



KODEKLOUD

DOCKER

Certified Associate Course



MUMSHAD
MANNAMBE
TH

YOGESH
RAHEJA





CERTIFIED
ASSOCIATE

Objectives



Installation and Configuration



Image Management



Storage and Volumes



Networking



Security



Orchestration

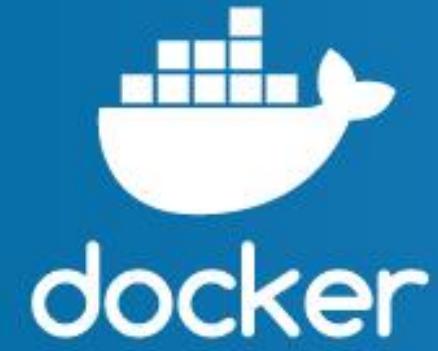




KODEKLOUD

Exam

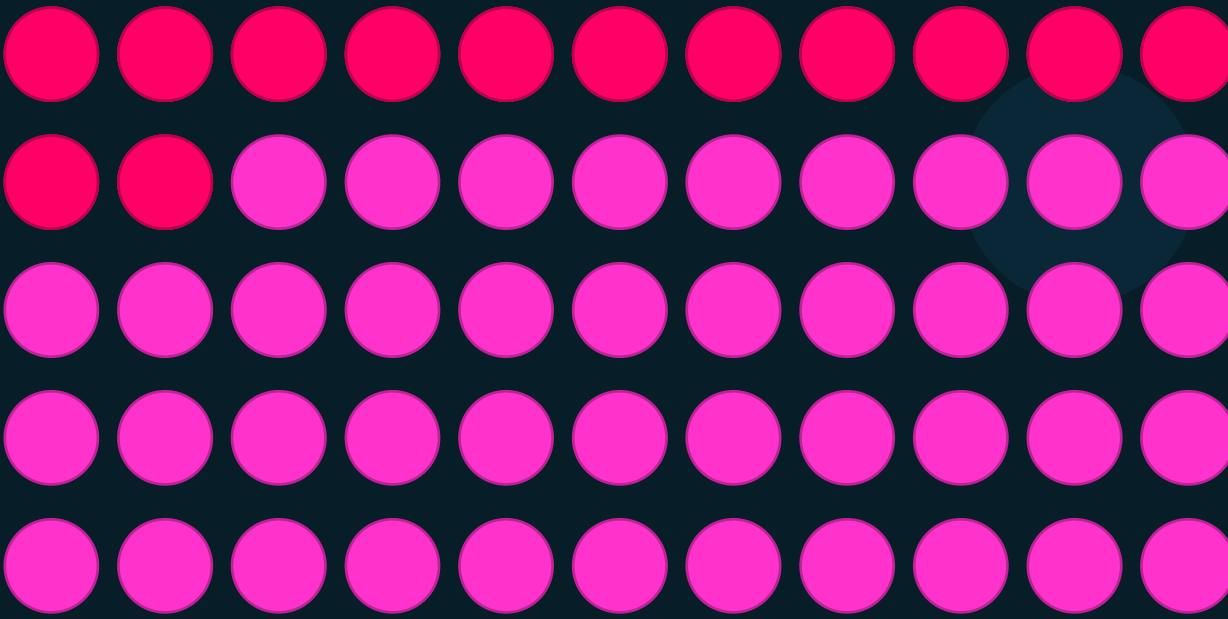
Details



CERTIFIED
ASSOCIATE



KODEKLOUD



90
minutes



MCQ vs DOMC

Q. What is the default network driver used when a container is provisioned?

- overlay
- bridge
- None
- host

Submit



MCQ vs DOMC

Q. What is the default network driver used when a container is provisioned?

- overlay Yes No
- bridge Yes No
- None Yes No
- host Yes No



Frequently Asked Questions

Q. Can we take the exam from home or a testing center?

A. Home (Proctored)

Q. Fee for the exam

A. \$195

Q. Passing score

A. N/A

Q. When will I get the results

A. Immediately



Register

<https://training.mirantis.com/dca-certification-exam/>





KODEKLOUD

Curriculum

Curriculum



Installation and Configuration



Image Management



Storage and Volumes



Networking



Security



Orchestration



Curriculum



Installation and Configuration



Image Management



Storage and Volumes



Networking



Security



Orchestration

- Sizing Requirements
- Docker Engine Installation
- Swarm Installation
- Docker Enterprise – UCP, DTR
- Manage Users & Teams
- Daemon Configuration
- Certificate based auth
- Namespaces & Cgroups
- Troubleshoot issues
- Configure Backups



Curriculum



Installation and Configuration



Image Management



Storage and Volumes



Networking



Security



Orchestration

- Dockerfile
- Dockerfile Instructions
- Create efficient image
- Docker Image CLI
- Push,Pull,Delete images
- Inspect Images
- Tag Images
- Display Layers
- Registry Functions
- Deploy & Search in Registry



Curriculum



Installation and Configuration



Image Management



Storage and Volumes



Networking



Security



Orchestration

- Drivers for various OS
- Compare Objects vs Block
- Image layers and filesystem
- Volumes
- Cleanup unused images
- PV, PVCs on Kubernetes
- Storage Classes



Curriculum



Installation and Configuration



Image Management



Storage and Volumes



Networking



Security



Orchestration

- Container Network Model
- Built-in Network Drivers
- Traffic flow between Docker Engine, Registry & UCP
- Docker Bridge Network
- Publish Ports
- External DNS
- Deploy a service on a docker overlay network
- Troubleshoot container and engine logs
- Kubernetes traffic using Cluster IP and NodePort Services
- Kubernetes Network Policies



Curriculum



Installation and Configuration



Image Management



Storage and Volumes



Networking



Security



Orchestration

- Image signing
- Docker Engine Security
- Docker Swarm Security
- Identity Roles
- UCP Workers vs Managers
- Security scan in images
- Docker Content Trust
- RBAC with UCP
- UCP with LDAP/AD
- UCP Client Bundles



Curriculum



Installation and Configuration



Image Management



Storage and Volumes



Networking



Security



Orchestration

- Docker Swarm:
 - Setup Swarm Cluster
 - Quorum in a Swarm Cluster
 - Stack in swarm
 - Scale up and down replicas
 - Networks, Publish Ports
 - Replicated vs Global Services
 - Placements
 - Healthchecks
- Kubernetes
 - PODS, Deployments
 - Services
 - ConfigMaps, Secrets
 - Liveness and Readiness Probes



Curriculum

Docker Engine

Docker Swarm

Kubernetes

Docker Enterprise



Installation and Configuration



Image Management



Storage and Volumes



Networking



Security



Orchestration



Pre-Requisite

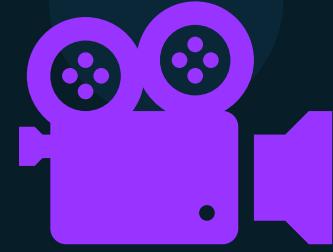
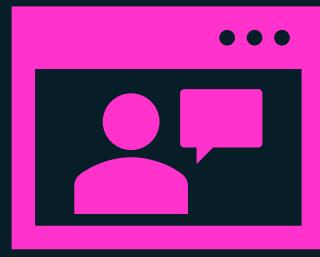




KODEKLOUD

Course & Exam Tips

Learning Format



Research Questions

Docker Engine - Architecture

- 📄 Section Introduction
- ▶ Docker Engine Architecture (9:09)
- 📄 Docker Engine Setup
- 💡 Research Questions
- 📄 Docker Service Configuration
- 📄 Basic Container Operations
- 💡 Research Questions
- 📄 Interacting with a Running Container
- ▶ Inspecting a Container (5:54)
- 📄 Stopping and Removing a Container

1 / 16

What is the command to start docker daemon manually?

`docker`
`dockerd`
`docker-engine`
`docker --start-engine`

- Open Book
- Refer to Lecture and Documentation
- Research
- Get familiar with the MCQ format

Notes

- Note the most difficult/confusing concepts for you
- Don't write large notes



Revision

Docker Engine - Architecture

-  Section Introduction
-  Docker Engine Architecture (9:09)
-  Docker Engine Setup
-  Research Questions
-  Docker Service Configuration
-  Basic Container Operations
-  Research Questions
-  Interacting with a Running Container
-  Inspecting a Container (5:54)
- Stopping and Removing a Container

Revision

- Research Questions
- Troubleshooting Docker Daemon
- Docker Debug Mode
- Logging Driver
- Logging Driver
- Research Questions
- Practice Test

Docker Engine - Images

- Section Introduction Draft
- Docker Image Registry Draft

Revision

 Research Questions

 Practice Test

Docker Engine - Images

 Section Introduction Draft

 Docker Image Registry Draft

Mock Exams

 Mock Exam 1

 Mock Exam 2

 Mock Exam 3

Learning Schedule				
Section	Learning Time (Hours)	Days (2 Hours)	Days (4 Hours)	Days (6 Hours)
Docker Architecture	20	10	5	3
Images	20	10	5	3
Security	8	4	2	1
Networking	14	7	3.5	2
Storage	8	4	2	1
Compose	12	6	3	2
Docker Swarm	26	13	6.5	4
Kubernetes	32	16	8	5
Docker Engine Enterprise	12	6	3	2
Docker Trusted Registry	6	3	1.5	1
Disaster Recovery	8	4	2	1
Mock Exams	28	14	7	5
Total Duration	194 Hours	97 Days	48.5 Days	30 Days

Quest

Q5. You have

By default

- (A) /
- (B) /
- (C) /
- (D) /

What port on pods matching this service's selector will receive traffic sent to the service on port 8080?

- (A) 8080/tcp
- (B) 8080/udp
- (C) 80/tcp
- (D) 80/udp

```
yaml
apiVersion: v1
kind: Service
metadata:
  name: dca
spec:
  type: clusterIP
  selector:
    app: nginx
  ports:
    - port: 8080
      targetPort: 80
    - port: 4443
      targetPort: 443
```

manager logs?



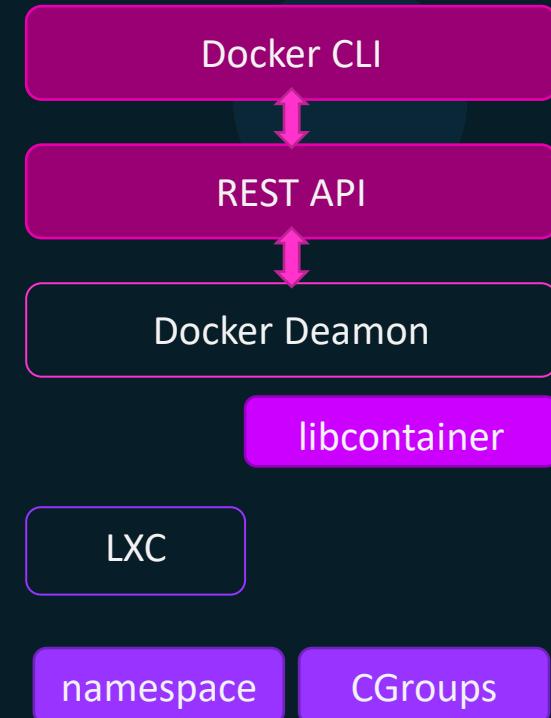
Architecture

Docker Engine

Docker Engine Architecture

2013

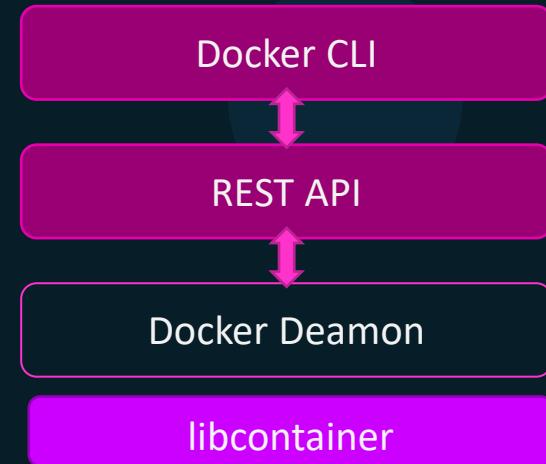
2014 (v0.9)



Docker Engine Architecture

2013

2014 (v0.9)



namespace

CGroups



Docker Engine Architecture

201
201

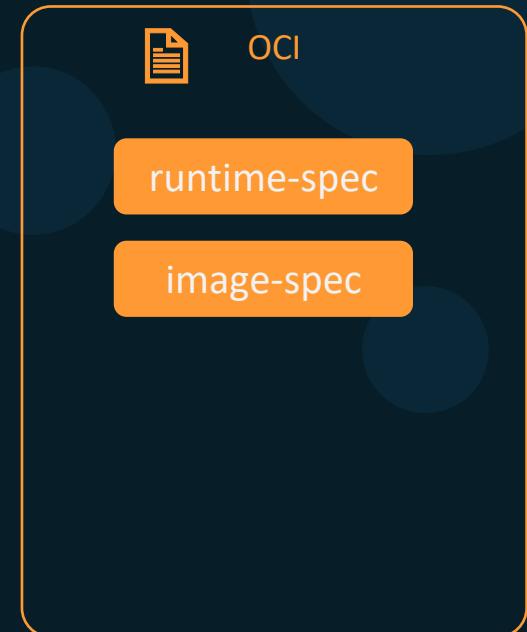
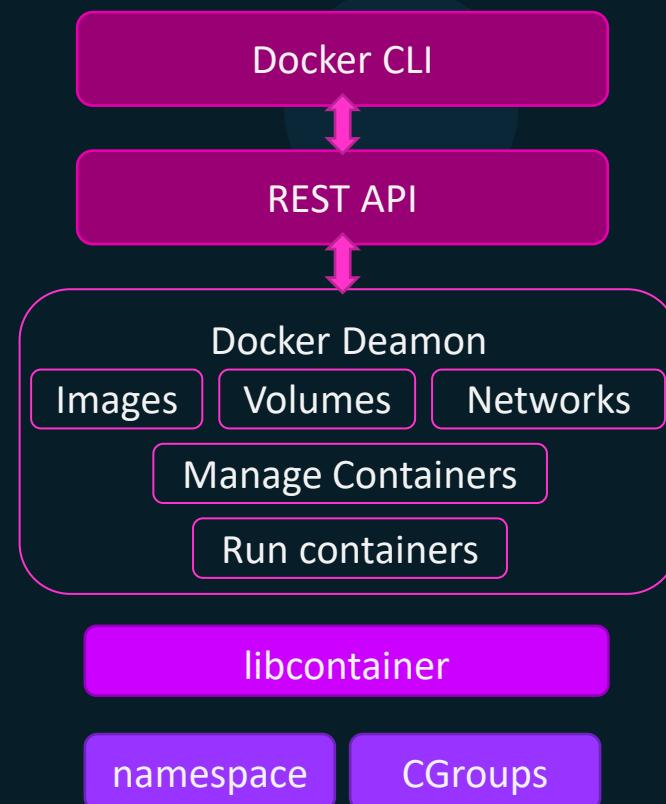
Lifecycle <https://github.com/opencontainers/runtime-spec/blob/master/runtime.md>

The lifecycle describes the timeline of events that happen from when a container is created to when it ceases to exist.

