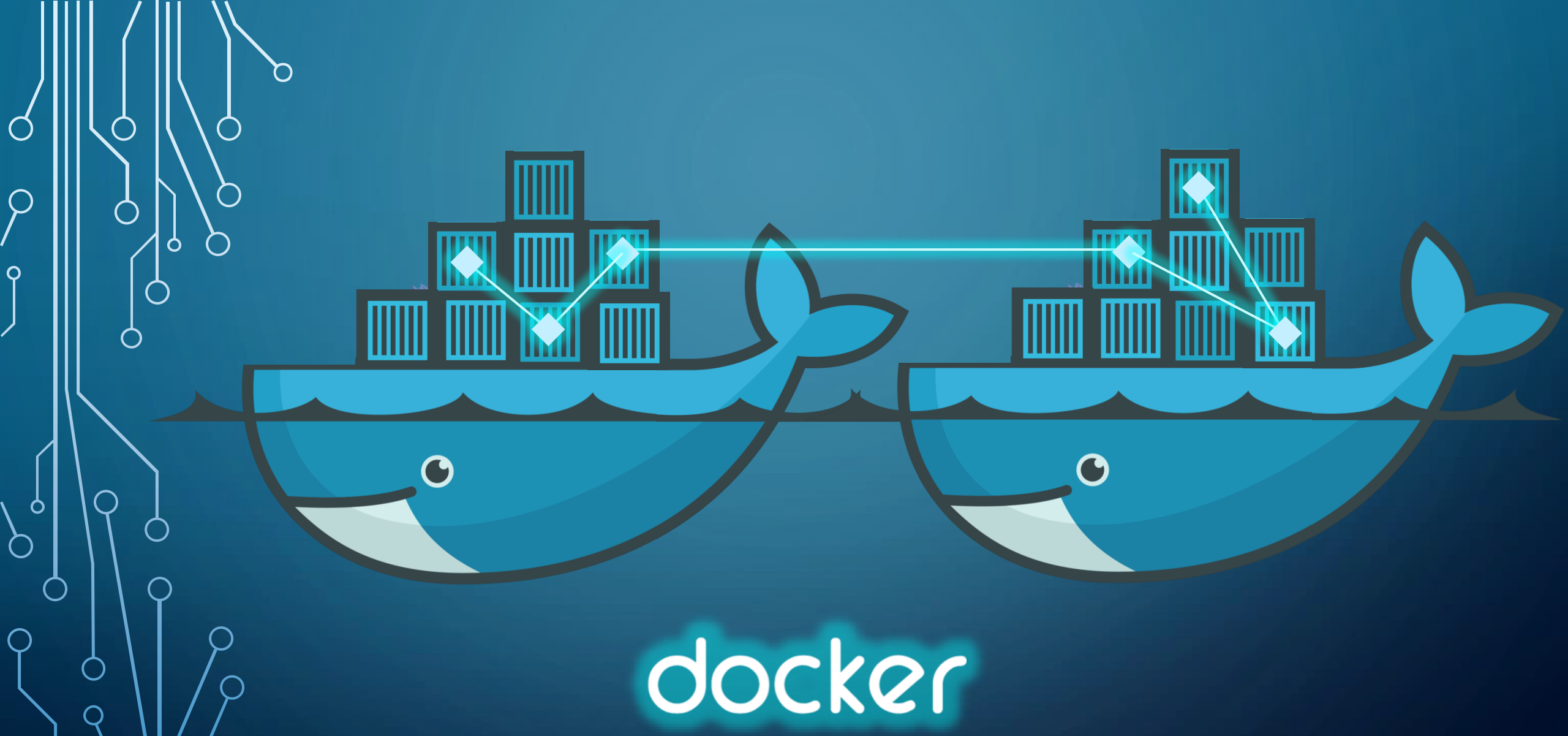
KodeKloud

Learn more about DevOps and Cloud courses with KodeKloud: https://kode.wiki/3N3A4kt

docker

Advanced



docker

Advanced

DOCKER ADVANCED

Mumshad Mannambeth | mmumshad@gmail.com

INTRODUCTION

- Lecture
- Demos
- Coding Exercises
- Assignment

PRE-REQUISITES

- Basic System Administration
- Basic Docker Commands
- Docker Files
- Docker Compose
- Docker Networking

OBJECTIVES

- ☑ Docker Overview
- ☑ Running Docker Containers
- ☑ Creating a Docker Image
- ☑ Docker Compose
- ☑ Docker Swarm
- ☑ Networking in Docker
- ☐ Docker Architecture
- ☐ Docker For Windows
- ☐ Docker Service
- ☐ Docker Swarm
- ☐ Overlay Networks
- ☐ Load Balancing
- ☐ CI/CD Integration

DOCKER STORY

Founder: Solomon Hykes

Release: March 2013

Downloads: 13 Billion

dotCloud

DOCKER ON WINDOWS

Mumshad Mannambeth | mmumshad@gmail.com

DOCKER ON WINDOWS

- Docker on Windows using Docker Toolbox
- Docker for Windows

1. DOCKER TOOLBOX

- 64-bit operating
- Windows 7 or higher.
- Virtualization is enabled

- Oracle Virtualbox
- Docker Engine
- Docker Machine
- Docker Compose
- Kitematic GUI

2. DOCKER FOR WINDOWS

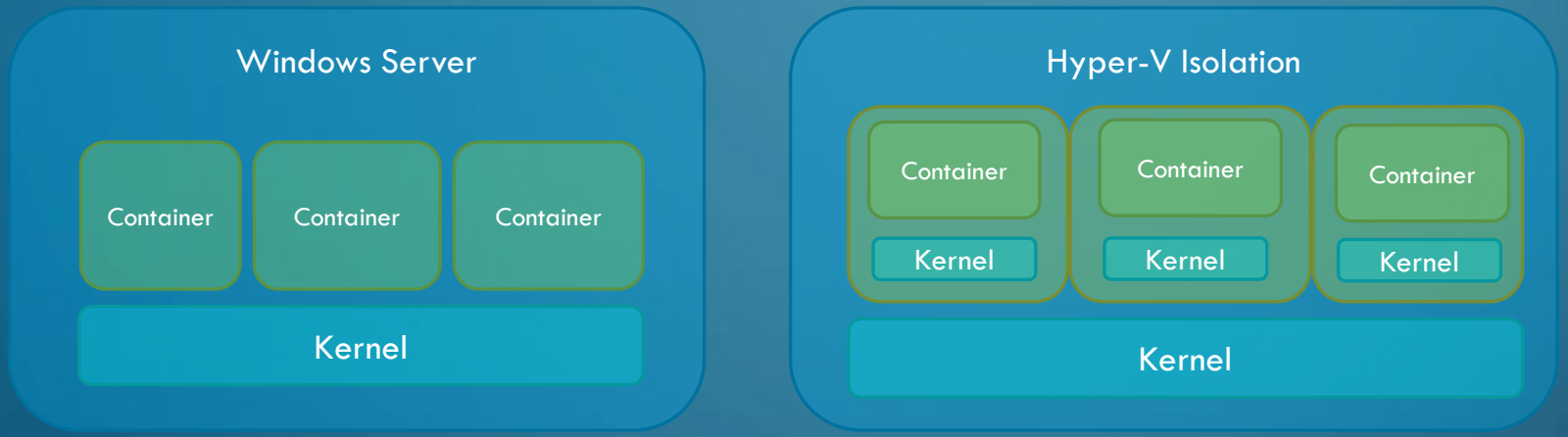
Microsoft
Hyper-V

Support: Windows 10 Enterprise/Professional Edition
Windows Server 2016

Linux Containers (Default)
Or
Windows Containers

WINDOWS CONTAINERS

Container Types:



Base Images:

- Windows Server Core
- Nano Server

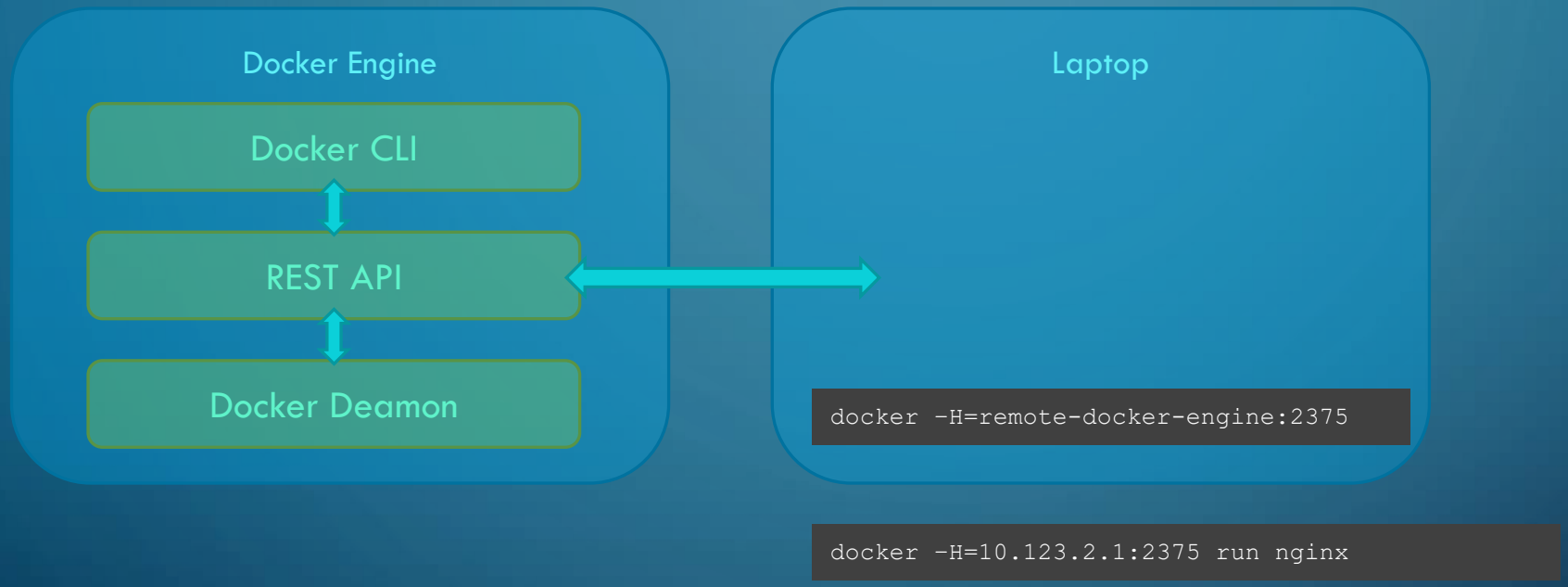
Support

- Windows Server 2016
- Nano Server
- Windows 10 Professional and Enterprise (Hyper-V Isolated Containers)

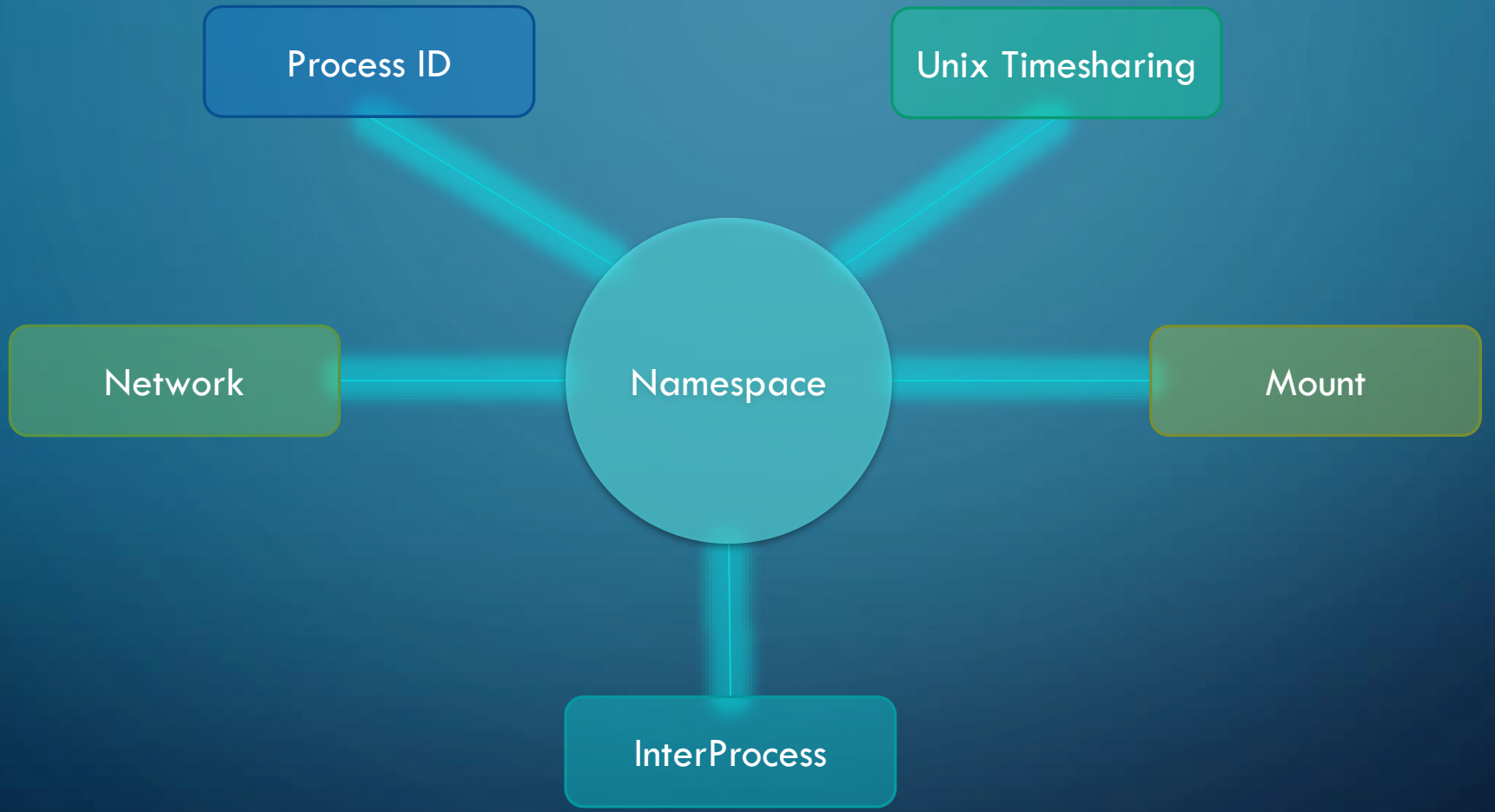
DOCKER ENGINE

Mumshad Mannambeth | mmumshad@gmail.com

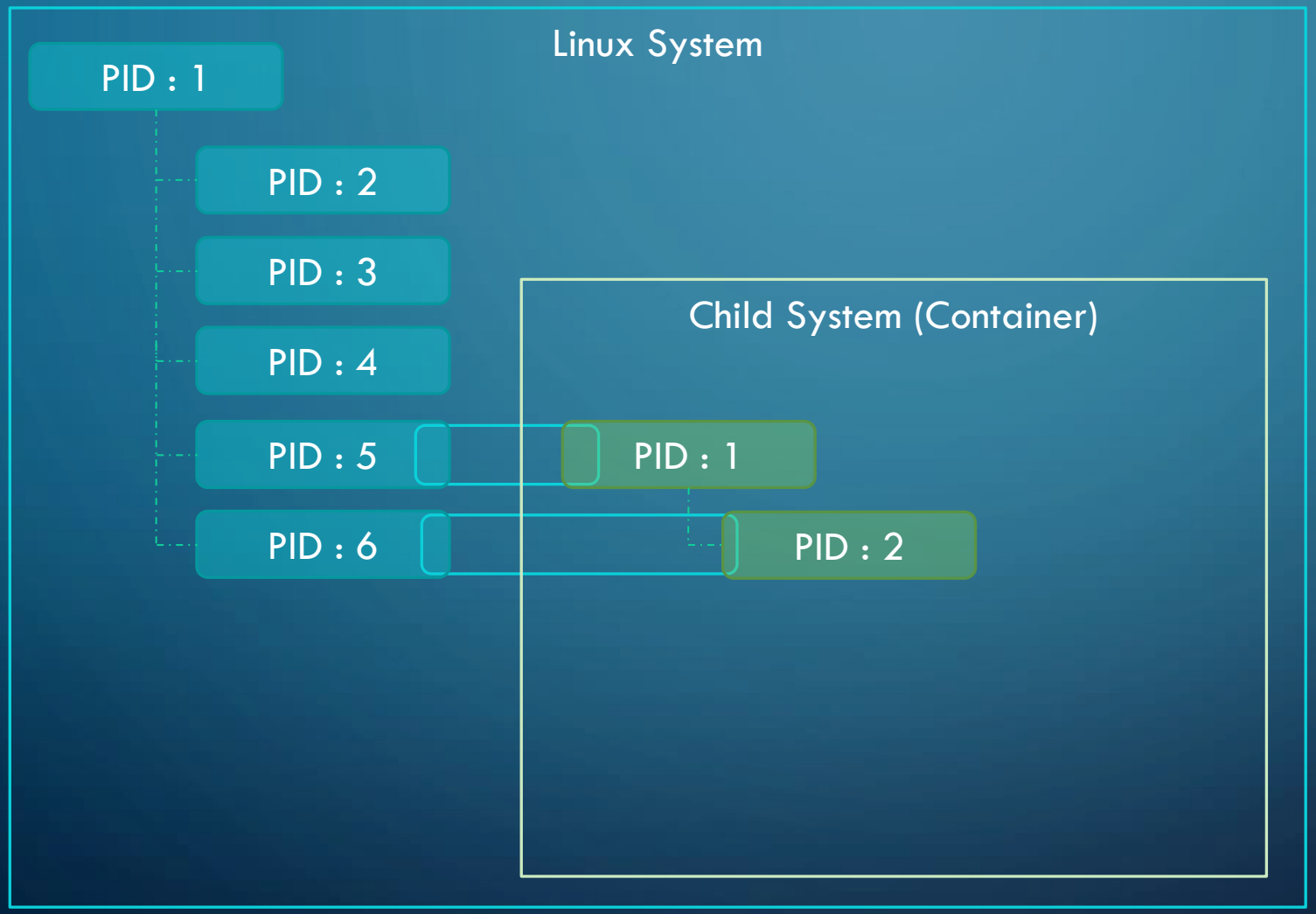
DOCKER ENGINE



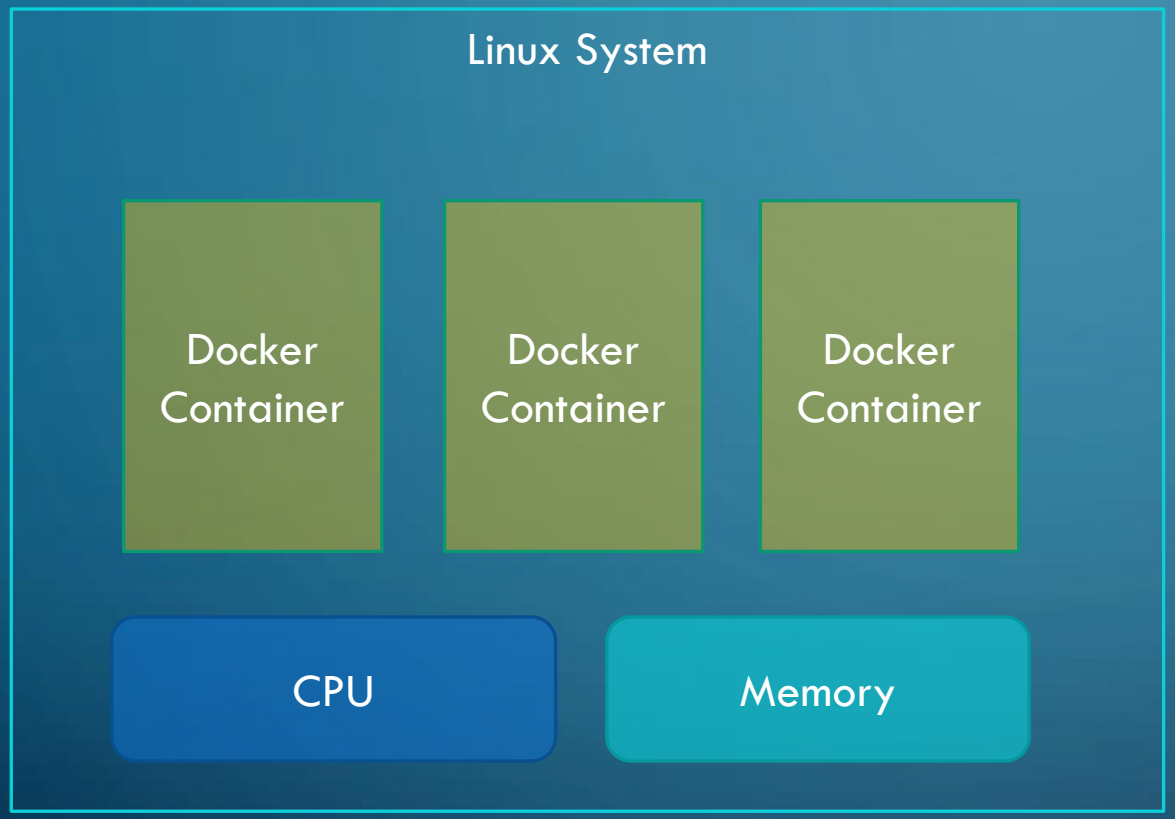
CONTAINERIZATION



NAMESPACE - PID



CGROUPS




```
docker run --cpus=.5 ubuntu
```

```
docker run --memory=100m ubuntu
```

