Docker Advanced Concepts

Shailendra Chauhan

Microsoft MVP, Technical Consultant & Corporate Trainer

Agenda

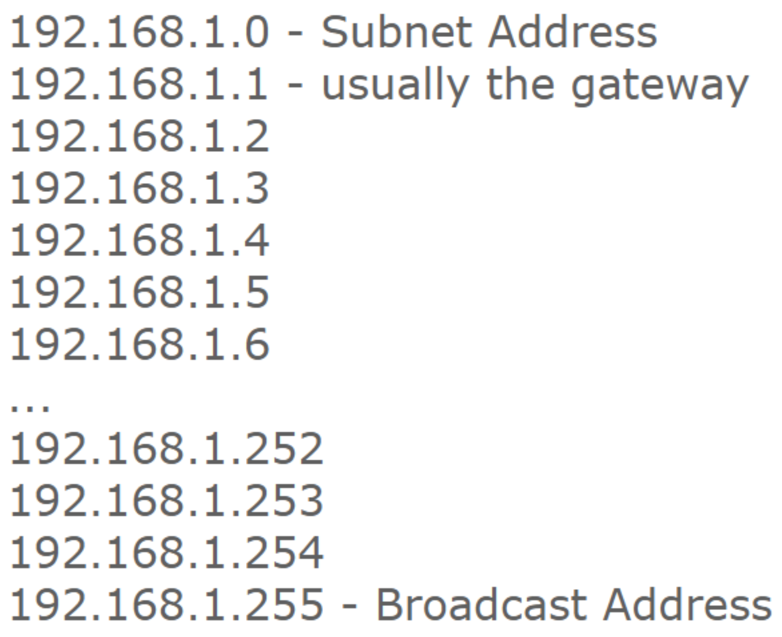
- Networking Basics
- IP Address
- Subnet
- Docker Networking
- Container Orchestration
- Docker Swarm
- Docker Service
- Docker Compose

IP Address

- IP address is a series of four numbers separated by periods. Generally it looks like 147.181.183.86.
- Each number will range from 0 to 255. So 255 would be the largest number in an ip address and 0 would be the smallest number.
- Each number is an eight bit number and looks like:
  - 10010011.10110101.10110111.01010110 = 147.181.183.86

