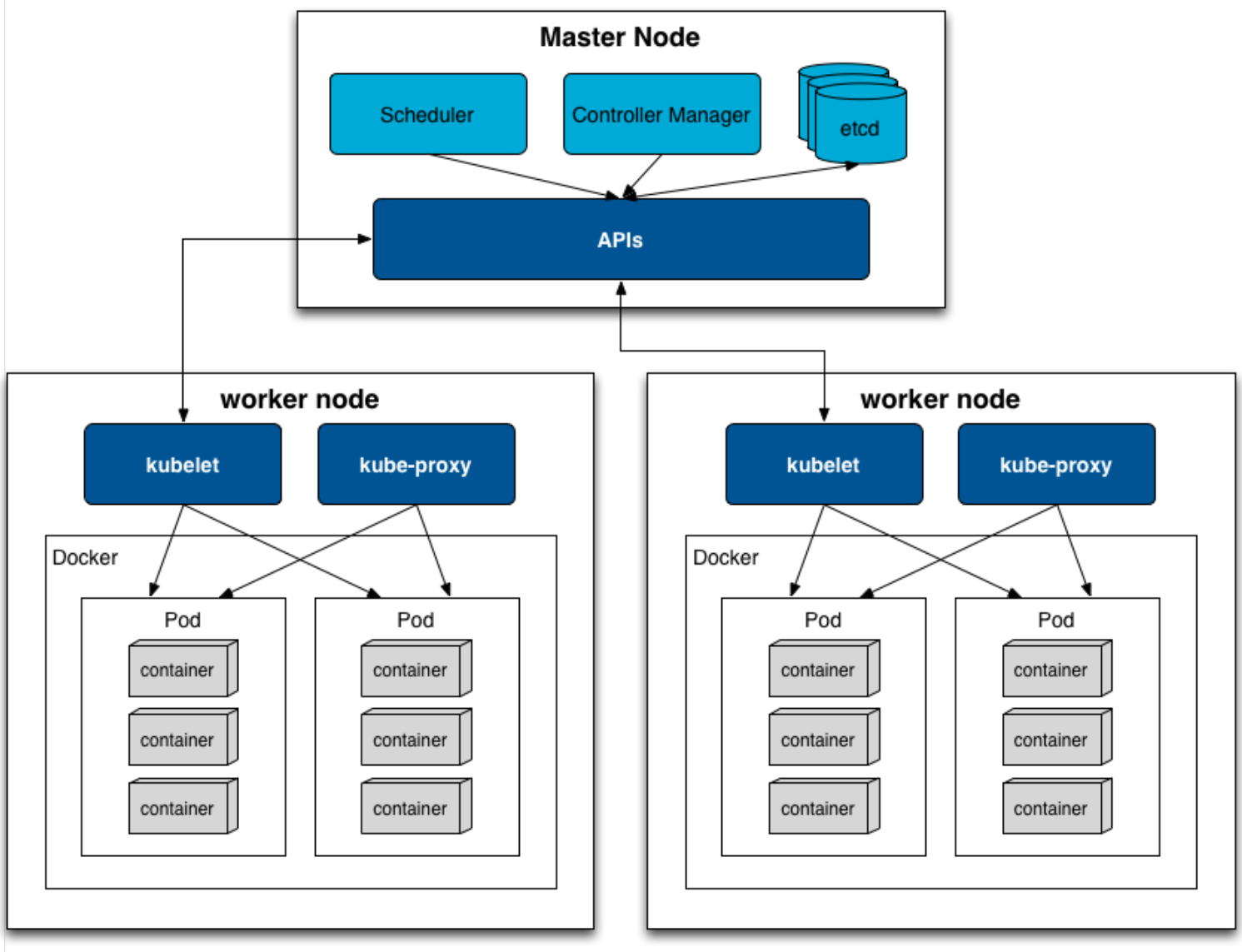


What is Kubernetes?

- Open-source system for automating deployment, scaling, and management of containerized applications across clusters of physical or virtual machines
- Container-centric infrastructure
- Groups containers into logical units for easy management

https://kubernetes.io/

Kubernetes Architecture



Kubernetes Objects

- **Pod**
  - Smallest unit in Kubernetes object model
  - Consists of a single container or multiple containers that are tightly coupled
- **Service**
  - Abstraction defining a logical set of Pods and how to access them
  - Provides network connection
- **Volume**
  - Defined at Pod level
  - Has same lifetime as the Pod that encloses it
  - Can be used for sharing data between containers inside Pod
  - Many different types of Volumes:
    - emptyDir
    - hostDir
    - …
- **Namespace**
  - Logical partition of cluster for resources
  - Resource names are unique within a namespace but not across namespaces

Kubernetes
Create a Deployment

- Example Deployment definition to bring up 3 nginx Pods (nginx-deployment.yaml)

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
   name: nginx-deployment
spec:
   replicas: 3
   template:
      metadata:
         labels:
            app: nginx
      spec:
         containers:
         - name: nginx
           image: nginx:1.7.9
           ports:
           - containerPort: 80
```

- Create Deployment by running kubectl create

```
$ kubectl create -f nginx-deployment.yaml
```

Kubernetes
Create a Service

- Example Service definition to allow access to nginx Pods (nginx-service.yaml)

```
apiVersion: v1
kind: Service
metadata:
   name: nginx-service
spec:
   type: NodePort
   ports:
     - port: 80
       nodePort: 30080
   selector:
     app: nginx
```

- Create Service by running kubectl create

```
$ kubectl create -f nginx-service.yaml
```

Summary

- Containers and virtual machines (VMs) are complementary
- Containers not a new idea but Docker popularized it by standardizing the container image format and distributing image through Docker Registry
- Docker accelerates and optimizes Continuous Integration/Continuous Deployment (CI/CD) process
- Container orchestration platforms enable automatic scaling, deployment and management of containers