KodeKloud

Learn more about DevOps and Cloud courses with KodeKloud: https://kode.wiki/3N3A4kt

DOCKER STORAGE

Mumshad Mannambeth | mmumshad@gmail.com

FILE SYSTEM

LAYERED ARCHITECTURE

Dockerfile

```
FROM Ubuntu

RUN apt-get update && apt-get -y install python

RUN pip install flask flask-mysql

COPY . /opt/source-code

ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

```
docker build Dockerfile -t mmumshad/my-custom-app
```

Layer 1. Base Ubuntu Layer	120 MB
Layer 2. Changes in apt packages	306 MB
Layer 3. Changes in pip packages	6.3 MB
Layer 4. Source code	229 B
Layer 5. Update Entrypoint	0 B

Dockerfile2

```
FROM Ubuntu

RUN apt-get update && apt-get -y install python

RUN pip install flask flask-mysql

COPY app2.py /opt/source-code

ENTRYPOINT FLASK_APP=/opt/source-code/app2.py flask run
```

```
docker build Dockerfile2 -t mmumshad/my-custom-app-2
```

Layer 1. Base Ubuntu Layer	0 MB
Layer 2. Changes in apt packages	0 MB
Layer 3. Changes in pip packages	0 MB
Layer 4. Source code	229 B
Layer 5. Update Entrypoint	0 B

LAYERED ARCHITECTURE

Container Layer

Read Write

Layer 6. Container Layer

```
docker run mmumshad/my-custom-app
```

Image Layers

Read Only

Layer 5. Update Entrypoint with “flask” command

Layer 4. Source code

Layer 3. Changes in pip packages

Layer 2. Changes in apt packages

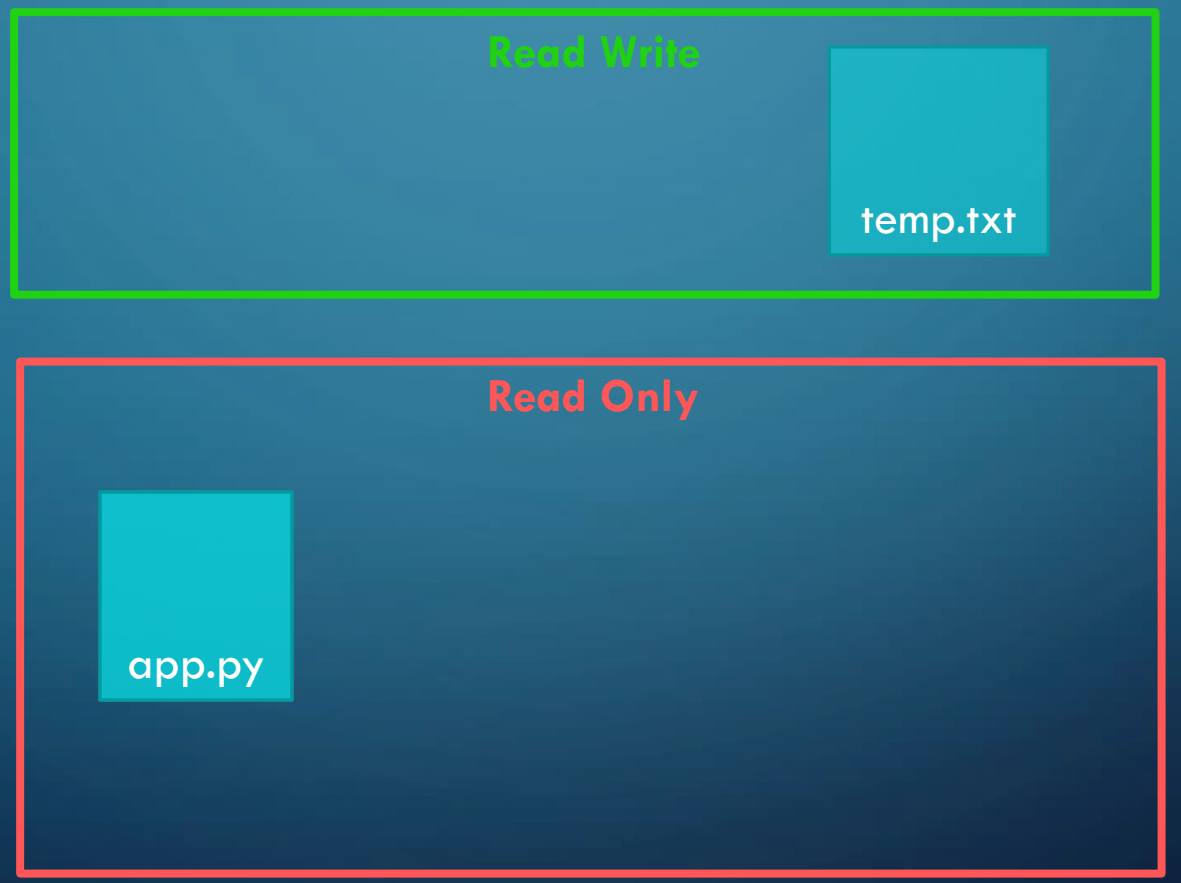
Layer 1. Base Ubuntu Layer

```
docker build Dockerfile -t mmumshad/my-custom-app
```

COPY-ON-WRITE

Container Layer

Image Layers



VOLUMES

```
docker volume create data_volume
```

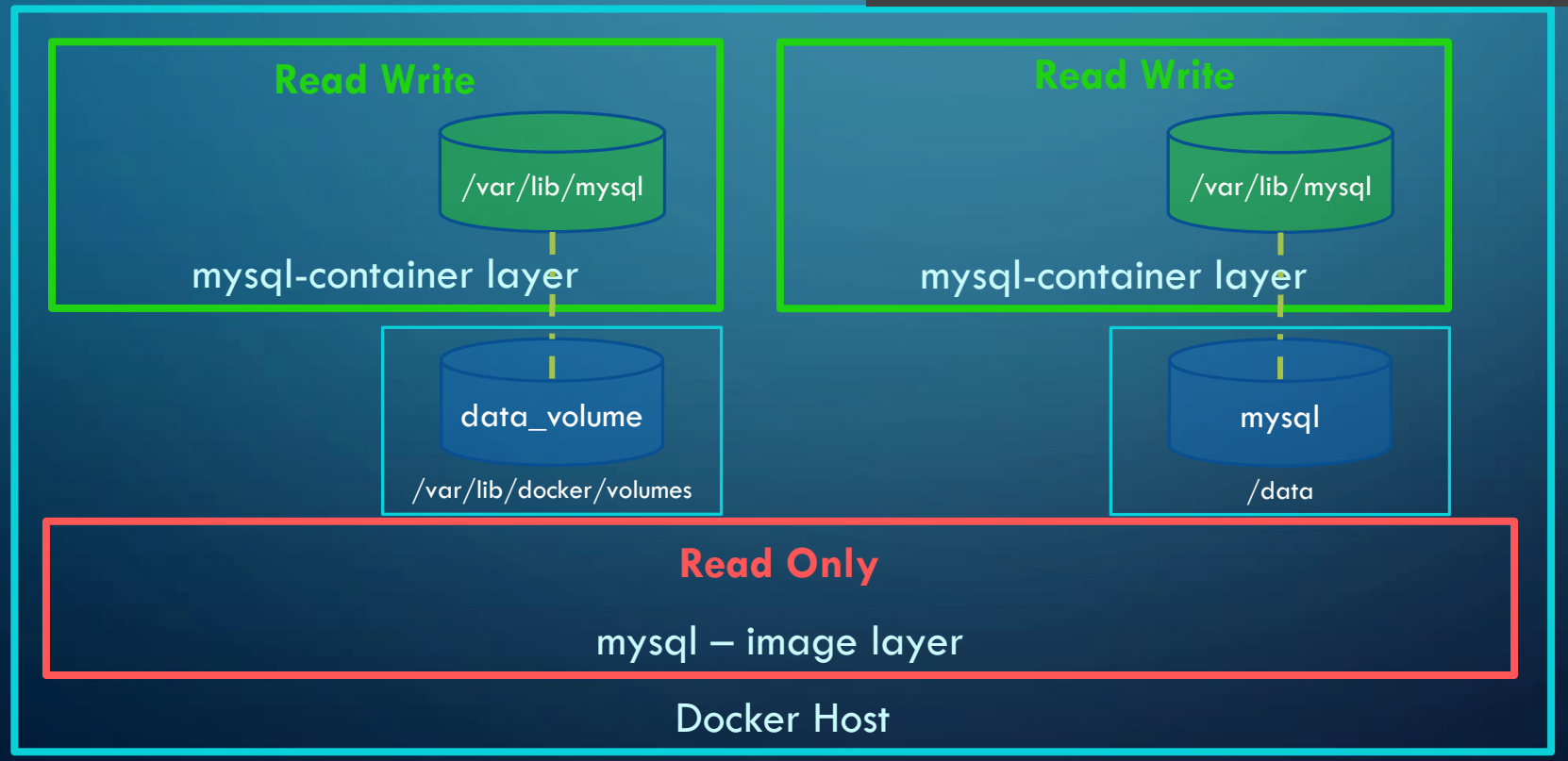
- /var/lib/docker
  - volumes
    - data_volume

```
docker run -v data_volume:/var/lib/mysql mysql
```

```
docker run -v data_volume2:/var/lib/mysql mysql
```

```
docker run -v /data/mysql:/var/lib/mysql mysql
```

```
docker run \
--mount type=bind,source=/data/mysql,target=/var/lib/mysql mysql
```



STORAGE DRIVERS

- AUFS
- ZFS
- BTRFS
- Device Mapper
- Overlay
- Overlay2

DOCKER COMPOSE

Mumshad Mannambeth | mmumshad@gmail.com

DOCKER COMPOSE

```
docker run mmumshad/simple-webapp
```

```
docker run mongodb
```

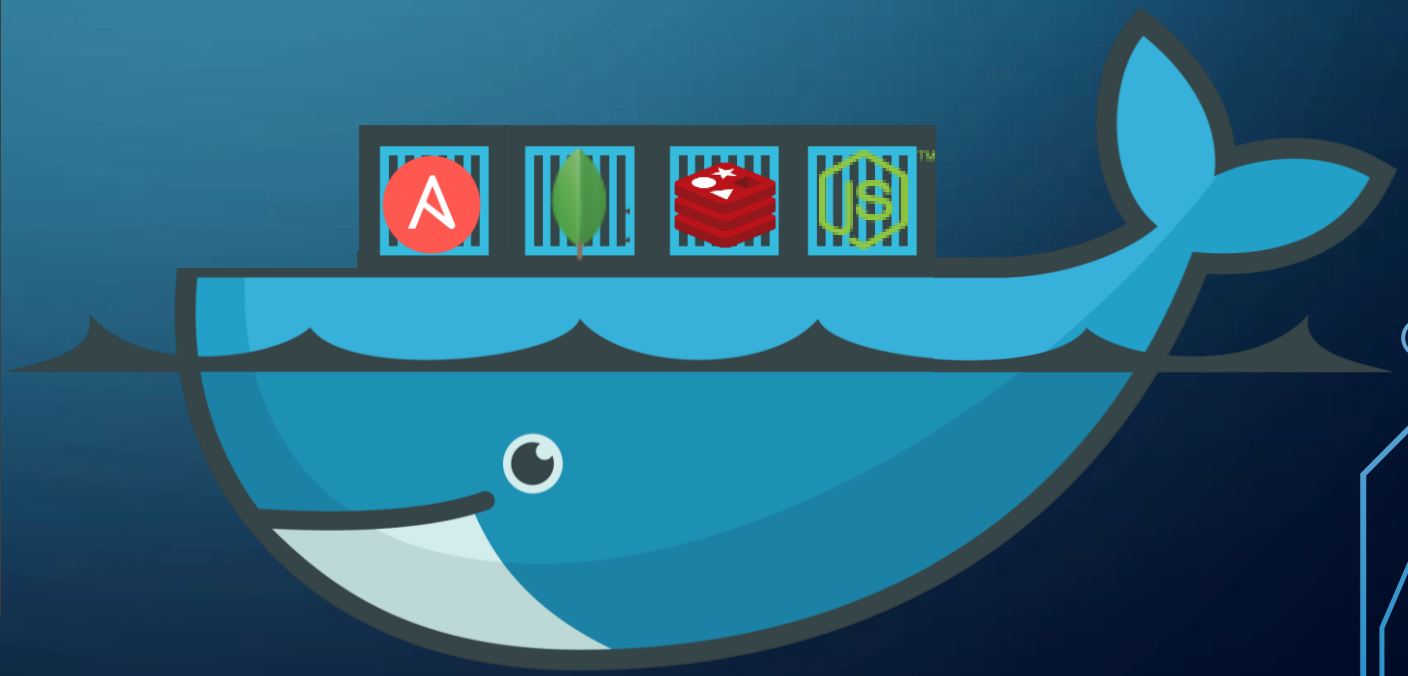
```
docker run redis:alpine
```

```
docker run ansible
```

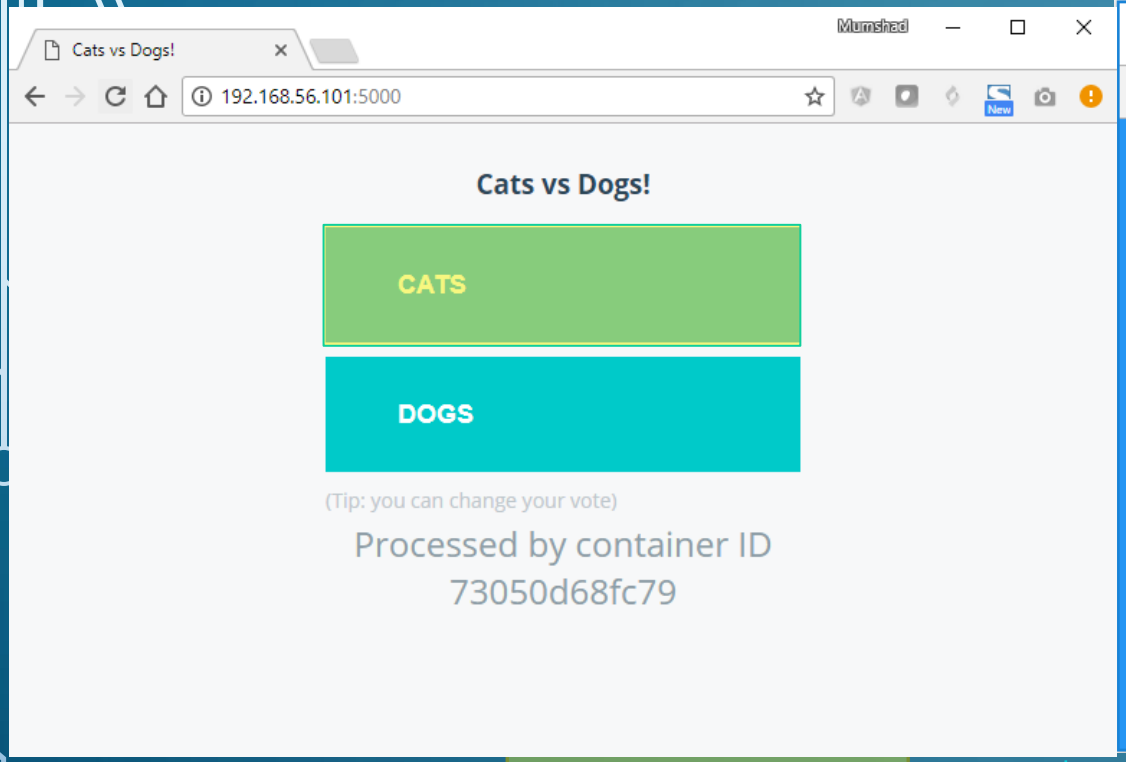
Public Docker registry - dockerhub

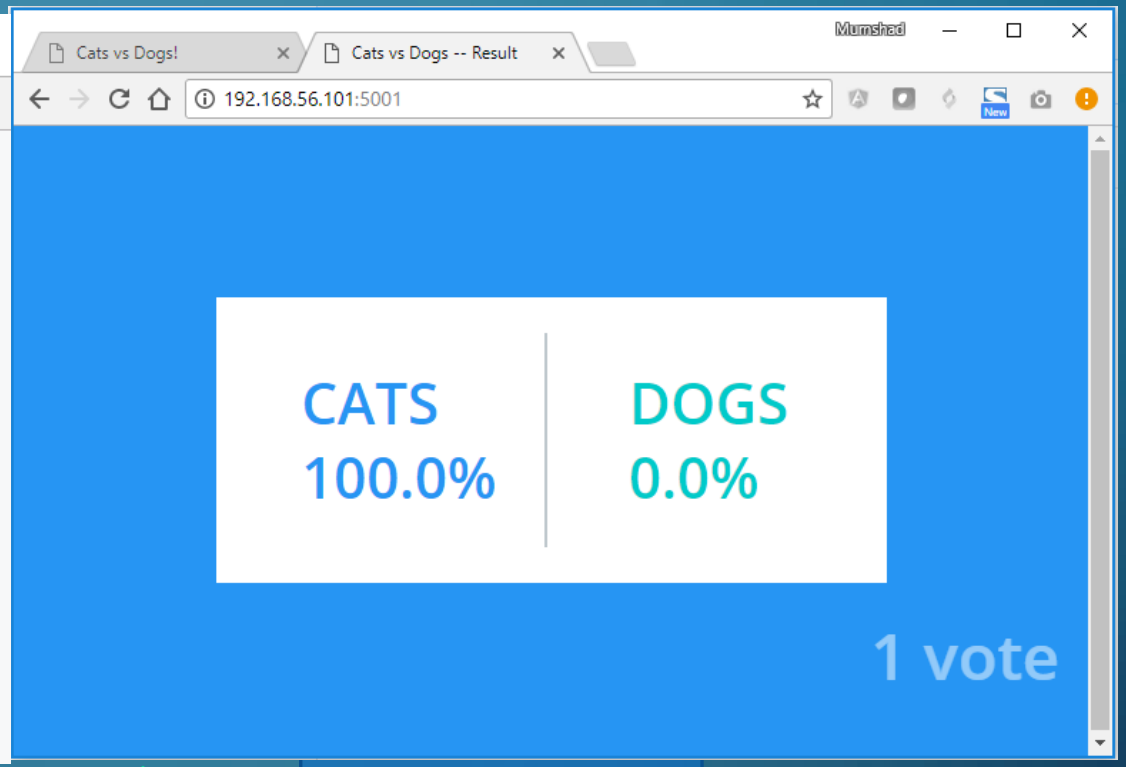
docker-compose.yml