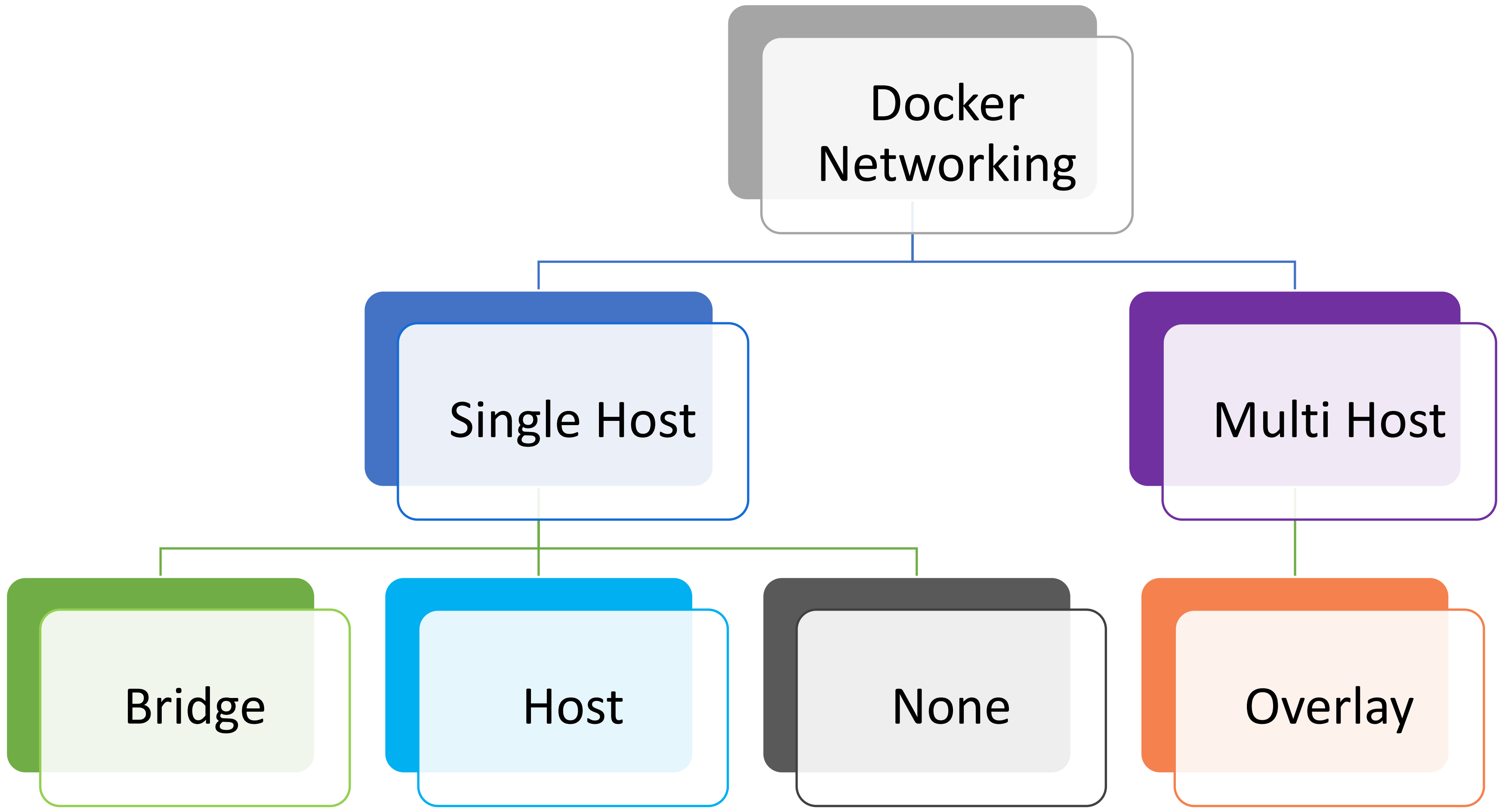
Subnet

- Used to group IP addresses and looks a lot like an ip address. A typical subnet mask is 255.255.255.0.
- These numbers range from 0 to 255.
- Each number is an eight bit number and looks like:
  - 11111111.11111111.11111111.00000000= 255.255.255.0
  - Every group of ip addresses, has a Subnet Address, Broadcast Address, and Gateway.
  - Both the Subnet and the Broadcast are used to send information to every ip address in the group. The Gateway acts sort of like the group's controller.



```
192.168.1.0 - Subnet Address
192.168.1.1 - usually the gateway
192.168.1.2
192.168.1.3
192.168.1.4
192.168.1.5
192.168.1.6
...
192.168.1.252
192.168.1.253
192.168.1.254
192.168.1.255 - Broadcast Address
```

