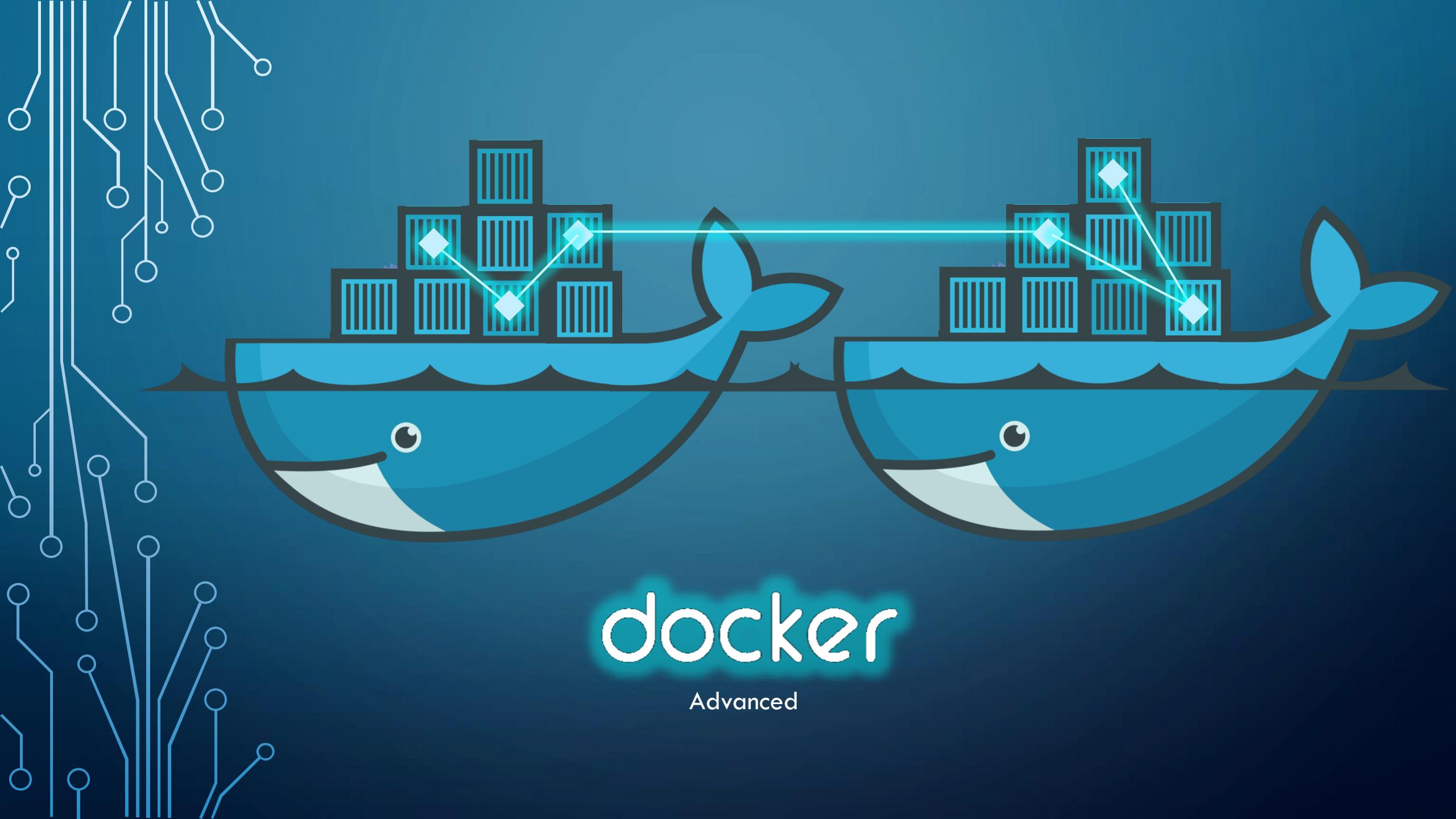


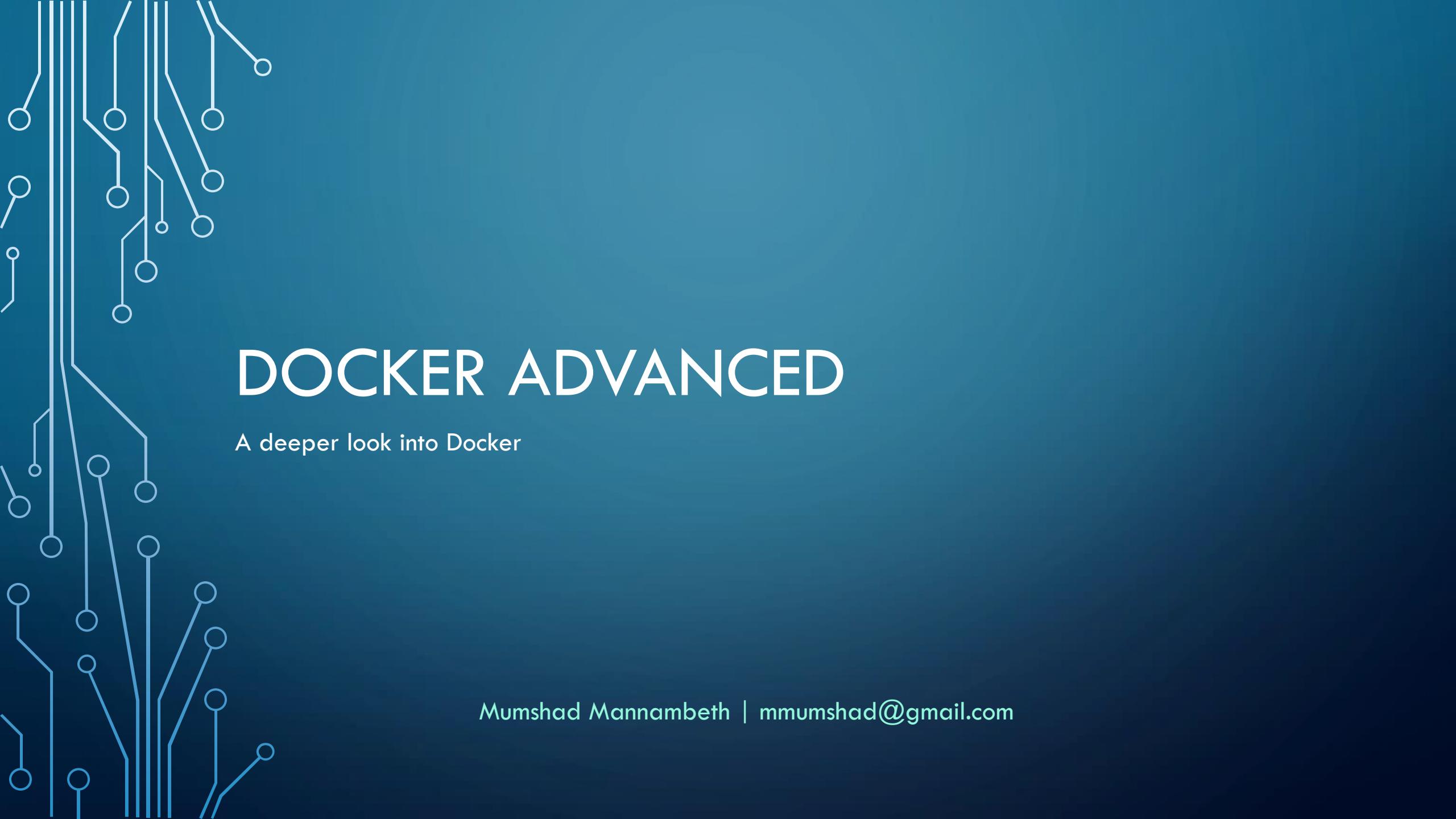


**docker**  
Advanced





**docker**  
Advanced



# DOCKER ADVANCED

A deeper look into Docker

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

## Docker for the Absolute Beginner - Hands On

Learn Docker with Hands On Coding Exercises

NEW ★★★★★ 4.6 (116 ratings) 6,004 students enrolled

Created by Mumshad Mannambeth Last updated 9/2017 English

### What Will I Learn?

- ✓ Beginner level introduction to Docker
- ✓ Build Docker images using Dockerfiles with Hands-On Exercises
- ✓ Build Application stack using Docker Compose Files with Hands-On Exercises
- ✓ Basic Docker Commands with Hands-On Exercises
- ✓ Understand what Docker Compose is
- ✓ Understand what Docker Swarm is



\$12 \$40 70% off

2 days left at this price!

Buy Now

Add to Cart

30-Day Money-Back Guarantee

### Includes:

2.5 hours on-demand video

28 Coding exercises

# OBJECTIVES

- Docker Overview
- Running Docker Containers
- Creating a Docker Image
- Docker Compose
- Docker Swarm
- Networking in Docker
- Docker Concepts in Depth
- 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