1. OCI compliant runtime's `create` command is invoked with a reference to the location of the bundle and a unique identifier.
2. The container's runtime environment MUST be created according to the configuration in `config.json`. If the runtime is unable to create the environment specified in the `config.json`, it MUST **generate an error**. While the resources requested in the `config.json` MUST be created, the user-specified program (from `process`) MUST NOT be run at this time. Any updates to `config.json` after this step MUST NOT affect the container.
3. The `prestart hooks` MUST be invoked by the runtime. If any `prestart` hook fails, the runtime MUST **generate an error**, stop the container, and continue the lifecycle at step 12.
4. The `createRuntime hooks` MUST be invoked by the runtime. If any `createRuntime` hook fails, the runtime MUST **generate an error**, stop the container, and continue the lifecycle at step 12.
5. The `createContainer hooks` MUST be invoked by the runtime. If any `createContainer` hook fails, the runtime MUST **generate an error**, stop the container, and continue the lifecycle at step 12.
6. Runtime's `start` command is invoked with the unique identifier of the container.
7. The `startContainer hooks` MUST be invoked by the runtime. If any `startContainer` hook fails, the runtime MUST **generate an error**, stop the container, and continue the lifecycle at step 12.
8. The runtime MUST run the user-specified program, as specified by `process`.
9. The `poststart hooks` MUST be invoked by the runtime. If any `poststart` hook fails, the runtime MUST **log a warning**, but the remaining hooks and lifecycle continue as if the hook had succeeded.
10. The container process exits. This MAY happen due to erroring out, exiting, crashing or the runtime's `kill` operation being invoked.
11. Runtime's `delete` command is invoked with the unique identifier of the container.
12. The container MUST be destroyed by undoing the steps performed during create phase (step 2).
13. The `poststop hooks` MUST be invoked by the runtime. If any `poststop` hook fails, the runtime MUST **log a warning**, but the remaining hooks and lifecycle continue as if the hook had succeeded.

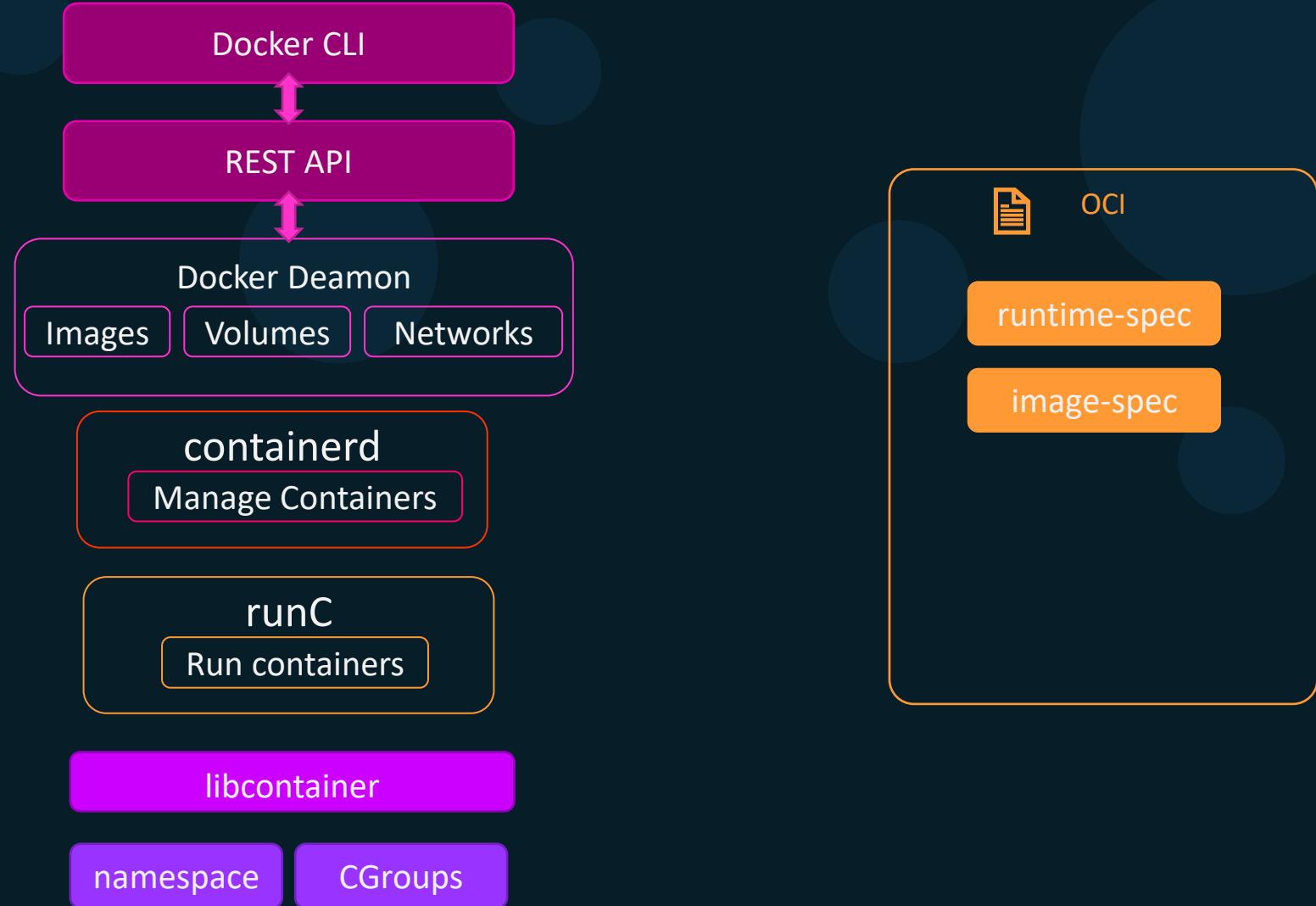
Docker Engine Architecture

- 2013
- 2014 (v0.9)
- 2016 (v1.11)



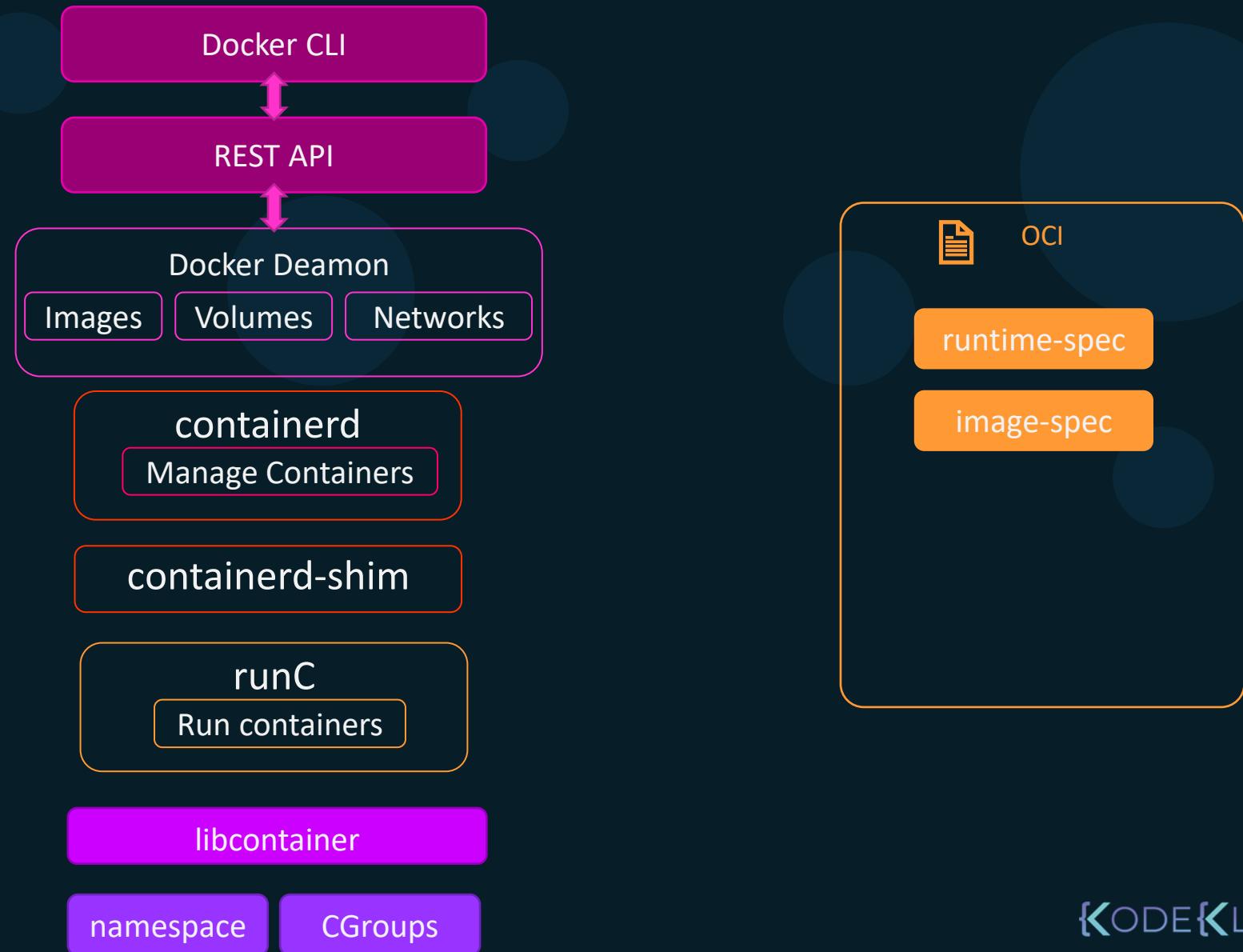
Docker Engine Architecture

- 2013
- 2014 (v0.9)
- 2016 (v1.11)



Docker Engine Architecture

- 2013
- 2014 (v0.9)
- 2016 (v1.11)



Docker Objects

Images

Networks

Containers

Volumes



Docker Objects

Images



Networks



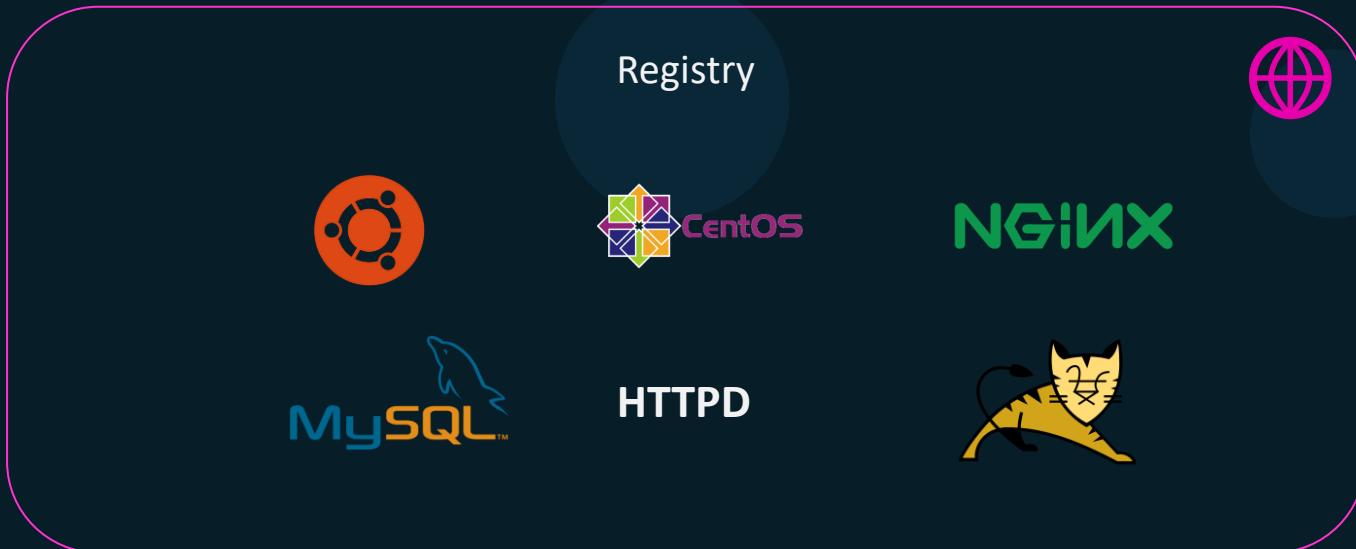
Containers



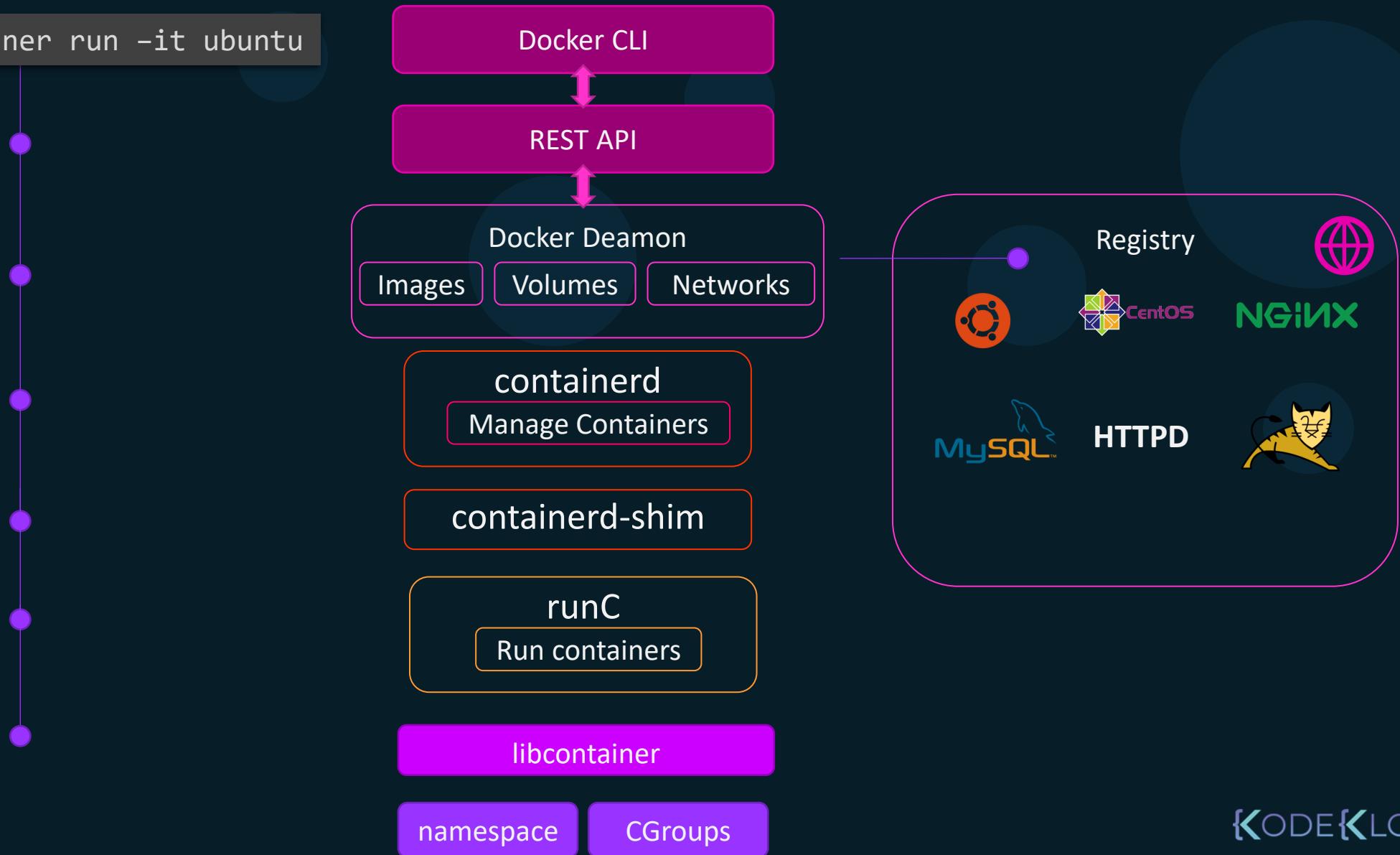
Volumes



Registry



```
▶ docker container run -it ubuntu
```



Docker Engine Installation

```
▶ docker version
```

```
Client: Docker Engine - Community
Version: 19.03.5
API version: 1.40
Go version: go1.12.12
Git commit: 633a0ea
Built: Wed Nov 13 07:25:41 2019
OS/Arch: linux/amd64
Experimental: false
```

```
Server: Docker Engine - Community
```

```
Engine:
  Version: 19.03.5
  API version: 1.40 (minimum version 1.12)
  Go version: go1.12.12
  Git commit: 633a0ea
  Built: Wed Nov 13 07:24:18 2019
  OS/Arch: linux/amd64
  Experimental: false
containerd:
  Version: 1.2.10
  GitCommit: b34a5c8af56e510852c35414db4c1f4fa6172339
runc:
  Version: 1.0.0-rc8+dev
  GitCommit: 3e425f80a8c931f88e6d94a8c831b9d5aa481657
docker-init:
  Version: 0.18.0
  GitCommit: fec3683
```

```
▶ docker --version
```

```
Docker version 19.03.5, build 633a0ea
```

```
▶ docker system info
```

```
Client:
  Debug Mode: false
```

```
Server:
  Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
  Images: 0
  Server Version: 19.03.5
  Storage Driver: overlay2
    Backing Filesystem: xfs
  .
  .
  .
  Experimental: false
```

```
Insecure Registries:
```

```
  127.0.0.0/8
```

```
Live Restore Enabled: false
```