Docker Networking

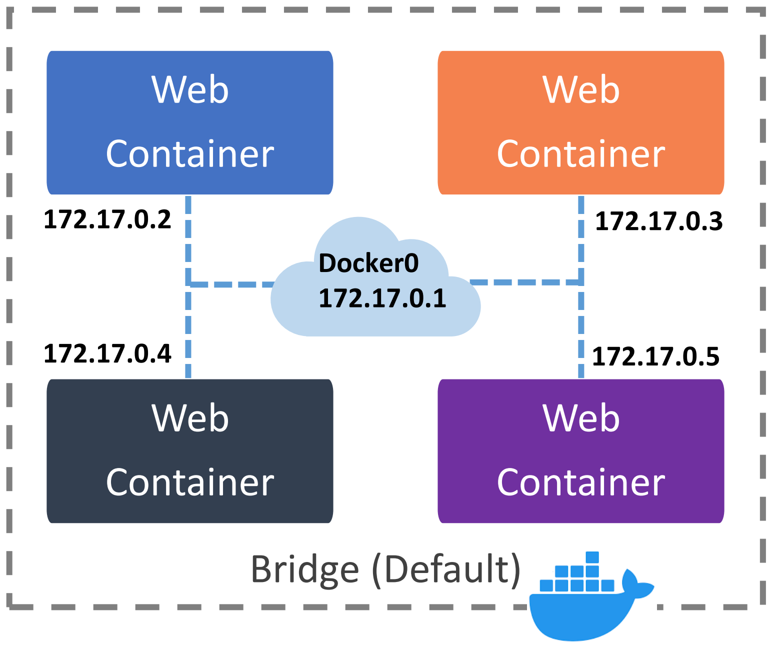


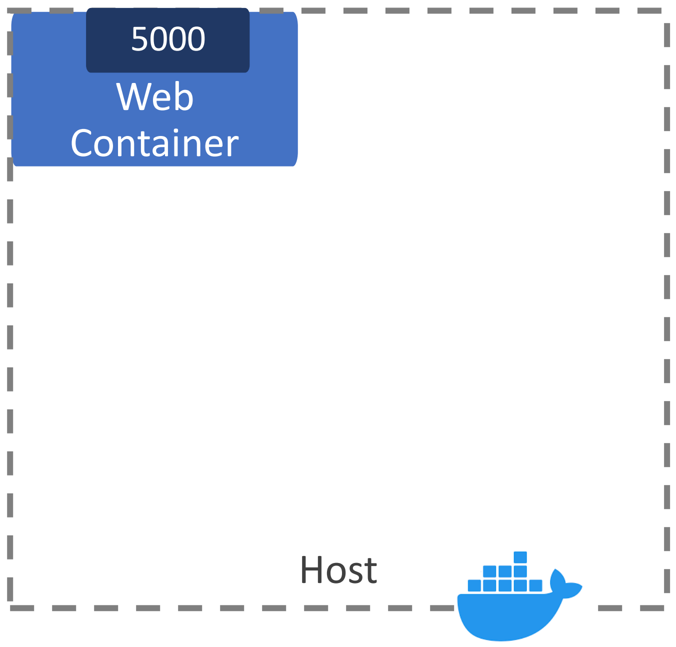
Docker Single Host Networking

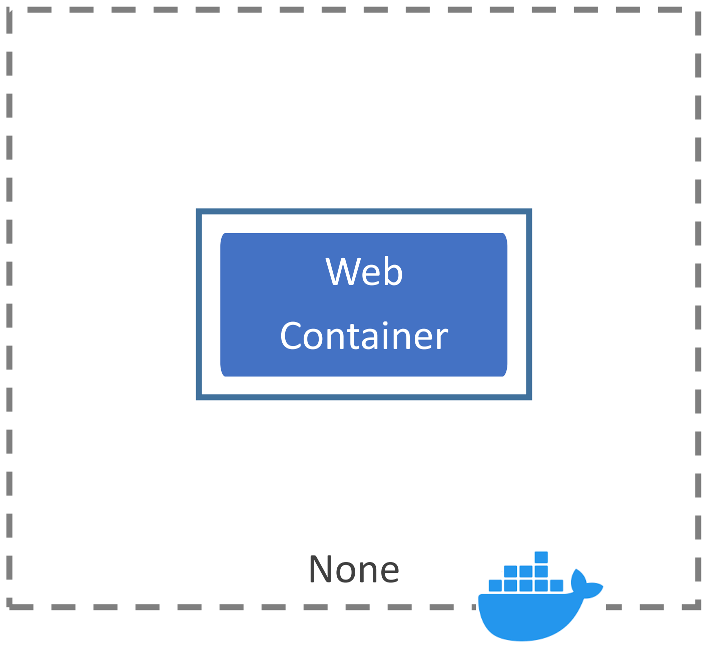
```
docker run myapp
```

```
docker run myapp --network host
```

```
docker run myapp --network none
```







DotNetTricks

Bridge

- Default driver
- The bridge is a private network restricted to a single docker host
- Each container is placed in its own network namespace
- The bridge driver creates a bridge(virtual switch) on a single Docker host.
- All containers on a bridge network can communicate with each others but containers on different bridge network cannot communicate with each other.
- Offers external access to containers through the port mapping