# 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

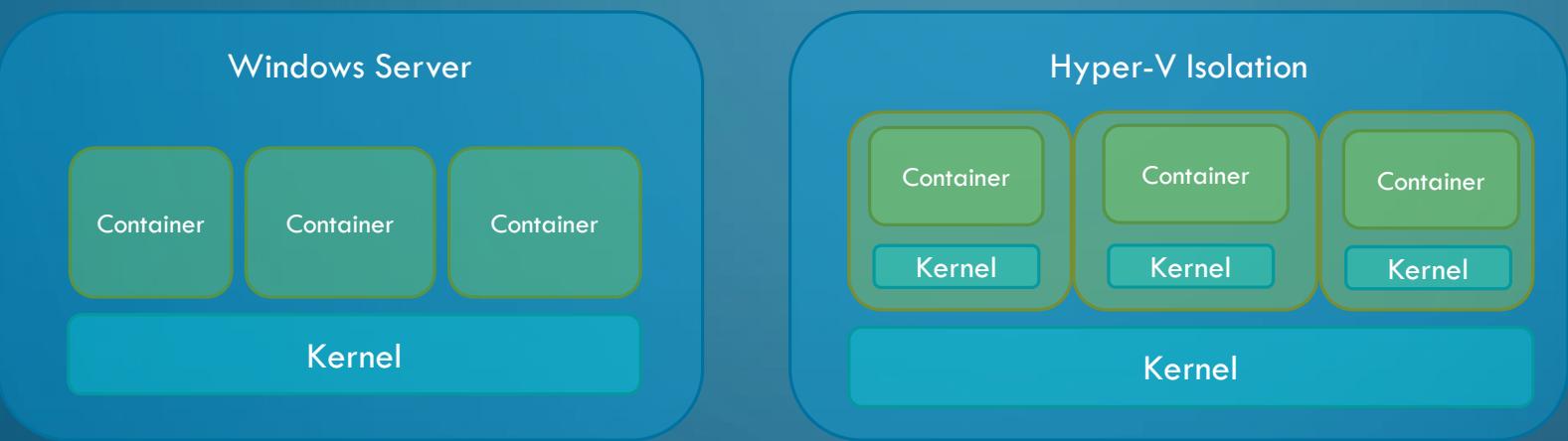


Support: Windows 10 Enterprise/Professional Edition  
Windows Server 2016

Linux Containers (Default)  
Or  
Windows Containers

# WINDOWS CONTAINERS

## Container Types:



## Base Images:

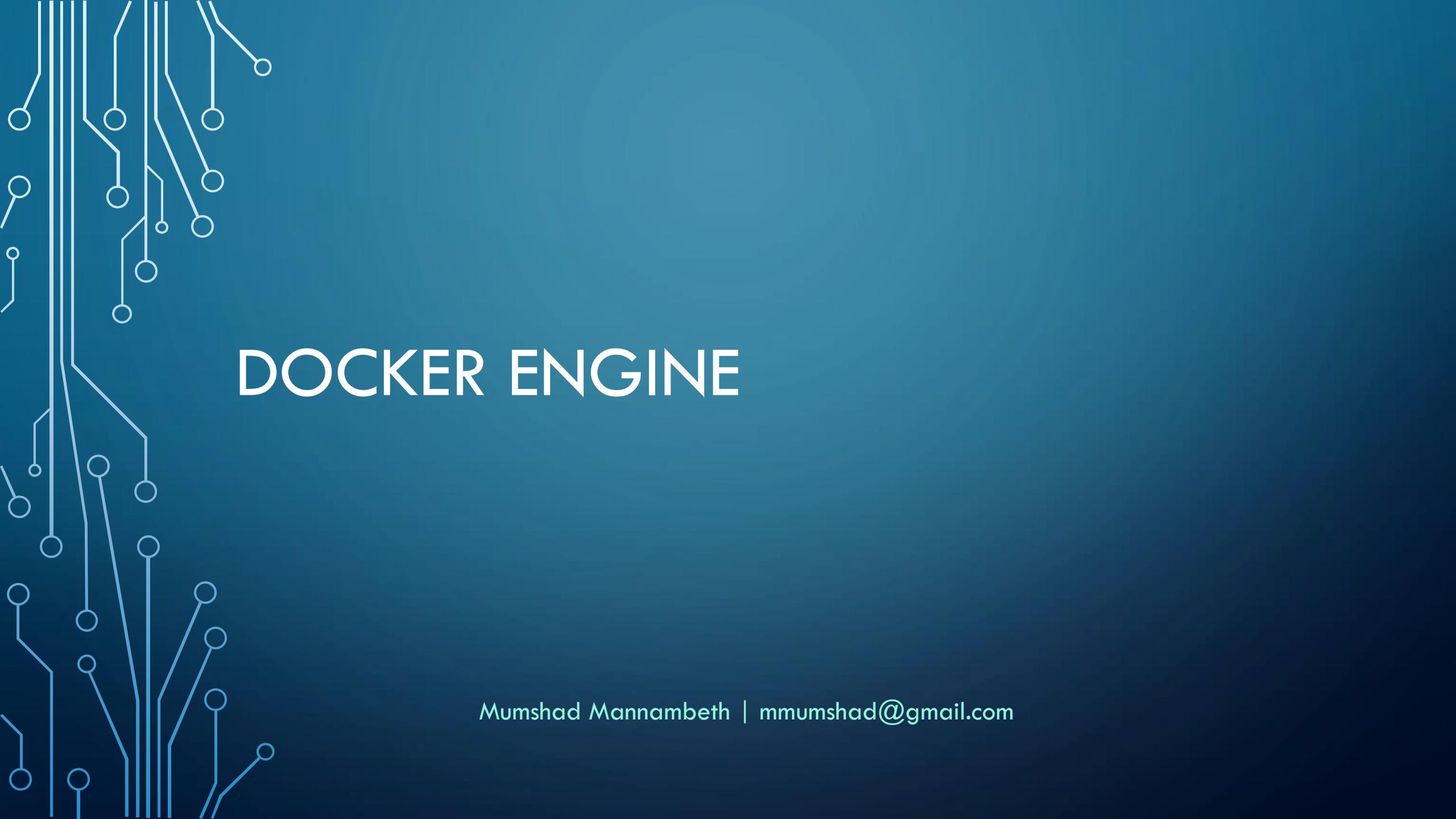
- Windows Server Core
- Nano Server

## Support

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



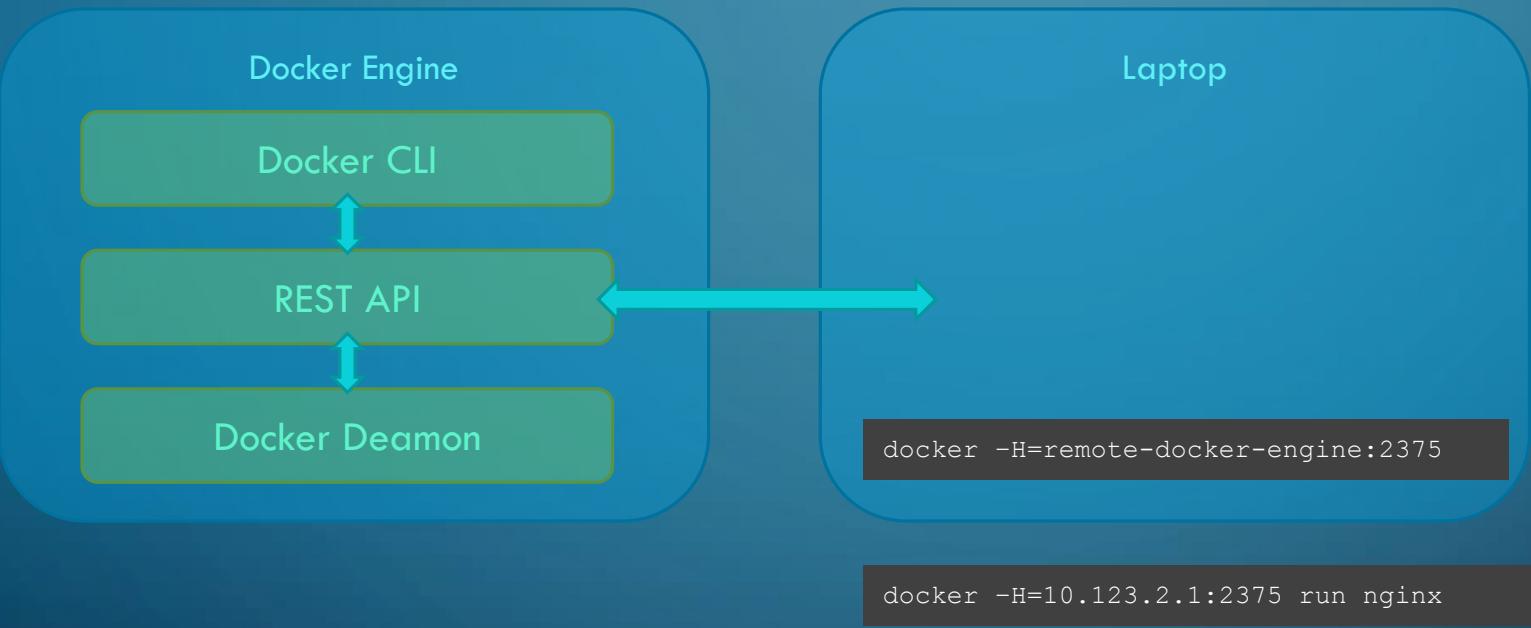
DEMO



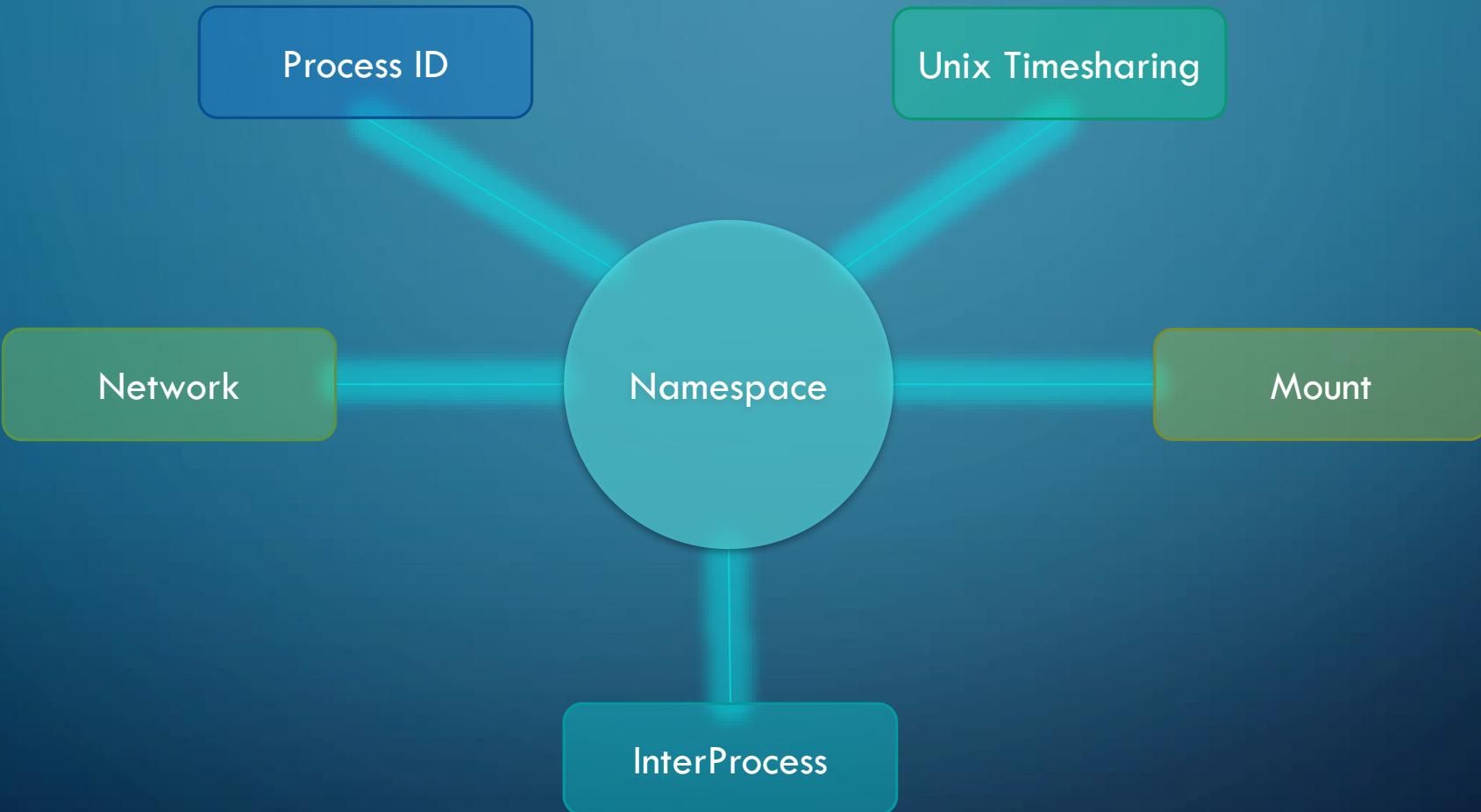
# DOCKER ENGINE

Mumshad Mannambeth | mmumshad@gmail.com

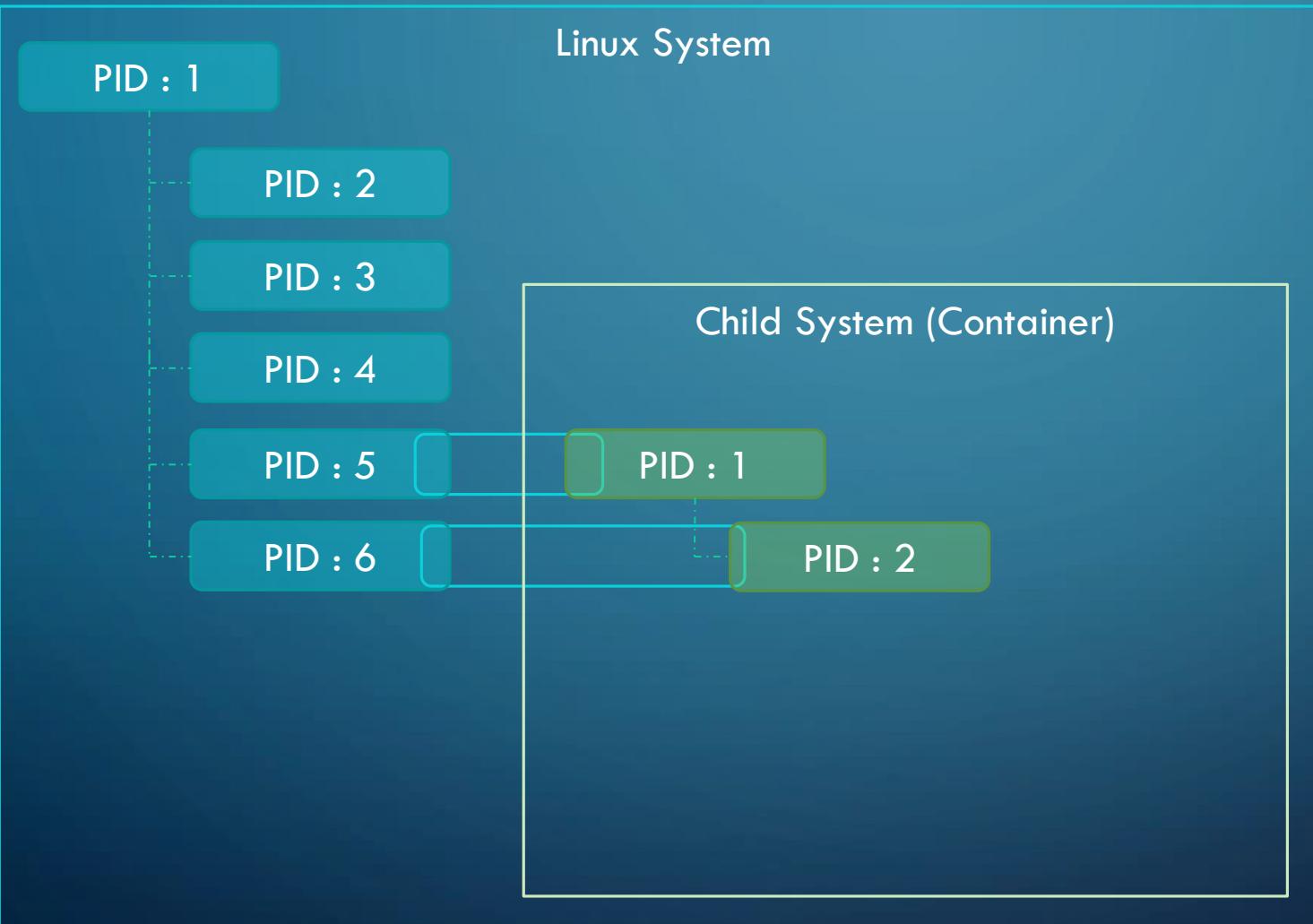
# DOCKER ENGINE



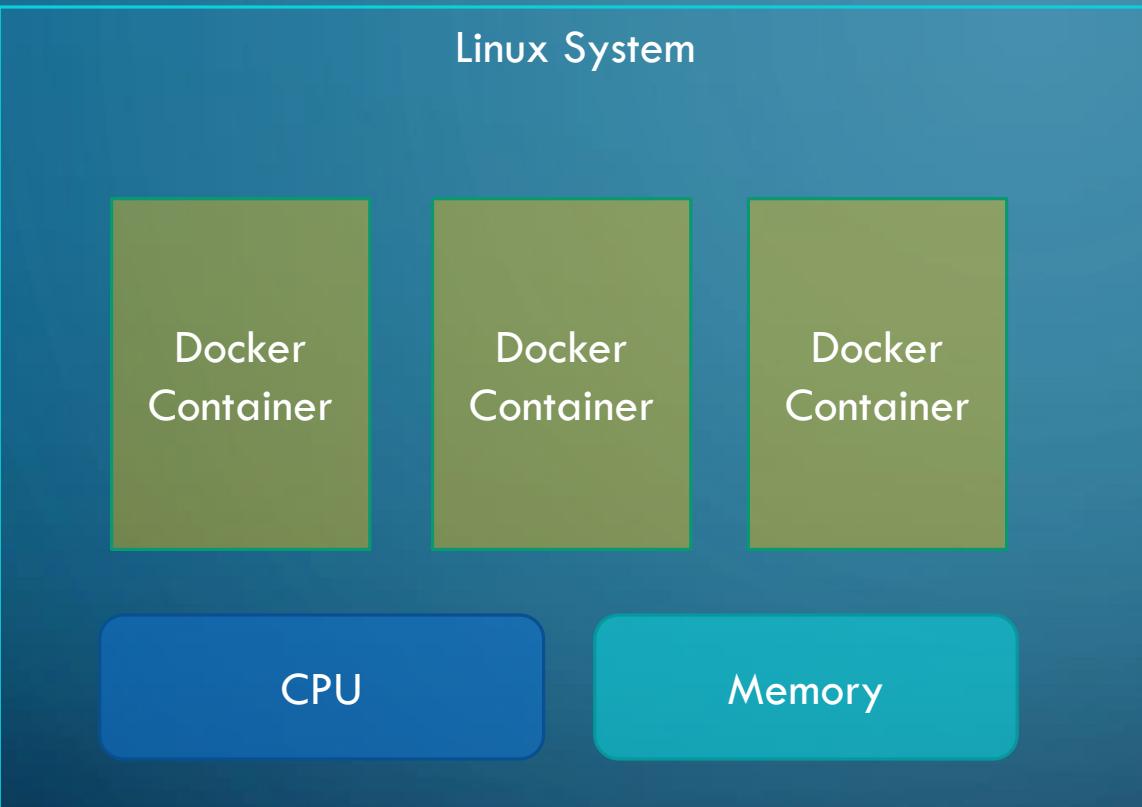
# CONTAINERIZATION



# NAMESPACE - PID



# CGROUPS



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

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



# 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

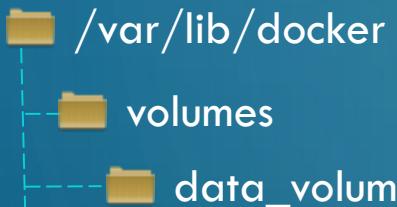


Image Layers



# VOLUMES

```
docker volume create data_volume
```

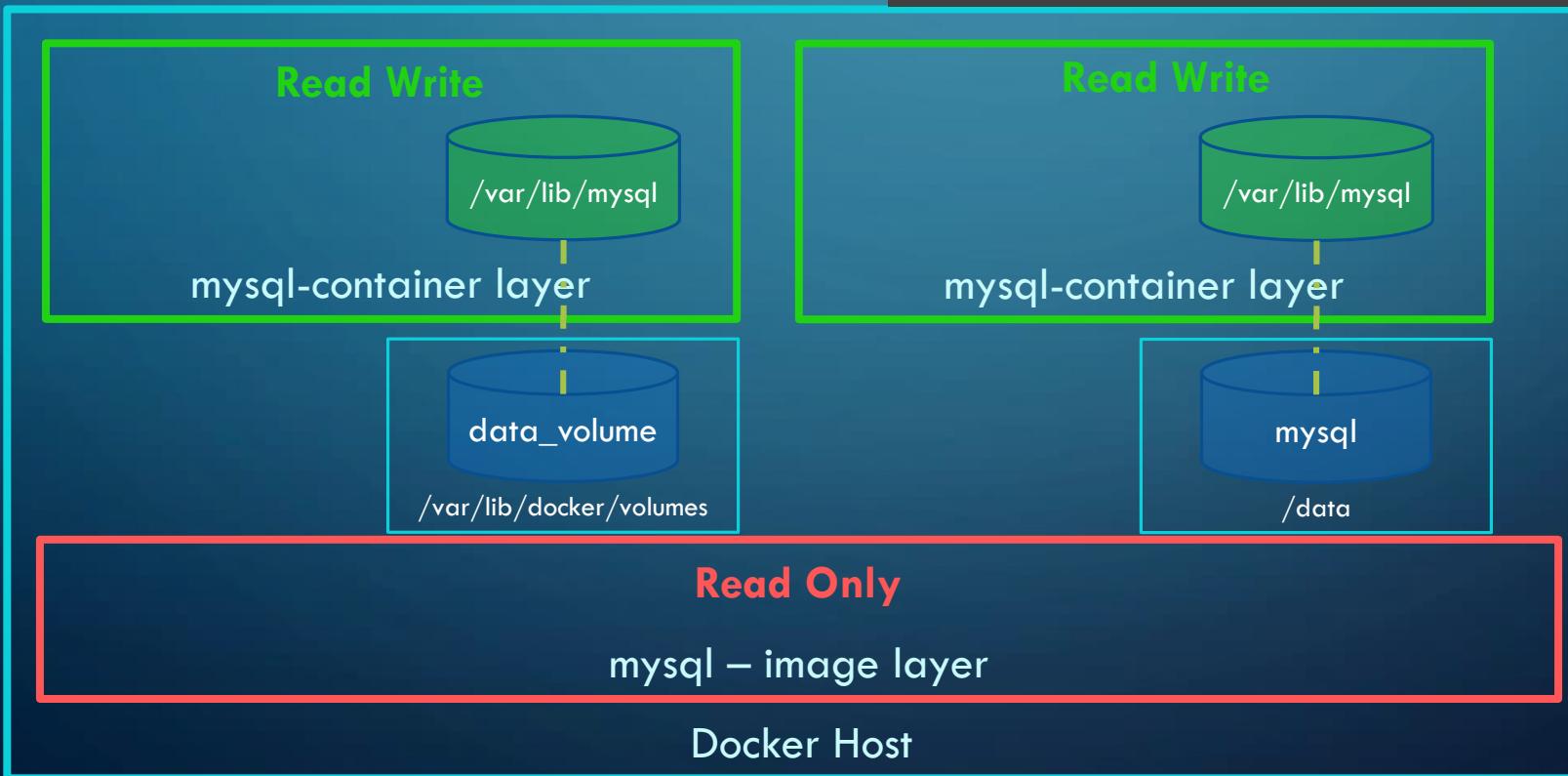


```
docker run -v data_volume:/var/lib/mysql mysql
```

```
