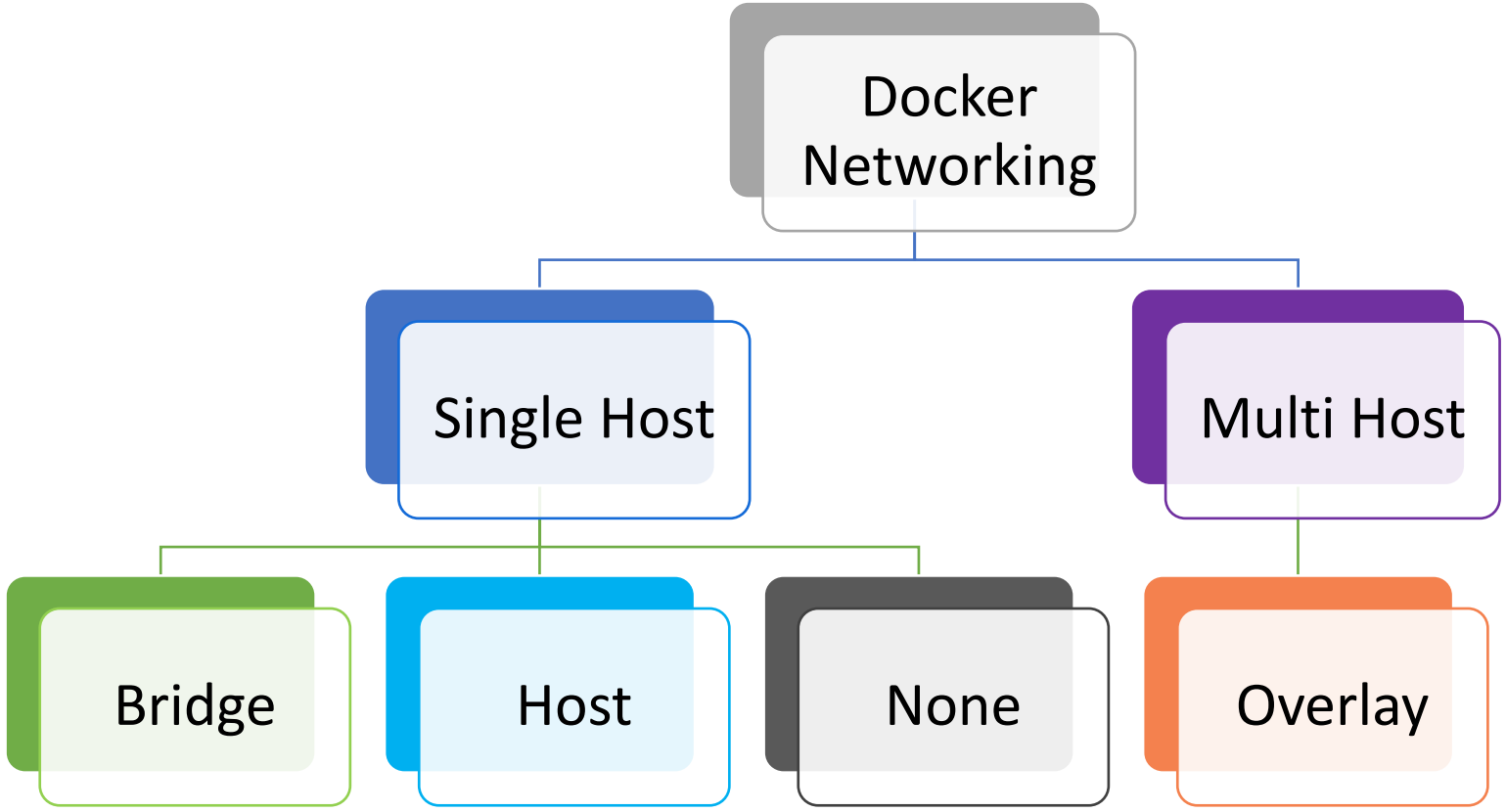
Agenda

Docker Advanced Concepts

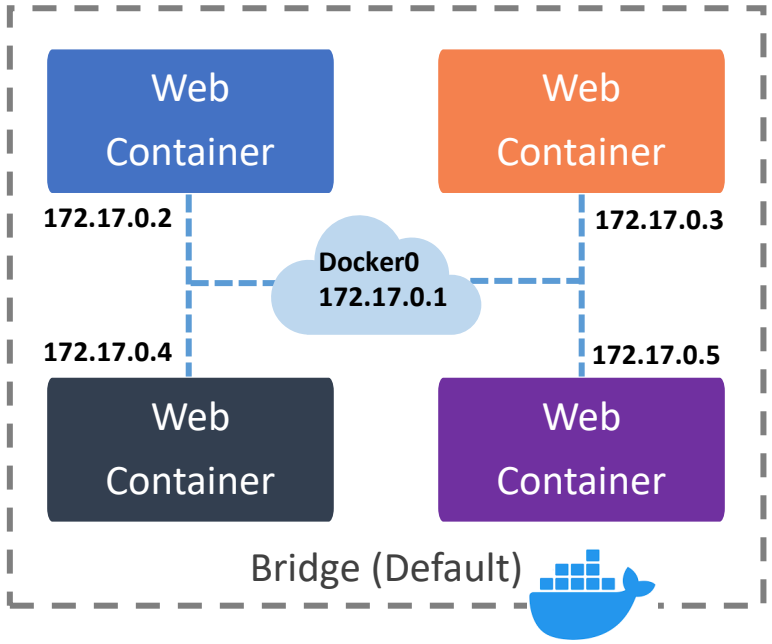
- Docker Networking
- Container Orchestration
- Docker Swarm
- Docker Compose