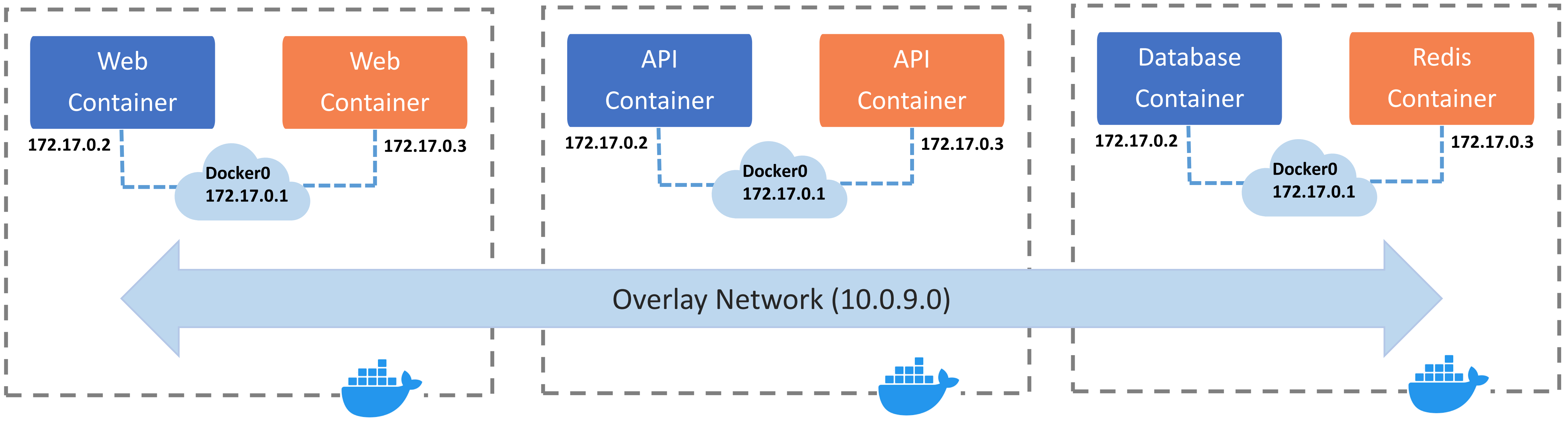
Host

- The Host Network Driver allows containers to use the host’s network stack directly.
- Removes the network isolation between the docker host and the docker containers to use the hosts networking directly.
- No two containers can use the same port(s).
- Used to setup, one or only a few containers on a single host.

None

- Completely disable the networking stack on a container.
- By using this mode, it will not configure any IP for the container and have no access to the external network as well as for other containers.

Docker Multi Host Networking



DotNetTricks

Overlay

- Manage communications among the Docker daemons participating in the swarm.
- The overlay driver enables simple and secure multi-host networking.
- All containers on the overlay network can communicate with each other.

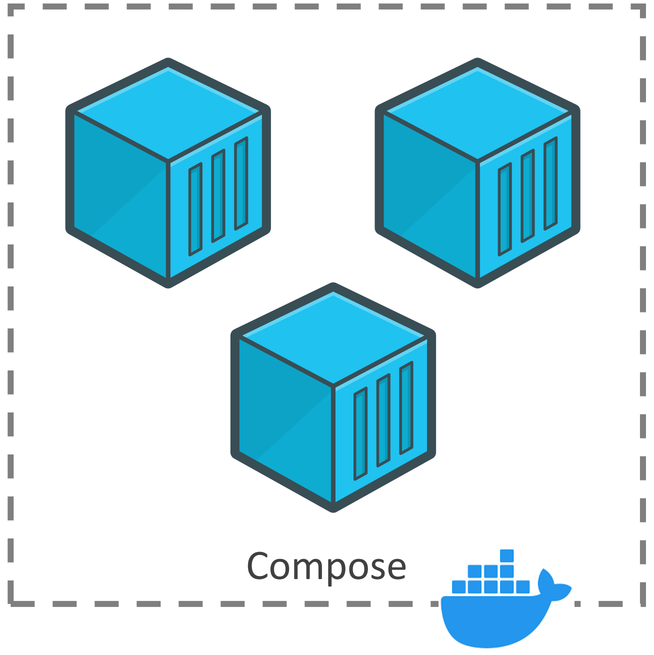
Docker Network Commands

```
> docker network create --driver bridge mybridge
> docker network inspect mybridge

> docker run --network mybridge --name=myapp  aspnetcrud
> docker network connect mybridge myapp
> docker network disconnect mybridge myapp

> docker network rm mybridge
> docker inspect -f "{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}" <container_id>
```

DotNetTricks

Docker Compose

- Compose is a tool for defining and running multi-container applications with Docker.
- With Compose, a multi-container application is defined using a single file and then spin your application up using a single command.
- All of that can be done by Docker Compose in the scope of a single host.
- The Docker Compose is useful for setting up development and testing workflows.