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 SWARM

Mumshad Mannambeth | mmumshad@gmail.com

# DOCKER SWARM



# SETUP SWARM

Swarm Manager

```
docker swarm init
```



Node  
Worker

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



Node  
Worker

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



Node  
Worker

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

```
root@osboxes:/root/simple-webapp-docker # docker swarm init --advertise-addr 192.168.1.12  
Swarm initialized: current node (0j76dum2r56plxfne4ub1ps2c) is now a manager.
```

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

```
docker swarm join --token SWMTKN-1-35va8b3fi5krpdskefqqxgttmulw3z828daucr7y526ne0sgu-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



Swarm Manager



Swarm Manager



Worker



Worker



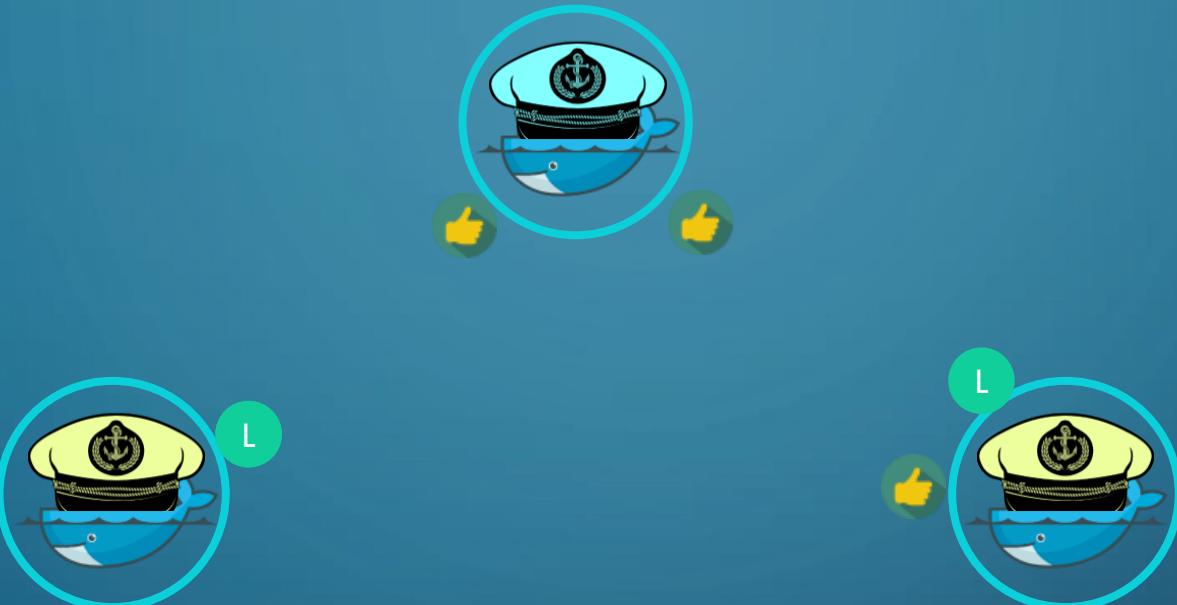
Worker



Worker



# 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

Quorum of N =

$$\frac{N + 1}{2}$$

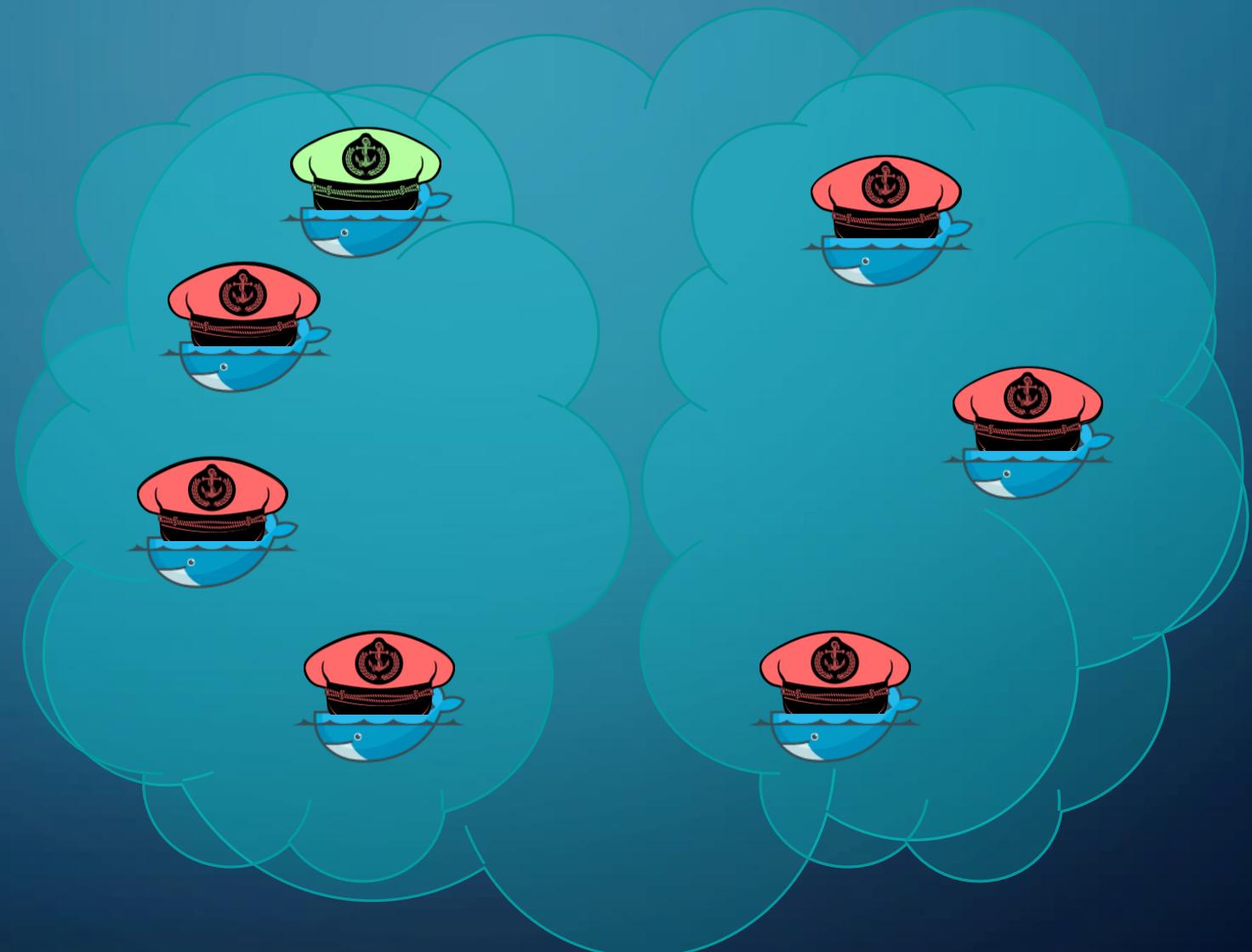
Quorum of 5 =

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

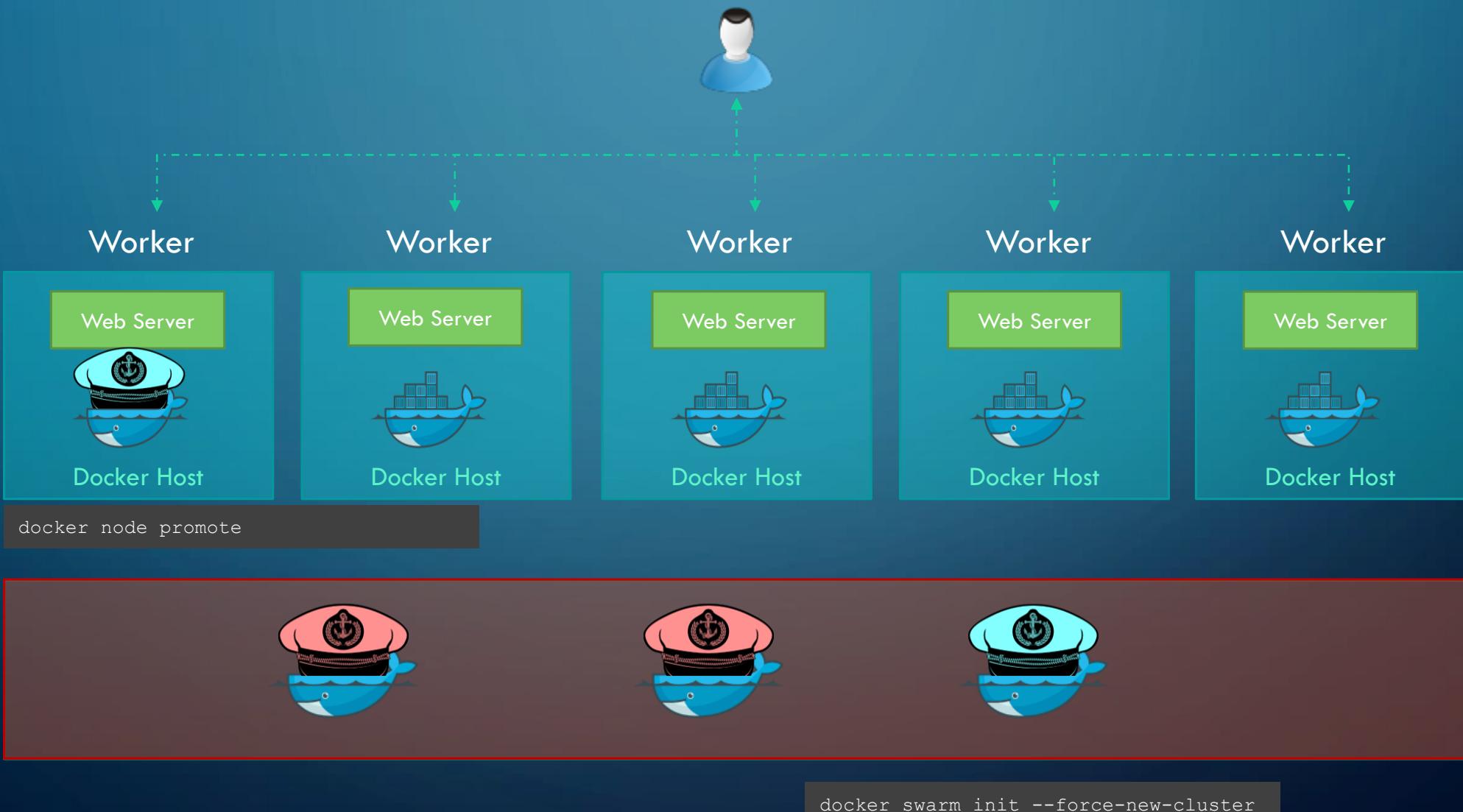
Fault Tolerance of N =  $\frac{N - 1}{2}$