```
services:
     web:
        image: "mmumshad/simple-webapp"
     database:
        image: "mongodb"
     messaging:
        image: "redis:alpine"
     orchestration:
        image:  "ansible"
```



```
docker-compose up
```





CATS	DOGS
1	0

worker

.NET

DOCKER RUN --LINKS

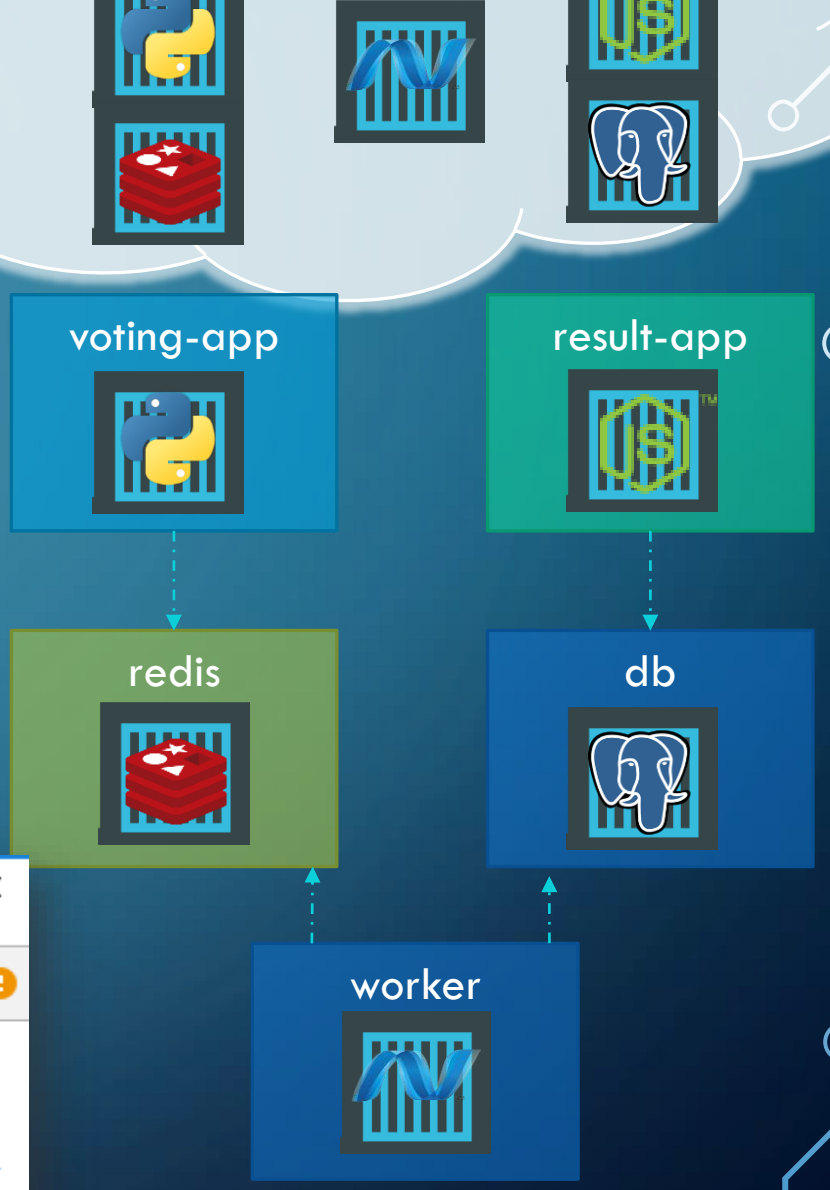
```
docker run -d --name=redis redis
```

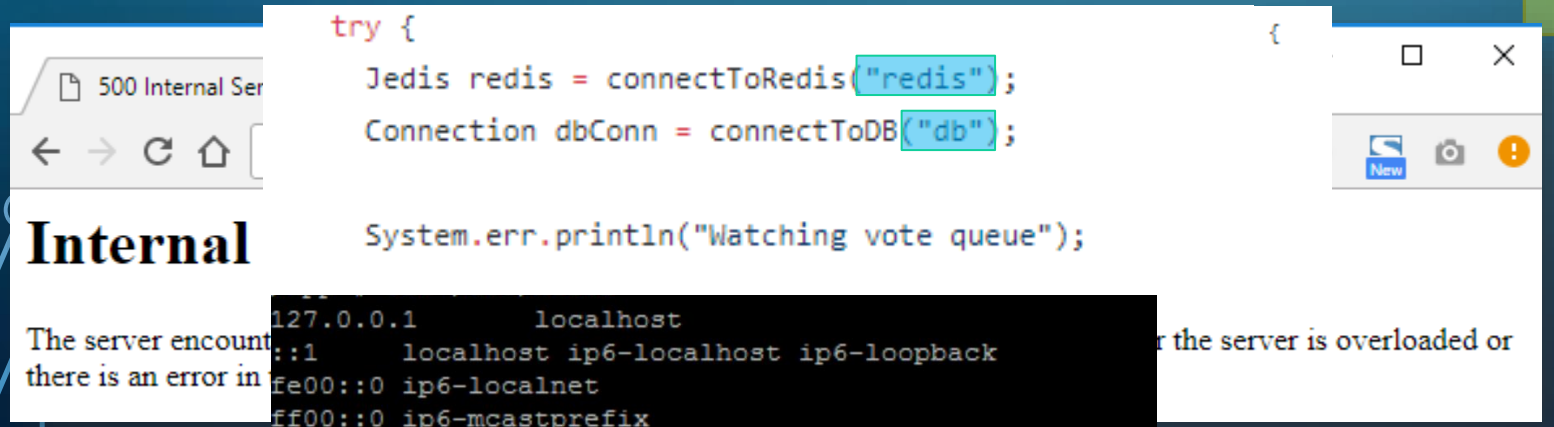
```
docker run -d --name=db postgres:9.4
```

```
docker run -d --name=vote -p 5000:80 --link redis:redis
```

```
docker run -d --name=result -p 5001:80 --link db:db
```

```
docker run -d --name=worker worker db:db ---link redis:redis
```






```
try {
  Jedis redis = connectToRedis("redis");
  Connection dbConn = connectToDB("db");

  System.err.println("Watching vote queue");
```

```
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.2      redis 89cd8eb563da
172.17.0.3      ebcae9eb46bf
```

! Deprecation Warning

DOCKER COMPOSE

```
docker run -d --name=redis redis
```

```
docker run -d --name=db postgres:9.4
```

```
docker run -d --name=vote -p 5000:80 --link redis:redis voting-app
```

```
docker run -d --name=result -p 5001:80 --link db:db result-app
```

```
docker run -d --name=worker --link db:db --link redis:redis worker
```

db:db = db

```
docker-compose up
```

docker-compose.yml

```
image: redis

image: postgres:9.4

ports:
  - 5000:80
links:
  - redis
image: voting-app

ports:
  - 5001:80
links:
  - db
image: result-app

links:
  - redis
  - db
image: worker
```

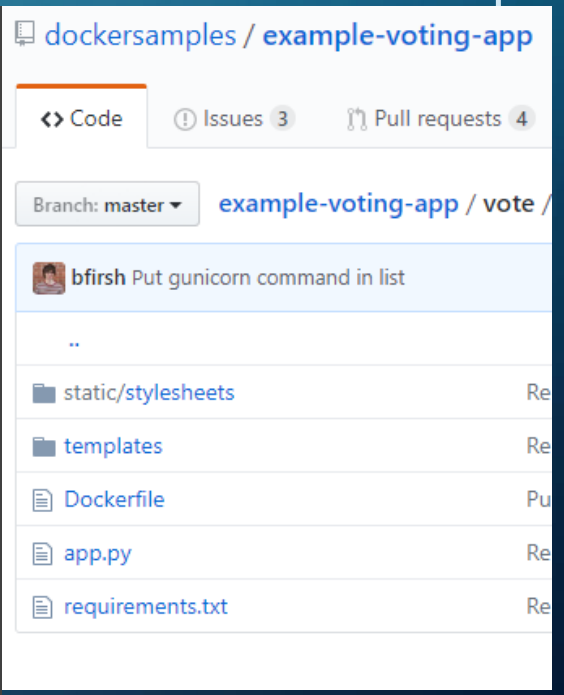
DOCKER COMPOSE - BUILD

docker-compose.yml

```
redis:
    image: redis
db:
    image: postgres:9.4
vote:
    image: voting-app
    ports:
        - 5000:80
    links:
        - redis
result:
    image: result
    ports:
        - 5001:80
    links:
        - db
worker:
    image: worker
    links:
        - db
        - redis
```

docker-compose.yml

```
redis:
    image: redis
db:
    image: postgres:9.4
vote:
    build: ./vote
    ports:
        - 5000:80
    links:
        - redis
result:
    build: ./result
    ports:
        - 5001:80
    links:
        - db
worker:
    build: ./worker
    links:
        - db
        - redis
```



DOCKER COMPOSE - VERSIONS

docker-compose.yml

```
redis:
    image: redis
db:
    image: postgres:9.4
vote:
    image: voting-app
    ports:
        - 5000:80
    links:
        - redis
```

version: 1

docker-compose.yml

```
version: 2
services:
   redis:
        image: redis
   db:
        image: postgres:9.4
   vote:
        image: voting-app
        ports:
            - 5000:80
        depends_on:
            - redis
```

version: 2

docker-compose.yml

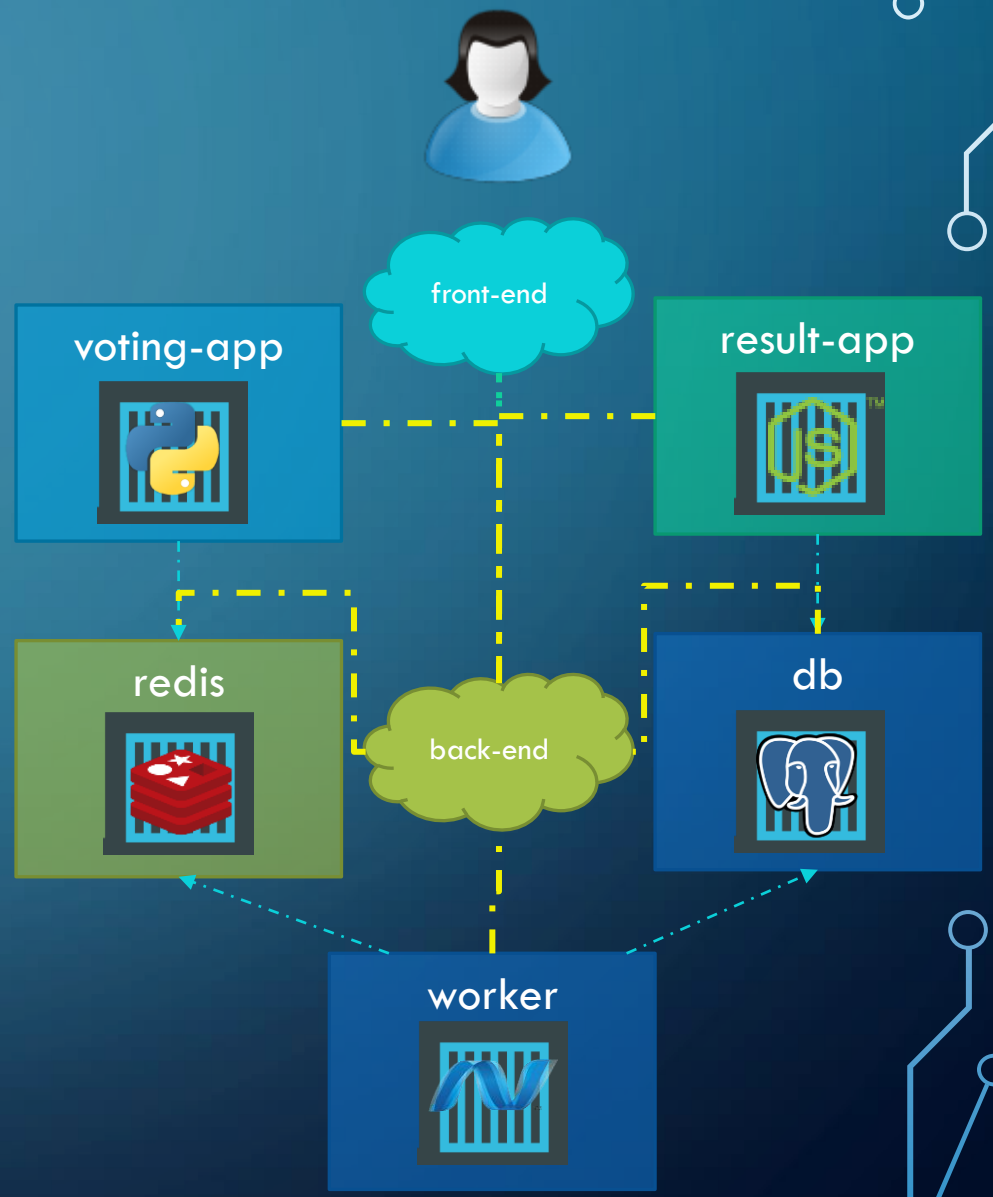
```
version: 3
services:
```

version: 3

DOCKER COMPOSE

docker-compose.yml

```
version: 2
services:
    redis:
        image: redis
        networks:
            - back-end
    db:
        image: postgres:9.4
        networks:
            - back-end
    vote:
        image: voting-app
        networks:
            - front-end
            - back-end
    result:
        image: result
        networks:
            - front-end
            - back-end

networks:
    front-end:
    back-end:
```



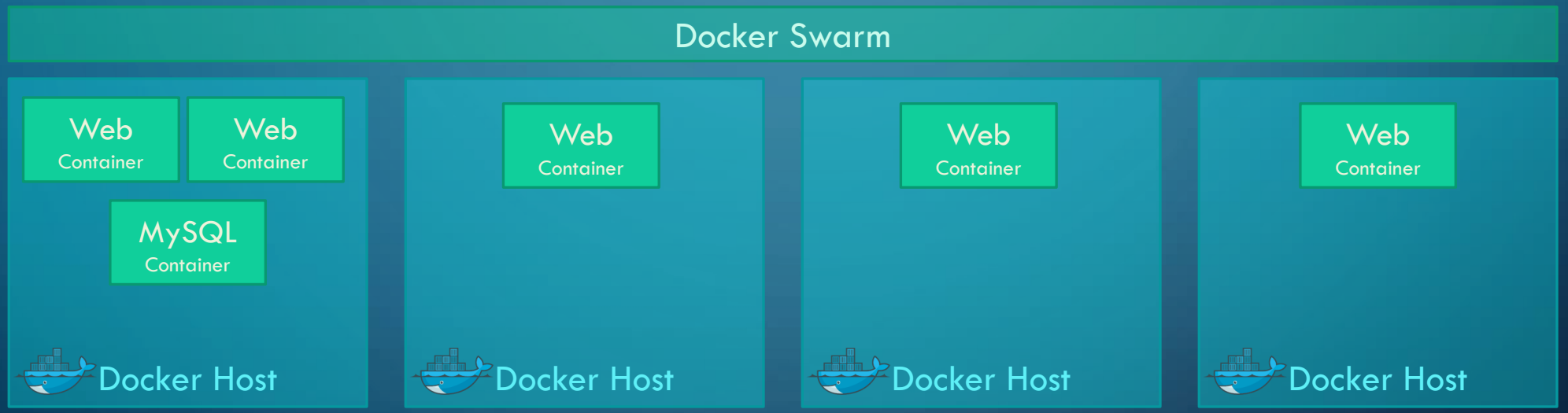
CODING EXERCISES

- Develop Docker compose files
- Practice different versions of files
- Docker compose with networking

DOCKER SWARM

Mumshad Mannambeth | mmumshad@gmail.com

DOCKER SWARM



SETUP SWARM

Swarm Manager

```
docker swarm init
```

Docker Host

Node

Worker

```
docker swarm join
  --token <token>
```

Docker Host

Node

Worker

```
docker swarm join
  --token <token>
```

Docker Host

Node

Worker

```
docker swarm join
  --token <token>
```

Docker Host

```
root@osboxes:/root/simple-webapp-docker # docker swarm init --advertise-addr 192.168.1.12
Swarm initialized: current node (0j76dum2r56p1xfne4ub1ps2c) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-35va8b3fi5krpdskefqqxgttmulw3z828daucri7y526ne0sgu-2eek9qm33d4lxzoq6we9i8izp 192.16
8.1.12:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