Docker Networking

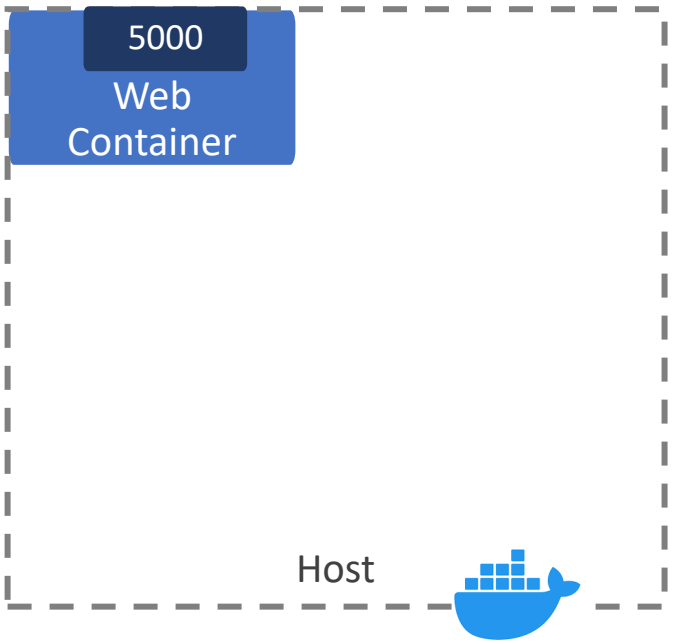


Docker Single Host Networking

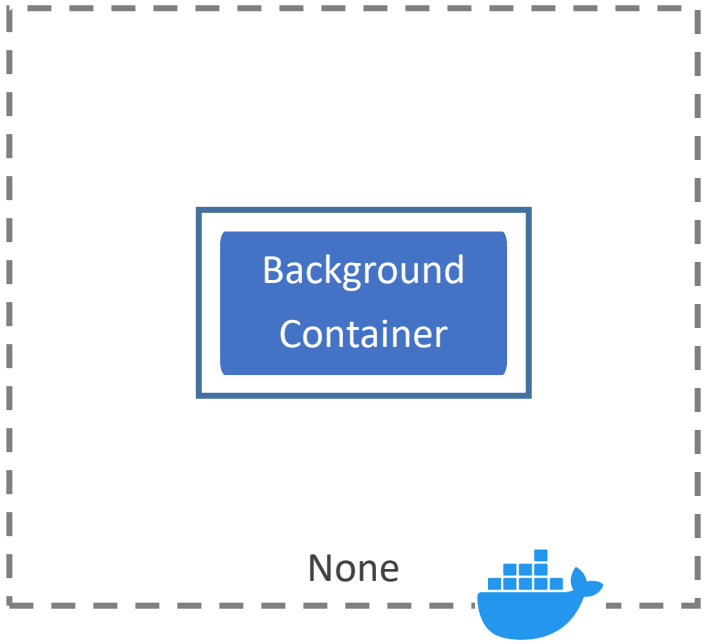
docker run myapp



docker run myapp --network host



docker run myapp --network none



ScholarHat

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

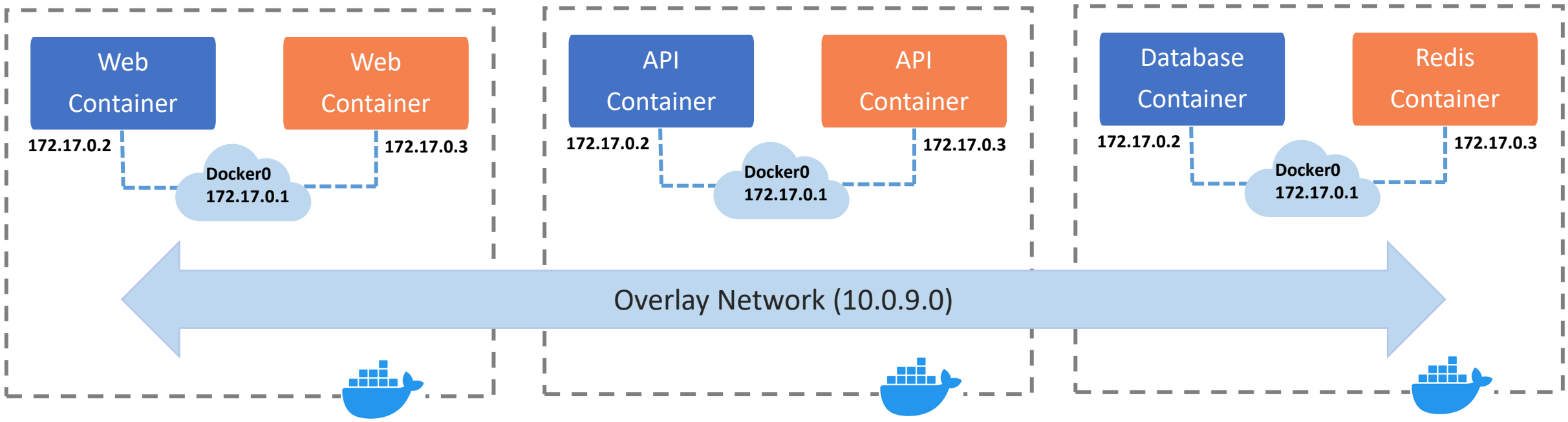
Host