# 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



# WHAT HAPPENS WHEN IT FAILS?



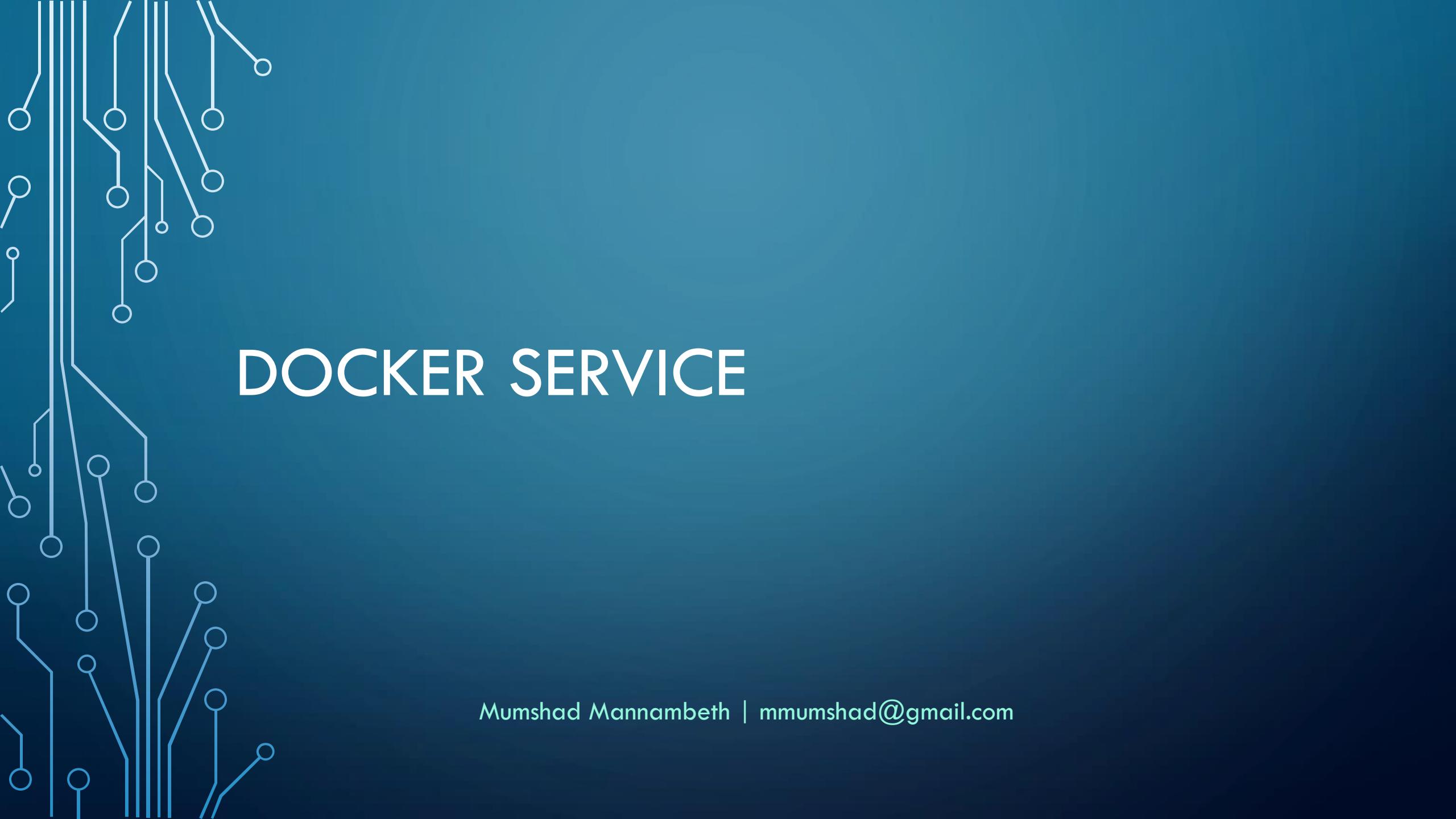
# CAN MANAGER WORK?



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



# QUIZ



# DOCKER SERVICE

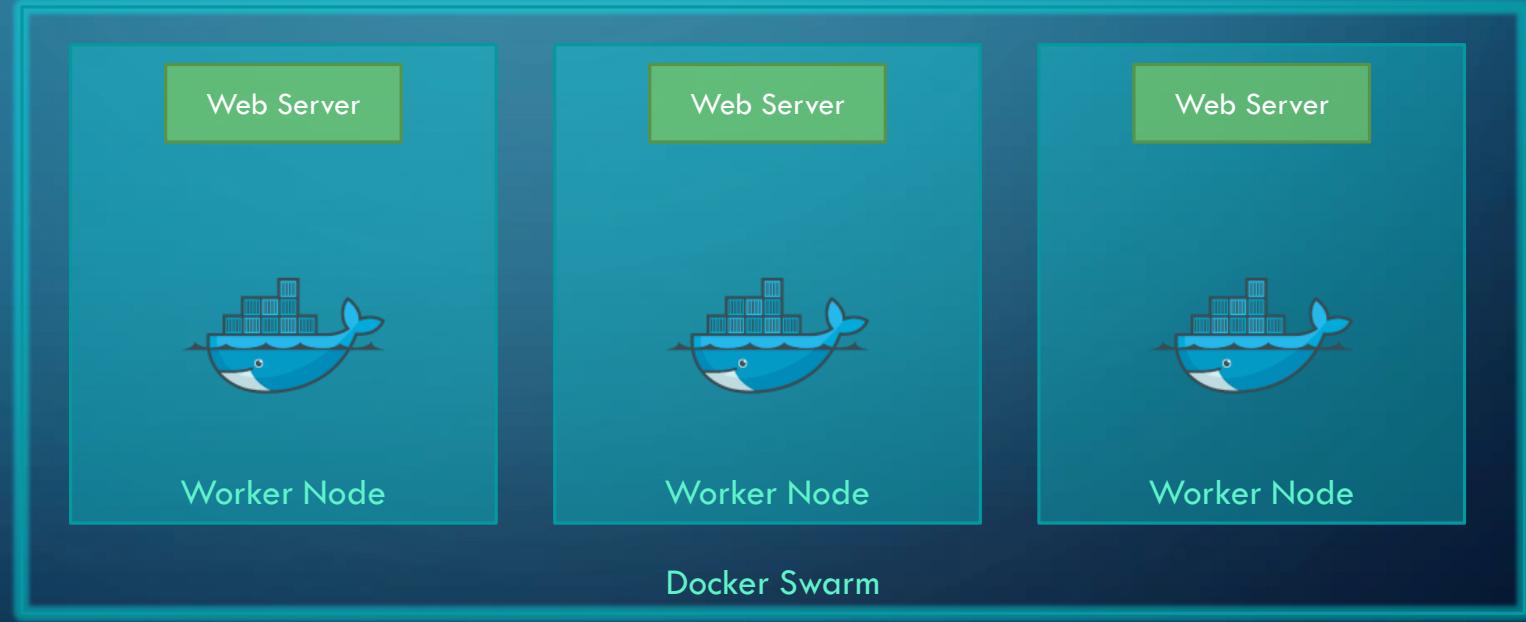
Mumshad Mannambeth | mmumshad@gmail.com

# DOCKER SERVICE

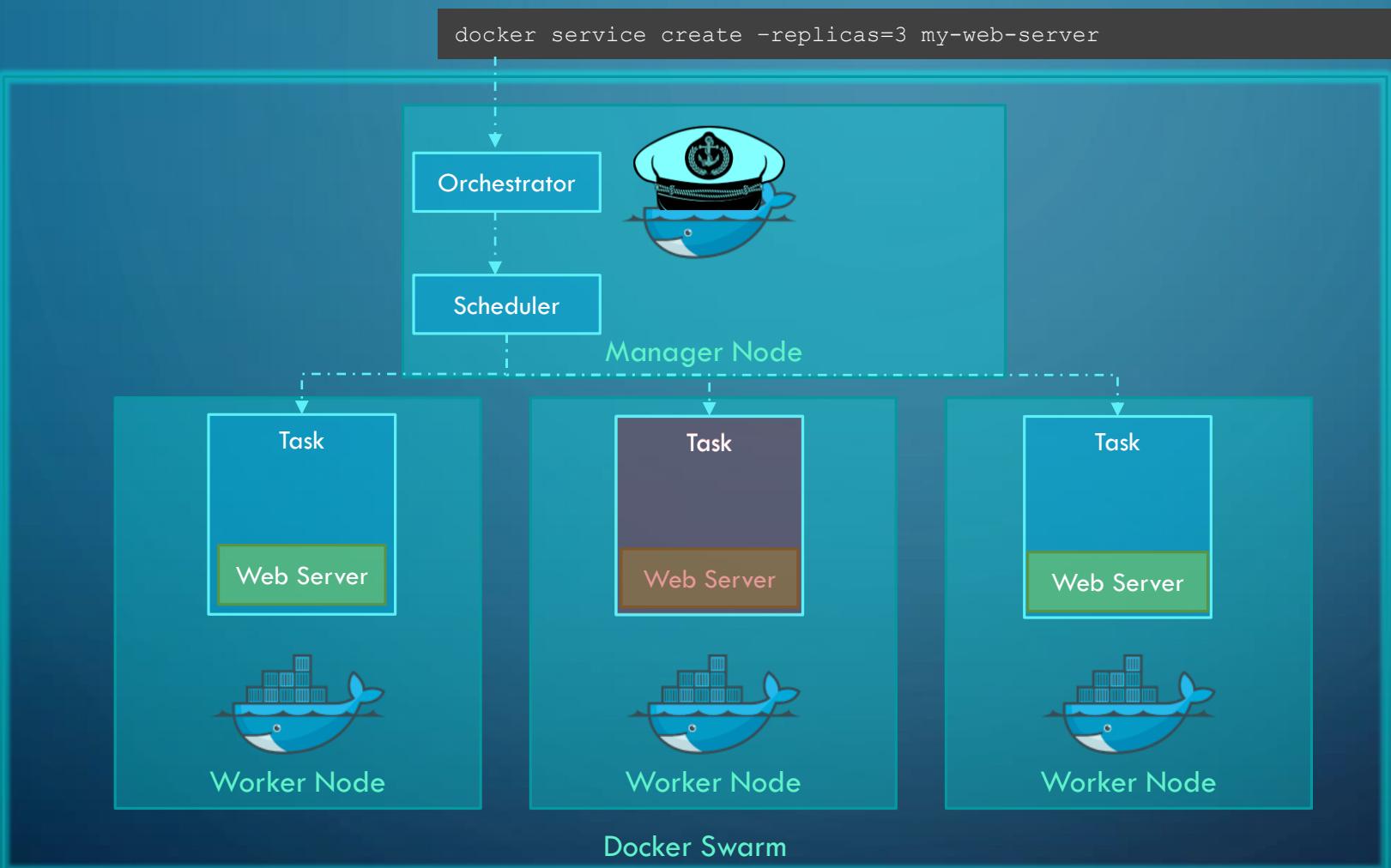
```
docker run my-web-server
```



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



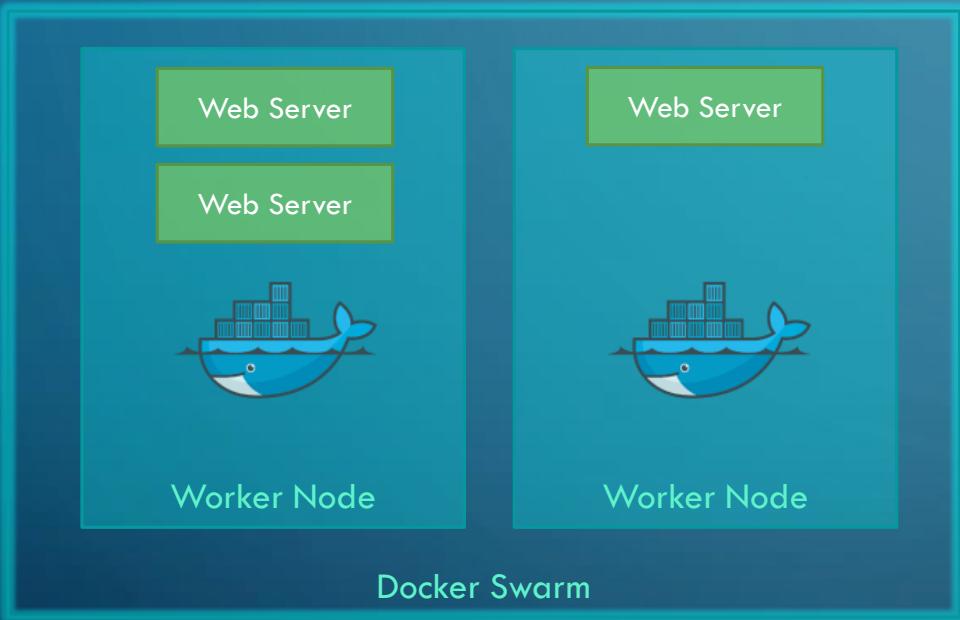
# TASKS



# REPLICAS



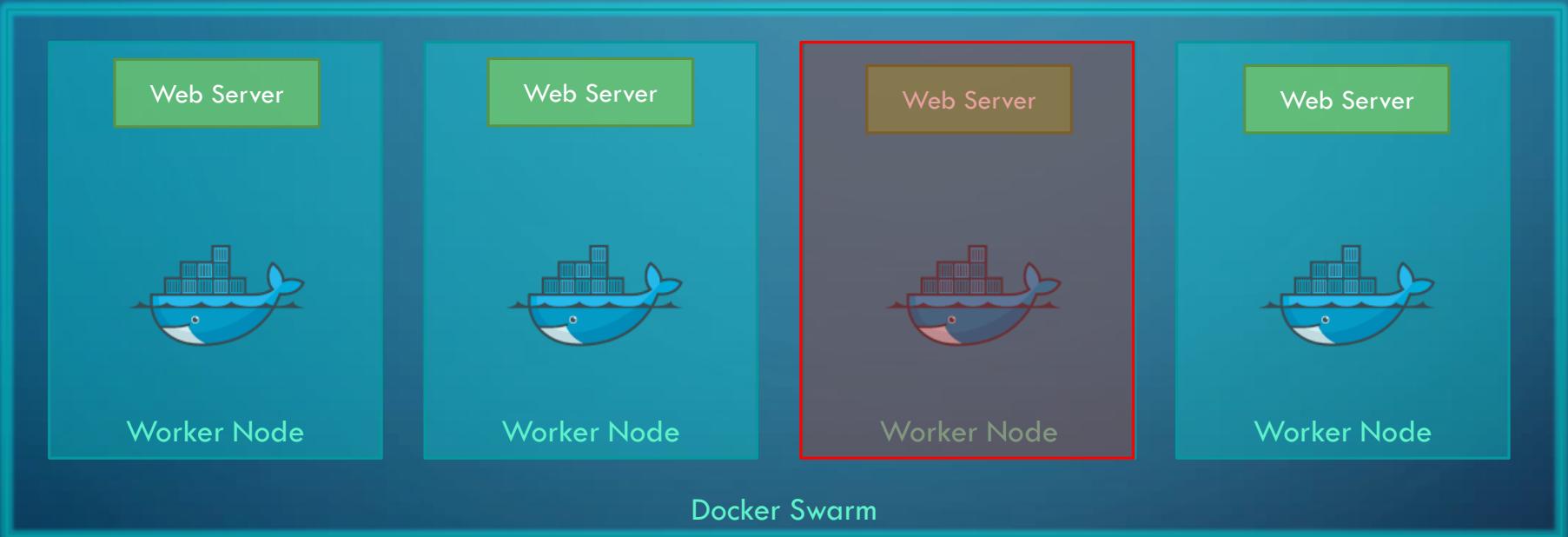
```
docker service create --replicas=3 my-web-server
```