DOCKER MANAGERS

MANAGER NODES

Swarm Manager

Docker Host

Swarm Manager

Leader

Docker Host

Swarm Manager

Docker Host

Worker

Docker Host

Worker

Docker Host

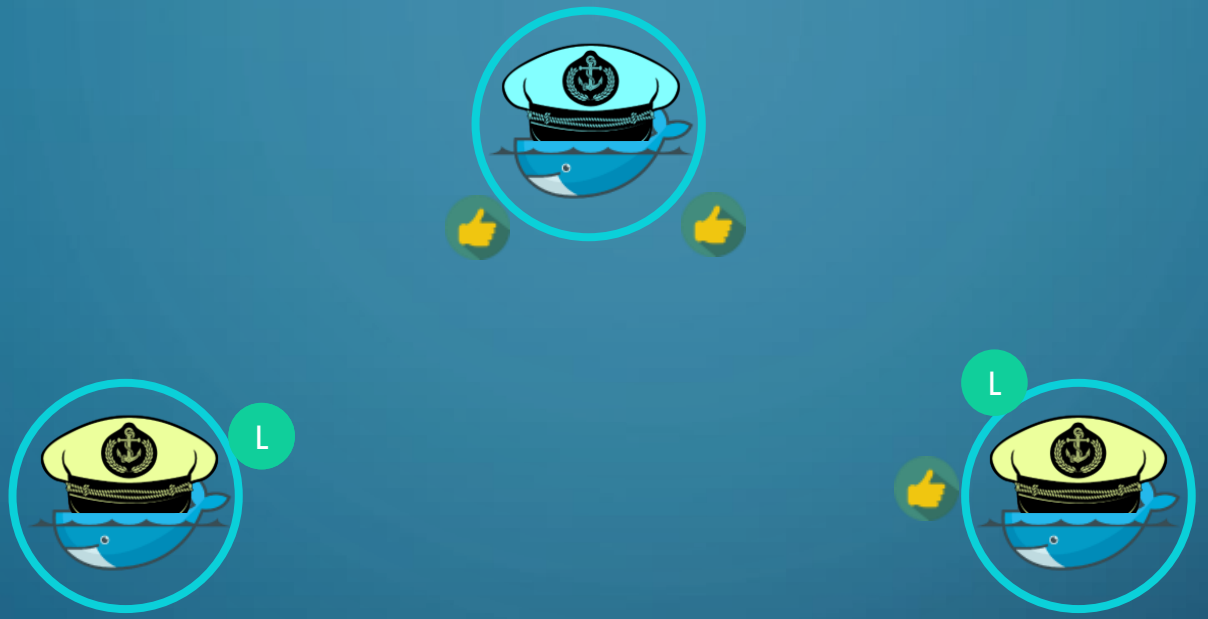
Worker

Docker Host

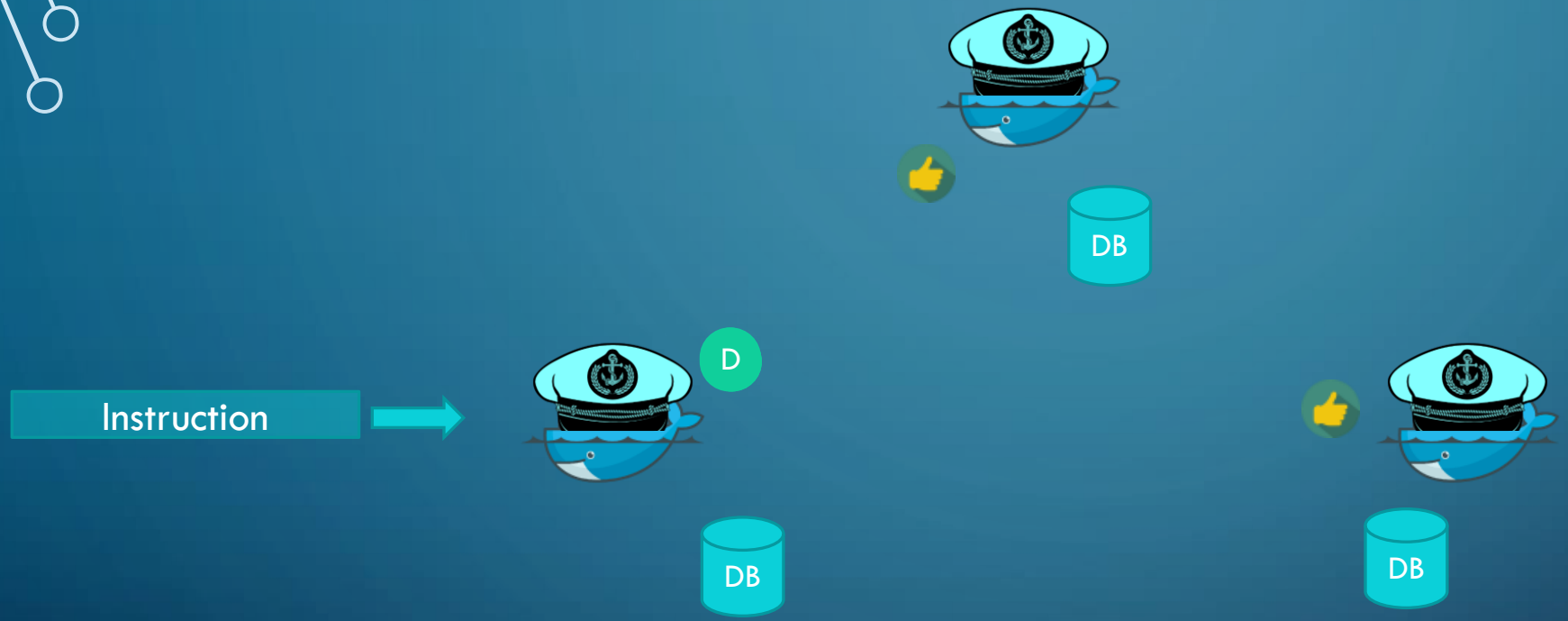
Worker

Docker Host

DISTRIBUTED CONSENSUS - RAFT



DISTRIBUTED CONSENSUS - RAFT



HOW MANY MANAGER NODES?







- Docker Recommends – 7 Managers
- No limit on Managers

Managers	Majority	Fault Tolerance
1	1	0
2	2	0
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3

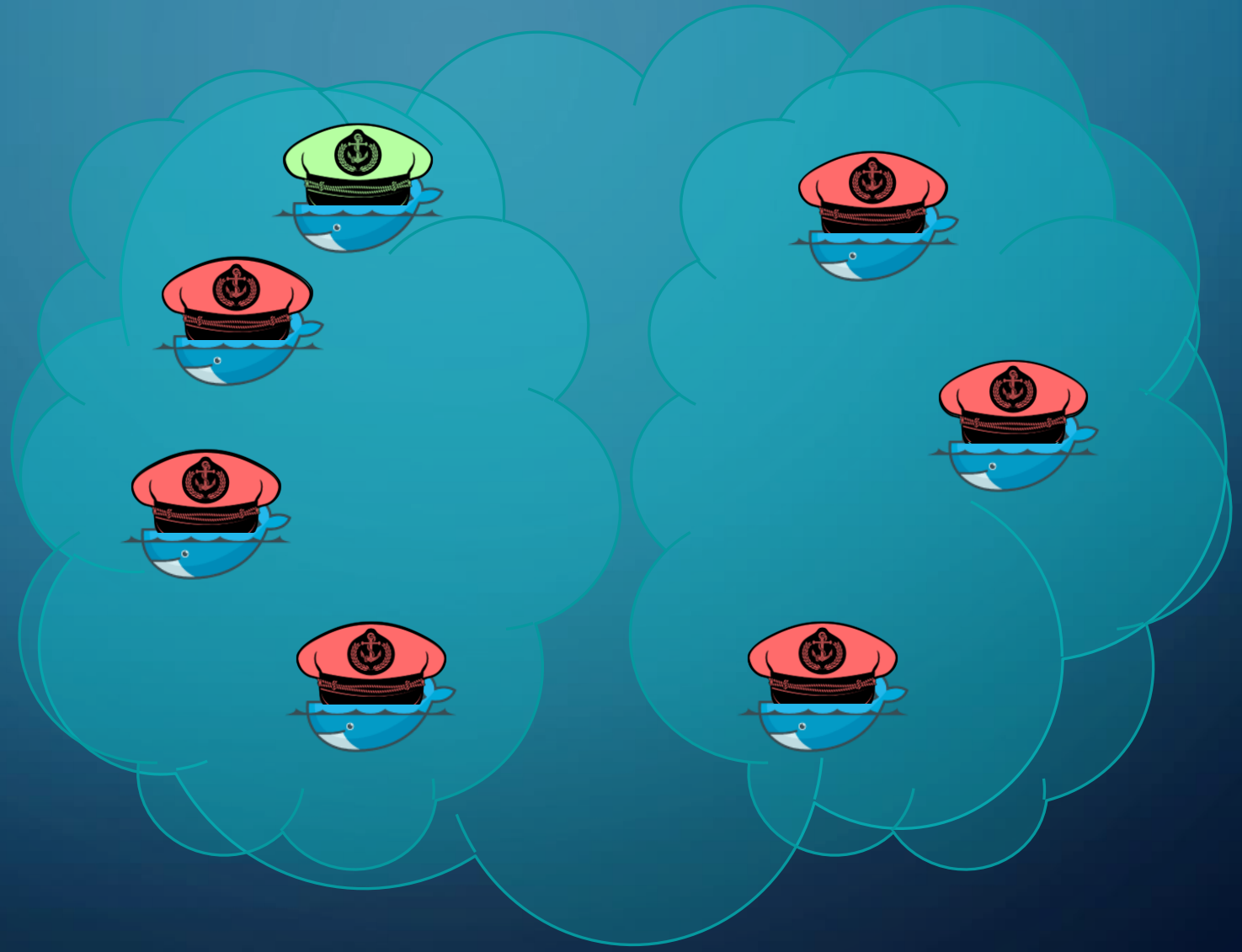
$$\text{Quorum of } N = \frac{N + 1}{2}$$

$$\text{Quorum of } 5 = \frac{5 + 1}{2} = 3.5 = 3$$

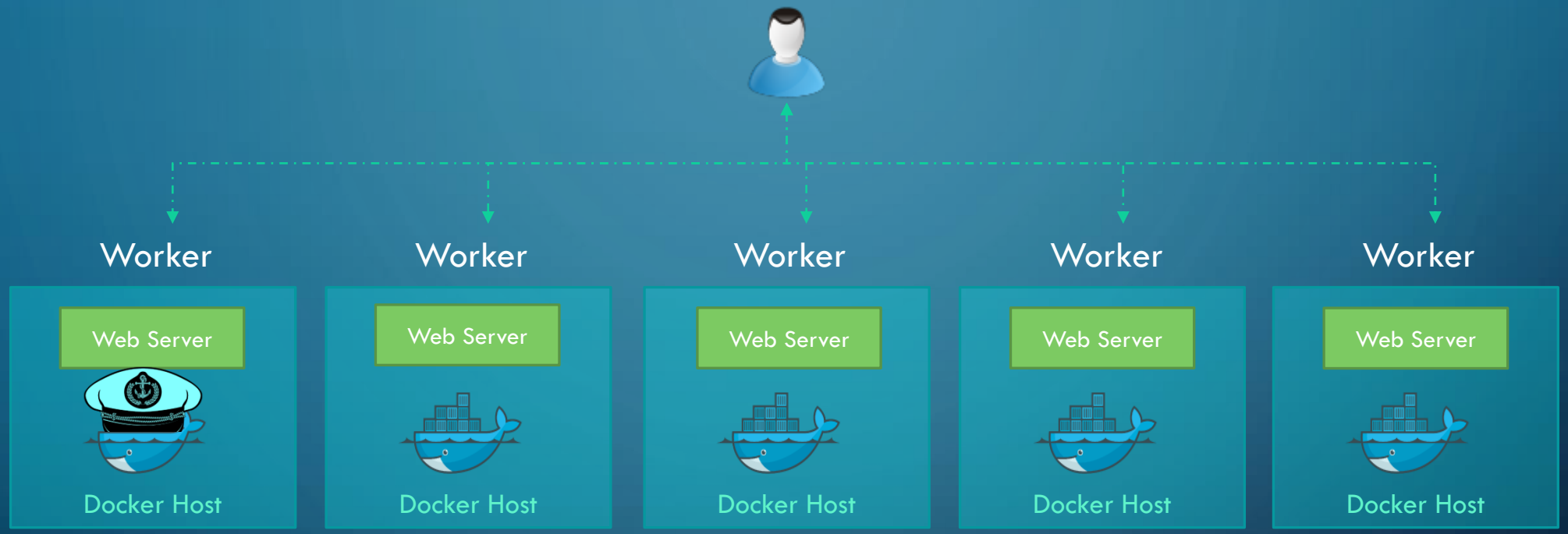
$$\text{Fault Tolerance of } N = \frac{N - 1}{2}$$

ODD OR EVEN?

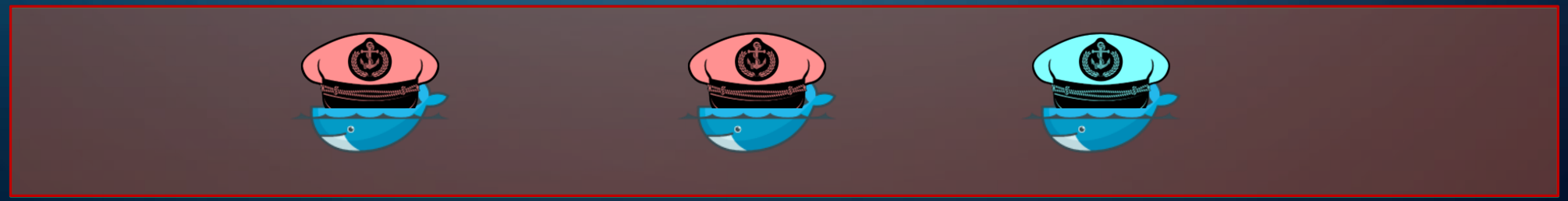
Managers	Majority	Fault Tolerance
1	1	0
2	2	0
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3



CLUSTER FAILURE




```
docker node promote
```



```
docker swarm init --force-new-cluster
```

CAN MANAGER WORK?


```
docker node update --availability drain <Node>
```

KodeKloud

Learn more about DevOps and Cloud courses with KodeKloud: https://kode.wiki/3N3A4kt

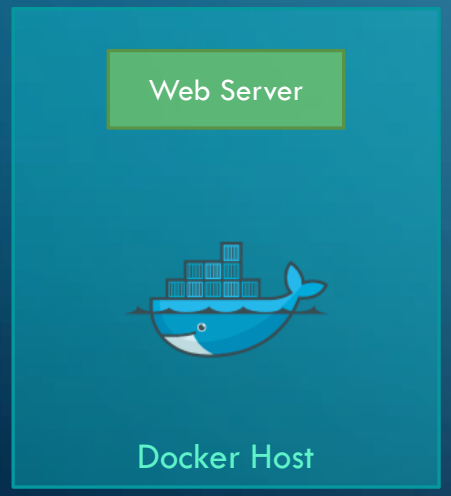
DOCKER SERVICE

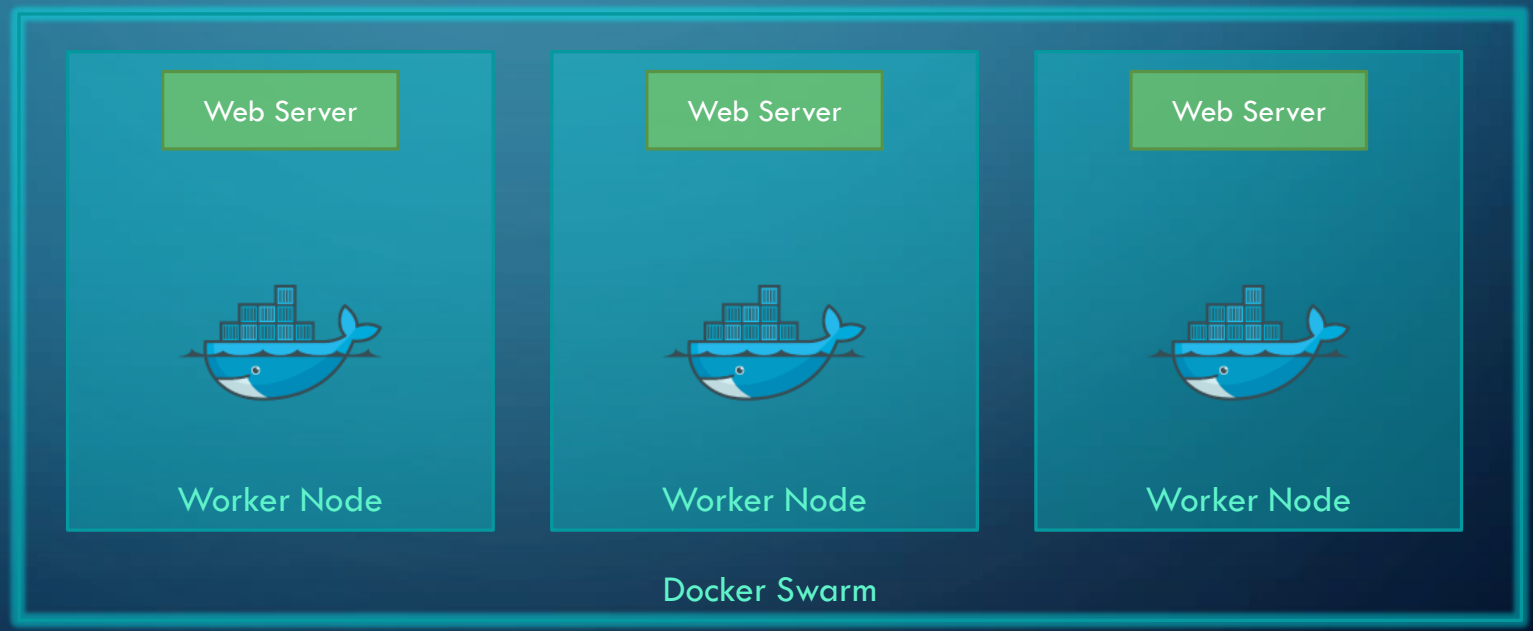
Mumshad Mannambeth | mmumshad@gmail.com

DOCKER SERVICE

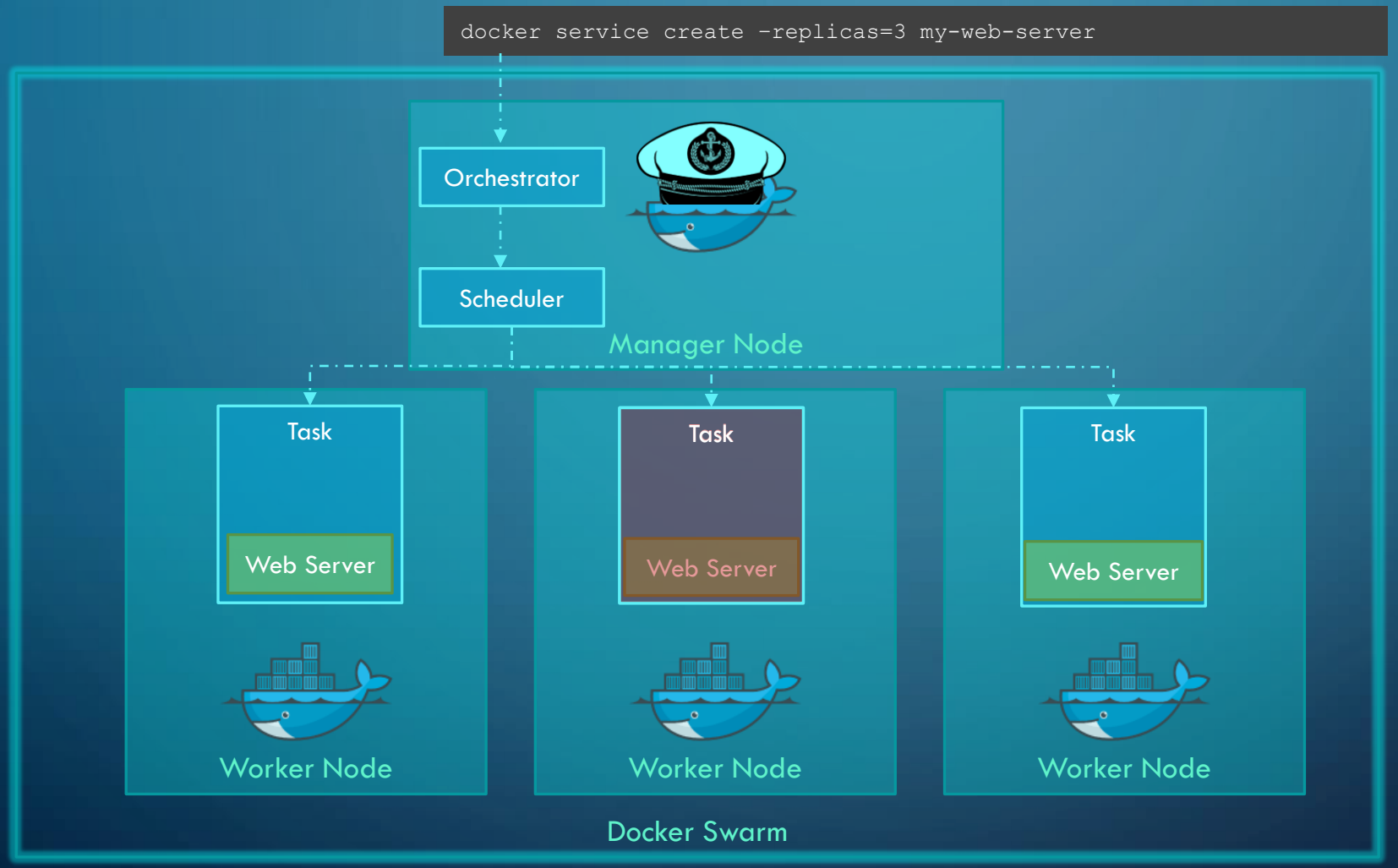
```
docker run my-web-server
```

```
docker service create --replicas=3 --network frontend my-web-server
docker service create --replicas=3 -p 8080:80 my-web-server
docker service create --replicas=3 my-web-server
```