Docker

Service Configuration

Check Service Status

```
▶ systemctl start docker
```

```
▶ systemctl status docker
```

```
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2020-10-21 04:21:01 UTC; 3 days ago
     Docs: https://docs.docker.com
 Main PID: 4197 (dockerd)
    Tasks: 13
   Memory: 129.7M
      CPU: 9min 6.980s
    CGroup: /system.slice/docker.service
            └─4197 /usr/bin/dockerd -H fd:// -H tcp://0.0.0.0 --containerd=/run/containerd/containerd.sock
```

```
▶ systemctl stop docker
```



Start Manually

► dockerd

```
INFO[2020-10-24T08:20:40.372653463Z] Starting up
INFO[2020-10-24T08:20:40.375298351Z] parsed scheme: "unix"                                     module=grpc
INFO[2020-10-24T08:20:40.375510773Z] scheme "unix" not registered, fallback to default scheme   module=grpc
INFO[2020-10-24T08:20:40.375657667Z] ccResolverWrapper: sending update to cc: {{[{"unix://run/containerd/containerd.sock": <nil>}]} <nil>}  module=grpc
INFO[2020-10-24T08:20:40.375973480Z] ClientConn switching balancer to "pick_first"  module=grpc
INFO[2020-10-24T08:20:40.377210185Z] parsed scheme: "unix"                                     module=grpc
INFO[2020-10-24T08:20:40.377304998Z] scheme "unix" not registered, fallback to default scheme  module=grpc
INFO[2020-10-24T08:20:40.377491827Z] ccResolverWrapper: sending update to cc: {{[{"unix://run/containerd/containerd.sock": <nil>}]} <nil>}  module=grpc
INFO[2020-10-24T08:20:40.377762558Z] ClientConn switching balancer to "pick_first"  module=grpc
INFO[2020-10-24T08:20:40.381198263Z] [graphdriver] using prior storage driver: overlay2
WARN[2020-10-24T08:20:40.572888603Z] Your kernel does not support swap memory limit
WARN[2020-10-24T08:20:40.573014192Z] Your kernel does not support cgroup rt period
WARN[2020-10-24T08:20:40.573404879Z] Your kernel does not support cgroup rt runtime
```

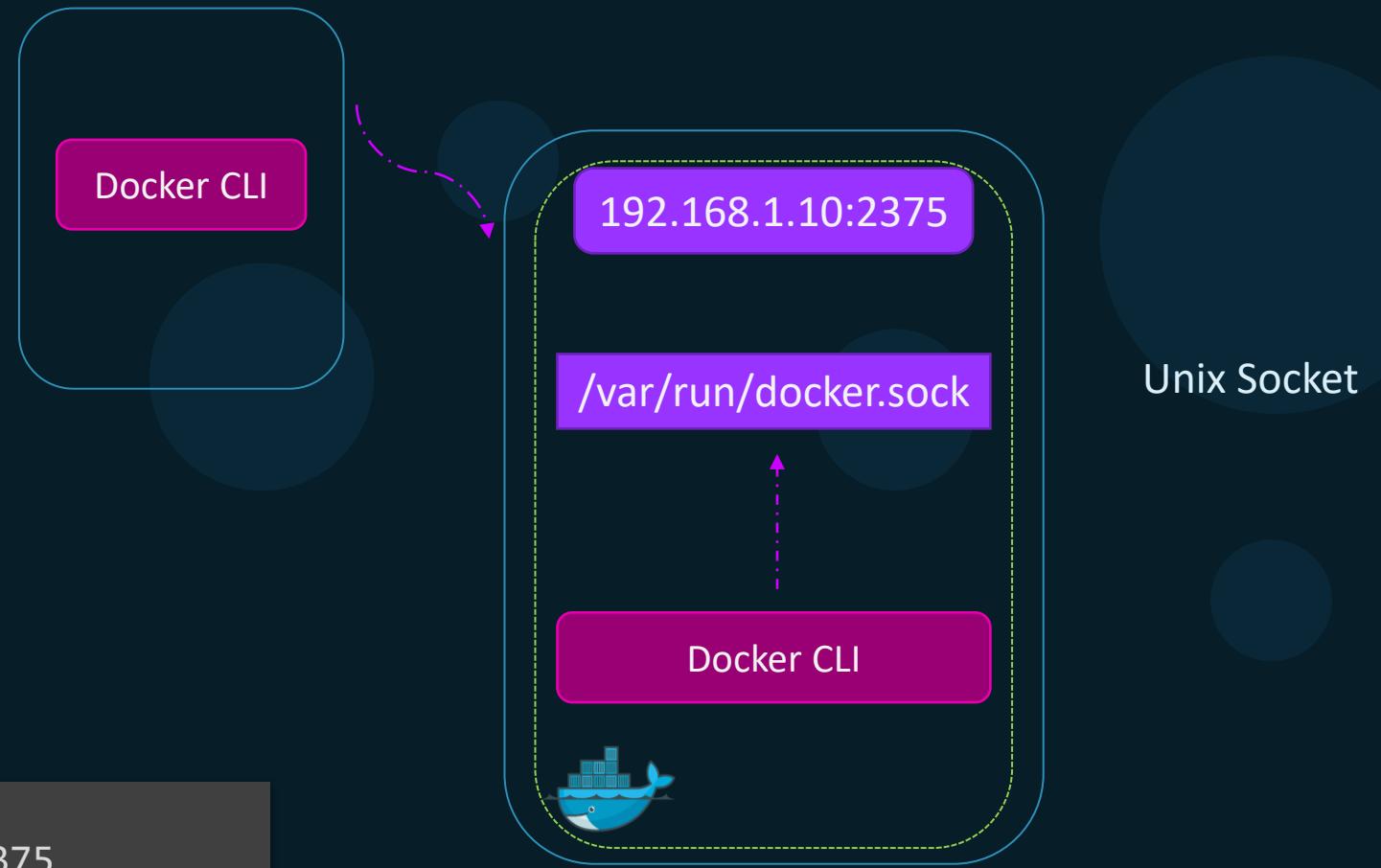


Start Manually With Debug

► dockerd --debug

```
INFO[2020-10-24T08:29:00.331925176Z] Starting up
DEBU[2020-10-24T08:29:00.332463203Z] Listener created for HTTP on unix (/var/run/docker.sock)
DEBU[2020-10-24T08:29:00.333316936Z] Golang's threads limit set to 6930
INFO[2020-10-24T08:29:00.333659056Z] parsed scheme: "unix" module=grpc
INFO[2020-10-24T08:29:00.333685921Z] scheme "unix" not registered, fallback to default scheme module=grpc
INFO[2020-10-24T08:29:00.333705237Z] ccResolverWrapper: sending update to cc: {{[{"unix:///run/containerd/containerd.sock": <nil>}]} <nil>} module=grpc
INFO[2020-10-24T08:29:00.333715024Z] ClientConn switching balancer to "pick_first" module=grpc
INFO[2020-10-24T08:29:00.334889983Z] parsed scheme: "unix" module=grpc
INFO[2020-10-24T08:29:00.334914951Z] scheme "unix" not registered, fallback to default scheme module=grpc
INFO[2020-10-24T08:29:00.334931237Z] ccResolverWrapper: sending update to cc: {{[{"unix:///run/containerd/containerd.sock": <nil>}]} <nil>} module=grpc
INFO[2020-10-24T08:29:00.334940958Z] ClientConn switching balancer to "pick_first" module=grpc
DEBU[2020-10-24T08:29:00.335626982Z] Using default logging driver json-file
DEBU[2020-10-24T08:29:00.335808043Z] [graphdriver] priority list: [btrfs zfs overlay2 aufs overlay devicemapper vfs]
DEBU[2020-10-24T08:29:00.335969923Z] processing event stream module=libcontainerd
namespace=plugins.moby
DEBU[2020-10-24T08:29:00.337633503Z] backingFs=extfs, projectQuotaSupported=false, indexOff="" storage-driver=overlay2
INFO[2020-10-24T08:29:00.337658643Z] [graphdriver] using prior storage driver: overlay2
DEBU[2020-10-24T08:29:00.337674607Z] Initialized graph driver overlay2
WARN[2020-10-24T08:29:00.364649284Z] Your kernel does not support swap memory limit
WARN[2020-10-24T08:29:00.364679148Z] Your kernel does not support cgroup rt period
WARN[2020-10-24T08:29:00.364687757Z] Your kernel does not support cgroup rt runtime
```

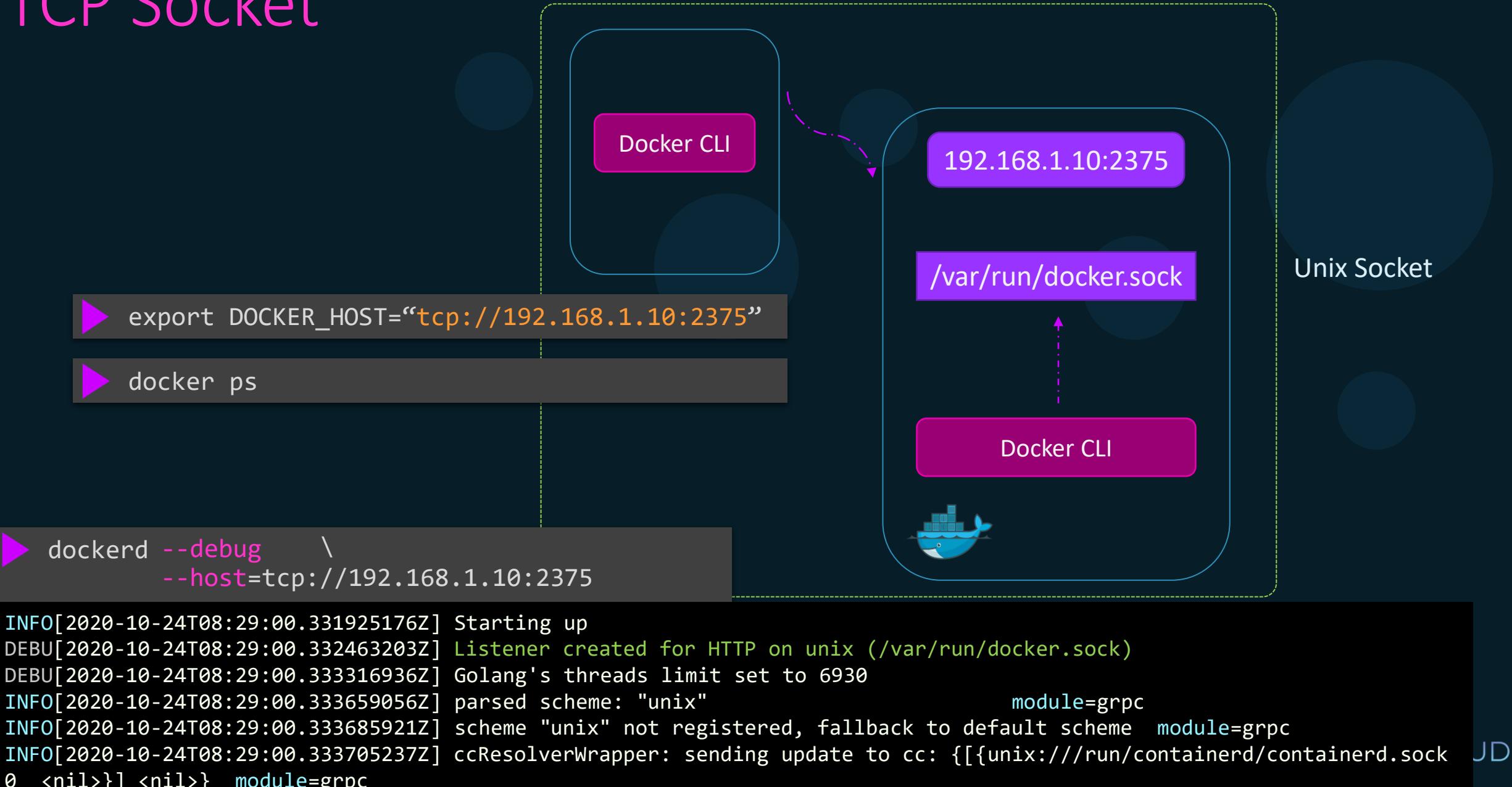
Unix Socket



```
▶ dockerd --debug \
--host=tcp://192.168.1.10:2375
```

```
INFO[2020-10-24T08:29:00.331925176Z] Starting up
DEBU[2020-10-24T08:29:00.332463203Z] Listener created for HTTP on unix (/var/run/docker.sock)
DEBU[2020-10-24T08:29:00.333316936Z] Golang's threads limit set to 6930
INFO[2020-10-24T08:29:00.333659056Z] parsed scheme: "unix"                                     module=grpc
INFO[2020-10-24T08:29:00.333685921Z] scheme "unix" not registered, fallback to default scheme module=grpc
INFO[2020-10-24T08:29:00.333705237Z] ccResolverWrapper: sending update to cc: {{<nil>}} <nil> module=grpc
```