# REPLICAS

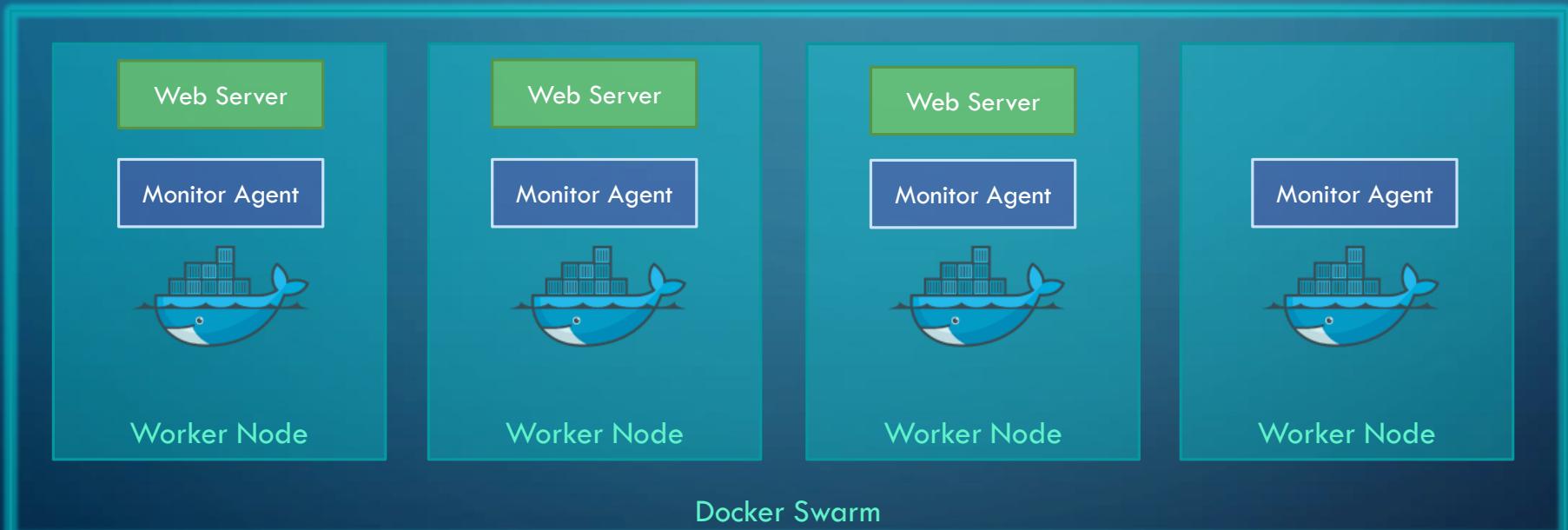


```
docker service create --replicas=3 my-web-server
```



# 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 myname webserver
```



# SERVICE UPDATE



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

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

web-server.1



Worker Node

web-server.2



Worker Node

web-server.3



Worker Node

web-server.4



Worker Node

Docker Swarm

# CODING EXERCISES

- Docker Service Commands



# QUIZ



# DOCKER STACKS

Mumshad Mannambeth | mmumshad@gmail.com

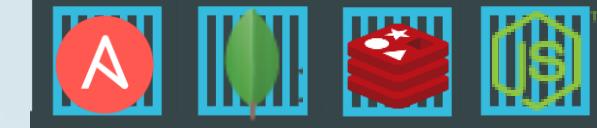
# DOCKER COMPOSE

```
docker run mmumshad/simple-webapp  
docker run mongodb  
docker run redis:alpine  
docker run ansible
```

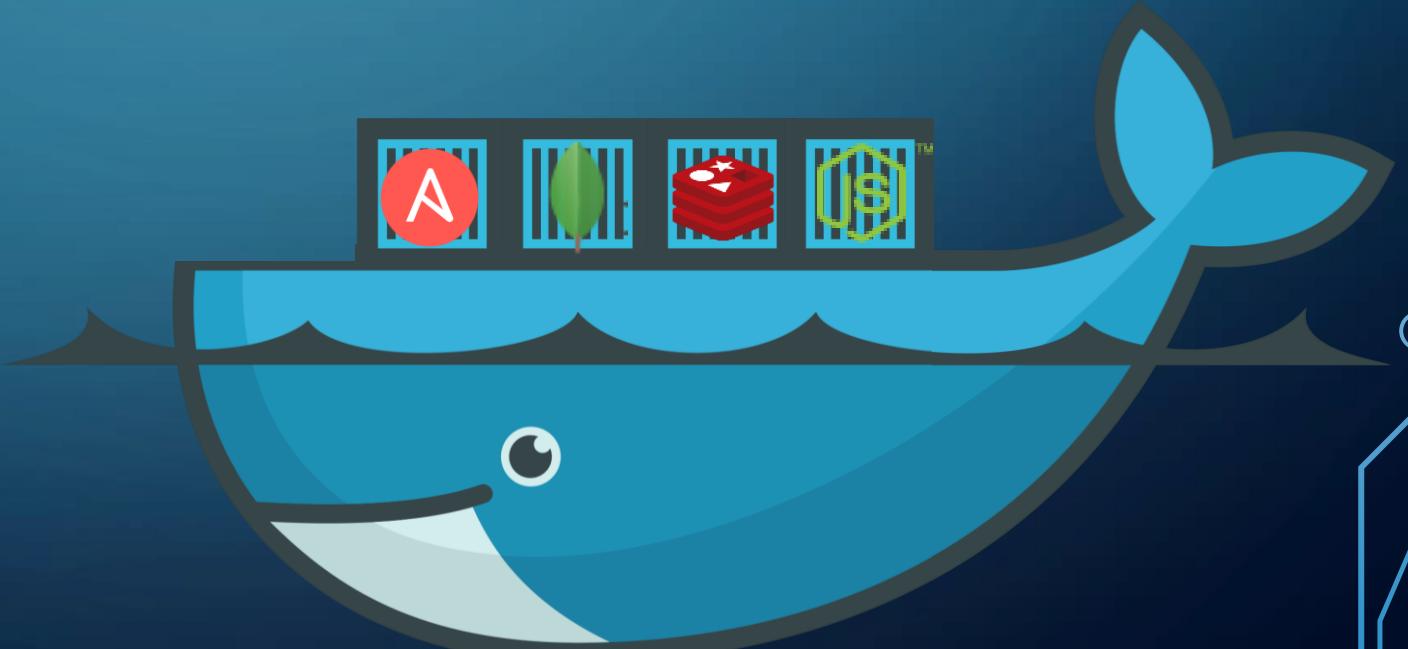
## docker-compose.yml

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

```
docker-compose up
```



Public Docker registry - dockerhub



# DOCKER COMPOSE

```
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
```

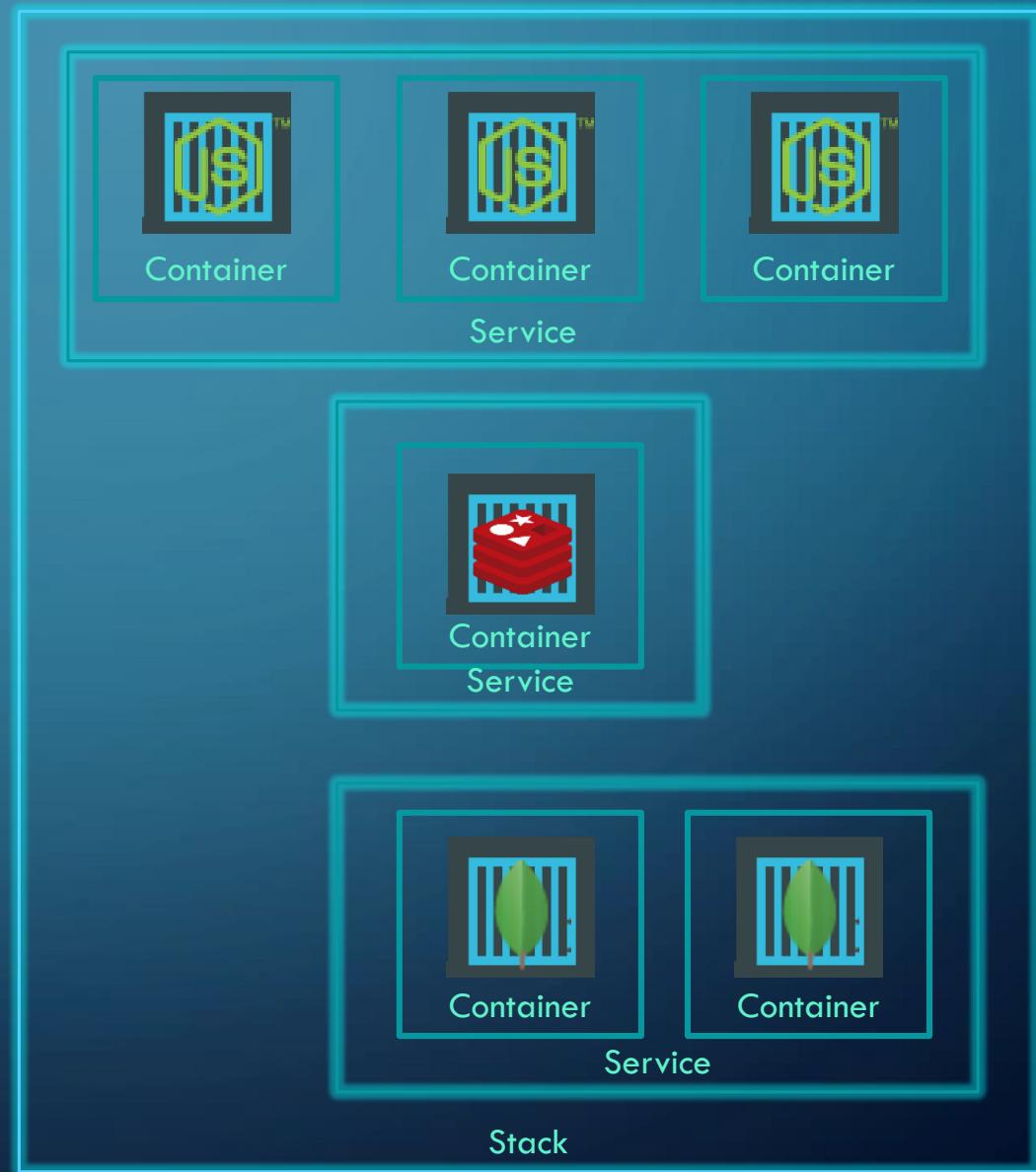
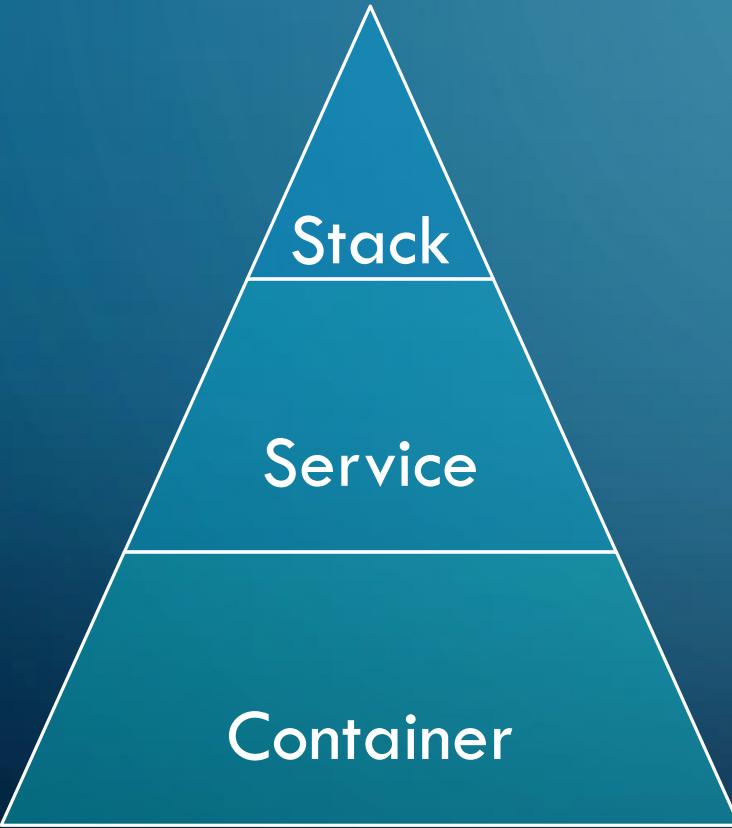
```
docker service create mmumshad/simple-webapp  
docker service create mongodb  
docker service create redis  
docker service create ansible
```

## docker-compose.yml

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

```
docker stack deploy
```

**STACK**



# STACK DEFINITION

```
docker service create mmumshad/simple-webapp  
docker service create mongodb  
docker service create redis  
docker service create ansible
```

## docker-compose.yml

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