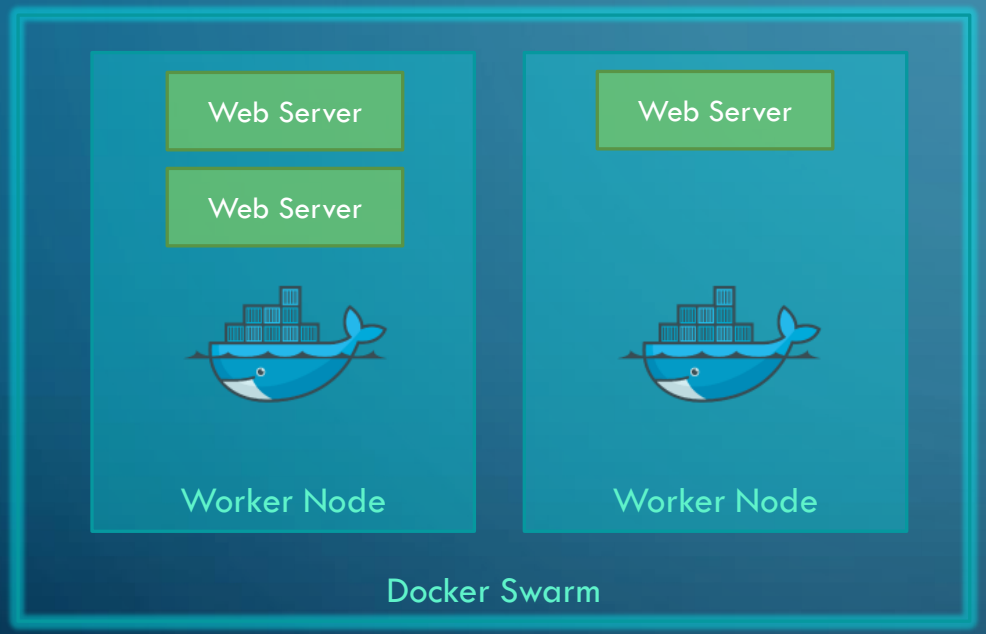


TASKS


docker service create -replicas=3 my-web-server
Orchestrator
Scheduler
Manager Node
Task
Web Server
Worker Node
Task
Web Server
Worker Node
Task
Web Server
Worker Node
Docker Swarm

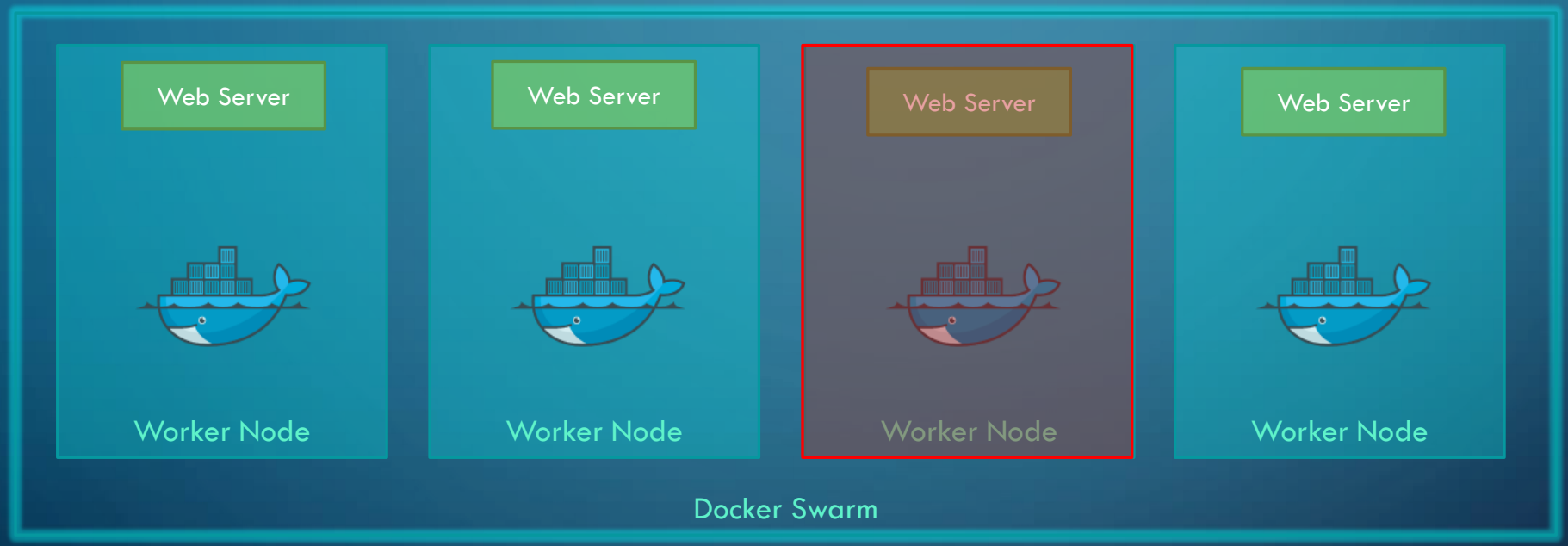
REPLICAS

```
docker service create -replicas=3 my-web-server
```



REPLICAS

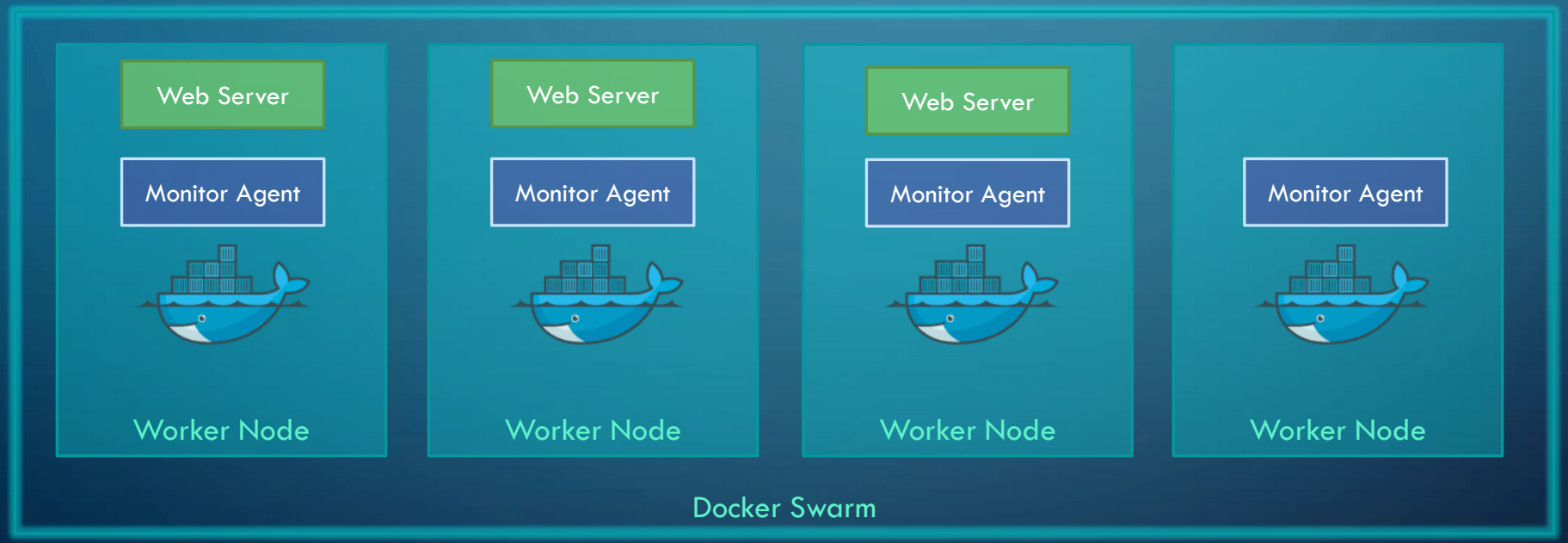
```
docker service create -replicas=3 my-web-server
```



REPLICAS VS GLOBAL

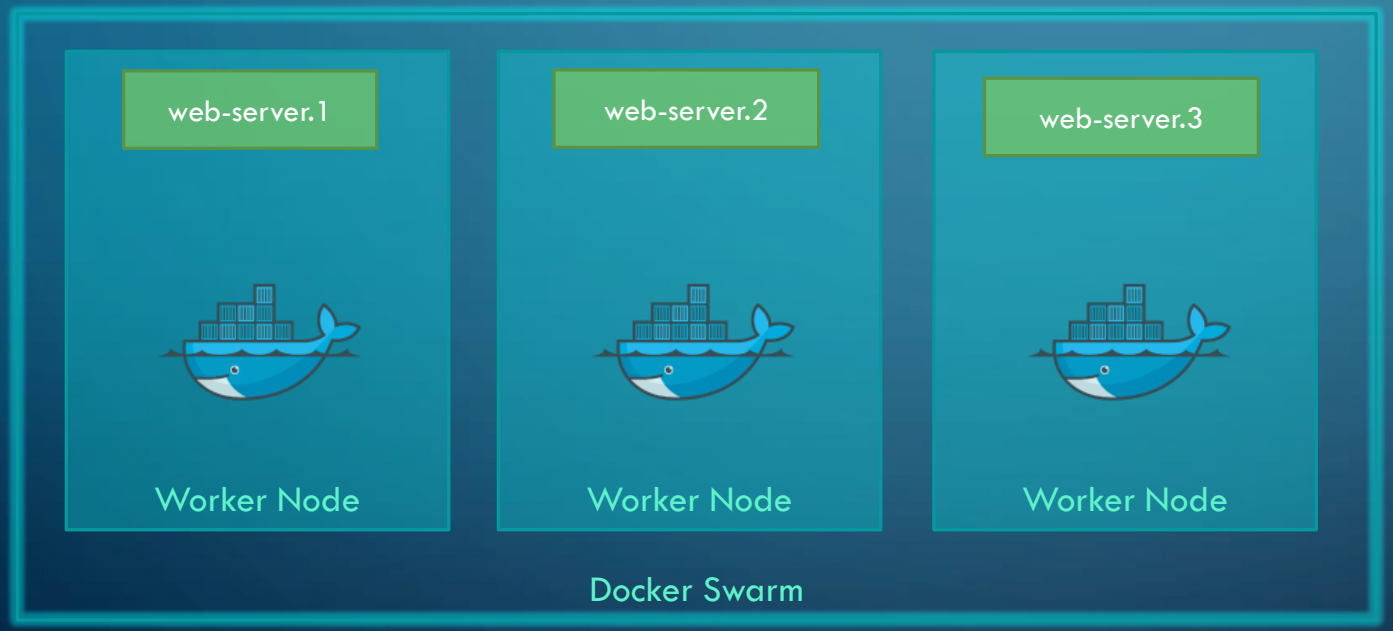
```
docker service create --replicas=3 my-web-server
```

```
docker service create --mode global my-monitoring-agent
```



SERVICE NAME

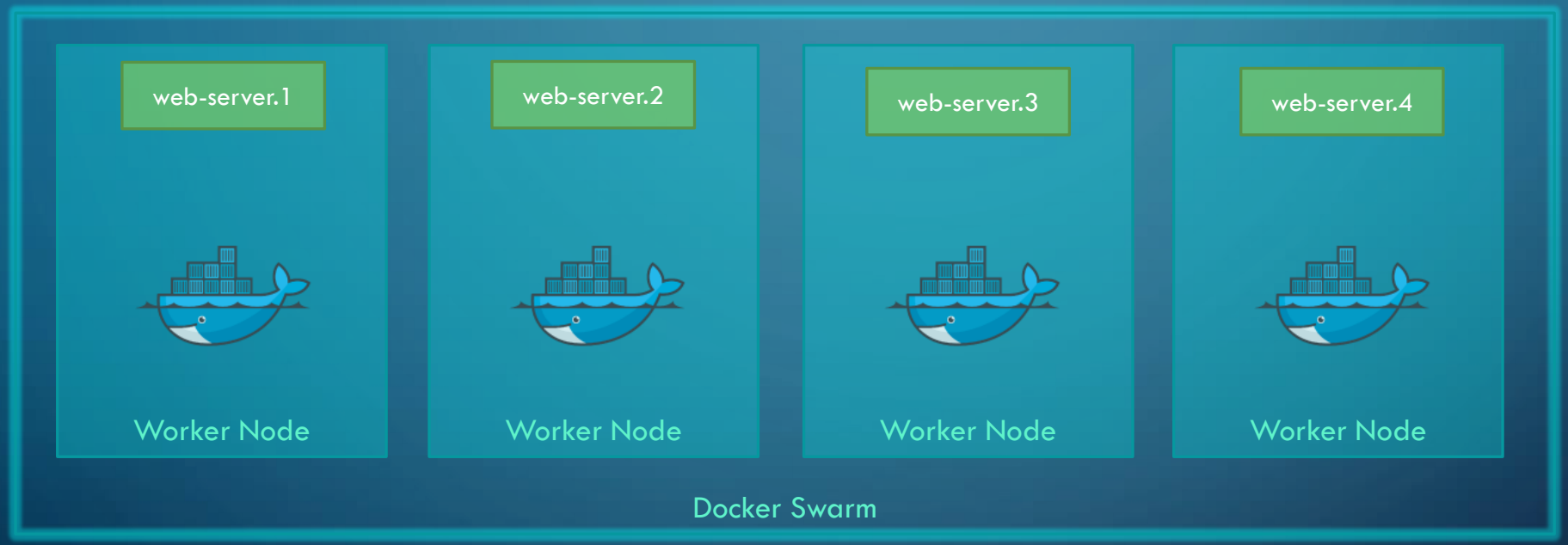
```
docker service create --replicas=3 --name web-server nginx
```



SERVICE UPDATE

```
docker service create -replicas=3 --name web-server my-web-server
```

```
docker service update -replicas=4 web-server
```