TCP Socket

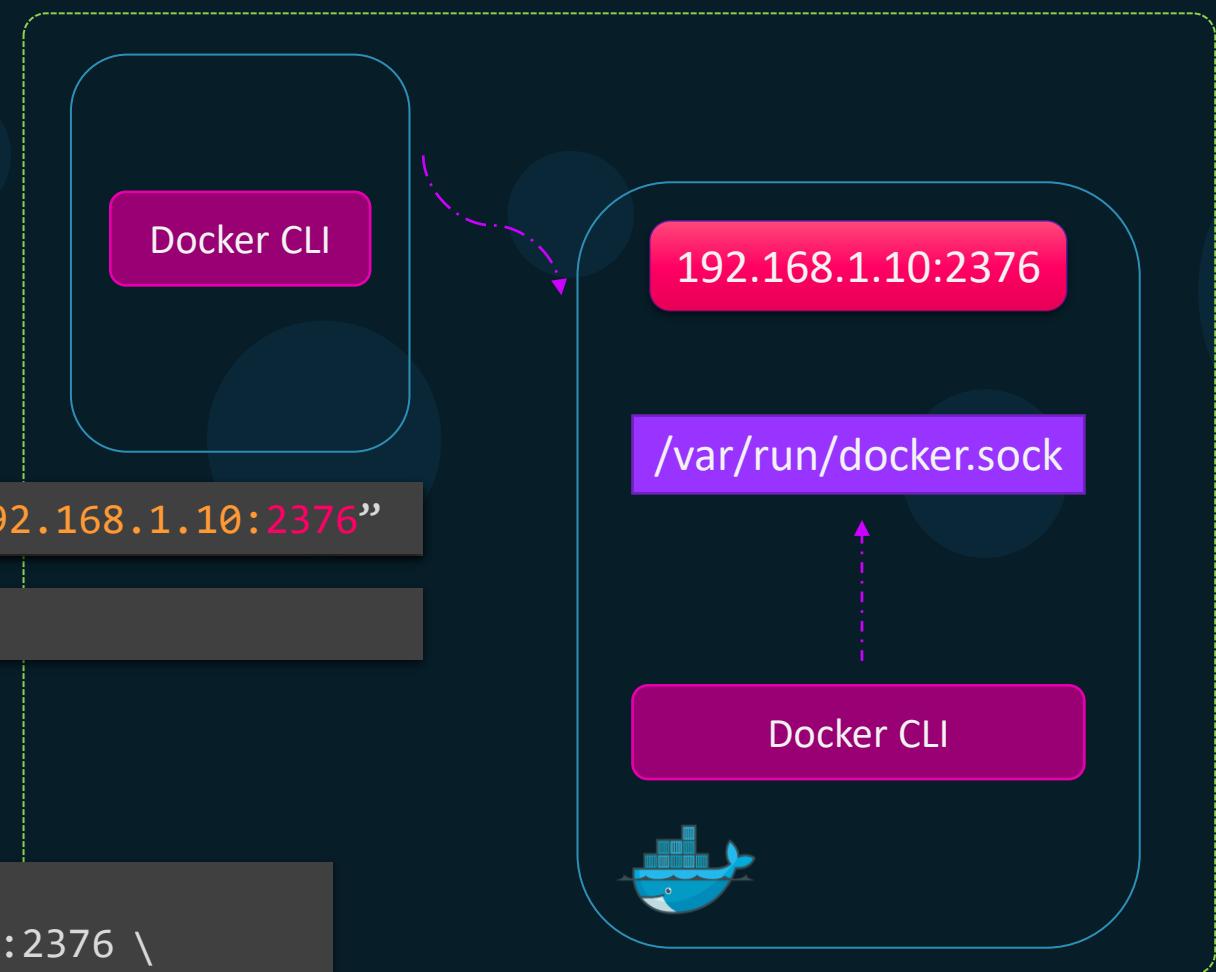


TLS Encryption

```
▶ export DOCKER_HOST="tcp://192.168.1.10:2376"
```

```
▶ docker ps
```

```
▶ dockerd --debug \
    --host=tcp://192.168.1.10:2376 \
    --tls=true \
    --tlscert=/var/docker/server.pem \
    --tlskey=/var/docker/serverkey.pem
```



2375	Un-encrypted
2376	Encrypted



Daemon Configuration File

```
▶ dockerd --debug \
--host=tcp://192.168.1.10:2376 \
--tls=true \
--tlscert=/var/docker/server.pem \
--tlskey=/var/docker/serverkey.pem
```

```
▶ dockerd --debug=false
```

unable to configure the Docker daemon with file /etc/docker/daemon.json: the following directives are specified both as a flag and in the configuration file: debug: (from flag: false, from file: true)

```
▶ systemctl start docker
```

```
/etc/docker/daemon.json
```

```
{
  "debug": true,
  "hosts": ["tcp://192.168.1.10:2376"]
  "tls": true,
  "tlscert": "/var/docker/server.pem",
  "tlskey": "/var/docker/serverkey.pem"
}
```



References

- <https://docs.docker.com/engine/reference/commandline/dockerd/#daemon-configuration-file>
- <https://docs.docker.com/config/daemon/>
- <https://docs.docker.com/engine/reference/commandline/dockerd/>
- <https://docs.docker.com/engine/security/https/>





KODEKLOUD

Basic Container Operations

Docker Objects

► docker <docker-object> <sub-command> [options] <Arguments/Commands>

Images



► docker image ls

Networks



► docker network ls

Containers



► docker container ls

Volumes



► docker volume ls



Docker Engine Command

```
▶ docker <docker-object> <sub-command> [options] <Arguments/Commands> s>
```

```
▶ docker container run -it ubuntu
```

```
▶ docker run -it ubuntu
```

```
▶ docker image build .
```

```
▶ docker build .
```

```
▶ docker container attach ubuntu
```

```
▶ docker attach ubuntu
```

```
▶ docker container kill ubuntu
```

```
▶ docker kill ubuntu
```



Container Create - Create a new container

```
▶ docker container create httpd
```

```
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
8ec398bc0356: Pull complete
354e6904d655: Pull complete
36412f6b2f6e: Pull complete
Digest:
sha256:769018135ba22d3a7a2b91cb89b8de711562cdf51ad6621b2b9b13e95f3798de
Status: Downloaded newer image for httpd:latest
36a391532e10d45f772f2c9430c2cc38dad4b441aa7a1c44d459f6fa3d78c6b6
```

```
▶ ls /var/lib/docker/
```

builder	containers	network	plugins	swarm	trust
buildkit	image	overlay2	runtimes	tmp	volumes

```
▶ ls -lrt /var/lib/docker/containers/
```

```
36a391532e10d45f772f2c9430c2cc38dad4b441aa7a1c44d459f6fa3d78c6b6
```

```
▶ ls -lrt /var/lib/docker/containers/36a391532e10*
```

```
Checkpoint          hostconfig.json    config.v2.json
```



Container ls - List the details for container

```
▶ docker container ls
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

```
▶ docker container ls -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
36a391532e10	httpd	"httpd-foreground"	2 minutes ago	Created		charming_wiles

```
▶ docker container ls -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
36a391532e10	httpd	"httpd-foreground"	2 minutes ago	Created		charming_wiles

```
▶ docker container ls -q
```

```
36a391532e10
```



Container Start - Start a container

```
▶ docker container start 36a391532e10
```

```
36a391532e10
```

```
▶ docker container ls
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
36a391532e10	httpd	"httpd-foreground"	6 minutes ago	Up 1 minutes	80/tcp	charming_wiles



Container Run – Create and Start a container

```
▶ docker container create httpd
```



```
▶ docker container start 36a391532e10
```

```
▶ docker container run ubuntu
```

```
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
8ec398bc0356: Pull complete
354e6904d655: Pull complete
36412f6b2f6e: Pull complete
Digest: sha256:769018135ba22d3a7a2b91cb89b8de711562cdf51ad6621b2b9b13e95f3798de
Status: Downloaded newer image for httpd:latest
36a391532e10d45f772f2c9430c2cc38dad4b441aa7a1c44d459f6fa3d78c6b6
```



Container Run – Create and Start a container

```
▶ docker container run ubuntu
```

```
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
2746a4a261c9: Pull complete
4c1d20cdee96: Pull complete
0d3160e1d0de: Pull complete
c8e37668deea: Pull complete
Digest: sha256:250cc6f3f3ffc5cdaa9d8f4946ac79821aafb4d3afc93928f0de9336eba21aa4
Status: Downloaded newer image for ubuntu:latest
```

```
▶ docker container ls
```

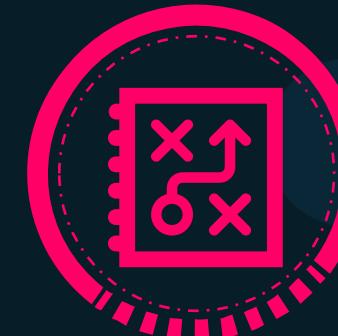
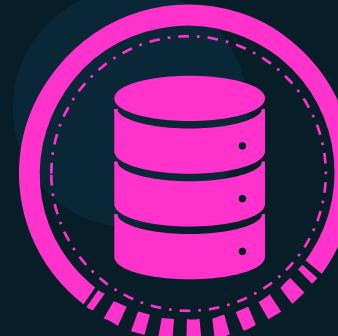
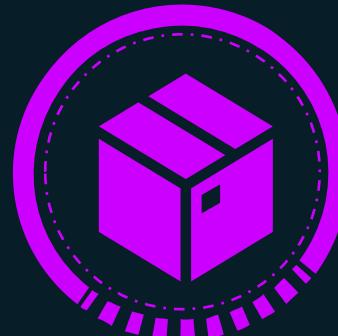
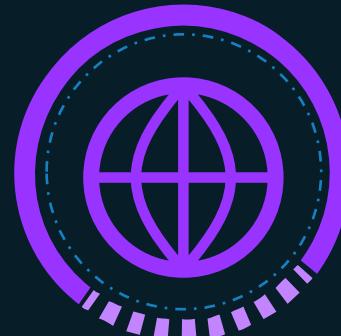
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

```
▶ docker container ls -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
d969ecdb44ea	ubuntu	"/bin/bash"	2 minutes ago	Exited (0) 2 minutes ago		intelligent_almeida



Container Run – Create and Start a container



```
▶ docker container run ubuntu
```

```
▶ docker container ls -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
d969ecdb44ea	ubuntu	"/bin/bash"	2 minutes ago	Exited (0) 2 minutes ago		intelligent_almeida



Container Run – With Options

```
▶ docker container run -it ubuntu
```

```
root@6caba272c8f5:/#
root@6caba272c8f5:/# hostname
6caba272c8f5
root@6caba272c8f5:/#
```

```
▶ docker container ls
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
6caba272c8f5	ubuntu	"/bin/bash"	About a minute ago	Up About a minute		quizzical_austin

```
▶ docker container run -it ubuntu
```



```
▶ docker container run ubuntu -it
```



Container Run – exiting a running process

```
▶ docker container run -it ubuntu
```

```
root@6caba272c8f5:/#
root@6caba272c8f5:/# hostname
6caba272c8f5
root@6caba272c8f5:/# exit
exit
```

```
▶ docker container ls -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
6caba272c8f5	ubuntu	"/bin/bash"	8 minutes ago	Exited (0) 37 seconds ago		quizzical_austin



Container Run – Container Name

```
▶ docker container ls -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
6caba272c8f5	ubuntu	"/bin/bash"	8 minutes ago	Exited (0) 37 seconds ago		quizzical_austin

```
▶ docker container run -itd --name=webapp ubuntu
```

```
59aa5eacd88c42970754cd6005ce315944a2efcd32288df998b29267ae54c152
```

```
▶ docker container ls -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
59aa5eacd88c	ubuntu	"/bin/bash"	20 seconds ago	Up 19 seconds		webapp

```
▶ docker container rename webapp custom-webapp
```

```
▶ docker container ls -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
59aa5eacd88c	ubuntu	"/bin/bash"	About a minute ago	Up About a minute		custom-webapp

Container Run – Detached Mode

```
▶ docker container run httpd
```

```
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Set the 'ServerName' directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Set the 'ServerName' directive globally to suppress this message
[Thu Sep 17 15:39:31.138134 2020] [mpm_event:notice] [pid 1:tid 139893041316992] AH00489: Apache/2.4.46 (Unix) configured -- resuming normal operations
[Thu Sep 17 15:39:31.138584 2020] [core:notice] [pid 1:tid 139893041316992] AH00094: Command line: 'httpd -D FOREGROUND'
```

```
▶ docker container run -d httpd
```

```
11cbd7fe7e65a9da453e159ed0fe163592dccc8a7845abc91b8305c78f50ac70
```

```
▶ docker container attach 11cb
```





KODEKLOUD

Interacting with a Container

Container Run – Escape Sequence

```
▶ docker container run -it ubuntu
```

```
root@6caba272c8f5:/# exit  
exit
```

```
▶ docker container ls -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
6caba272c8f5	ubuntu	"/bin/bash"	8 minutes ago	Exited (0) 37 seconds ago		quizzical_austin

```
▶ docker container run -it ubuntu
```

```
root@b71f15d33b60:/# [PRESS CTRL+p+q]
```

```
▶ docker container ls -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
b71f15d33b60	ubuntu	"/bin/bash"	3 minutes ago	Up 3 minutes		magical_babbage

Container Exec – Executing Commands

```
▶ docker container exec b71f15d33b60 hostname  
b71f15d33b60
```

```
▶ docker container exec -it b71f15d33b60 /bin/bash  
root@b71f15d33b60:#  
root@b71f15d33b60:# ps -ef  
UID      PID  PPID  C STIME TTY          TIME CMD  
root        1      0  0 12:53 pts/0    00:00:00 /bin/bash  
root       86      1  0 13:10 pts/0    00:00:00 ps -ef  
root@b71f15d33b60:# tty  
/dev/pts/0  
root@b71f15d33b60:# exit  
exit
```

```
▶ docker container attach b71f15d33b60  
root@b71f15d33b60:#
```





KODEKLOUD

Inspecting a Container

Container Inspect

```
▶ docker container inspect webapp
```

```
[  
 {  
   "Id": "59aa5eacd88c42970754cd6005ce315944a2efcd32288df998b29267ae54c152",  
   "Created": "2020-01-14T13:23:01.225868339Z",  
   "Path": "/bin/bash",  
   "Args": [],  
   "State": {  
     "Status": "running",  
     "Running": true,  
     "Paused": false,  
     "Restarting": false,  
     ...  
     "IPAddress": "172.17.0.5",  
     "IPPrefixLen": 16,  
     "IPv6Gateway": "",  
     "GlobalIPv6Address": "",  
     "GlobalIPv6PrefixLen": 0,  
     "MacAddress": "02:42:ac:11:00:05",  
     "DriverOpts": null  
   }  
 }  
 ]
```

Container Stats

```
▶ docker container stats
```

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK
PIDS						
59aa5eacd88c	webapp	50.00%	400KiB / 989.4MiB	0.04%	656B / 0B	0B / 0B
a00b5535783d	epic_leavitt	0.00%	404KiB / 989.4MiB	0.04%	656B / 0B	0B / 0B
616f80b0f026	elegant_cohen	0.00%	404KiB / 989.4MiB	0.04%	656B / 0B	0B / 0B
36a391532e10	charming_wiles	0.01%	8.363MiB / 989.4MiB	0.85%	656B / 0B	0B / 0B



Container Top

```
▶ docker container top webapp
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	17001	16985	0	13:23	?	00:00:00	stress



Container Logs

```
▶ docker container logs logtest
```

```
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.6. Set the 'ServerName' directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.6. Set the 'ServerName' directive globally to suppress this message
[Tue Jan 14 13:38:15.699310 2020] [mpm_event:notice] [pid 1:tid 140610463122560] AH00489: Apache/2.4.41 (Unix) configured -- resuming normal operations
[Tue Jan 14 13:38:15.699520 2020] [core:notice] [pid 1:tid 140610463122560] AH00094: Command line: 'httpd -D FOREGROUND'
```

```
▶ docker container logs -f logtest
```