```
docker stack deploy
```



# DOCKER NETWORKING

Mumshad Mannambeth | mmumshad@gmail.com

# DEFAULT NETWORKS



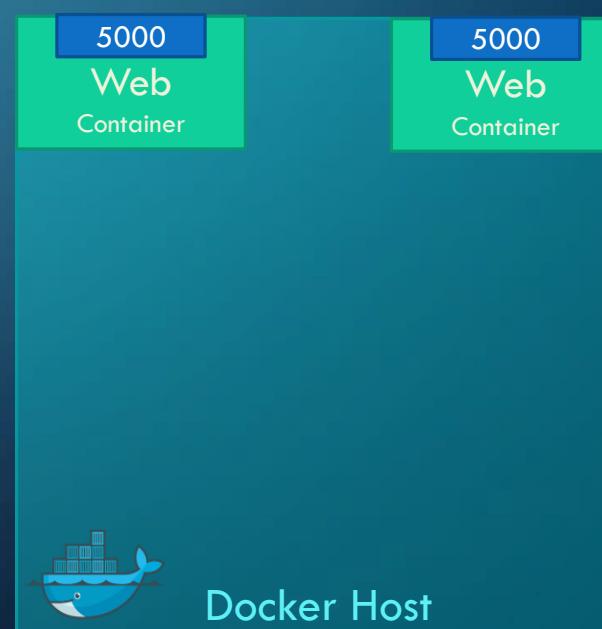
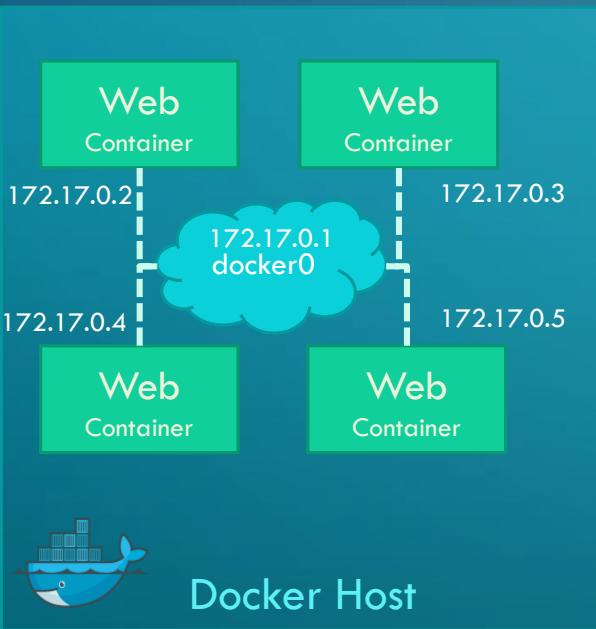
```
docker run ubuntu
```



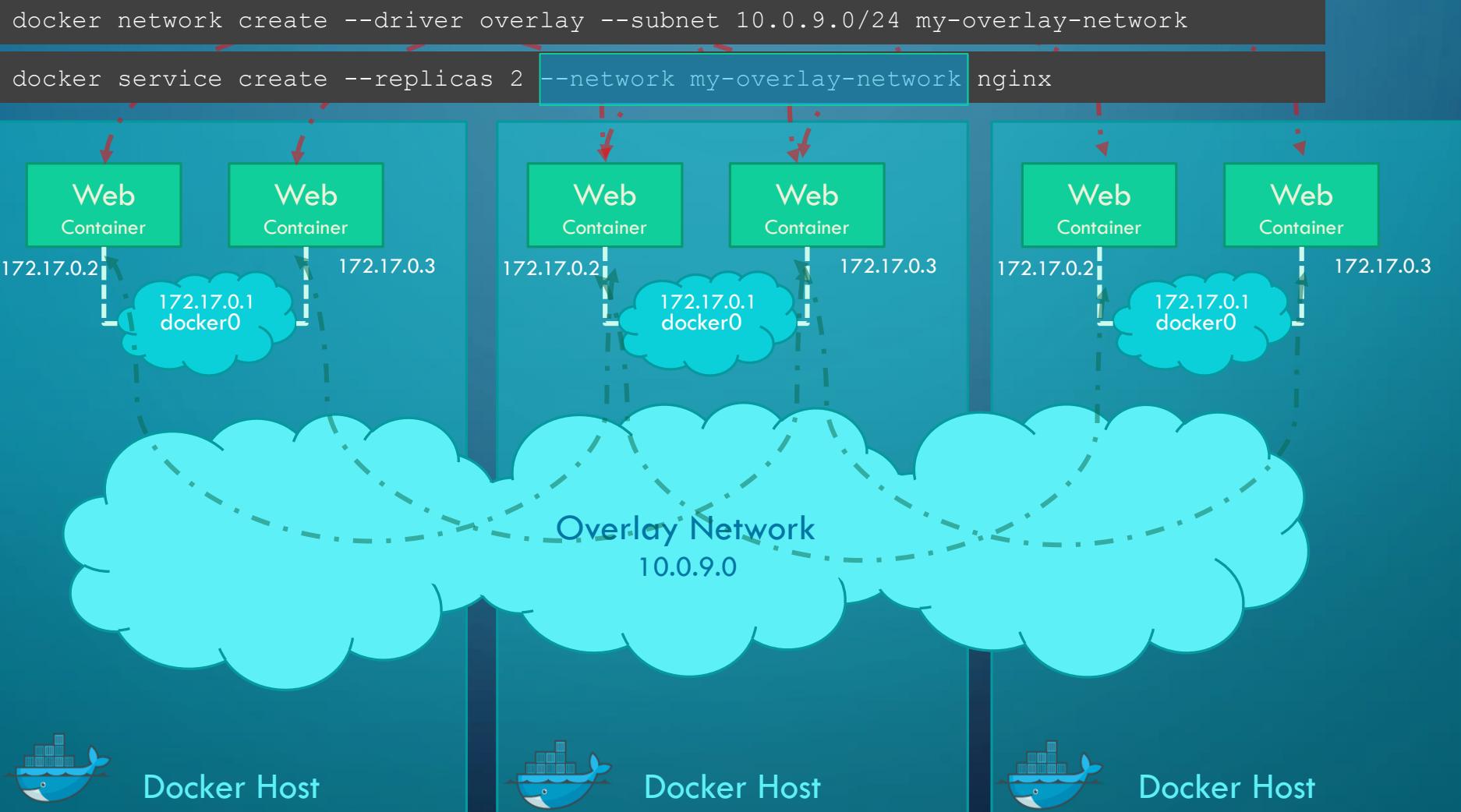
```
docker run Ubuntu --network=none
```



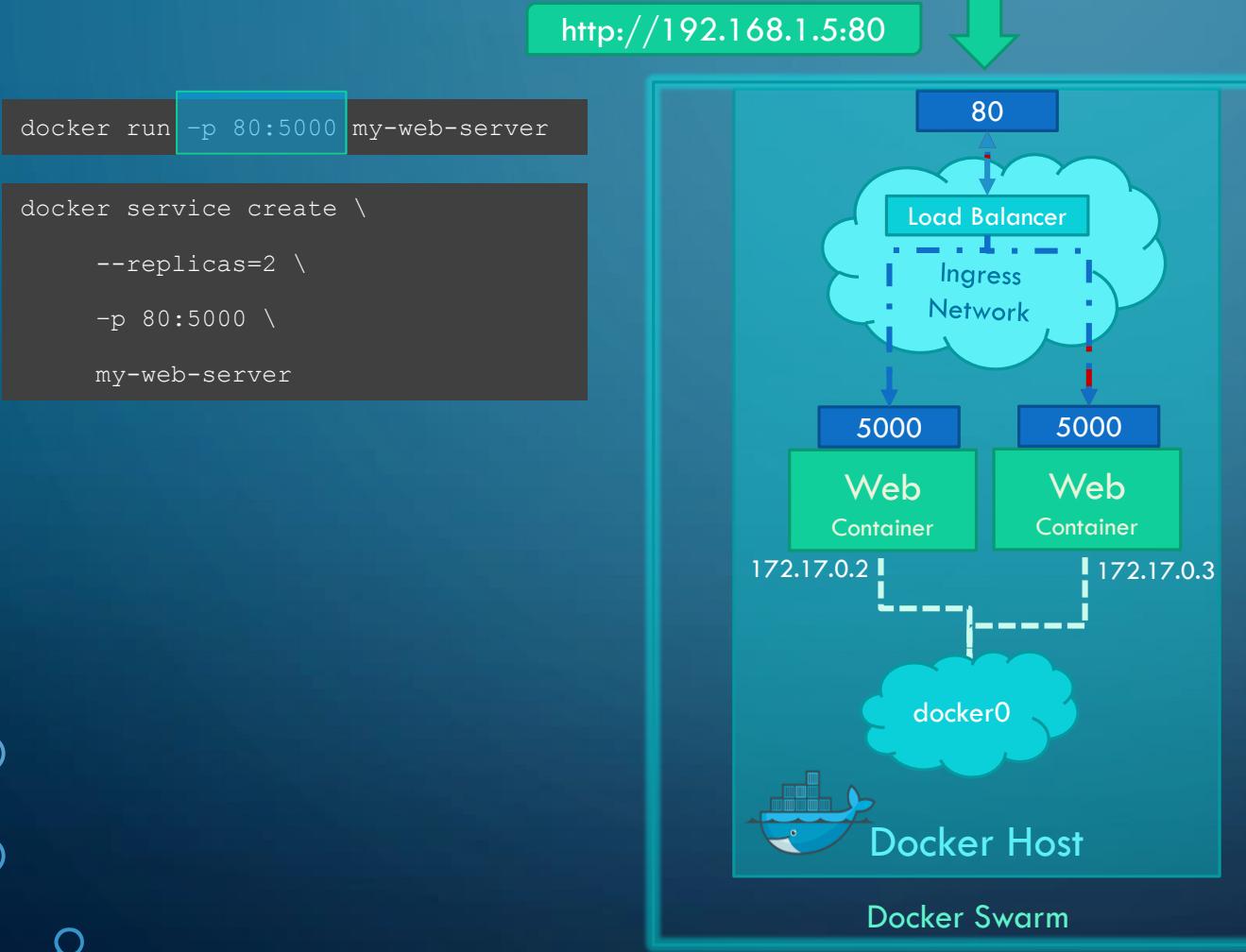
```
docker run Ubuntu --network=host
```



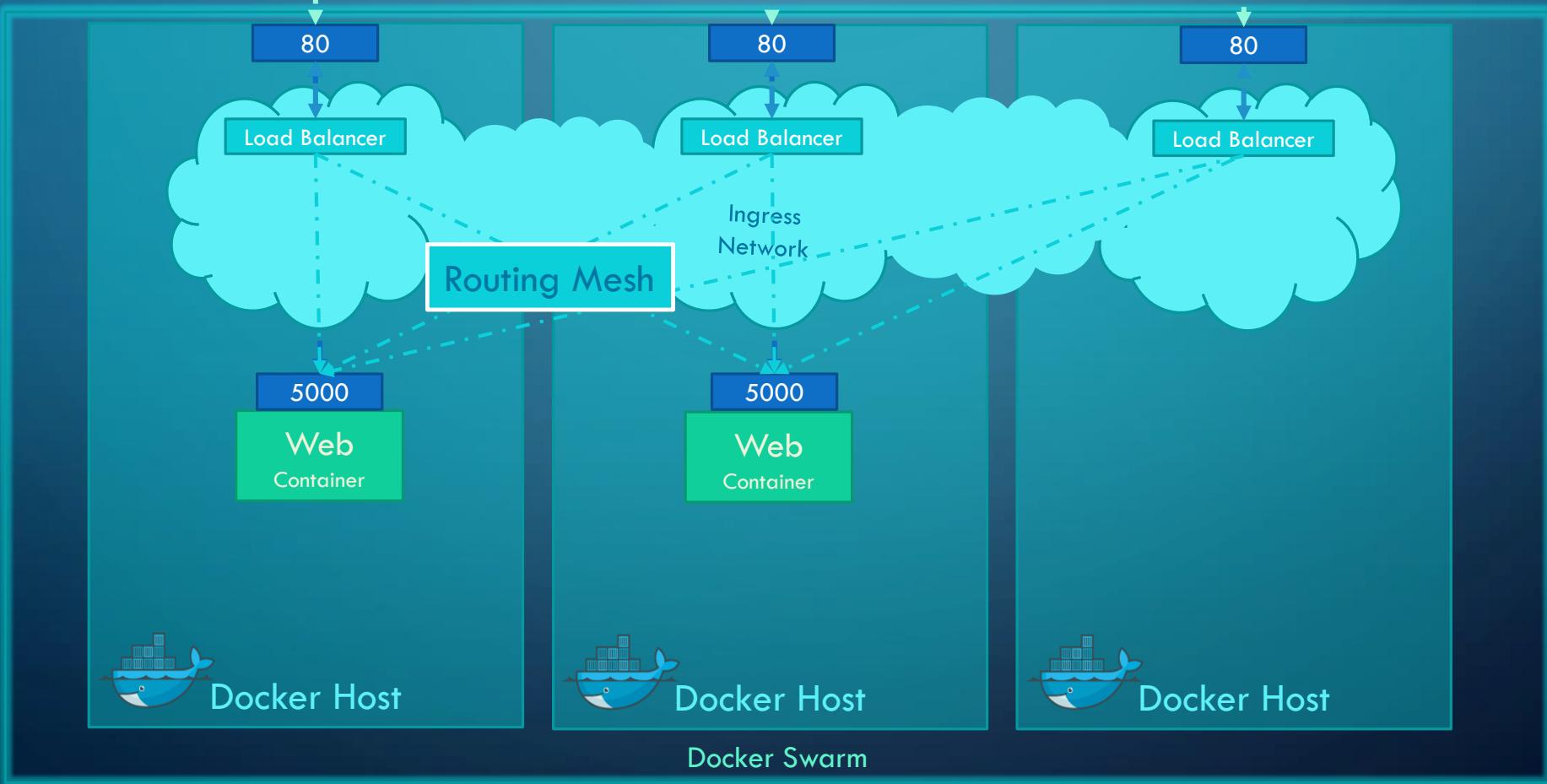
# OVERLAY NETWORK



# INGRESS NETWORK

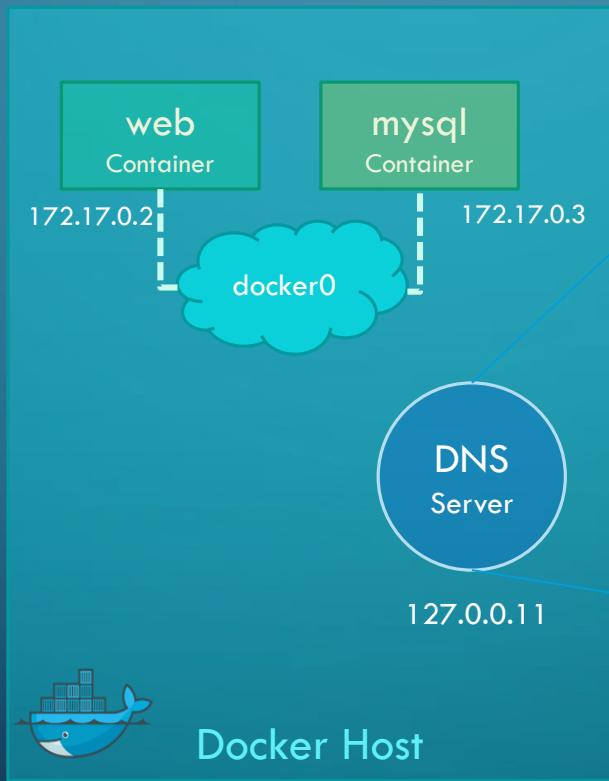