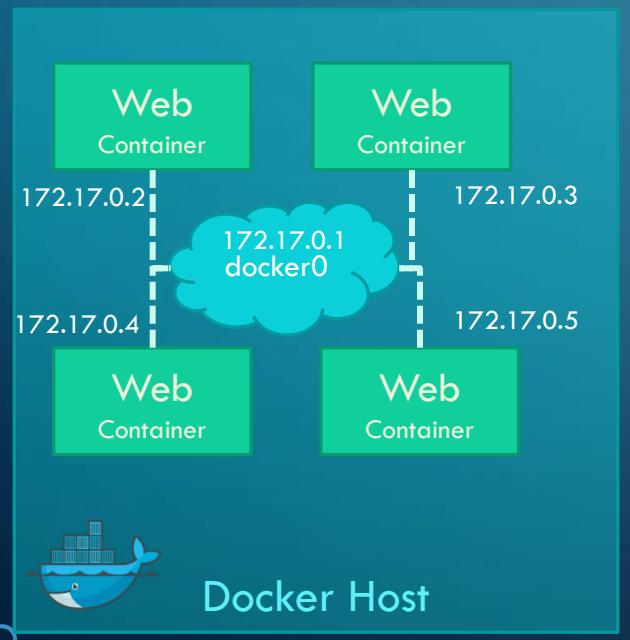
DOCKER NETWORKING

Mumshad Mannambeth | mmumshad@gmail.com

DEFAULT NETWORKS

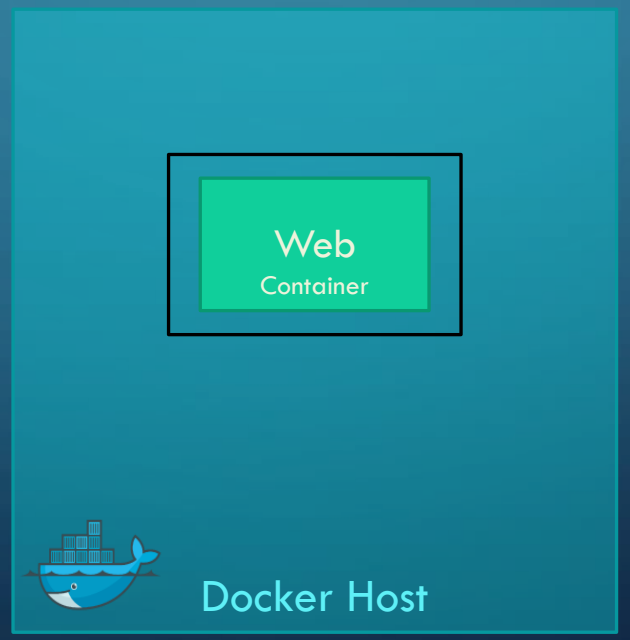
Bridge

```
docker run ubuntu
```



none

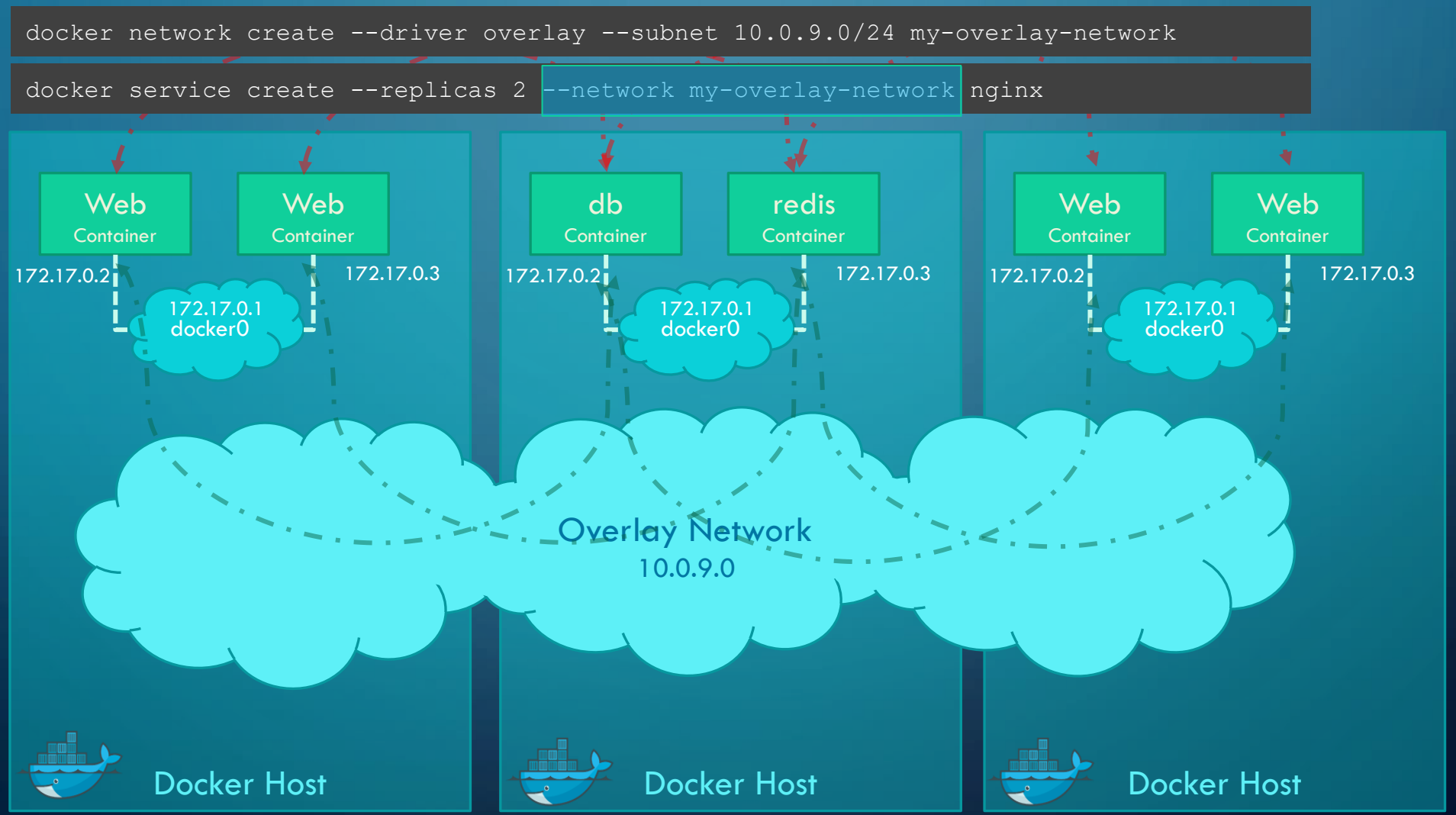
```
docker run Ubuntu --network=none
```



host

```
docker run Ubuntu --network=host
```

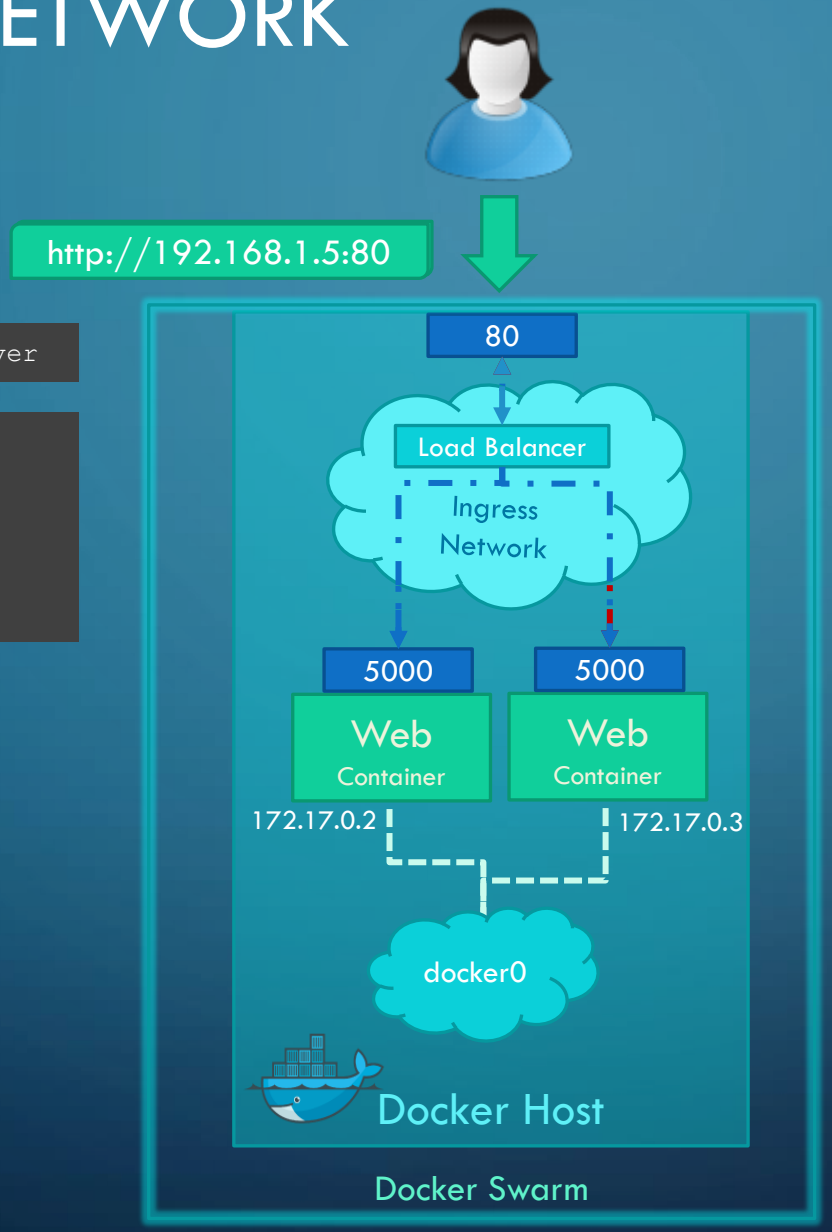
OVERLAY NETWORK



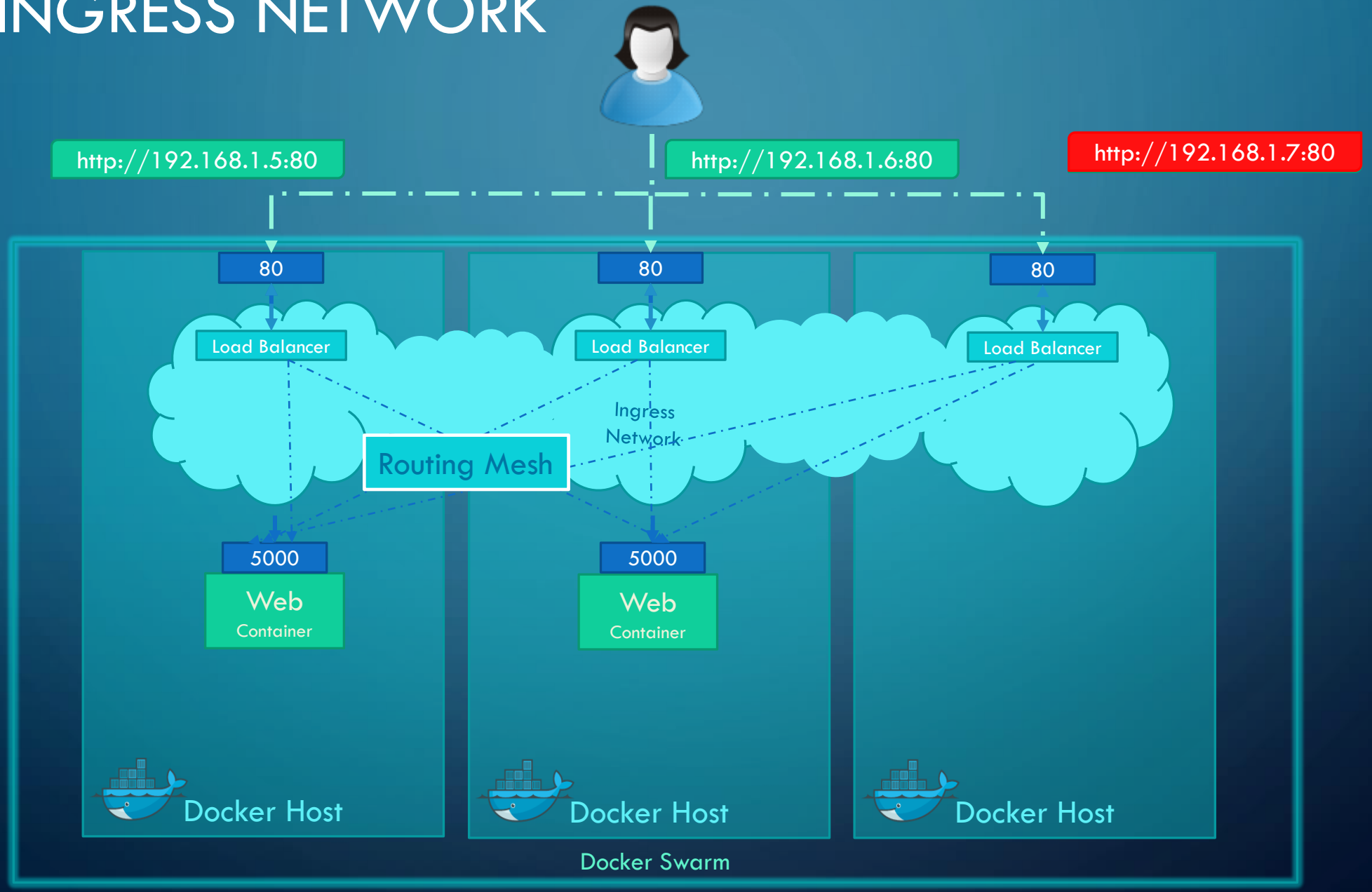
INGRESS NETWORK

```
docker run -p 80:5000 my-web-server
```

```
docker service create \
      --replicas=2 \
      -p 80:5000 \
      my-web-server
```

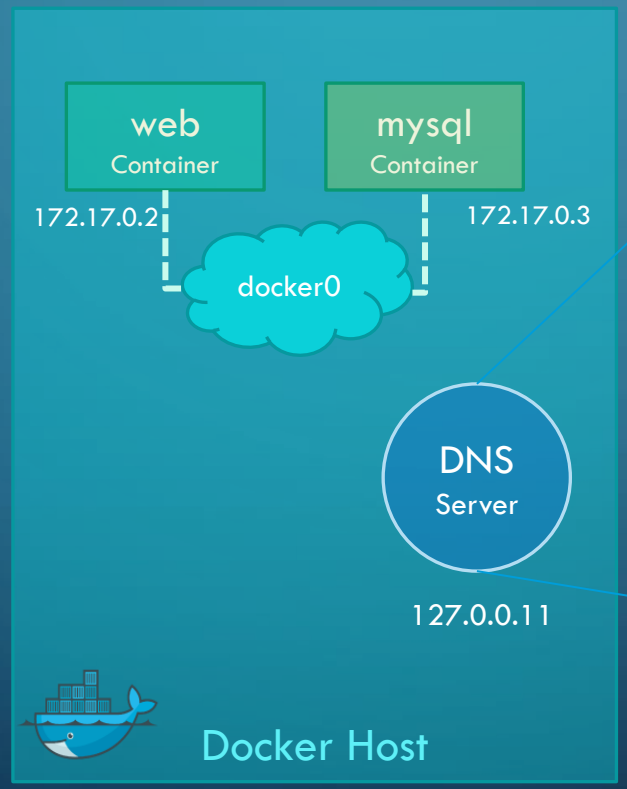


INGRESS NETWORK



EMBEDDED DNS

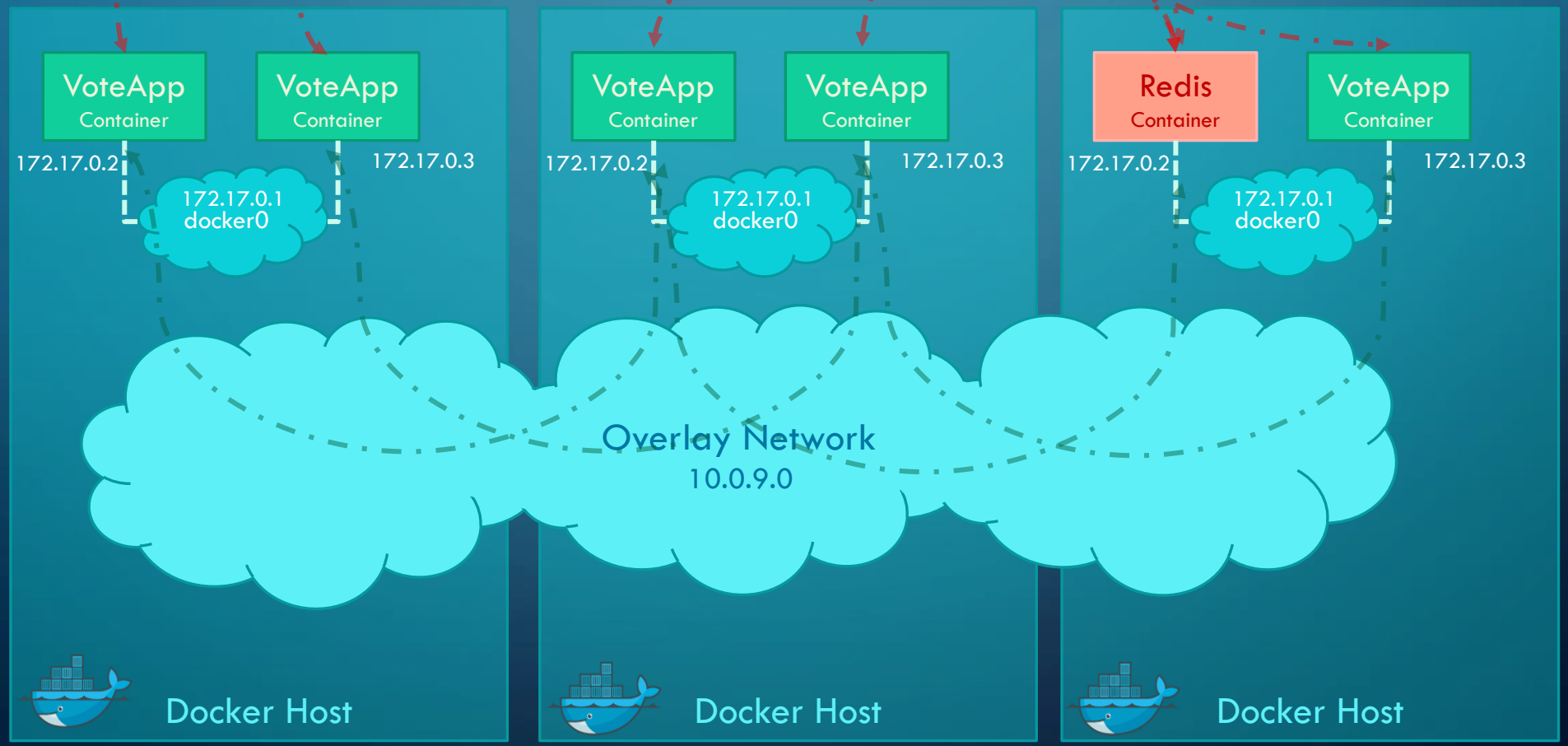
```
mysql.connect( mysql        )
```



Host	IP
web	172.17.0.2
mysql	172.17.0.3

OVERLAY NETWORK

```
docker network create --driver overlay --subnet 10.0.9.0/24 my-overlay-network
docker service create --replicas 2 --network my-overlay-network nginx
```



KodeKloud

Learn more about DevOps and Cloud courses with KodeKloud: https://kode.wiki/3N3A4kt

DOCKER STACKS

Mumshad Mannambeth | mmumshad@gmail.com

DOCKER STACK

```
docker run mmumshad/simple-webapp
```

```
docker run mongodb
```

```
docker run redis:alpine
```

```
docker run ansible
```

docker-compose.yml

```
services:
     web:
        image: "mmumshad/simple-webapp"
     database:
        image: "mongodb"
     messaging:
        image: "redis:alpine"
     orchestration:
        image:  "ansible"
```

```
docker-compose up
```

```
docker service create mmumshad/simple-webapp
```

```
docker service create mongodb
```

```
docker service create redis
```