DotNetTricks

```yaml
version: '3.4'
networks:
 docker_app:
   driver: bridge
   ipam:
      driver: default
      config:
        - subnet: 172.16.238.0/24
services:
  db:
    image: "mcr.microsoft.com/mssql/server:2019-CU4-ubuntu-16.04"
    environment:
       ACCEPT_EULA: 'Y'
       SA_PASSWORD: 'YourStrong@Passw0rd'
    container_name: 'sql_db'
    networks:
       docker_app:
         ipv4_address: 172.16.238.2
    ports:
      - "5020:1433"
  aspnetdockercrud:
    image: ${DOCKER_REGISTRY-}aspnetdockercrud
    build:
      context: .
      dockerfile: ASPNetDockerCRUD/Dockerfile
    container_name: 'aspnet_app'
    networks:
       docker_app:
          ipv4_address: 172.16.238.3
    ports:
      - "5010:80"
```

```
docker-compose -f docker-compose.yml up
docker-compose -f docker-compose.yml down
```

```
docker-compose -f docker-compose.yml start
docker-compose -f docker-compose.yml stop
```

Container Orchestration

Docker Swarm

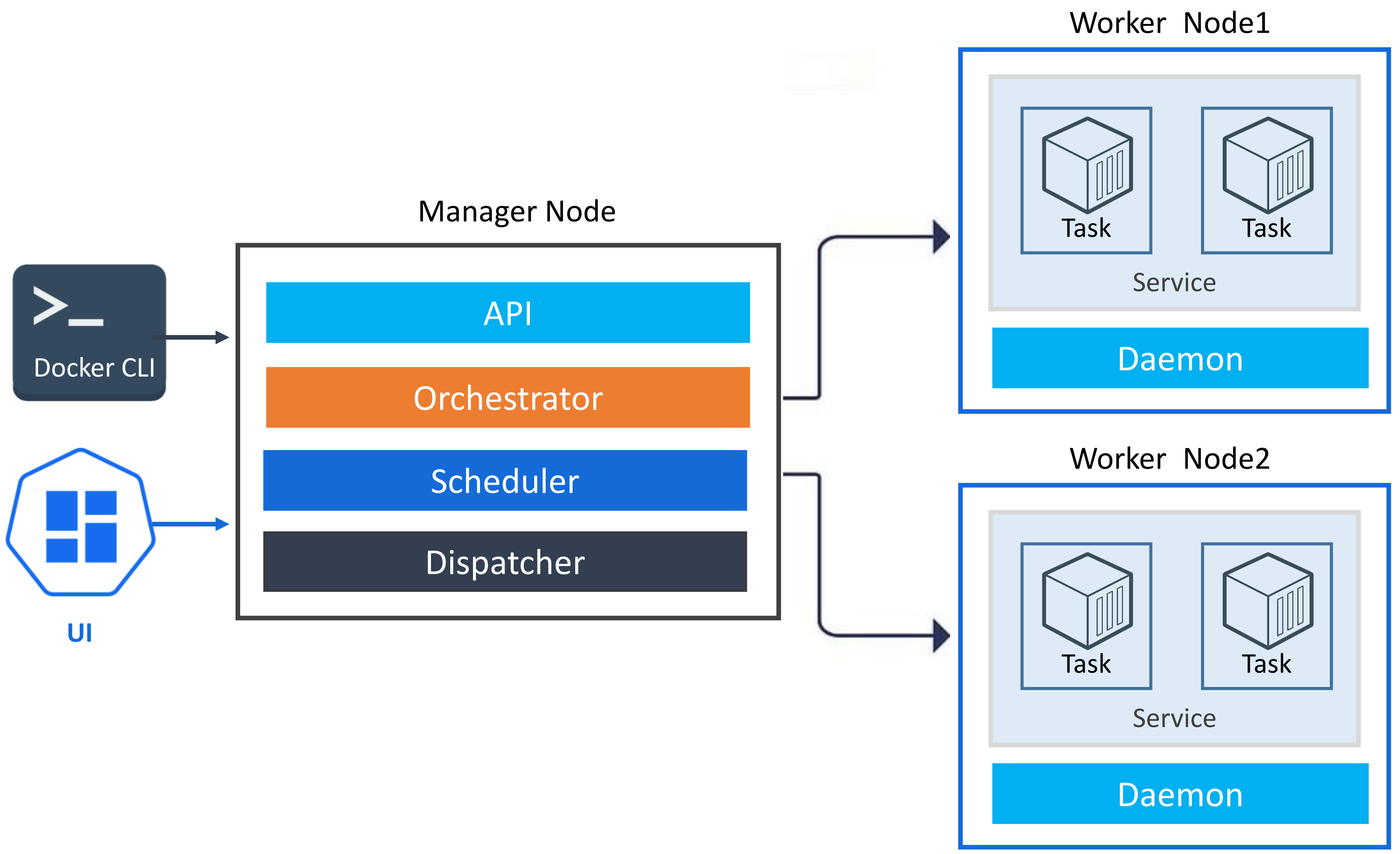
Kubernetes

MESOS

Docker Swarm

- Swarm is a cluster of Docker hosts that are called nodes.
- Docker Swarm is a native clustering tool for Docker that can turn a pool of Docker hosts into a single virtual host.
- The swarm cluster consists of a swarm manager and a set of workers.
- With swarm, you can deploy and scale your applications to multiple hosts.
- Swarm helps in containers management, scaling, service discovery, and load balancing between the nodes in the cluster.

Docker Swarm Architecture



DotNetTricks

Docker Swarm Architecture Contd..

- **Manager nodes** are used to perform control orchestration, cluster management and task distribution.
  - API - Accepts commands from CLI and create service object
  - Orchestrator - Reconciliation loop for service objects and create tasks
  - Scheduler - Assign Nodes to tasks
  - Dispatcher - Checks in on workers