Docker System Events

```
▶ docker container start webapp
```

```
webapp
```

```
▶ docker system events --since 60m
```

```
2020-01-14T18:30:30.423389441Z network connect d349c5984e7eebab74db57b8529df40e11a140f98a6b5e3ee1807aaeafa0e684  
(container=68649c8b359f89db7a3866ee0ebcc7261c0cb9697f3a624cd314c8f4f652f84b, name=bridge, type=bridge)  
2020-01-14T18:30:30.721669156Z container start 68649c8b359f89db7a3866ee0ebcc7261c0cb9697f3a624cd314c8f4f652f84b (image=ubuntu, name=casethree)  
2020-01-14T18:40:46.779320656Z network connect d349c5984e7eebab74db57b8529df40e11a140f98a6b5e3ee1807aaeafa0e684  
(container=71c90a19b9876c9ce2eb9d035355a062fdaceed4a714b61ddf0612651d47d3e2, name=bridge, type=bridge)  
2020-01-14T18:40:47.076482525Z container start 71c90a19b9876c9ce2eb9d035355a062fdaceed4a714b61ddf0612651d47d3e2 (image=ubuntu, name=webapp)
```





KODEKLOUD

Stopping & Removing Container

Linux Signals

▶ httpd

▶ kill -SIGSTOP \$(pgrep httpd)



Linux Signals

```
▶ httpd
```

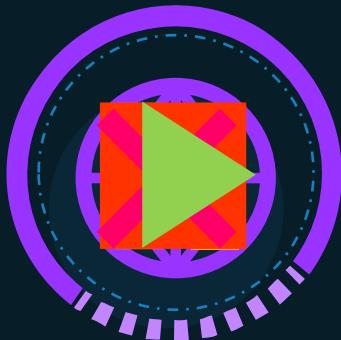
```
▶ kill -SIGSTOP $(pgrep httpd)
```

```
▶ kill -SIGCONT $(pgrep httpd)
```

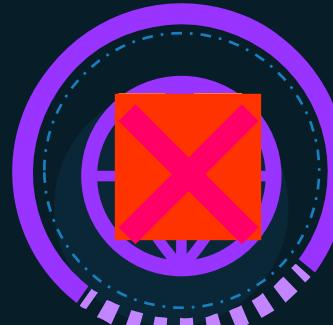
```
▶ kill -SIGTERM $(pgrep httpd)
```

```
▶ kill -SIGKILL $(pgrep httpd)
```

```
▶ kill -9 $(pgrep httpd)
```



Linux Signals



▶ httpd

▶ kill -**SIGSTOP** \$(pgrep httpd)

▶ kill -**SIGCONT** \$(pgrep httpd)

▶ kill -**SIGTERM** \$(pgrep httpd)

▶ kill -**SIGKILL** \$(pgrep httpd)

▶ kill -**9** \$(pgrep httpd)

▶ docker container **run** --name web httpd

▶ docker container **pause** web

freezer cgroup

▶ docker container **unpause** web

▶ docker container **stop** web

▶ docker container **kill** --signal=9 web



Removing a container

```
▶ docker container stop web
```

```
web
```

```
▶ ls -lrt /var/lib/docker/containers/
```

```
36a391532e10d45f772f2c9430c2cc38dad4b441aa7a1c44d459f6fa3d78c6b6
```

```
▶ docker container rm web
```

```
web
```

Error response from daemon: You cannot remove a running container

36c57f29b607460fc53dace758dac47afbf8cb698694d2fcfcbb0ab43a74f0d90. Stop the container before attempting removal or force remove

```
▶ ls -lrt /var/lib/docker/containers/
```



Remove All Container

```
▶ docker container ls -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
59aa5eacd88c	ubuntu	"/bin/bash"	23 minutes ago	Up 23 minutes		kodekloudagain
a00b5535783d	ubuntu	"/bin/bash"	25 minutes ago	Up 25 minutes		epic_leavitt
616f80b0f026	ubuntu	"/bin/bash"	31 minutes ago	Up 28 minutes		elegant_cohen
36a391532e10	httpd	"httpd-foreground"	About an hour ago	Up About an hour	80/tcp	charming_wiles

```
▶ docker container ls -q
```

```
59aa5eacd88c
a00b5535783d
616f80b0f026
36a391532e10
```

```
▶ docker container stop $(docker container ls -q)
```

```
59aa5eacd88c
a00b5535783d
616f80b0f026
36a391532e10
```

```
▶ docker container rm $(docker container ls -aq)
```

```
59aa5eacd88c
a00b5535783d
616f80b0f026
36a391532e10
```



Container Prune

```
▶ docker container ls -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
59aa5eacd88c	ubuntu	"/bin/bash"	23 minutes ago	Up 23 minutes		kodekloudagain
a00b5535783d	ubuntu	"/bin/bash"	25 minutes ago	Up 25 minutes		epic_leavitt
616f80b0f026	ubuntu	"/bin/bash"	31 minutes ago	Up 28 minutes		elegant_cohen
36a391532e10	httpd	"httpd-foreground"	About an hour ago	Up About an hour	80/tcp	charming_wiles

```
▶ docker container ls -q
```

```
59aa5eacd88c
a00b5535783d
616f80b0f026
36a391532e10
```

```
▶ docker container stop $(docker container ls -q)
```

```
59aa5eacd88c
a00b5535783d
616f80b0f026
36a391532e10
```

```
▶ docker container prune
```

WARNING! This will remove all stopped containers.

Are you sure you want to continue? [y/N] y

Deleted Containers:

```
59aa5eacd88c
a00b5535783d
616f80b0f026
36a391532e10
```

Total reclaimed space: 1223423

Remove Flag

```
▶ docker container run --rm ubuntu expr 4 + 5
```

9





Container Hostname

Container Hostname

```
▶ docker container run -it --name=webapp ubuntu
```

```
root@3484d738:/# hostname  
3484d738
```

```
▶ docker container run -it --name=webapp --hostname=webapp ubuntu
```

```
root@webapp :/# hostname  
webapp
```





KODEKLOUD

Restart Policy

Container – Restart Policies

NO

ON-FAILURE

ALWAYS

UNLESS STOPPED

```
▶ docker container run ubuntu expr 3 + 5  
ubuntu "expr 3 + 5" Exited (0) 11 seconds ago
```



```
▶ docker container run ubuntu expr three + 5  
ubuntu "expr three + 5" Exited (1) 2 seconds ago
```



```
▶ docker container stop httpd  
httpd "httpd-foreground" Exited (0) 4 days ago
```



```
▶ docker container run --restart=unless-stopped ubuntu
```



Live Restore

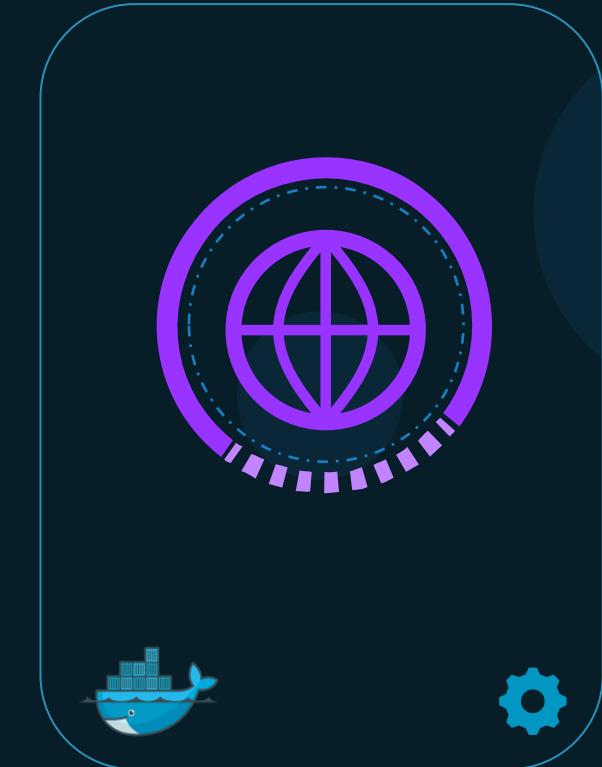
```
▶ docker container run --name web httpd
```

```
▶ systemctl stop docker
```

```
▶ systemctl start docker
```

```
▶ docker container run --name web httpd
```

```
▶ systemctl stop docker
```



```
/etc/docker/daemon.json
```

```
{  
  "debug": true,  
  "hosts": ["tcp://192.168.1.10:2376"],  
  "live-restore": true  
}
```

ODU



KODEKLOUD

Copy Files

Container cp – From Host to Container

SRC_PATH

DEST_PATH

▶ docker container cp /tmp/web.conf webapp:/etc/web.conf

▶ docker container cp webapp:/root/dockerhost /tmp/

▶ docker container cp /tmp/web.conf webapp:/etc/ 

▶ docker container cp /tmp/web.conf webapp:/etccc/ 

▶ docker container cp /tmp/app/ webapp:/opt/app



/etc/web.conf

Container - webapp



/tmp/web.conf





P u b l i s h i n g P o r t s

Run – PORT mapping

```
▶ docker run kodekloud/simple-webapp
```

* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)

http://172.17.0.2:5000

Internal IP

```
▶ docker run -p 80:5000 kodekloud/simple-webapp
```

```
▶ docker run -p 8000:5000 kodekloud/simple-webapp
```

```
▶ docker run -p 8001:5000 kodekloud/simple-webapp
```

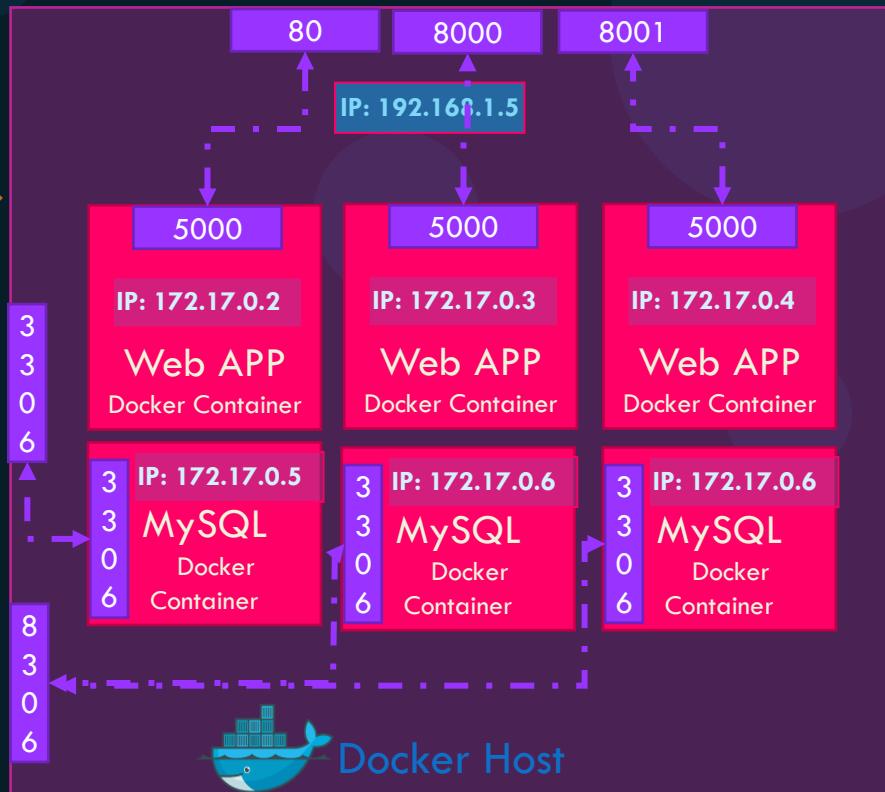
```
▶ docker run -p 3306:3306 mysql
```

```
▶ docker run -p 8306:3306 mysql
```

```
▶ docker run -p 8306:3306 mysql
```



http://192.168.1.5:80



Container PORT Publish

```
▶ docker run -p 8000:5000 kodekloud/simple-webapp
```

```
▶ docker run -p 192.168.1.5:8000:5000 kodekloud/simple-webapp
```

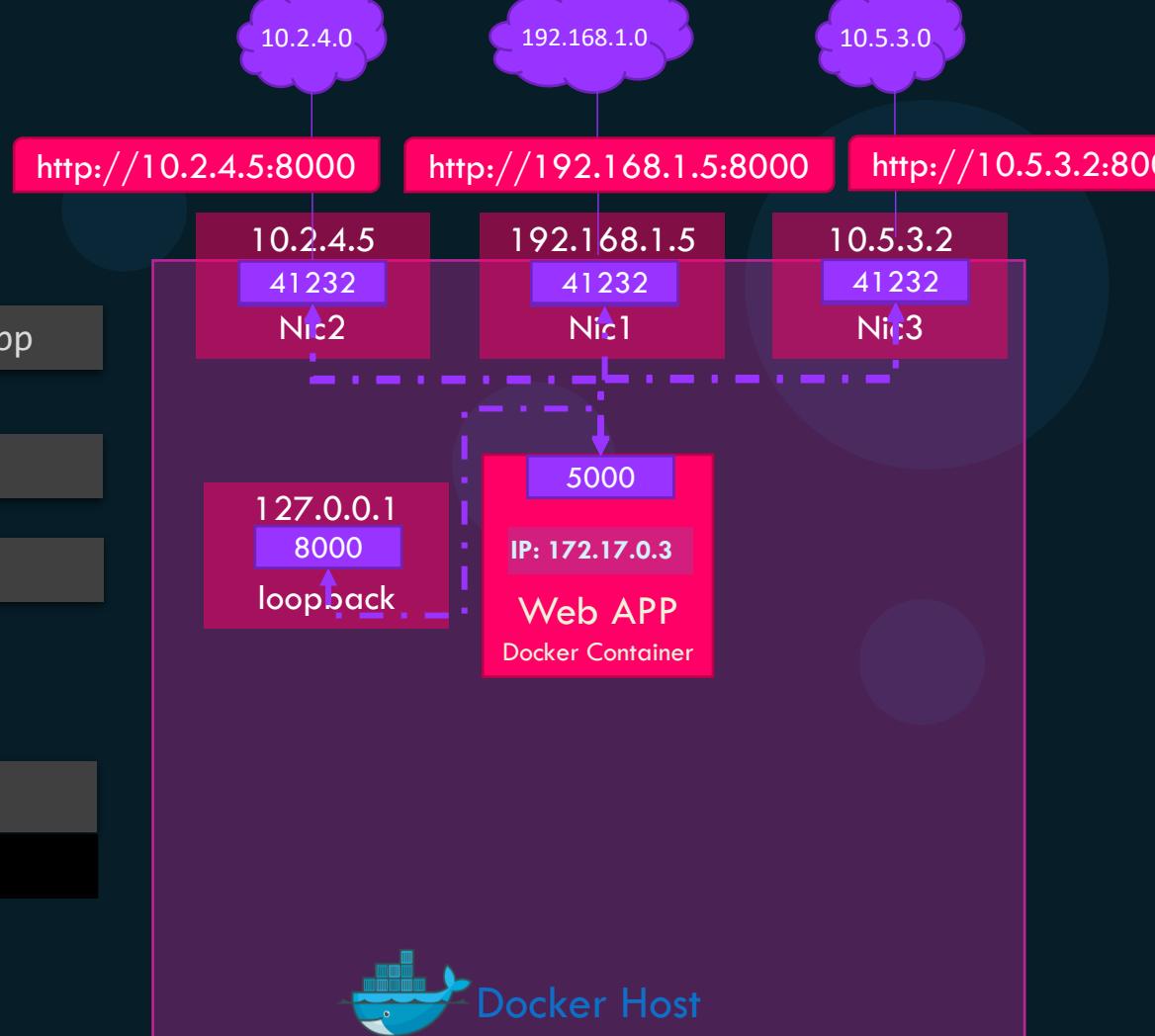
```
▶ docker run -p 127.0.0.1:8000:5000 kodekloud/simple-webapp
```

```
▶ docker run -p 5000 kodekloud/simple-webapp
```