- 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



ScholarHat

Overlay

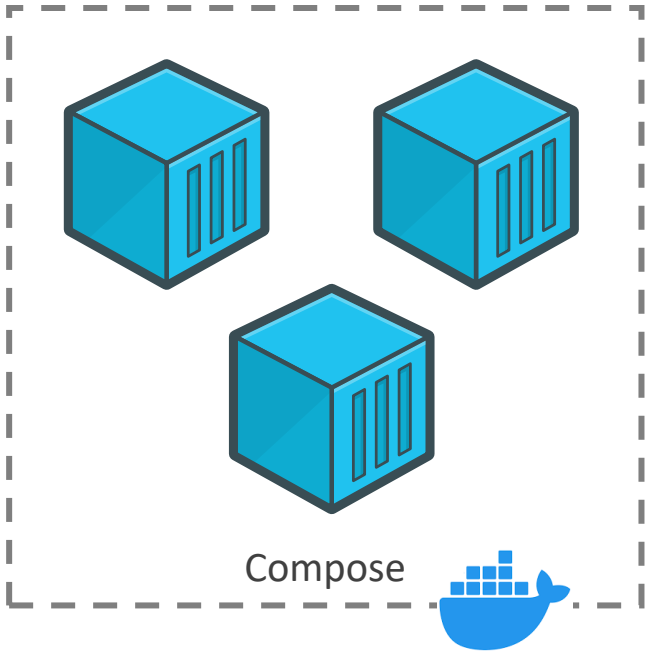
- Manage communications among the Docker daemons participating in the swarm.
- The overlay driver enables simple and secure multi-host networking.
- All containers on the overlay network can communicate with each other.