- **Worker nodes** are used for running containers whose tasks are assigned by Manager nodes. Each node can be configured as a Manager node, Worker node, or as both.

Docker Swarm Architecture Contd..

- **Tasks** - A task is a slot in which a single container is running. Tasks are the part of a Swarm service.
- **Service** is one or more containers with the same configuration running under docker's swarm mode.

Docker Service

- A docker "service" is one or more containers with the same configuration running under docker's swarm mode.
- With docker service you manage a group of containers from the same image.
- You can scale them (start multiple containers).
- Docker service is useful for a microservices based application

```
docker service create --publish 8080:80 --name myapp myapp:v1
docker service rm myapp
docker service scale myapp=3
```

Docker Container vs. Docker Service

- "docker run" command create a single independent container which is not a part of any orchestration such as docker swarm.
- Not scalable to share the excessive load.
- Containers running through Docker run will be of different configuration, such as Network adapters etc.
- High availability is less as compared to the Cluster.
- Simple networking to communicate between multiple containers.
- Don't support auto recovery in case of failure.

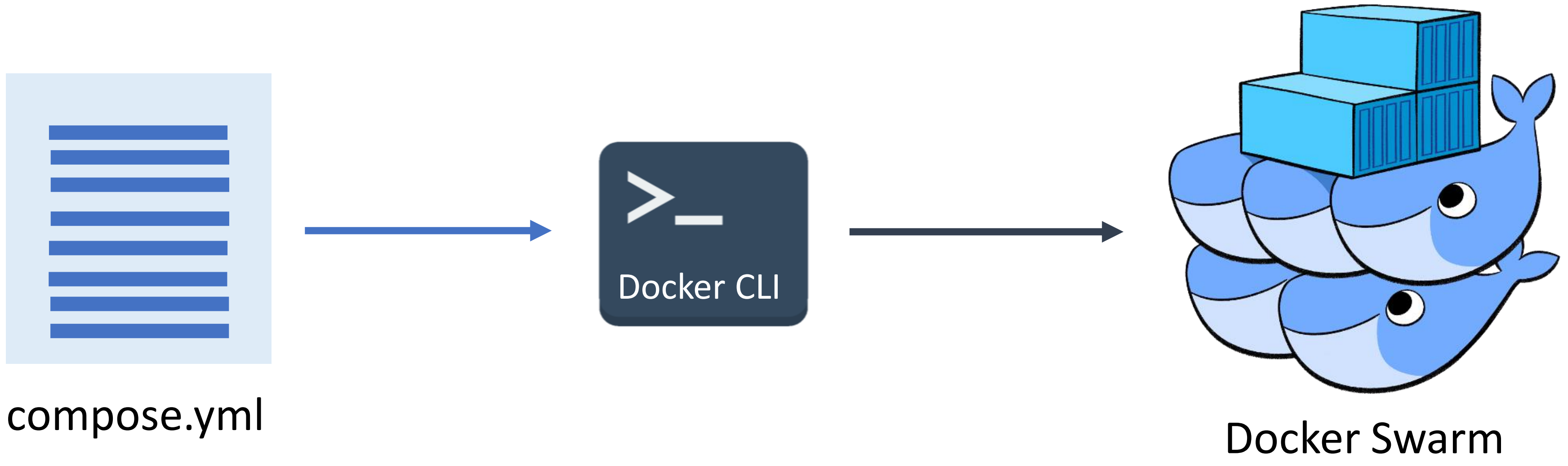
- "docker service create" command creates a service which is a part of a orchestration such as docker swarm or Kubernetes.
- Supports scaling & load balancer to manage workloads.
- Containers running through the Docker swarm will be of the same configuration.
- For service, there is a master node and another are worker nodes to supports high availability.
- Docker swarm uses mesh networking to communicate within the cluster.
- Docker Swarm can auto recover the tasks in case of failure.