Ephemeral Port Range => 32768 - 60999

```
▶ cat /proc/sys/net/ipv4/ip_local_port_range
```

32768 60999



Container PORT Publish

```
▶ docker run -P kodekloud/simple-webapp
```

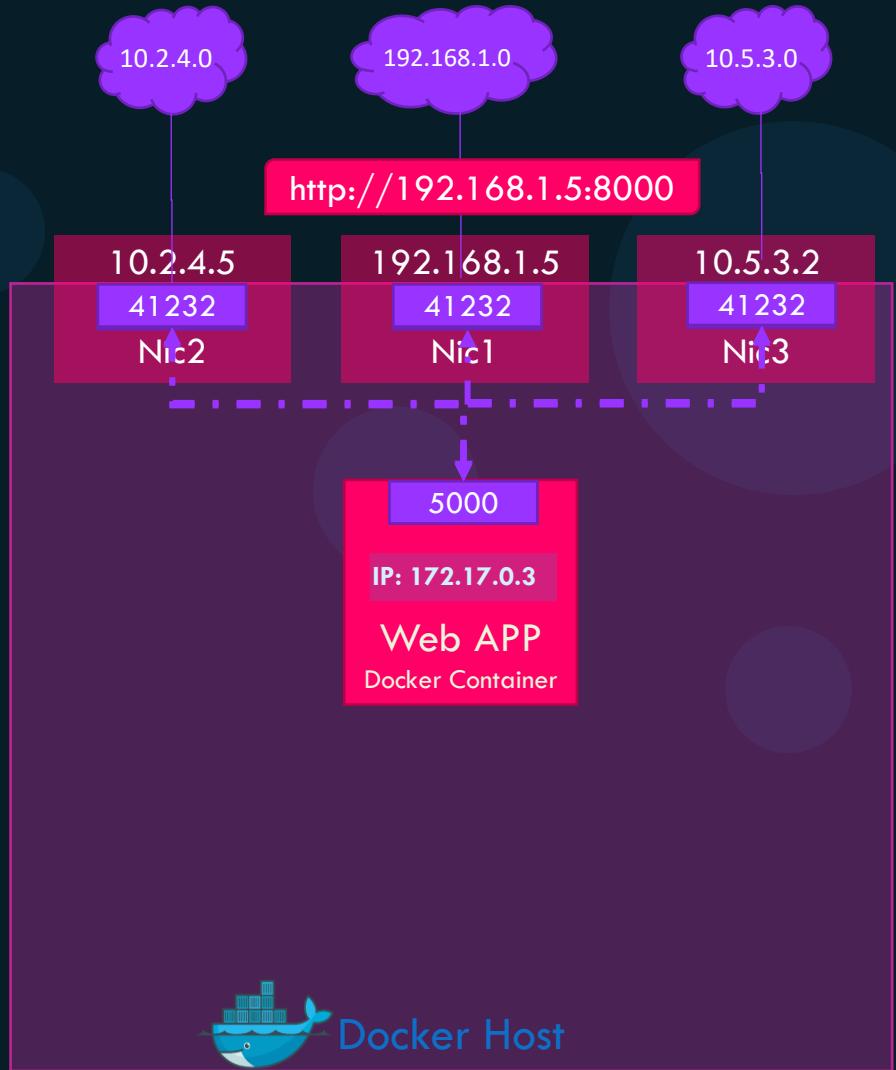
Dockerfile

```
FROM ubuntu:16.04
RUN apt-get update && apt-get install -y python python-pip
RUN pip install flask
COPY app.py /opt/
ENTRYPOINT flask run
EXPOSE 5000
```

```
▶ docker run -P --expose=8080 kodekloud/simple-webapp
```

```
▶ docker inspect kodekloud/simple-webapp
```

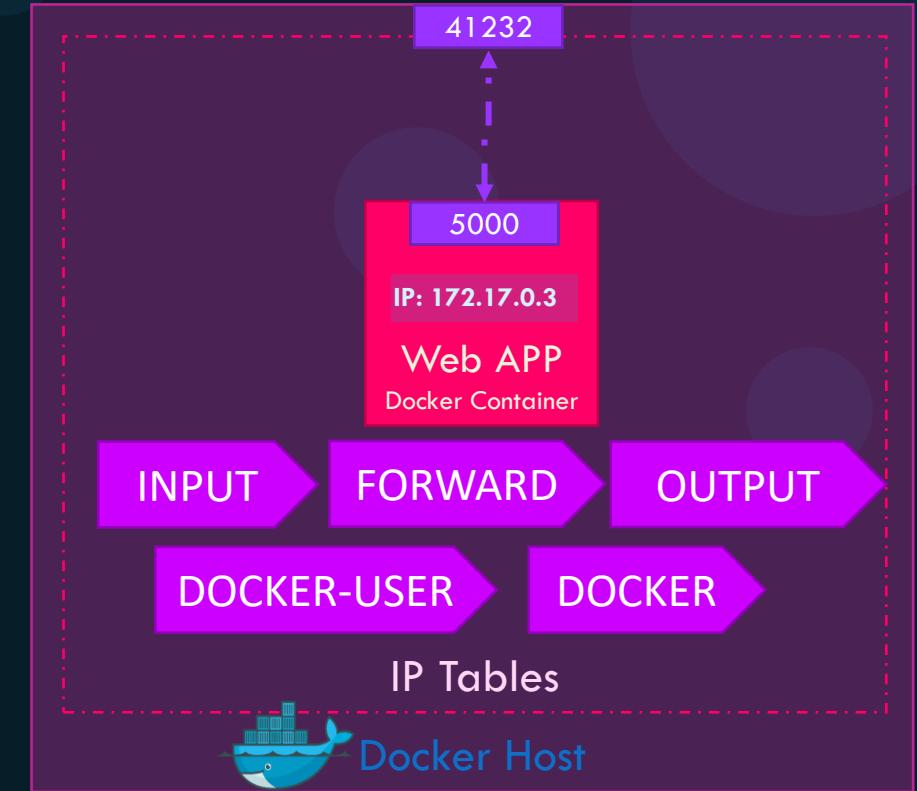
```
"ExposedPorts": {
    "5000/tcp": {},
    "8080/tcp": {}
},
```



IP Tables

```
▶ iptables -t nat -S DOCKER
```

```
-N DOCKER  
-A DOCKER ! -i docker0 -p tcp -m tcp --dport 41232 -j DNAT --to-destination 172.17.0.3:5000
```



References

<https://docs.docker.com/network/links/>

<https://docs.docker.com/engine/reference/run/#expose-incoming-ports>

<https://docs.docker.com/config/containers/container-networking/>

<https://docs.docker.com/network/iptables/>





Troubleshoot Docker Daemon

Check Service Status

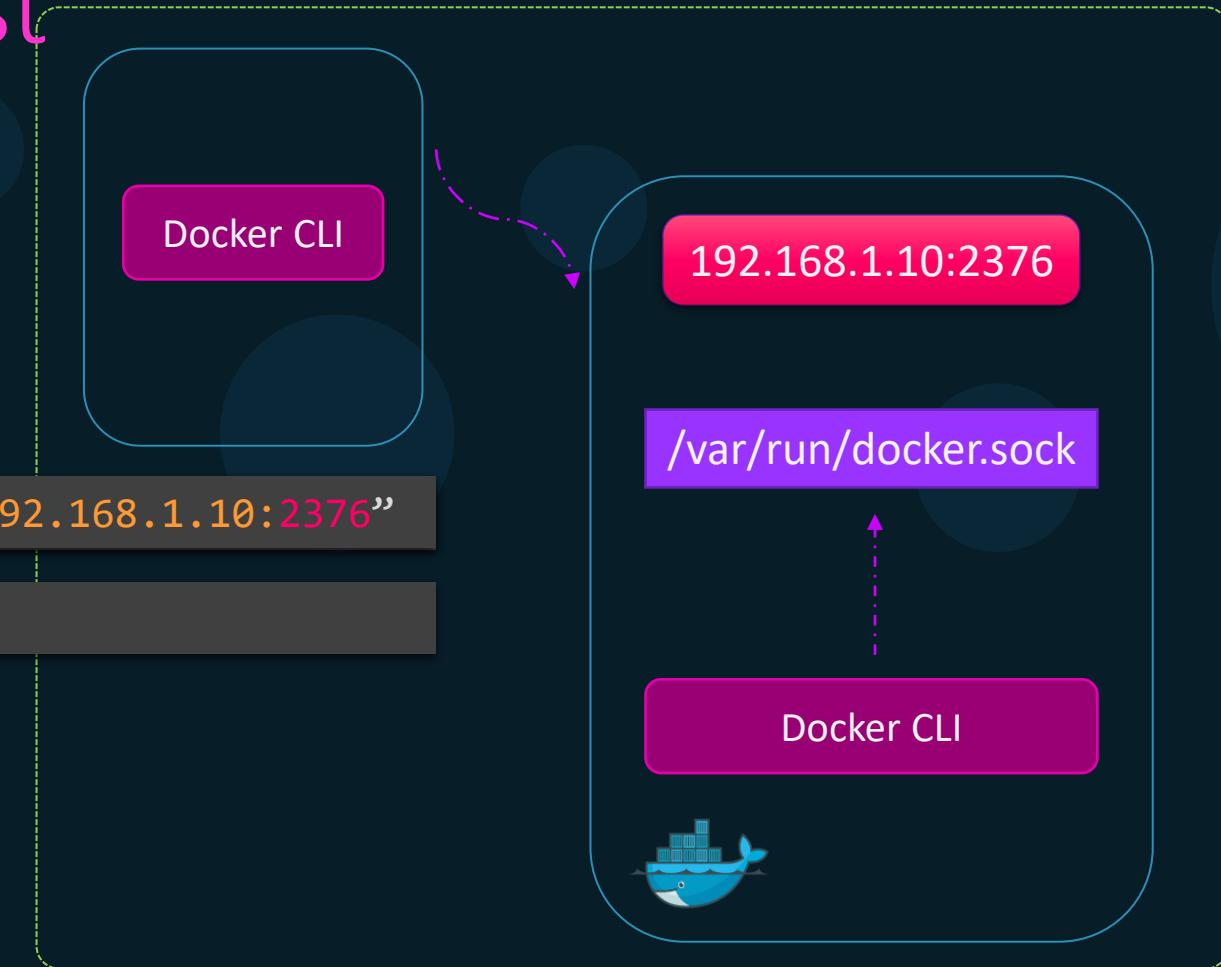
```
▶ docker ps
```

```
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?
```



Check Docker Host

```
▶ export DOCKER_HOST="tcp://192.168.1.10:2376"  
▶ docker ps
```



2375	Un-encrypted
2376	Encrypted



Check Service Status

```
▶ systemctl start docker
```

```
▶ systemctl status docker
```

```
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Sat 2020-10-24 07:42:08 UTC; 21s ago
     Docs: https://docs.docker.com
   Process: 4197 ExecStart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0 --containerd=/run/containerd/containerd.sock
           (code=exited, Main PID: 4197 (code=exited, status=0/SUCCESS)
```



View Logs

▶ journalctl -u docker.service

```
-- Logs begin at Wed 2020-10-21 04:05:39 UTC, end at Sat 2020-10-24 07:41:39 UTC. --
Oct 21 04:05:42 ubuntu-xenial systemd[1]: Starting Docker Application Container Engine...
Oct 21 04:05:42 time="2020-10-21T04:05:42.565473329Z" level=info msg="parsed scheme: \"unix\""
Oct 21 04:05:42 time="2020-10-21T04:05:42.565496428Z" level=info msg="scheme \"unix\" not register
Oct 21 04:05:42 time="2020-10-21T04:05:42.565554302Z" level=info msg="ccResolverWrapper: sending u
Oct 21 04:05:42 time="2020-10-21T04:05:42.565673967Z" level=info msg="ClientConn switching balance
Oct 21 04:05:42 time="2020-10-21T04:05:42.570967241Z" level=info msg="parsed scheme: \"unix\""
Oct 21 04:05:42 time="2020-10-21T04:05:42.570982918Z" level=info msg="scheme \"unix\" not register
Oct 21 04:05:42 time="2020-10-21T04:05:42.571027208Z" level=info msg="ccResolverWrapper: sending u
Oct 21 04:05:42 time="2020-10-21T04:05:42.571037442Z" level=info msg="ClientConn switching balance
Oct 21 04:05:42 time="2020-10-21T04:05:42.629609680Z" level=info msg="[graphdriver] using prior st
Oct 21 04:05:42 time="2020-10-21T04:05:42.847722164Z" level=warning msg="Your kernel does not supp
Oct 21 04:05:42 time="2020-10-21T04:05:42.847808687Z" level=warning msg="Your kernel does not supp
Oct 21 04:05:42 time="2020-10-21T04:05:42.847816072Z" level=warning msg="Your kernel does not supp
Oct 21 04:05:42 time="2020-10-21T04:05:42.848125012Z" level=info msg="Loading containers: start."
Oct 21 04:05:43 time="2020-10-21T04:05:43.610553801Z" level=info msg="Removing stale sandbox ae1f6
Oct 21 04:05:43 time="2020-10-21T04:05:43.618004459Z" level=warning msg="Error (Unable to complete
Oct 21 04:05:43 time="2020-10-21T04:05:43.865861594Z" level=info msg="Removing stale sandbox c1138
Oct 21 04:05:43 time="2020-10-21T04:05:43.872335497Z" level=warning msg="Error (Unable to complete
Oct 21 04:05:44 time="2020-10-21T04:05:44.135363994Z" level=info msg="Removing stale sandbox ingre
Oct 21 04:05:44 time="2020-10-21T04:05:44.136680822Z" level=warning msg="Error (Unable to complete
```

Daemon Configuration File

```
/etc/docker/daemon.json
```

```
{  
  "debug": true,  
  "hosts": ["tcp://192.168.1.10:2376"]  
  "tls": true,  
  "tlscert": "/var/docker/server.pem",  
  "tlskey":  "/var/docker/serverkey.pem"  
}
```

unable to configure the Docker daemon with file /etc/docker/daemon.json: the following directives are specified both as a flag and in the configuration file: debug: (from flag: true, from file: false)



Free Disk Space on Host

▶ df -h

Filesystem	Size	Used	Avail	Use%	Mounted on
dev	364M	0	364M	0%	/dev
run	369M	340K	369M	1%	/run
/dev/sda1	19G	14.7G	15M	99%	/
tmpfs	369M	0	369M	0%	/dev/shm
tmpfs	369M	0	369M	0%	/sys/fs/cgroup
tmpfs	369M	4.0K	369M	1%	/tmp
tmpfs	74M	0	74M	0%	/run/user/0

▶ docker container prune

▶ docker image prune



Debug in Docker

```
▶ docker system info
```

```
Client:  
  Debug Mode: false
```

```
Server:  
  Containers: 0  
    Running: 0  
    Paused: 0  
    Stopped: 0
```

```
  Images: 0  
  Server Version: 19.03.5  
  Storage Driver: overlay2  
    Backing Filesystem: xfs
```

```
.
```

```
.
```

```
.
```

```
  Experimental: false  
  Insecure Registries:  
    127.0.0.0/8  
  Live Restore Enabled: false
```



References

<https://docs.docker.com/config/daemon/>

<https://docs.docker.com/engine/reference/commandline/dockerd/>





Logging Drivers

Logging Drivers

```
▶ docker run -d --name nginx nginx
```

```
▶ docker logs nginx
```

```
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
```

```
▶ docker system info
```

Server:

...