Docker Network Commands

```
> docker network --help
> docker network ls

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

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

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

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.



```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-CU14-ubuntu-20.04"
    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"
```

// if swarm enabled then leave it

```
docker swarm leave --force
```

```
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

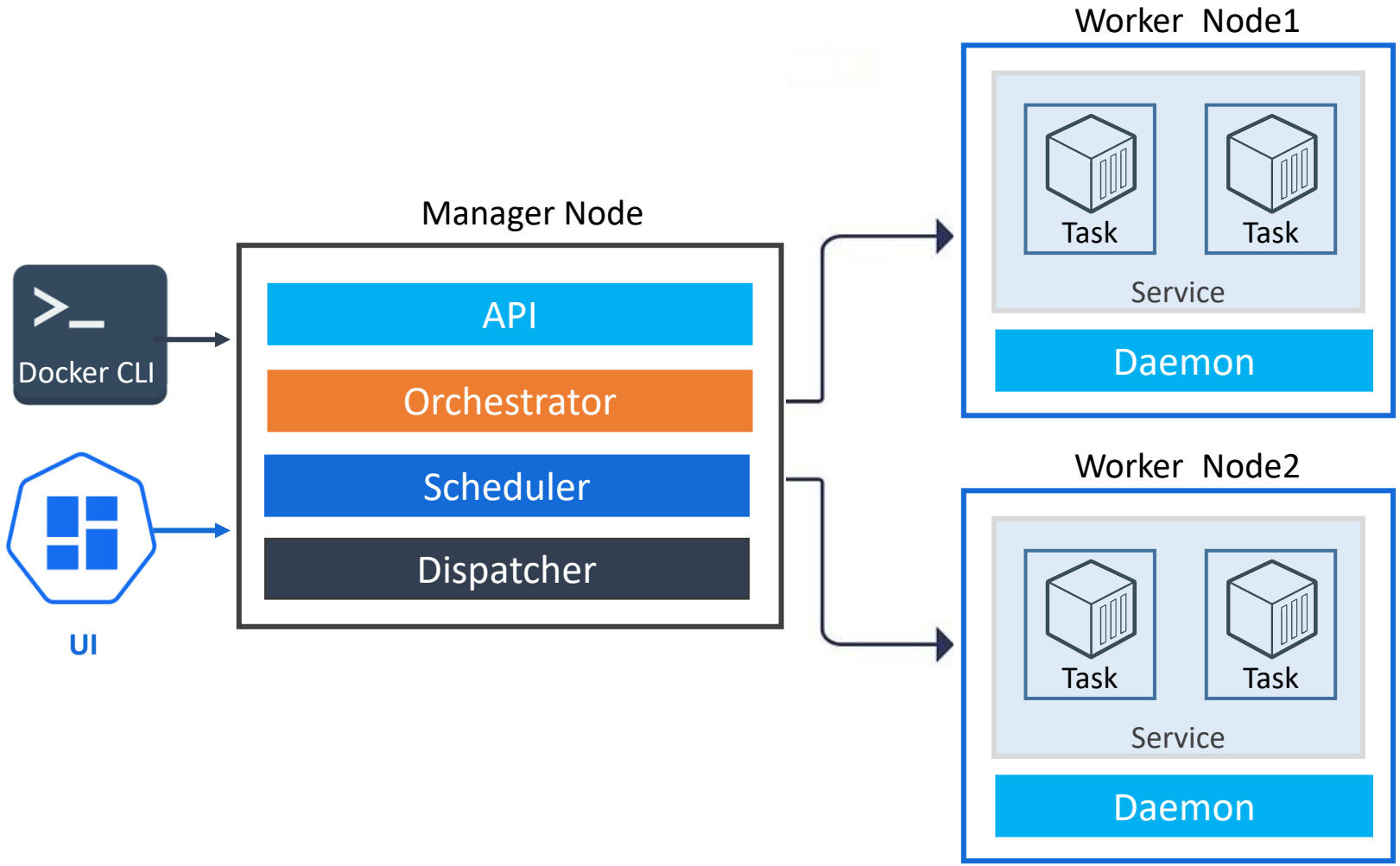
Kubernetes

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



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

CI/CD Pipeline