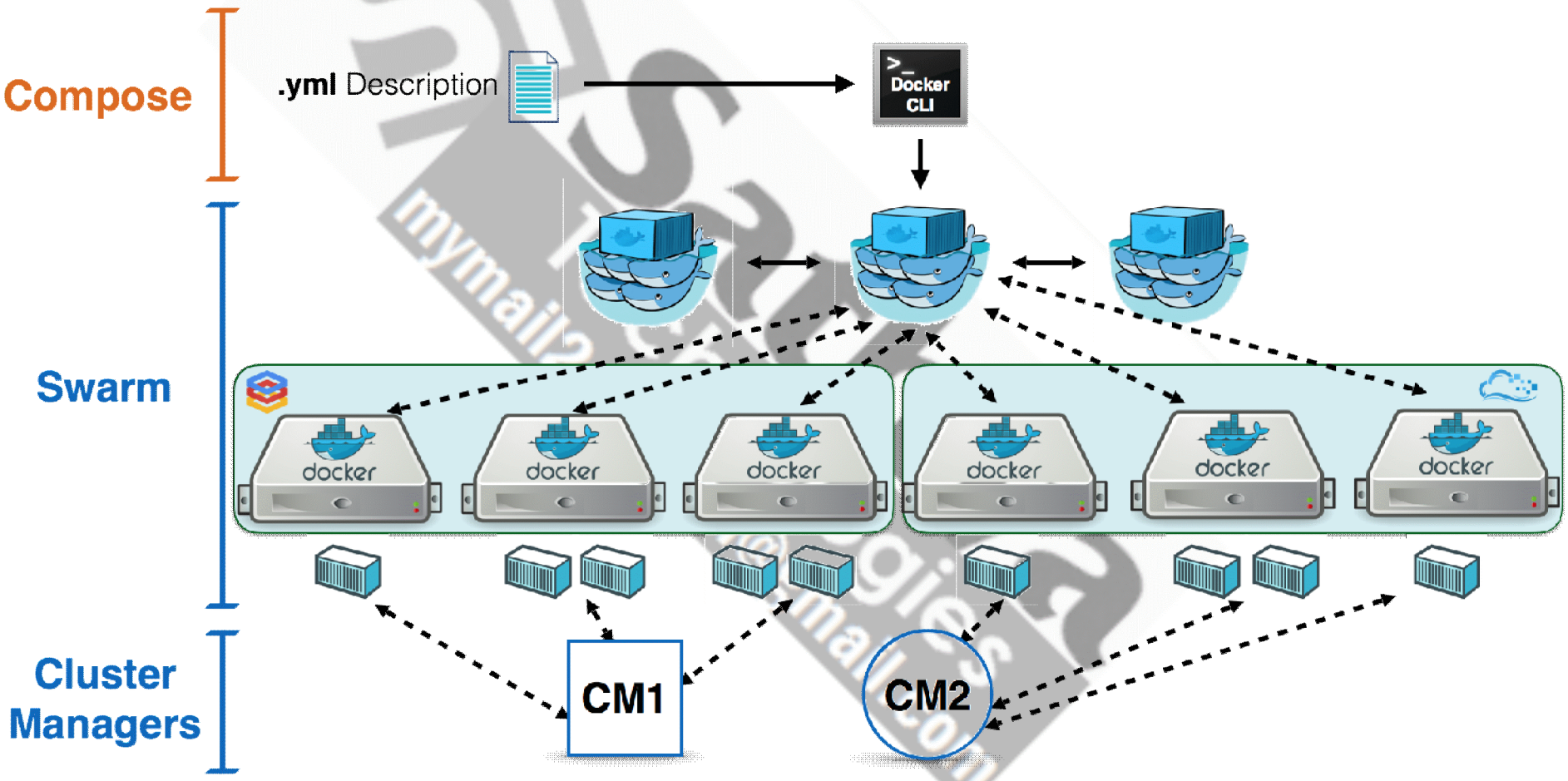
Docker Swarm



To Install docker on Nodes: (using Script)

```
nodes# curl -fsfl https://test.docker.com | sh
```

To create a docker swarm:

```
mgr# docker swarm init --advertise-addr 172.31.32.86
```

[OR]

```
mgr# docker swarm init  --listen-addr 172.31.32.86:2377
```

To add docker swarm Manager:

```
mgr# docker swarm join-token manager
mgr# docker node list
mgr# docker info
```

To Add Nodes to Docker Swarm: (worker nodes)

node# docker swarm join \ --token SWMTKN-1-49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8vxv8rssmk743ojnwacrr2e7c \
172.31.32.86:2377

To Leave Docker Swarm:

 # docker swarm leave

 # docker swarm leave –force

To Manager Docker Services:

```
mgr# docker service ls
mgr#  docker service create --replicas 3
          --name demo nginx
mgr# docker service create -p 80:80
          --name webserver nginx
mgr# docker service scale webserver=5
```

To update services:

```
mgr#  docker service create --name mytom -p
   8080:8080 tomcat
mgr# docker service update --replicas 10 mytom
mgr# netstat -lntp   ---> (you can find port: 8080)
nd1# netstat -lntp   ---> (you can find port: 8080)
nd2# netstat -lntp   ---> (you can find port: 8080)
```

To Inspect services:

```
#docker service ls
#docker service inspect <service id>
#docker service inspect <service id> --pretty
```

To Remove a Service:

docker service rm <service id>

To Remove a Node:

docker node rm node1

docker node rm --force node2

To Inspect Nodes :

Syntax:

#docker node inspect <Node Id>

Ex: #docker node inspect self --pretty

To Promote/Demote a Node to Manager :

Syntax:

#docker node promote <Node Id>

#reboot (manager)

#docker node ls

(you can find a new node as leader)

#docker node demote <Node Id>

To get Docker Logs:

Syntax:

```
# docker logs -f <cid>
```

To Create a Docker Network:

```
# docker network create dev
# docker network ls
```

To Apply a Docker Network:

```
# docker run –it --net dev nginx
# docker inspect  <cid>
```

Docker Network

- Docker Containers to communicate with each other and the outside world via the host machine, there has to be a layer of networking involved.
- Docker supports different types of networks, each fit for certain use cases.

Docker Network

Network Types

Docker comes with network drivers geared towards different use cases.

The most common network types being: **bridge**, **overlay**, and **macvlan**.

Docker Network

Bridge Networks:

Bridge networking is the most common network type. It is limited to containers within a single host running the Docker engine.

Bridge networks are easy to create, manage and troubleshoot.

Ex:

 # docker network create -d bridge my-bridge-net

Docker Network

Overlay Networks:

An overlay network uses software virtualization to create additional layers of network abstraction running on top of a physical network.

In Docker, an overlay network driver is used for multi-host network communication.

Ex:

```
# docker network create -d overlay
        --subnet=192.168.10.0/24 my-overlay-net
```

Docker Network

Macvlan Networks:

The macvlan driver is used to connect Docker containers directly to the host network interfaces through layer 2 segmentation. .

Ex:

```
# docker network create -d macvlan \
  --subnet=192.168.40.0/24 \
  --gateway=192.168.40.1 \
  -o parent=eth0 my-macvlan-net
```