Images: 54

Server Version: 19.03.6

...

Logging Driver: json-file

Cgroup Driver: cgroupfs

Plugins:

Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog

...

Logging Drivers

```
▶ docker ps
```

f3997637c0df	nginx	"/docker-entrypoint...."	37 minutes ago	Up 37	nginx
--------------	-------	--------------------------	----------------	-------	-------

```
▶ cd /var/lib/docker/containers; ls
```

38781779e9aa15c190746784ba23d1ae237f03b58e0479286259e275d4c8820a
c5ab1dba9b51486e0e69386c137542be2e4315a56b4ee07c825e2d41c99f89b4
f3997637c0df66becf4dd4662d3c172bf16f916a3b9289b95f0994675102de17

```
▶ cat f3997637c0df66becf4dd4662d3c172bf16f916a3b9289b95f0994675102de17.json
```

```
{"log":"/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration\n","stream":"stdout","time":"2020-10-25T05:59:43.832656488Z"}  
{"log":"/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/\n","stream":"stdout","time":"2020-10-25T05:59:43.832891838Z"}  
{"log":"/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh\n","stream":"stdout","time":"2020-10-25T05:59:43.833987067Z"}  
{"log":"/10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/default.conf\n","stream":"stdout","time":"2020-10-25T05:59:43.83695198Z"}  
{"log":"/10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf\n","stream":"stdout","time":"2020-10-25T05:59:43.84592186Z"}  
{"log":"/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh\n","stream":"stdout","time":"2020-10-25T05:59:43.846117966Z"}  
{"log":"/docker-entrypoint.sh: Configuration complete; ready for start up\n","stream":"stdout","time":"2020-10-25T05:59:43.850840102Z"}
```

Logging Drivers

```
▶ docker system info
```

```
Server:  
...  
Images: 54  
Server Version: 19.03.6  
...  
Logging Driver: json-file  
Cgroup Driver: cgroupfs  
Plugins:  
  Log: awslogs fluentd gcplogs gelf journald json-file local  
    logentries splunk syslog  
...
```

```
/etc/docker/daemon.json
```

```
{  
  "debug": true,  
  "hosts": ["tcp://192.168.1.10:2376"]  
  "tls": true,  
  "tlscert": "/var/docker/server.pem",  
  "tlskey": "/var/docker/serverkey.pem",  
  "log-driver": "awslogs"  
}
```



Logging Driver - Options

```
▶ docker system info
```

```
Server:  
...  
Images: 54  
Server Version: 19.03.6  
...  
Logging Driver: json-file  
Cgroup Driver: cgroupfs  
Plugins:  
  Log: awslogs fluentd gcplogs gelf journald json-file local  
    logentries splunk syslog  
...
```

```
/etc/docker/daemon.json
```

```
{  
  "debug": true,  
  "hosts": ["tcp://192.168.1.10:2376"]  
  "tls": true,  
  "tlscert": "/var/docker/server.pem",  
  "tlskey": "/var/docker/serverkey.pem",  
  "log-driver": "awslogs",  
  "log-opt": {  
    "awslogs-region": "us-east-1"  
  }  
}
```

```
export AWS_ACCESS_KEY_ID=<>  
export AWS_SECRET_ACCESS_KEY=<>  
export AWS_SESSION_TOKEN=<>
```



Logging Drivers

```
▶ docker run -d --log-driver json-file nginx
```

```
▶ docker container inspect nginx
```

```
[  
 {  
   "Id": "f3997637c0df66becf4dd4662d3c172bf16f916a3b9289b95f0994675102de17",  
   "Created": "2020-10-25T05:59:43.543296741Z",  
   "Path": "/docker-entrypoint.sh",  
   ...  
   "HostConfig": {  
     "Binds": null,  
     "ContainerIDFile": "",  
     "LogConfig": {  
       "Type": "json-file",  
       "Config": {}  
     },  
   },  
 }]
```

```
▶ docker container inspect -f '{{.HostConfig.LogConfig.Type}}' nginx
```

```
json-file
```



KODEKLOUD

Docker Images



Image Registry

The screenshot shows the Docker Hub homepage. On the left, there's a large blue banner with white text: "Build and Ship any Application Anywhere" and "Docker Hub is the world's easiest way to create, manage, and deliver your teams' container applications." At the top, there's a search bar with placeholder text "Search for great content (e.g., mysql)", and navigation links for "Explore", "Pricing", "Sign In", and a prominent "Sign Up" button. The main area features a "Sign Up Today" form with fields for "Docker ID", "Email", and "Password". It includes a checkbox for product updates, a reCAPTCHA verification, and a "Sign Up" button at the bottom. Below the form, small text links to "Terms of Service", "Privacy Policy", and "Data Processing Terms".

-  **Docker Trusted Registry**
-  **Google Container Registry**
-  **Amazon Container Registry**
-  **Azure Container Registry**



Image Registry

Official Images

Official Image

traefik  Updated 4 hours ago

Traefik, The Cloud Native Edge Router

OFFICIAL IMAGE 

10M+ Downloads 1.6K Stars

Container Windows Linux ARM 64 ARM x86-64

Application Infrastructure

postgres  Updated 4 hours ago

The PostgreSQL object-relational database system provides r...

OFFICIAL IMAGE 

10M+ Downloads 8.4K Stars

Container Linux PowerPC 64 LE 386 x86-64

mips64le ARM 64 ARM IBM Z Databases

mongo  Updated 4 hours ago

MongoDB document databases provide high availability and ...

OFFICIAL IMAGE 

10M+ Downloads 7.2K Stars

Container Linux Windows IBM Z ARM 64 x86-64

Databases

Verified Images

MySQL Server Enterprise Edition  VERIFIED PUBLISHER 

By Oracle • Updated 2 years ago 0 Stars

The world's most popular open source database system

Container Docker Certified Linux x86-64 Databases

Oracle Instant Client  VERIFIED PUBLISHER 

By Oracle • Updated 3 years ago 0 Stars

Oracle Database 12c Instant Client

Container Docker Certified Linux x86-64 Databases

Senzing Package Installer  VERIFIED PUBLISHER 

By Senzing, • Updated a year ago 0 Stars

Install the Senzing API onto mounted volumes.

Container Docker Certified Linux x86-64 Analytics

Splunk Universal Forwarder  VERIFIED PUBLISHER 

By Splunk • Updated a year ago 0 Stars

Collect data and send it to your Splunk instance.

Container Docker Certified Linux IBM Z x86-64

User Images

dlwrl207/keyword_task  By [dlwrl207](#) • Updated 3 hours ago 39 Downloads 0 Stars

Container Linux x86-64

projectopenubl/xsender-server  By [projectopenubl](#) • Updated 3 hours ago 136 Downloads 0 Stars

Container Linux x86-64

nautilusdeployedk/nautilus  By [nautilusdeployedk](#) • Updated 3 hours ago 154 Downloads 0 Stars

Container Linux x86-64

td0m/soton_cloud_panel  By [td0m](#) • Updated 3 hours ago 177 Downloads 0 Stars

Container Linux arm64

Registry: Searching an image

The screenshot shows the Docker Registry interface. At the top, there is a search bar with the text "Ubuntu". Below the search bar, there are navigation links: "Explore", "Pricing", "Sign In", and a "Sign Up" button. The main navigation tabs are "Docker", "Containers" (which is currently selected), and "Plugins". On the left side, there are filters for "Docker Certified" (with an option to filter by "Docker Certified" checked) and "Images" (with options for "Verified Publisher" and "Official Images"). The search results show 1 - 25 of 86,941 results for "Ubuntu". The results are sorted by "Most Popular". The first result is the "ubuntu" image, which is marked as an "OFFICIAL IMAGE". It has "10M+" downloads and "10K+" stars. The image details page shows the Ubuntu logo, the name "ubuntu", the last update ("Updated 4 hours ago"), a description ("Ubuntu is a Debian-based Linux operating system based on free software."), and a list of supported architectures: Container, Linux, x86-64, IBM Z, ARM 64, PowerPC 64 LE, 386, ARM. Below these are categories: Base Images and Operating Systems.

1 - 25 of 86,941 results for **Ubuntu**. [Clear search](#)

Most Popular

Docker Certified i

Docker Certified

Images

Verified Publisher i
Docker Certified And Verified Publisher Content

Official Images i
Official Images Published By Docker

ubuntu

Updated 4 hours ago

OFFICIAL IMAGE star

10M+ 10K+
Downloads Stars

Ubuntu is a Debian-based Linux operating system based on free software.

Container Linux x86-64 IBM Z ARM 64 PowerPC 64 LE 386 ARM

Base Images Operating Systems



Image Tags

► docker run ubuntu

► docker run ubuntu:**latest**

► docker run ubuntu:**18.04**

► docker run ubuntu:**trusty**



The screenshot shows the Docker Hub page for the official Ubuntu image. At the top, there's a large orange Ubuntu logo. To its right, the text "ubuntu ☆" is displayed, followed by "Docker Official Images". Below this, a brief description states: "Ubuntu is a Debian-based Linux operating system based on free software." A download button labeled "1B+" is shown. Underneath, there are several tabs: "Container", "Linux", "x86-64", "IBM Z", "ARM 64", "PowerPC 64 LE", "386", "ARM", "Base Images", "Operating Systems", and "Official Image". On the far right, there's a dropdown menu set to "Linux - ARM 64 (latest)", a "Copy and paste to pull this image" button, a command line input field containing "docker pull ubuntu", and a "View Available Tags" link.

Quick reference

- **Maintained by:** Canonical and Tianon (Debian Developer)
- **Where to get help:** the Docker Community Forums, the Docker Community Slack, or Stack Overflow

Supported tags and respective Dockerfile links

- [18.04](#), [bionic-20200903](#), [bionic](#)
- [20.04](#), [focal-20200916](#), [focal](#), [\[latest\]](#), [rolling](#)
- [20.10](#), [groovy-20200911](#), [groovy](#), [devel](#)
- [14.04](#), [trusty-20191217](#), [trusty](#)
- [16.04](#), [xenial-20200903](#), [xenial](#)

Image list: List Local Available Images

```
▶ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu	latest	549b9b86cb8d	4 weeks ago	64.2MB



Image Search: Search without GUI

```
▶ docker search httpd
```

NAME	DESCRIPTION	STARS	OFFICIAL
AUTOMATED			
httpd	The Apache HTTP Server Project	2815	[OK]
centos/httpd-24-centos7	Platform for running Apache httpd 2.4 or bui...	29	
centos/httpd		26	
[OK]			
armhf/httpd	The Apache HTTP Server Project	8	
salim1983hoop/httpd24	Dockerfile running apache config	2	
[OK]			

```
▶ docker search httpd --limit 2
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
httpd	The Apache HTTP Server Project	2815	[OK]	
centos/httpd-24-centos7	Platform for running Apache httpd 2.4 or bui...	29		

```
▶ docker search --filter stars=10 httpd
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
httpd	The Apache HTTP Server Project	2815	[OK]	
centos/httpd-24-centos7	Platform for running Apache httpd 2.4 or bui...	29		
centos/httpd		26		[OK]

```
▶ docker search --filter stars=10 --filter is-official=true httpd
```

Image Pull: Download latest Image

```
▶ docker image pull httpd
```

```
Using default tag: latest
latest: Pulling from library/httpd
8ec398bc0356: Pull complete
354e6904d655: Pull complete
27298e4c749a: Pull complete
10e27104ba69: Pull complete
36412f6b2f6e: Pull complete
Digest: sha256:769018135ba22d3a7a2b91cb89b8de711562cdf51ad6621b2b9b13e95f3798de
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
```

```
▶ docker image list
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
httpd	latest	c2aa7e16edd8	2 weeks ago	165MB
ubuntu	latest	549b9b86cb8d	4 weeks ago	64.2MB





Image Addressing Convention

Image Addressing Convention

```
▶ docker image pull httpd
```



Image Addressing

image: docker.io/httpd/httpd

Registry User/ Image/
 Account Repository

gcr.io/httpd/httpd





KODEKLOUD

Authenticat ing to Registries

Public/Private Registry

```
▶ docker pull ubuntu
```

```
▶ docker pull gcr.io/organization/ubuntu
```

Using default tag: latest

Error response from daemon: **pull access denied for gcr.io/organization/ubuntu**, repository does not exist or may require 'docker login': **denied: requested access to the resource is denied**

```
▶ docker push ubuntu
```

The push refers to repository [docker.io/library/ubuntu]

128fa0b0fb81: Layer already exists

c0151ca45f27: Layer already exists

b2fd17df2071: Layer already exists

[DEPRECATION NOTICE] registry v2 schema1 support will be removed in an upcoming release. Please contact admins of the docker.io registry NOW to avoid future disruption. More information at

<https://docs.docker.com/registry/spec/deprecated-schema-v1/>

errors:

denied: requested access to the resource is denied

unauthorized: authentication required



Public/Private Registry

```
▶ docker login docker.io
```

Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to <https://hub.docker.com> to create one.

Username: registry-user

Password:

WARNING! Your password will be stored unencrypted in /home/vagrant/.docker/config.json.

Login Succeeded

```
▶ docker login gcr.io
```

Username: registry-user

Password:

WARNING! Your password will be stored unencrypted in /home/vagrant/.docker/config.json.