Docker Stack

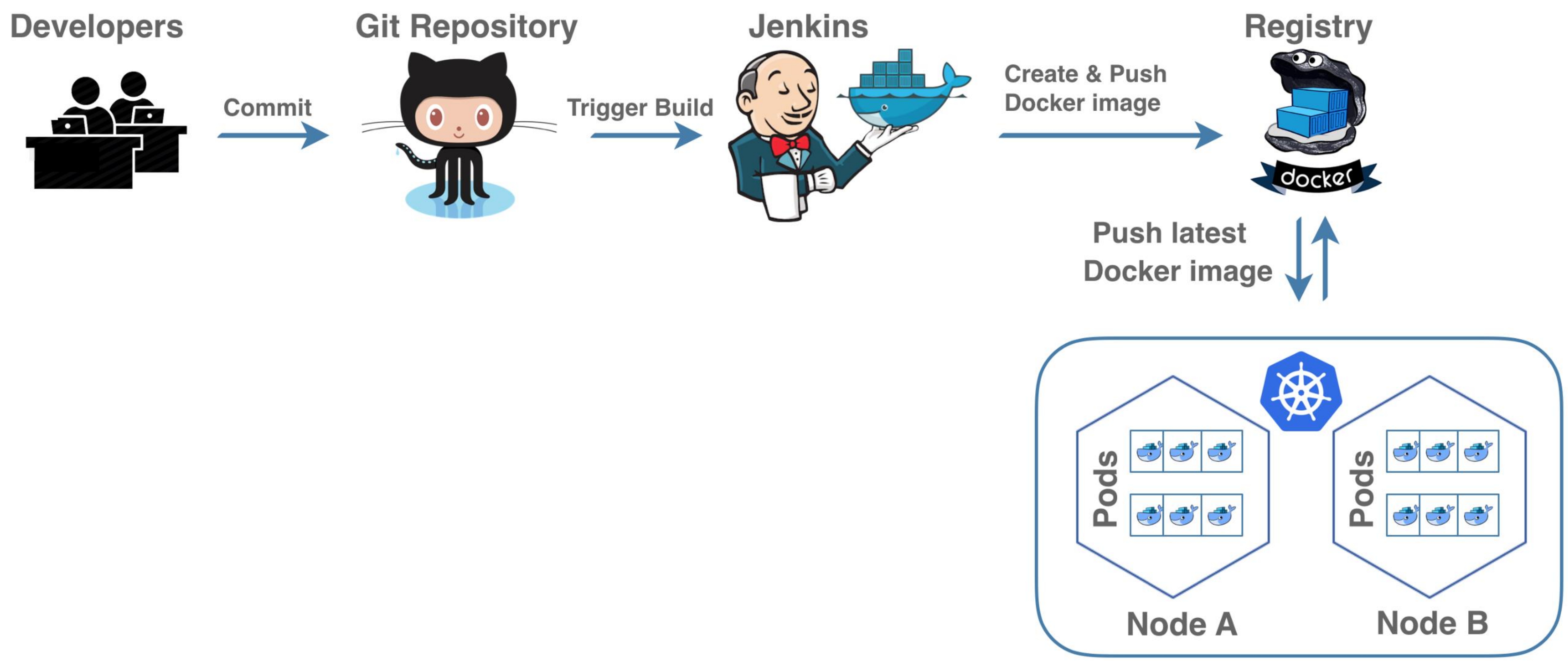
- Used to define and run multiple containers on a swarm cluster.
- Docker stack ignore “build” instructions. You **can’t build new images** using the stack commands.
- It needs pre-built images to deploy.

```
docker stack deploy -c my-stack.yaml mystack
docker stack ls
docker stack rm mystack
docker stack services mystack
```

Docker Stack with Swarm



CI/CD Pipeline



DotNetTricks