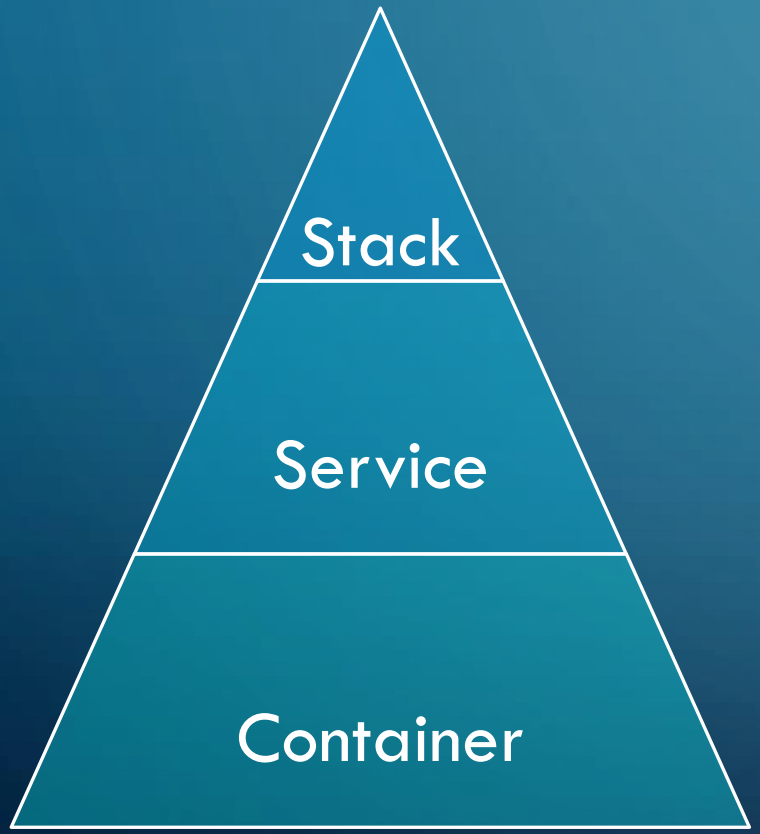
```
docker service create ansible
```

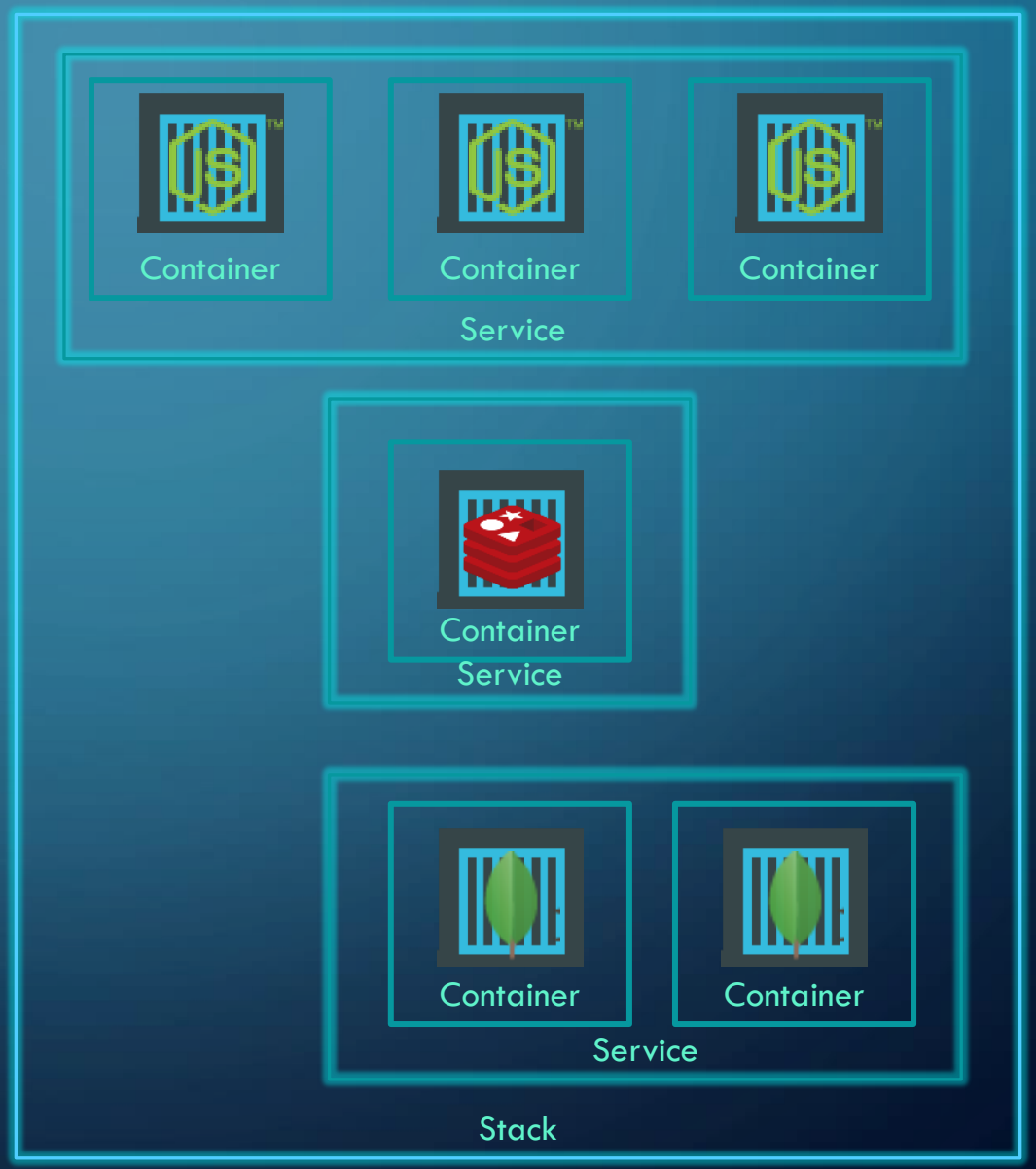
docker-compose.yml

```
version: 3
services:
     web:
        image: "mmumshad/simple-webapp"
     database:
        image: "mongodb"
     messaging:
        image: "redis:alpine"
     orchestration:
        image:  "ansible"
```

```
docker stack deploy
```

STACK


Stack
Service
Container


Container
Container
Container
Service
Container
Service
Container
Container
Service
Stack

SAMPLE APPLICATION IN SWARM

- Multiple Instances
- Placement Preferences
- Resource Constraints

STACK DEFINITION - REPLICA

docker-compose.yml

```
version: 3
services:
    redis:
        image: redis
        deploy:
            replicas: 1
    db:
        image: postgres:9.4
        deploy:
            replicas: 1
    vote:
        image: voting-app
        deploy:
            replicas: 2
    result:
        image: result
        deploy:
            replicas: 1
    worker:
        image: worker
        deploy:
            replicas: 1
```

STACK DEFINITION - PLACEMENT

docker-compose.yml

```
version: 3
services:
     redis:
          image: redis
          deploy:
               replicas: 1
     db:
          image: postgres:9.4
          deploy:
               placement:
                    constraints:
                      - node.hostname == node1
                      - node.role == manager
```

STACK DEFINITION - RESOURCES

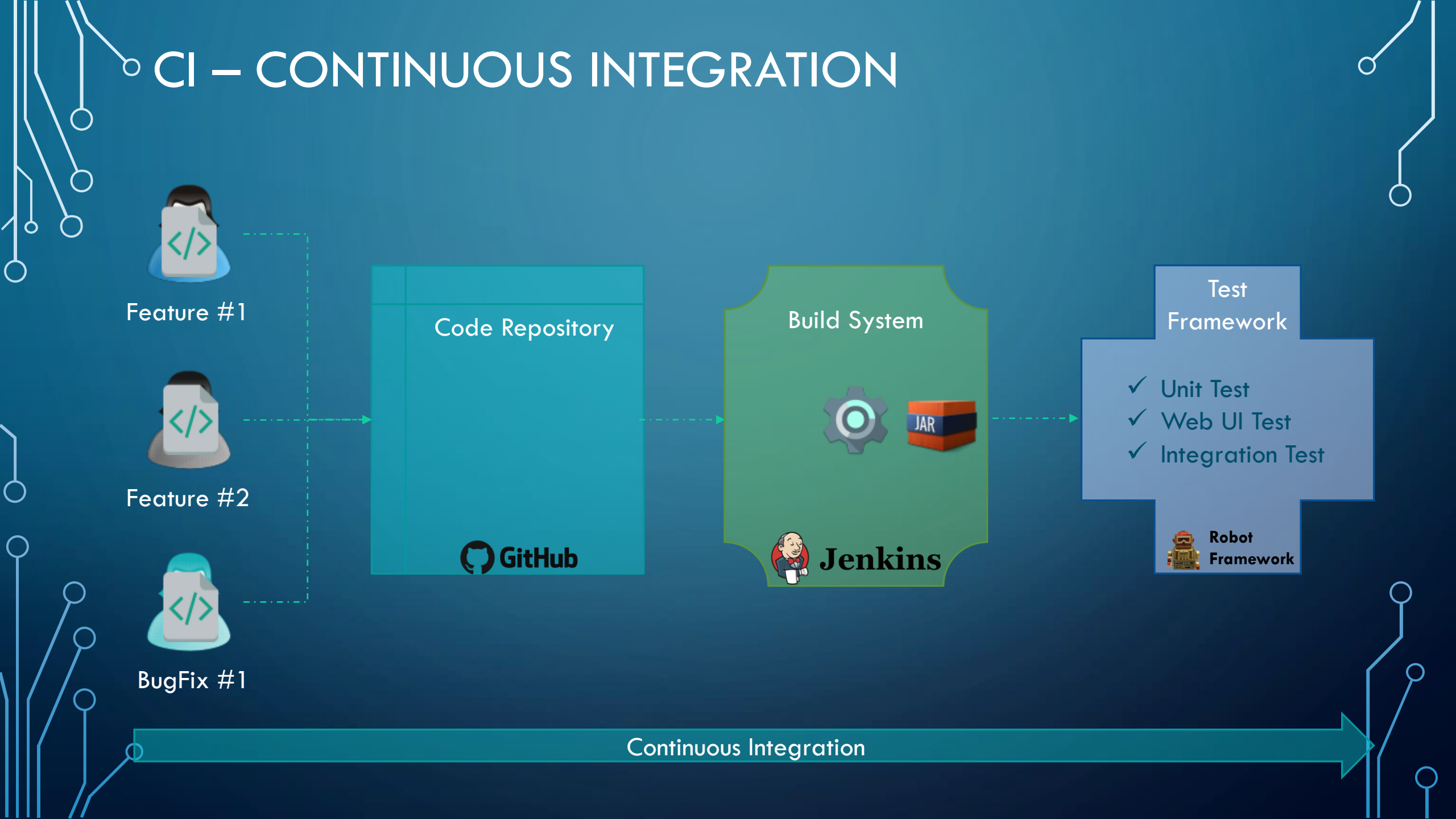
docker-compose.yml

```
version: 3
services:
     redis:
          image: redis
          deploy:
               replicas: 1
               resources:
                    limits:
                       cpus: 0.01
                       memory: 50M
```

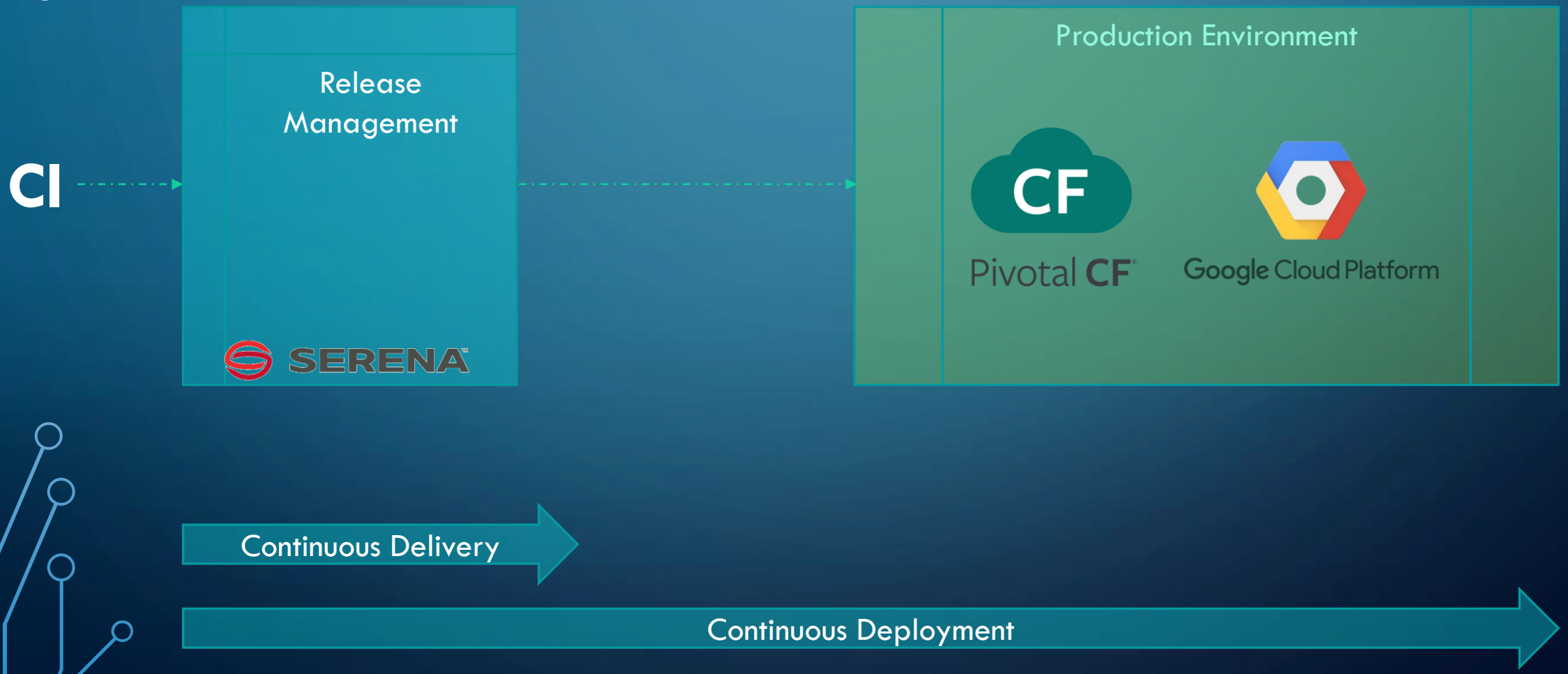
CI/CD

Mumshad Mannambeth | mmumshad@gmail.com

CI – CONTINUOUS INTEGRATION



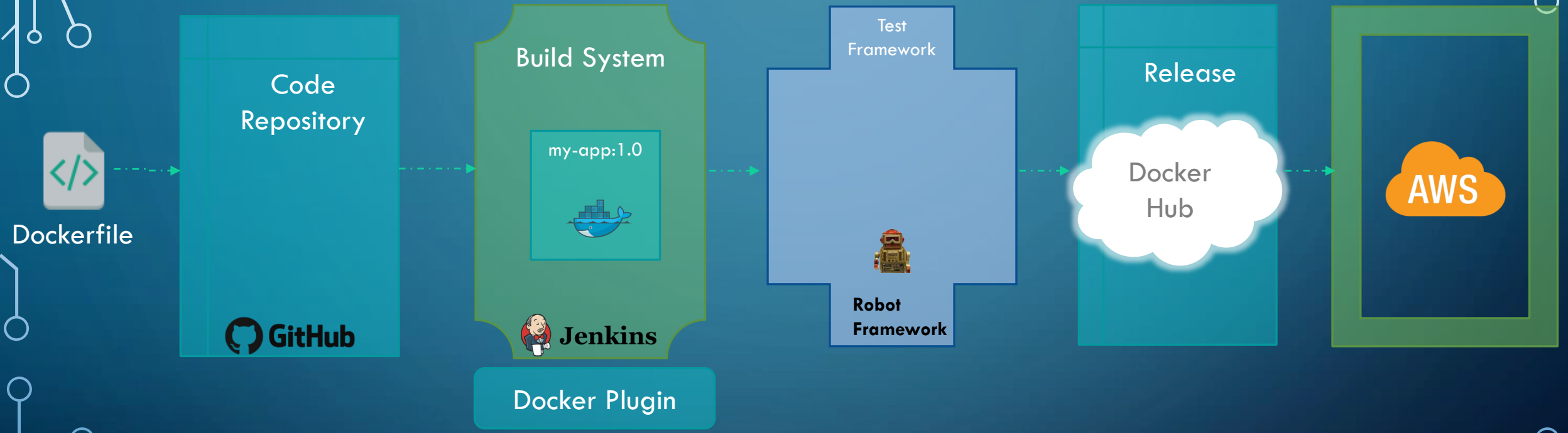
CD – CONTINUOUS DELIVERY/DEPLOYMENT



CI/CD - DOCKER

Mumshad Mannambeth | mmumshad@gmail.com

BUILD SYSTEMS – DOCKER SUPPORT



PUBLIC CLOUD – DOCKER SUPPORT

Google Cloud Platform

Google Container Engine
(GKE)

kubernetes

AWS

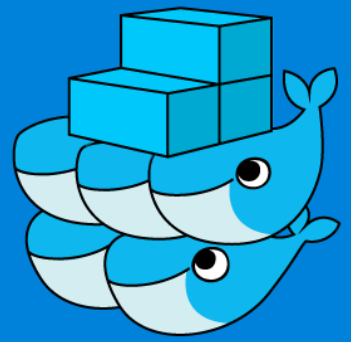
EC2 Container Service
(ECS)

CF

Pivotal CF

Pivotal Container Service
(PKS)

docker cloud



Docker Swarm

KodeKloud

Learn more about DevOps and Cloud courses with KodeKloud: https://kode.wiki/3N3A4kt