# INGRESS NETWORK

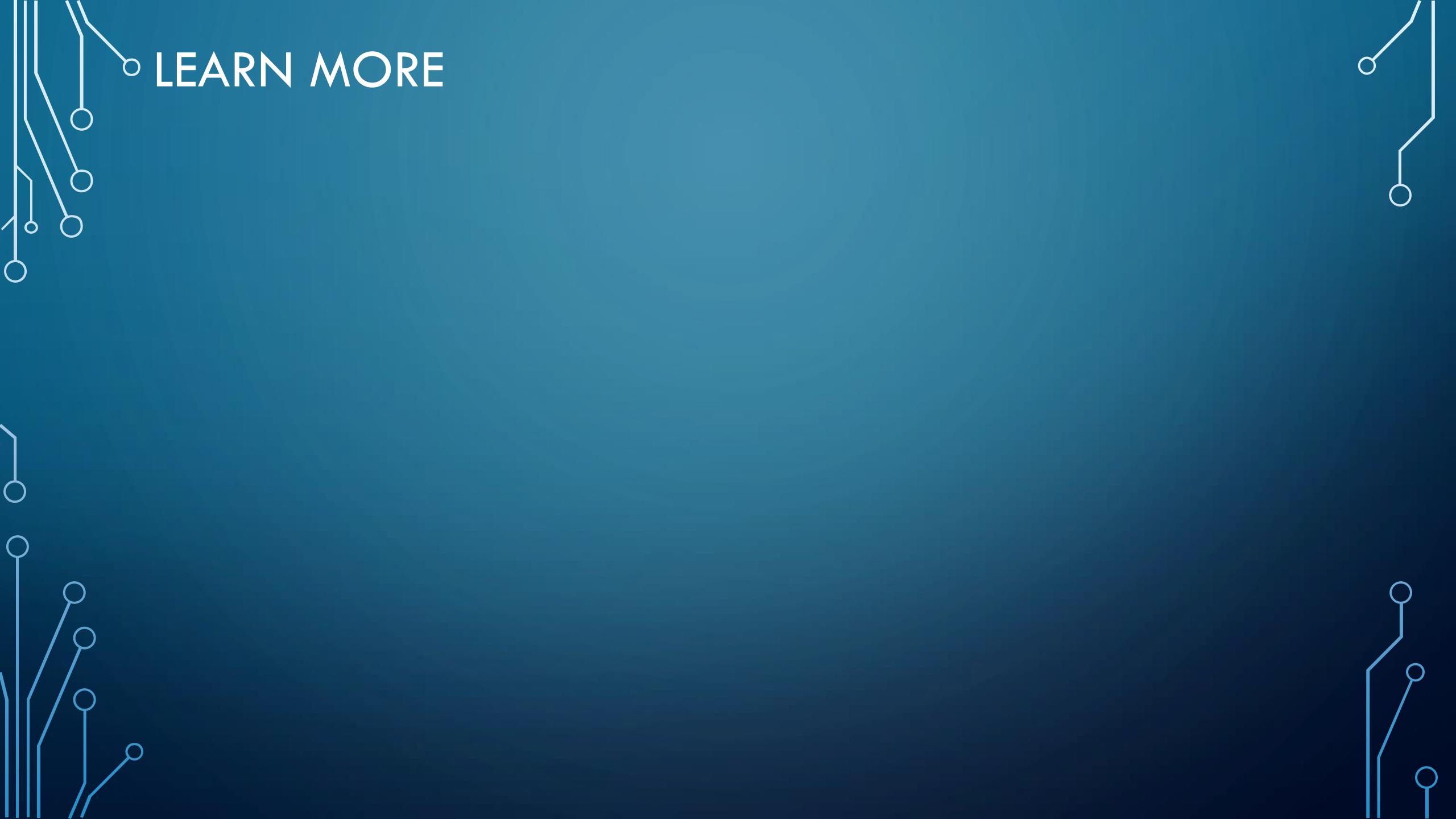


# EMBEDDED DNS

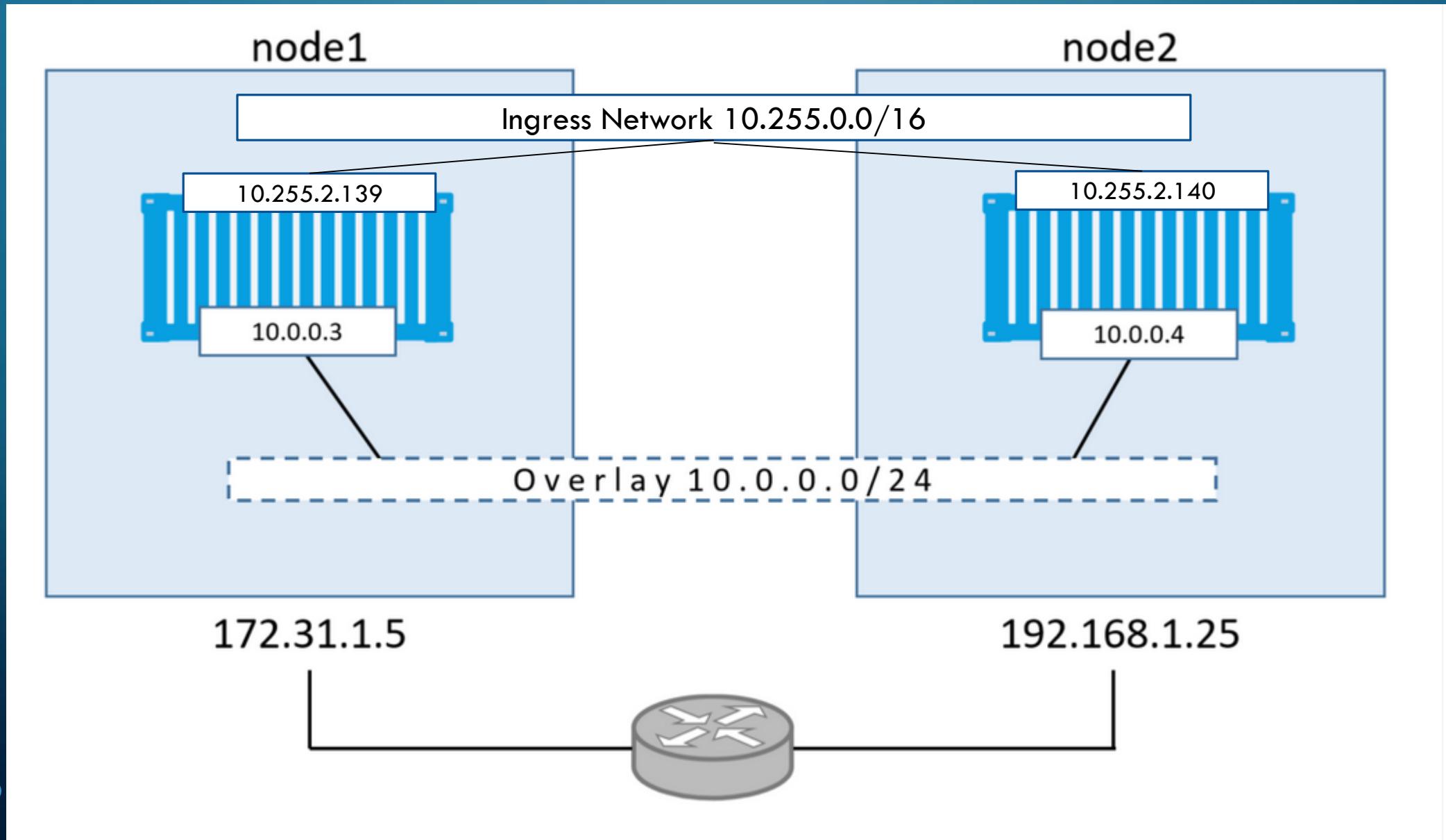
```
mysql.connect( mysql )
```

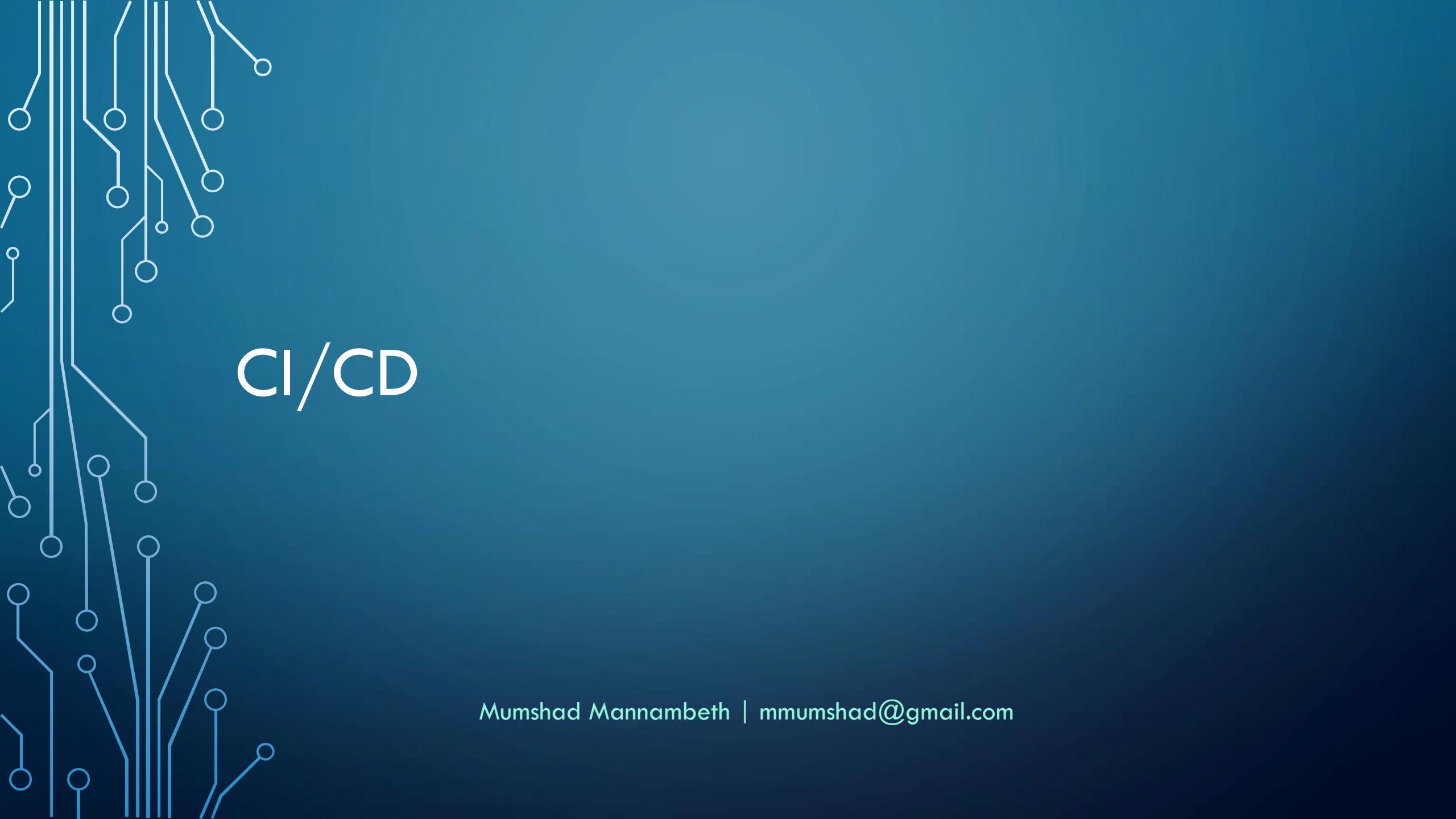


Host	IP
web	172.17.0.2
mysql	172.17.0.3



LEARN MORE

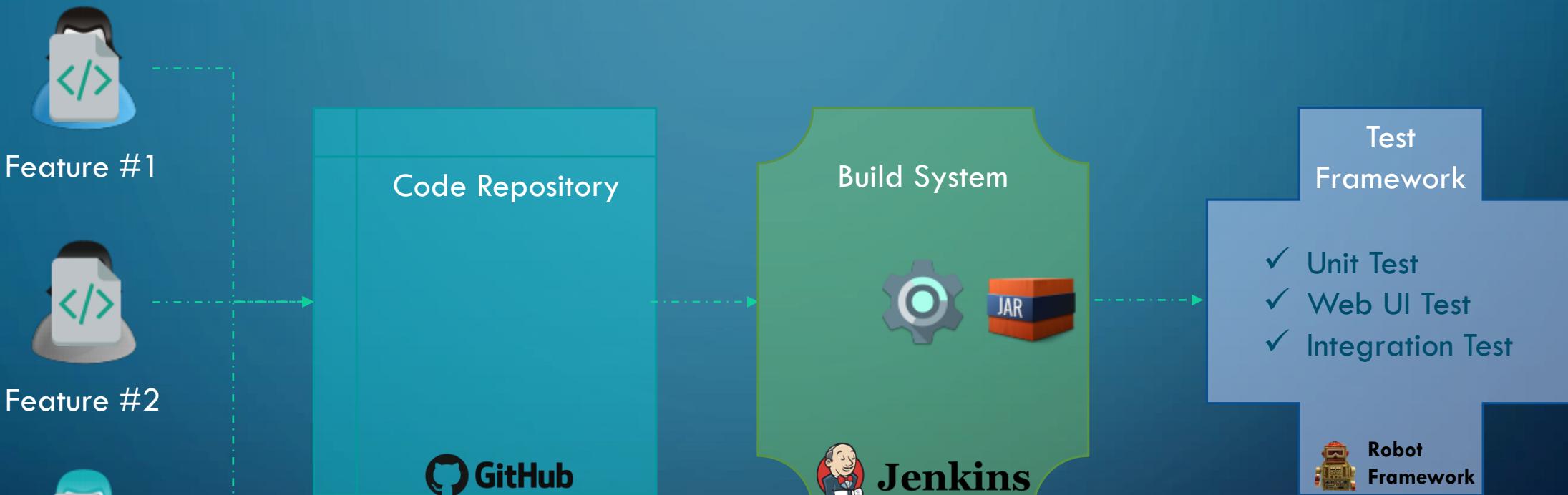




# CI/CD

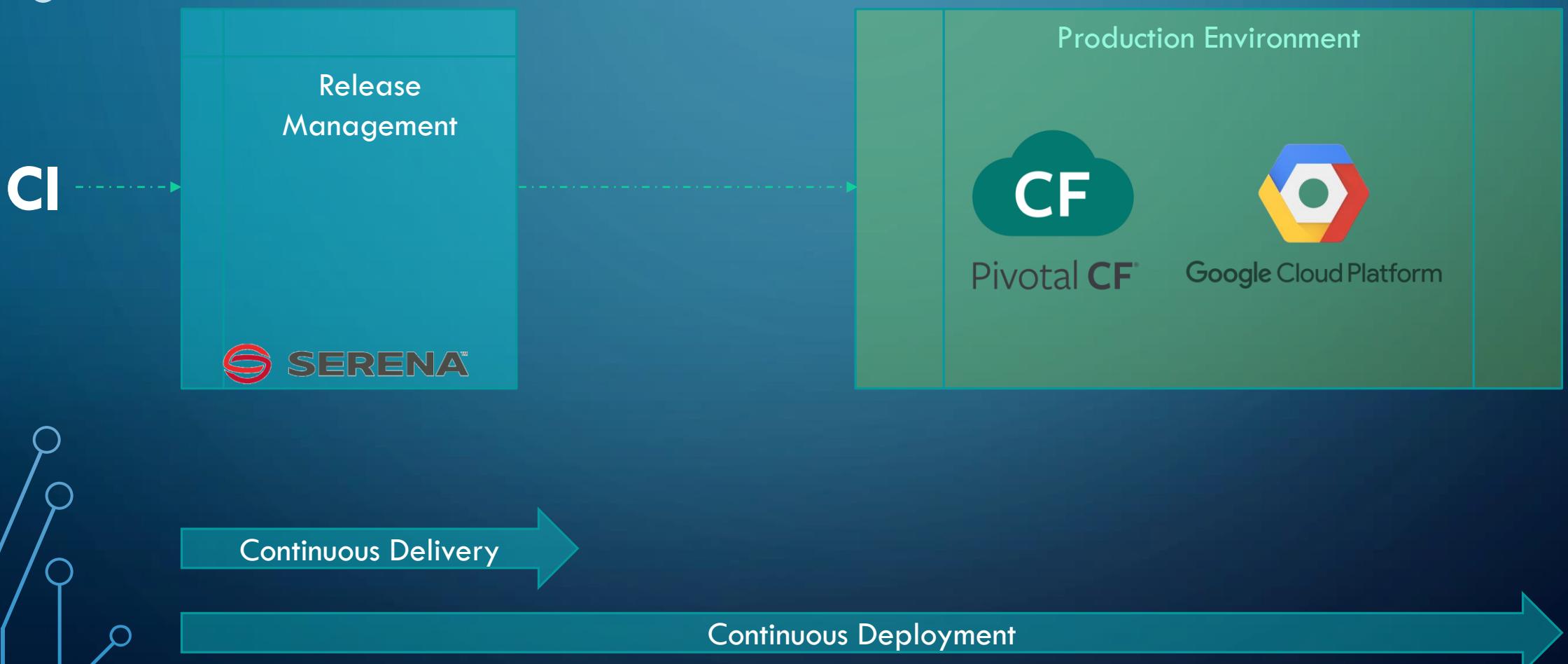
Mumshad Mannambeth | mmumshad@gmail.com

# CI – CONTINUOUS INTEGRATION



Continuous Integration

# CD – CONTINUOUS DELIVERY/DEPLOYMENT

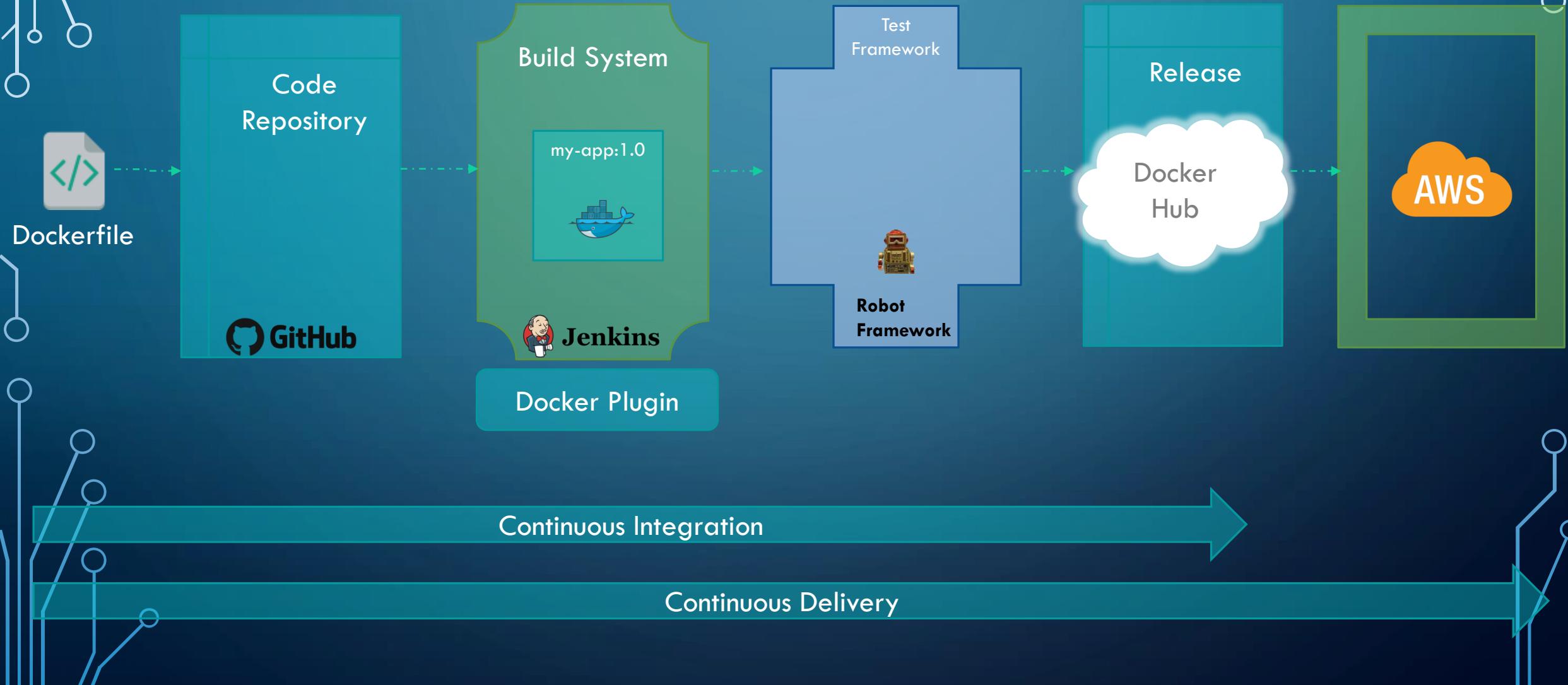




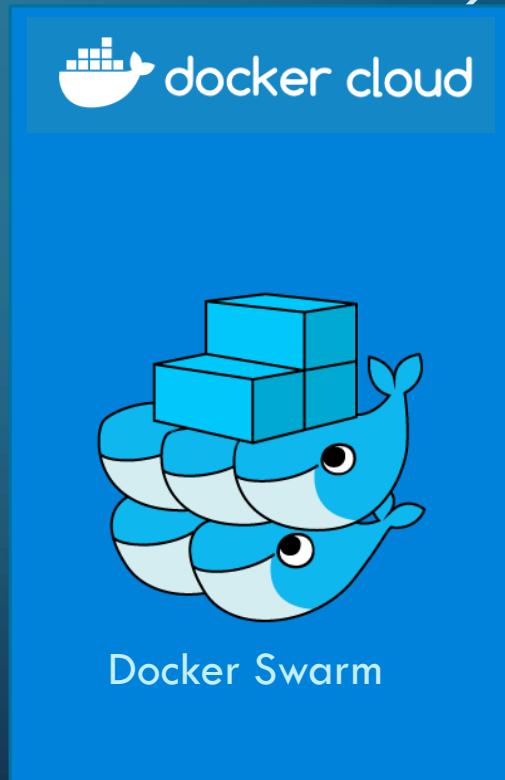
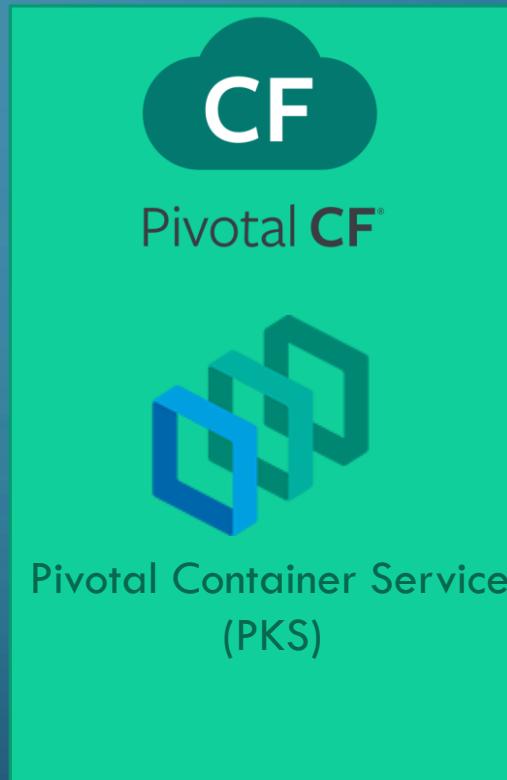
# CI/CD - DOCKER

Mumshad Mannambeth | mmumshad@gmail.com

# BUILD SYSTEMS – DOCKER SUPPORT

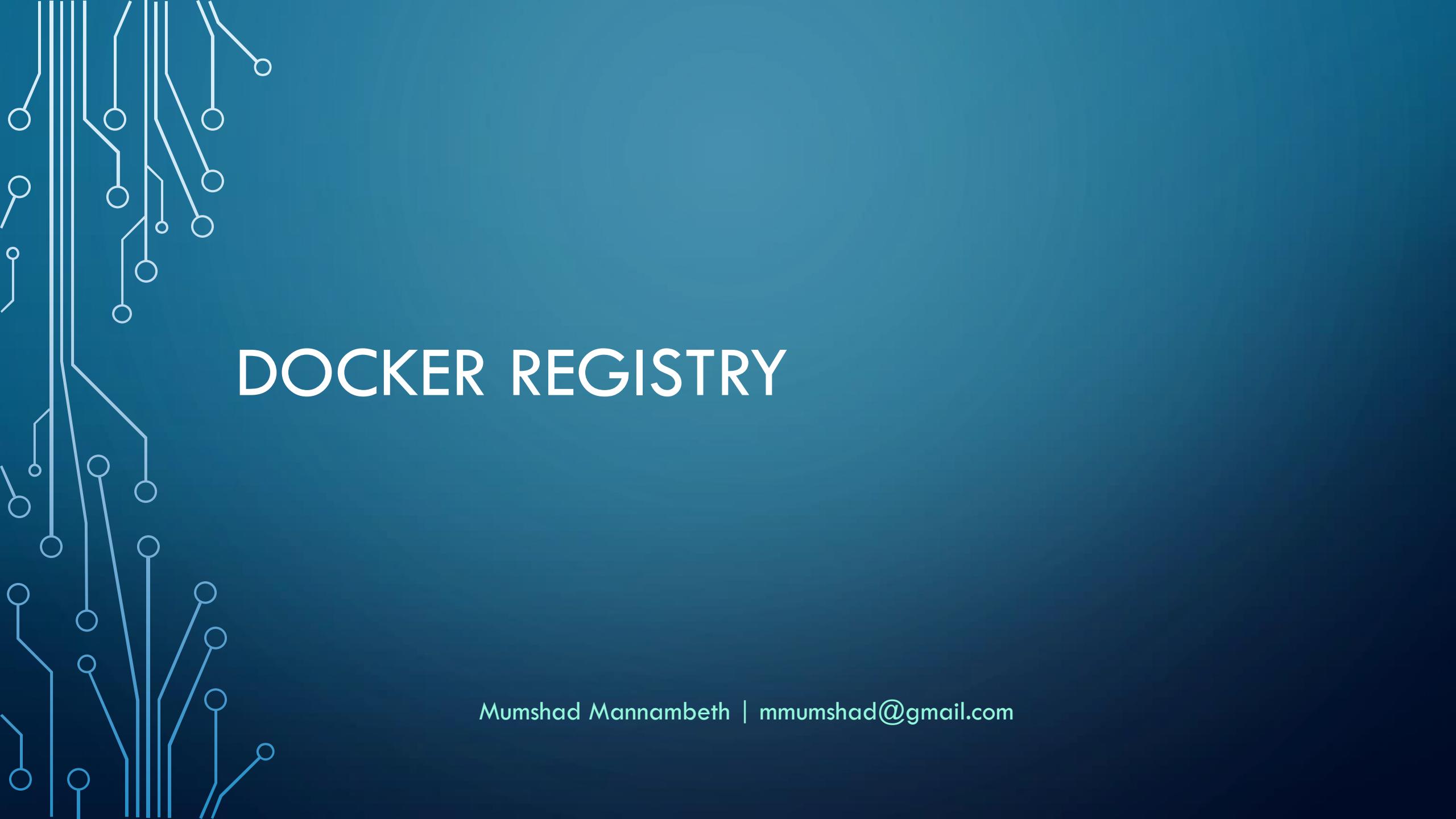


# PUBLIC CLOUD – DOCKER SUPPORT





DEMO



# DOCKER REGISTRY

Mumshad Mannambeth | mmumshad@gmail.com

# DOCKER REGISTRY

Private Docker Registry

Public Docker registry - dockerhub



```
docker push 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
```



DEMO

# CONCLUSION