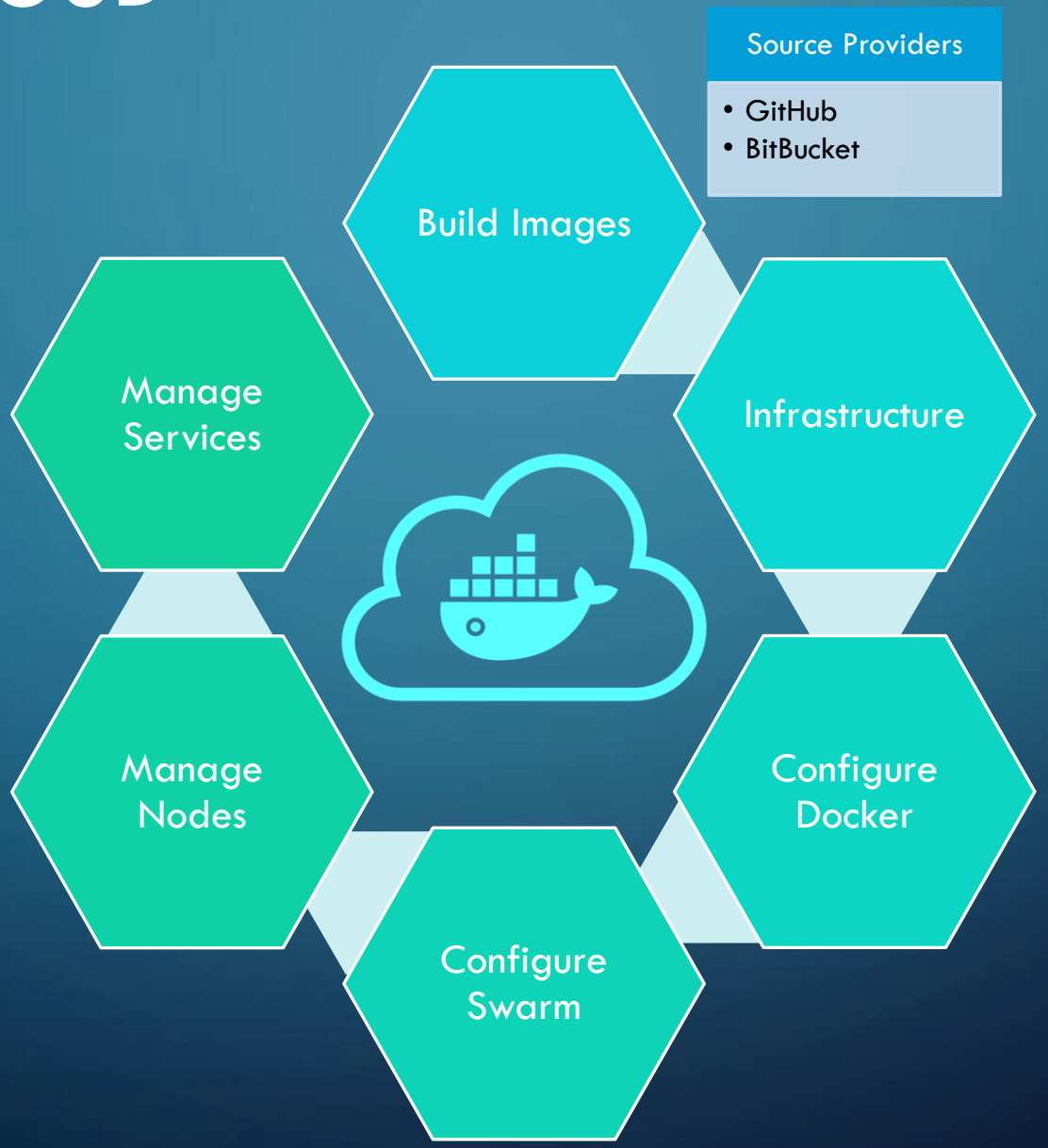
DOCKER ON CLOUD

Mumshad Mannambeth | mmumshad@gmail.com

DOCKER CLOUD

Mumshad Mannambeth | mmumshad@gmail.com

DOCKER CLOUD



Source Providers

- GitHub
- BitBucket

Cloud Providers

- AWS
- Digital Ocean
- Microsoft Azure
- SoftLayer
- Packet

DOCKER CLOUD - BUILD

Feature #1

Code Repository

GitHub

Build System

my-app:1.0

Docker Cloud

Release

Docker Hub

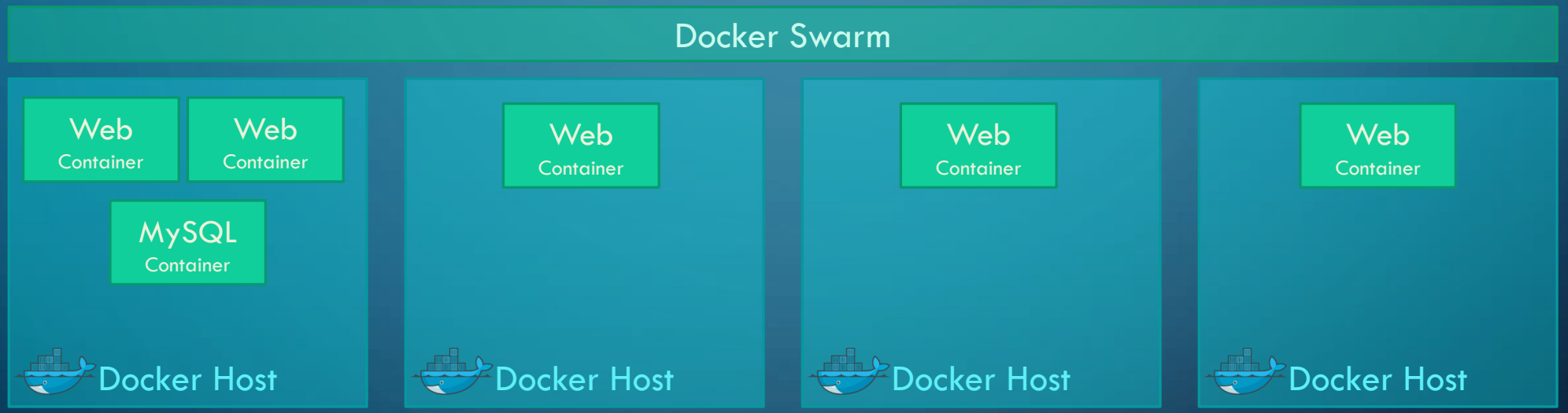
KUBERNETES

Mumshad Mannambeth | mmumshad@gmail.com

OBJECTIVE

- Introduction
- Architecture Basics
- Services
- Deployment
- GCP Kubernetes

CONTAINER ORCHESTRATION



CLUSTER

Docker Swarm

Kubernetes

PODS

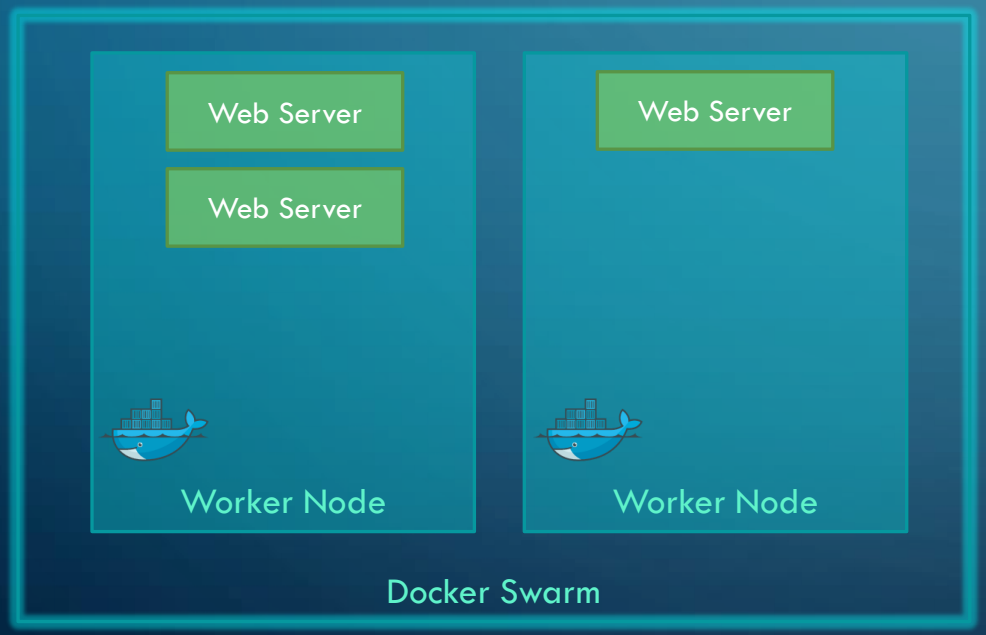

Container
Pod
Kube Worker
Container
Pod
Kube Worker
Container
Pod
Kube Worker

PODS

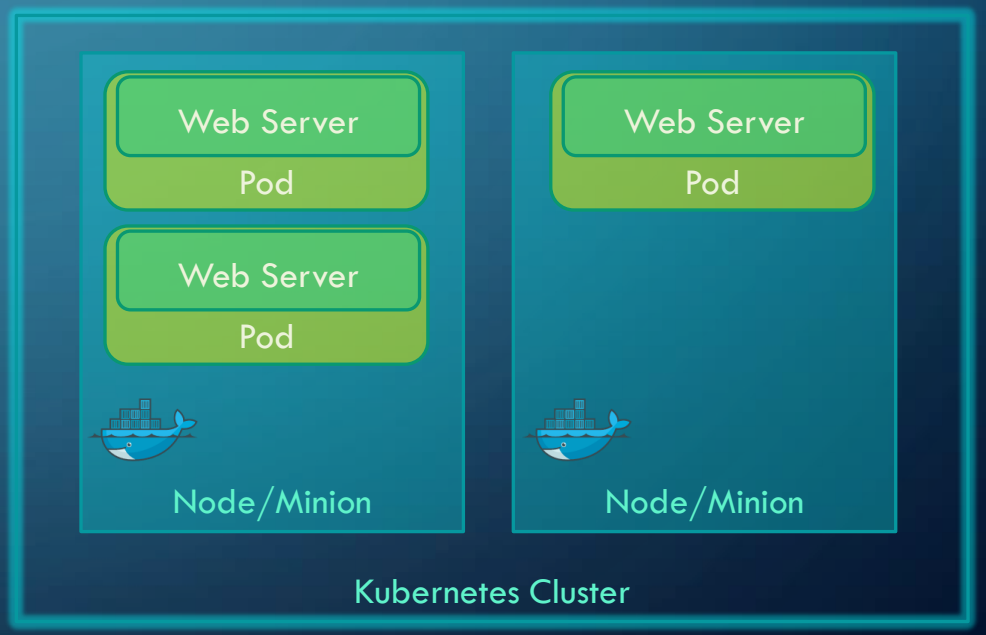

IP Address
Container1
Container2
Storage
Pod
IP Address
Container1
Container2
Storage
Pod
Kube Worker
IP Address
Container1
Container2
Storage
Pod
Kube Worker
IP Address
Container1
Container2
Storage
Pod
Kube Worker

KUBERNETES - DEPLOYMENT

Docker Swarm - Services

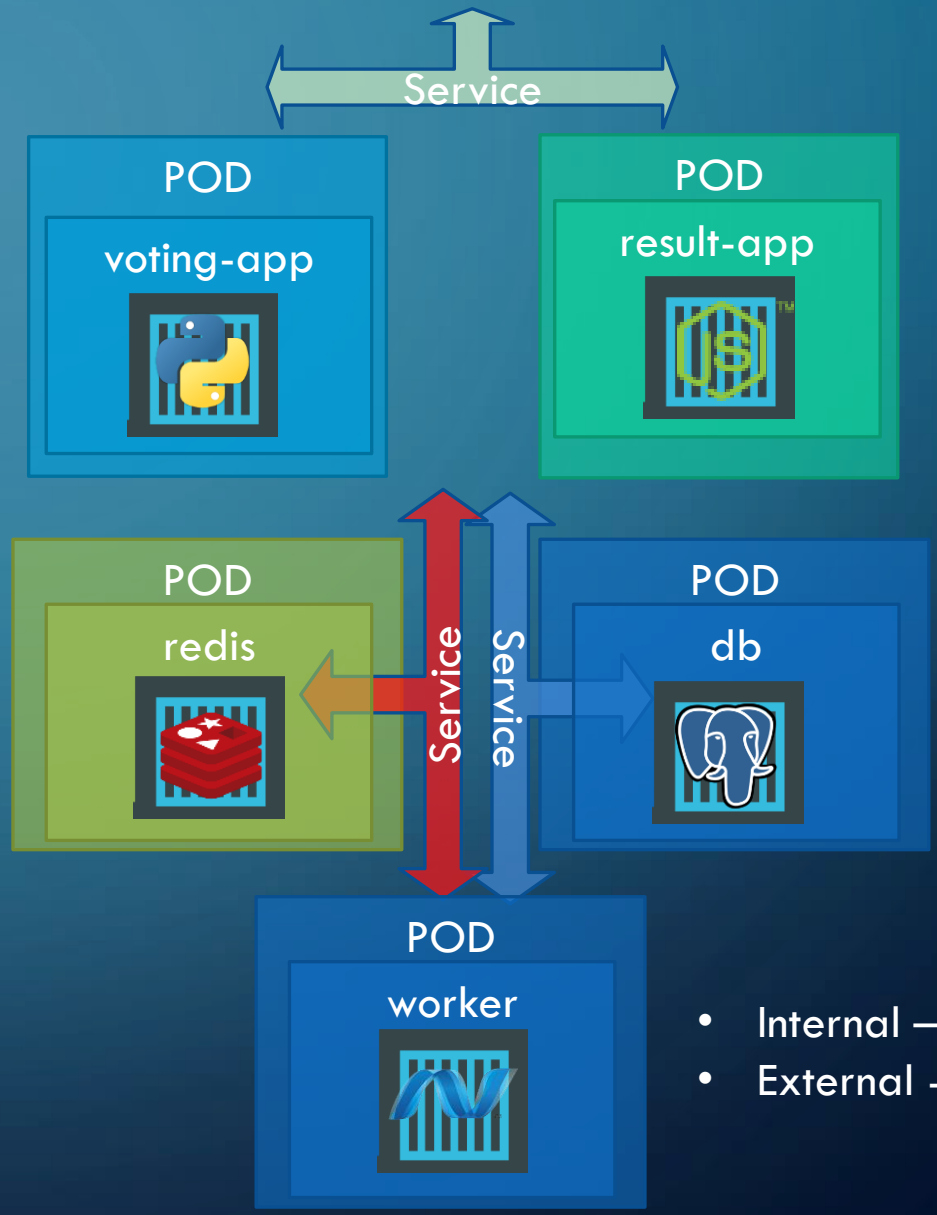


Kubernetes – Deployment
- Replica Sets



KUBERNETES SERVICES

Docker Swarm – Links



- Internal – ClusterIP
- External - LoadBalancer

KUBECTL

`kubectl create`

```
kubectl create -f pod-definition.yml
```

`kubectl get`

```
kubectl get pods
```

```
NAME                     READY     STATUS    RESTARTS   AGE
nginx-1423793266-fc31d   1/1       Running   0          19m
```

`kubectl describe`

```
kubectl describe pods
```

```
Name:           nginx-1423793266-fc31d
Namespace:      default
Node:           node1/10.0.41.4
Start Time:     Wed, 15 Nov 2017 19:46:03 +0000
Labels:         pod-template-hash=1423793266
                run=nginx
Annotations:    kubernetes.io/created-by={"kind":"SerializedReference","apiVersion
"9cd19eb1-ca3d-11e7-a82a-0242b894170f","...
Status:         Running
IP:             10.32.0.4
Created By:     ReplicaSet/nginx-1423793266
Controlled By:  ReplicaSet/nginx-1423793266
Containers:
  nginx:
    Container ID:       docker://10e663285e1a8f7a227833ca7ce6513cbe6bf48e10130b2d4
    Image:              nginx
    Image ID:           docker-pullable://nginx@sha256:9fca103a62af6db7f188ac3376c
    Port:               80/TCP
    State:              Running
      Started:          Wed, 15 Nov 2017 19:46:04 +0000
    Ready:              True
    Restart Count:      0
    Environment:        <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-9pjng (ro)
```

KUBERNETES DEFINITION FILE

Version	v1		
Kind	Pod	Deployment	Service
Metadata	Name	Labels	
Specification	containers	ports	

pod-definition.yml

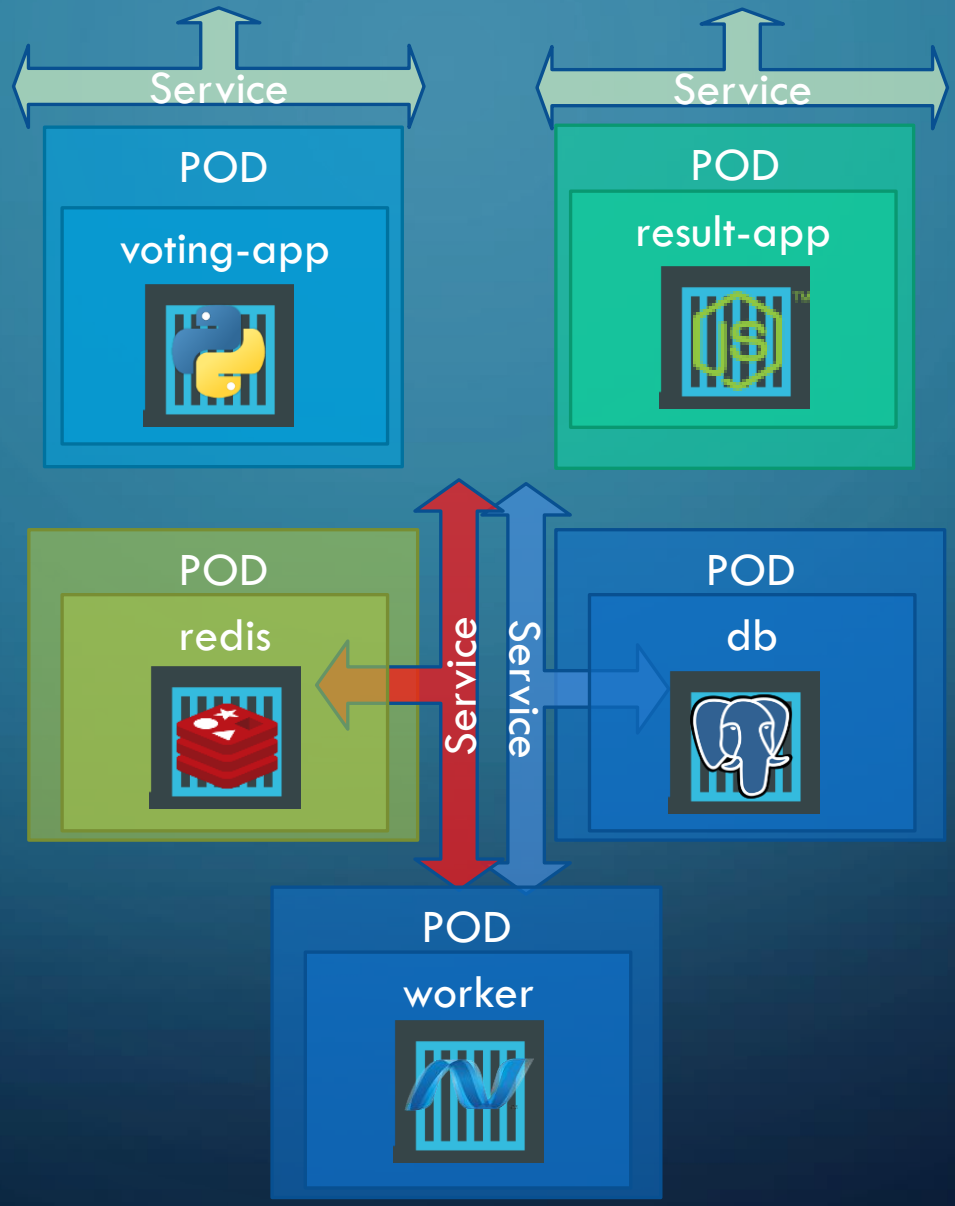
```
apiVersion: v1

kind: pod

metadata:
    name: my-nginx

spec:
    containers:
      - name: nginx
        image: nginx
        ports:
          - containerPort: 80
```

```
kubectl create -f pod-definition.yml
```

EXAMPLE VOTING APP

Goals:

1. Deploy Containers
2. Enable Connectivity
3. External Access



Steps:

1. Deploy PODs
2. Create Services (ClusterIP)
3. Create Services (LoadBalancer)

DEMO

1. Setup a Kubernetes Test Environment
2. Create Kubernetes PODs
3. Create Services – ClusterIP - Internal
4. Create Services – LoadBalancer - External

DEMO

1. Setup a Google Container Engine Environment
2. Create Kubernetes PODs
3. Create Services – ClusterIP - Internal
4. Create Services – LoadBalancer - External

DOCKER REGISTRY

Mumshad Mannambeth | mmumshad@gmail.com

DOCKER REGISTRY

Private Docker Registry

Public Docker registry - dockerhub

```
docker push localhost:5000/mmumshad/my-custom-app
```

```
docker build . -t mmumshad/my-custom-app
```

```
docker run -d -p 5000:5000 registry:2
```

```
docker push mmumshad/my-custom-app
```

```
docker build . -t mmumshad/my-custom-app
```

CONCLUSION

- ☑ Docker Architecture
- ☑ Docker For Windows
- ☑ Docker Service
- ☑ Docker Swarm
- ☑ Overlay Networks
- ☑ Load Balancing
- ☑ CI/CD Integration

KodeKloud

Learn more about DevOps and Cloud courses with KodeKloud: https://kode.wiki/3N3A4kt