Login Succeeded

```
▶ docker image push httpd
```

The push refers to repository [gcr.io/kodekloud/httpd]

2f159baeafde: Mounted from library/httpd

6b27de954cca: Mounted from library/httpd

httpd: digest: sha256:9a5e7d690fd4ca39ccdc9e6d39e3dc0f96bf3acda096a2567374b4c608f6dacc size: 1362



Image Tag: Retagging an image locally

```
▶ docker image list
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
httpd	alpine	52862a02e4e9	2 weeks ago	112MB
httpd	latest	c2aa7e16edd8	2 weeks ago	165MB
ubuntu	latest	549b9b86cb8d	4 weeks ago	64.2MB

```
▶ docker image tag httpd:alpine httpd:customv1
```

```
▶ docker image list
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
httpd	alpine	52862a02e4e9	2 weeks ago	112MB
httpd	customv1	52862a02e4e9	2 weeks ago	112MB
httpd	latest	c2aa7e16edd8	2 weeks ago	165MB
ubuntu	latest	549b9b86cb8d	4 weeks ago	64.2MB

```
▶ docker image tag httpd:alpine gcr.io/company/httpd:customv1
```

```
▶ docker image push gcr.io/company/httpd:customv1
```



Objects Size

```
▶ docker image list
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
httpd	alpine	52862a02e4e9	2 weeks ago	112MB
httpd	customv1	52862a02e4e9	2 weeks ago	112MB
httpd	latest	c2aa7e16edd8	2 weeks ago	165MB
ubuntu	latest	549b9b86cb8d	4 weeks ago	64.2MB

```
▶ docker system df
```

TYPE	TOTAL	ACTIVE	SIZE	RECLAIMABLE
Images	3	0	341.9MB	341.9MB (100%)
Containers	0	0	0B	0B
Local Volumes	0	0	0B	0B
Build Cache	0	0	0B	0B







Remove Images

Image Rm: Removing an Image Locally

```
▶ docker image list
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
httpd	alpine	52862a02e4e9	2 weeks ago	112MB
httpd	customv1	52862a02e4e9	2 weeks ago	112MB
httpd	latest	c2aa7e16edd8	2 weeks ago	165MB
ubuntu	latest	549b9b86cb8d	4 weeks ago	64.2MB

Note: An image cannot be removed if a container is dependent on it. All containers must be removed and deleted first.

```
▶ docker image rm httpd:customv1
```

```
Untagged: httpd:customv1
```

```
▶ docker image rm httpd:alpine
```

```
untagged: httpd:alpine
deleted: sha256:549b9b86cb8d75a2b668c21c50ee092716d070f129fd1493f95ab7e43767eab8
deleted: sha256:7c52cdc1e32d67e3d5d9f83c95ebe18a58857e68bb6985b0381ebdcec73ff303
deleted: sha256:a3c2e83788e20188bb7d720f36ebeef2f111c7b939f1b19aa1b4756791beece0
deleted: sha256:61199b56f34827cbab596c63fd6e0ac0c448faa7e026e330994818190852d479
deleted: sha256:2dc9f76fb25b31e0ae9d36adce713364c682ba0d2fa70756486e5cedfaf40012
```



Image Prune: removing all unused image

```
▶ docker image prune -a
```

```
WARNING! This will remove all images without at least one container associated to them.
```

```
Are you sure you want to continue? [y/N] y
```

```
Deleted Images:
```

```
untagged: ubuntu:latest
```

```
untagged: ubuntu@sha256:250cc6f3fffc5cdaa9d8f4946ac79821aafb4d3afc93928f0de9336eba21aa4
```

```
deleted: sha256:549b9b86cb8d75a2b668c21c50ee092716d070f129fd1493f95ab7e43767eab8
```

```
deleted: sha256:7c52cdc1e32d67e3d5d9f83c95ebe18a58857e68bb6985b0381ebdcec73ff303
```

```
deleted: sha256:a3c2e83788e20188bb7d720f36ebeef2f111c7b939f1b19aa1b4756791beece0
```

```
deleted: sha256:61199b56f34827cbab596c63fd6e0ac0c448faa7e026e330994818190852d479
```

```
deleted: sha256:2dc9f76fb25b31e0ae9d36adce713364c682ba0d2fa70756486e5cedfaf40012
```

```
untagged: httpd:latest
```

```
untagged: httpd@sha256:769018135ba22d3a7a2b91cb89b8de711562cdf51ad6621b2b9b13e95f3798de
```

```
deleted: sha256:c2aa7e16edd855da8827aa0ccf976d1d50f0827c08622c16e0750aa1591717e5
```

```
deleted: sha256:9fa170034369c33a4c541b38ba11c63c317f308799a46e55da9bea5f9c378643
```

```
deleted: sha256:9a41b3deb4609bec368902692dec63e858e6cd85a1312ee1931d421f51b2a07c
```

```
deleted: sha256:ed10451b31dfca751aa8d3e4264cb08ead23d4f2b661324eca5ec72b0e7c59fa
```

```
deleted: sha256:06020df9067f8f2547f53867de8e489fed315d964c9f17990c3e5e6a29838d98
```

```
deleted: sha256:556c5fb0d91b726083a8ce42e2faaed99f11bc68d3f70e2c7bbce87e7e0b3e10
```

```
Total reclaimed space: 229.4MB
```





Inspect Image

Image Layers: display image layers

```
▶ docker image list
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
httpd	latest	c2aa7e16edd8	2 weeks ago	165MB
ubuntu	latest	549b9b86cb8d	4 weeks ago	64.2MB

```
▶ docker image history ubuntu
```

IMAGE	CREATED	CREATED BY	SIZE
COMMENT			
549b9b86cb8d	4 weeks ago	/bin/sh -c #(nop) CMD ["/bin/bash"]	0B
<missing>	4 weeks ago	/bin/sh -c mkdir -p /run/systemd && echo 'do...	7B
<missing>	4 weeks ago	/bin/sh -c set -xe && echo '#!/bin/sh' > /...	745B
<missing>	4 weeks ago	/bin/sh -c [-z "\$(apt-get indextargets)"]	987kB
<missing>	4 weeks ago	/bin/sh -c #(nop) ADD file:53f100793e6c0adfc...	63.2MB



Image inspect

```
▶ docker image inspect httpd
```

```
[  
  {  
    "Parent": "",  
    "Comment": "",  
    "Created": "2020-09-15T23:05:57.348340124Z",  
    "ContainerConfig": {  
      "ExposedPorts": {  
        "80/tcp": {}  
      }  
    },  
    "DockerVersion": "18.09.7",  
    "Author": "",  
    "Architecture": "amd64",  
    "Os": "linux",  
    "Size": 137532780,  
    "VirtualSize": 137532780,  
    "Metadata": {  
      "LastTagTime": "0001-01-01T00:00:00Z"  
    }  
  }  
]
```

Parent/Base Image

Exposed Ports

Author Details

Size

All Configs in Dockerfile



Image inspect - with format

```
▶ docker image inspect httpd -f '{{.Os}}'
```

```
linux
```

```
▶ docker image inspect httpd -f '{{.Architecture}}'
```

```
amd64
```

```
▶ docker image inspect httpd -f '{{.Architecture}} {{.Os}}'
```

```
amd64 linux
```

```
▶ docker image inspect httpd -f  
      '{{.ContainerConfig.ExposedPorts}}'
```

```
map[80/tcp:{}]
```

```
▶ docker image inspect httpd
```

```
[  
 {  
   "Parent": "",  
   "Comment": "",  
   "Created": "2020-09-15T23:05:57.348340124Z",  
   "ContainerConfig": {  
     "ExposedPorts": {  
       "80/tcp": {}  
     }  
   },  
   "DockerVersion": "18.09.7",  
   "Author": "",  
   "Architecture": "amd64",  
   "Os": "linux",  
   "Size": 137532780,  
   "VirtualSize": 137532780,  
   "Metadata": {  
     "LastTagTime": "0001-01-01T00:00:00Z"  
   }  
 }  
 ]
```





Save and Load

Image Save and Load

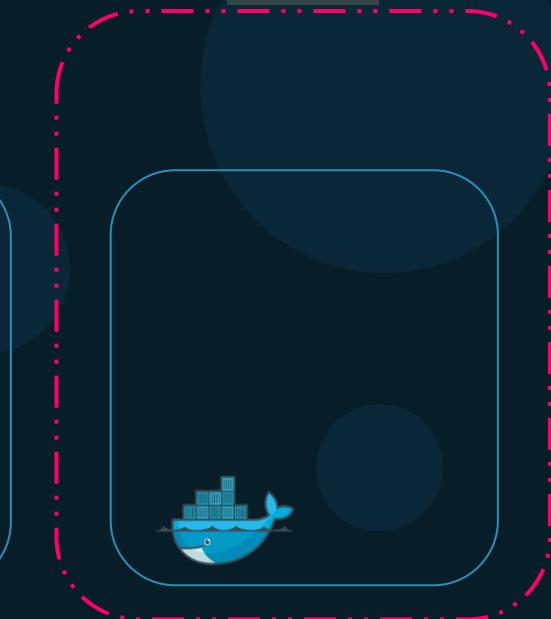
```
▶ docker image save alpine:latest -o alpine.tar
```

```
▶ docker image load -i alpine.tar
```

```
beee9f30bc1f: Loading layer [=====] 5.862MB/5.862MB
Loaded image: alpine:latest
```

```
▶ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
alpine	latest	a187dde48cd2	4 weeks ago	5.6MB



Import and Export Operations

```
▶ docker export <container-name> > testcontainer.tar
```

```
▶ docker image import testcontainer.tar newimage:latest
```

```
sha256:8090b7da236bb21aa2e52e6e04dff4b7103753e4046e15457a3daf6dfa723a12
```

```
▶ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
newimage	latest	8090b7da236b	2 minutes ago	5.6MB
alpine	latest	a187dde48cd2	4 weeks ago	5.6MB





Building Images Using Commit

Docker Container Commit

```
▶ docker run -d --name httpd httpd
```

```
▶ docker exec -it httpd bash  
root@3484d738:/# cat > htdocs/index.html  
Welcome to my custom web application
```

```
▶ docker containerimage registry and operations customhttpd
```

```
▶ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
customhttpd	latest	adac0f56a7df	5 seconds ago	138MB
httpd	latest	417af7dc28bc	8 days ago	138MB



Image-customhttpd



Save vs Load vs Import vs Export vs Commit

```
▶ docker run -d --name httpd httpd
```

```
▶ docker exec -it httpd bash  
root@3484d738:/# cat > htdocs/index.html  
Welcome to my custom web application
```

```
▶ docker container commit -a "Ravi" httpd customhttpd
```

```
▶ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
customhttpd	latest	adac0f56a7df	5 seconds ago	138MB
httpd	latest	417af7dc28bc	8 days ago	138MB





Build Context



Build Context

Dockerfile

```
FROM ubuntu

RUN apt-get update
RUN apt-get install python

RUN pip install flask
RUN pip install flask-mysql
-----  
COPY . /opt/source-code  
-----  
ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

Docker Daemon

Docker CLI

 /opt/my-custom-app

▶ docker build [.] -t my-custom-app

▶ docker build /opt/my-custom-app



Build Context

Dockerfile

```
FROM ubuntu

RUN apt-get update
RUN apt-get install python

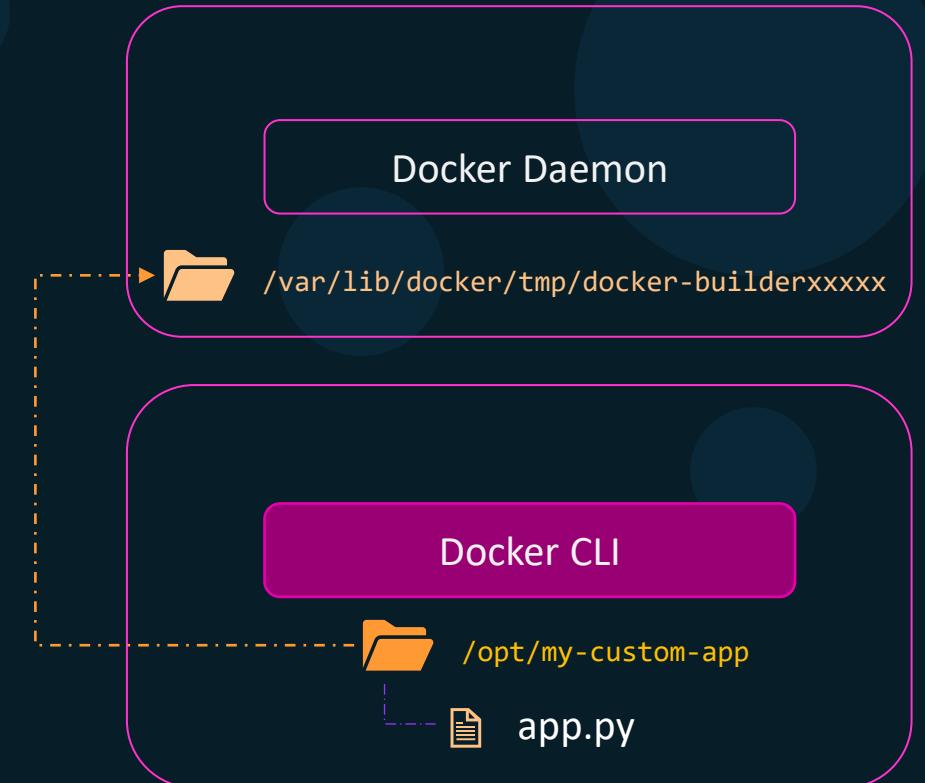
RUN pip install flask
RUN pip install flask-mysql

COPY . /opt/source-code

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

```
▶ docker build . -t my-custom-app
```

```
▶ docker build /opt/my-custom-app
Sending build context to Docker daemon 2.048kB
Step 1/7 : FROM ubuntu
```



.dockerignore

```
.dockerignore  
tmp  
logs  
build  
  
RUN apt-get install python  
  
RUN pip install flask  
RUN pip install flask-mysql  
  
COPY . /opt/source-code  
  
ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

```
▶ docker build . -t my-custom-app
```

```
▶ docker build /opt/my-custom-app
```

```
Sending build context to Docker daemon 2.048kB  
Step 1/7 : FROM ubuntu
```

Docker Daemon

▶ /var/lib/docker/tmp/docker-builderxxxxx

Docker CLI

▶ /opt/my-custom-app

▶ app.py

▶ tmp

▶ logs

▶ build

▶ .dockerignore

Build Context

```
▶ docker build . -t my-custom-app
```

```
▶ docker build /opt/my-custom-app
```

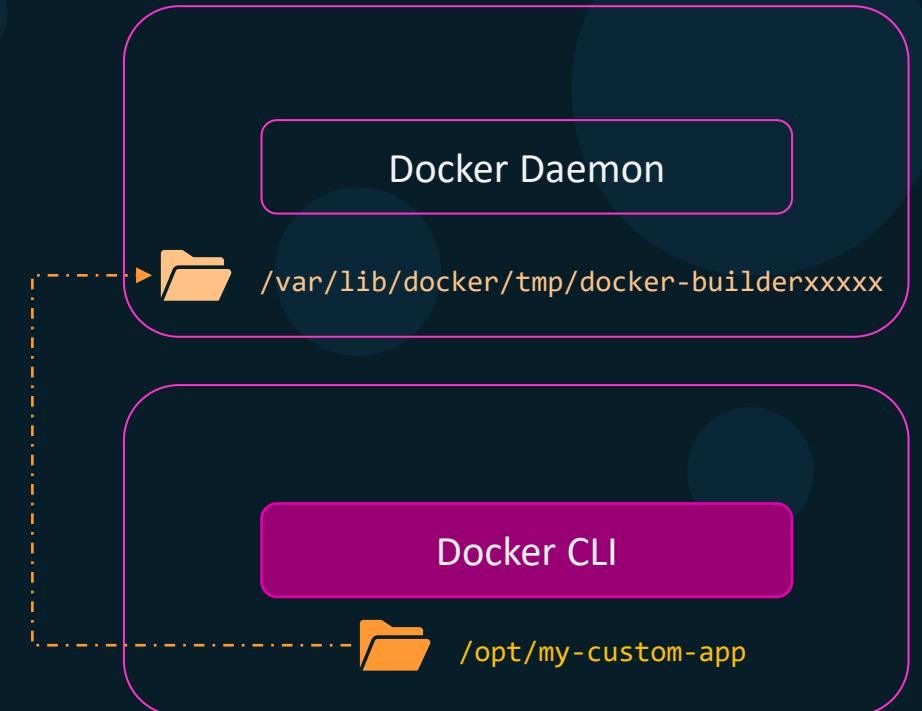
```
Sending build context to Docker daemon 2.048kB  
Step 1/7 : FROM ubuntu
```

```
▶ docker build https://github.com/myaccount/myapp
```

```
▶ docker build https://github.com/myaccount/myapp#<branch>
```

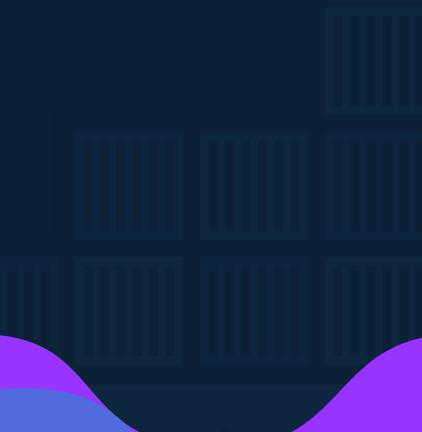
```
▶ docker build https://github.com/myaccount/myapp:<folder>
```

```
▶ docker build -f Dockerfile.dev https://github.com/myaccount/myapp
```





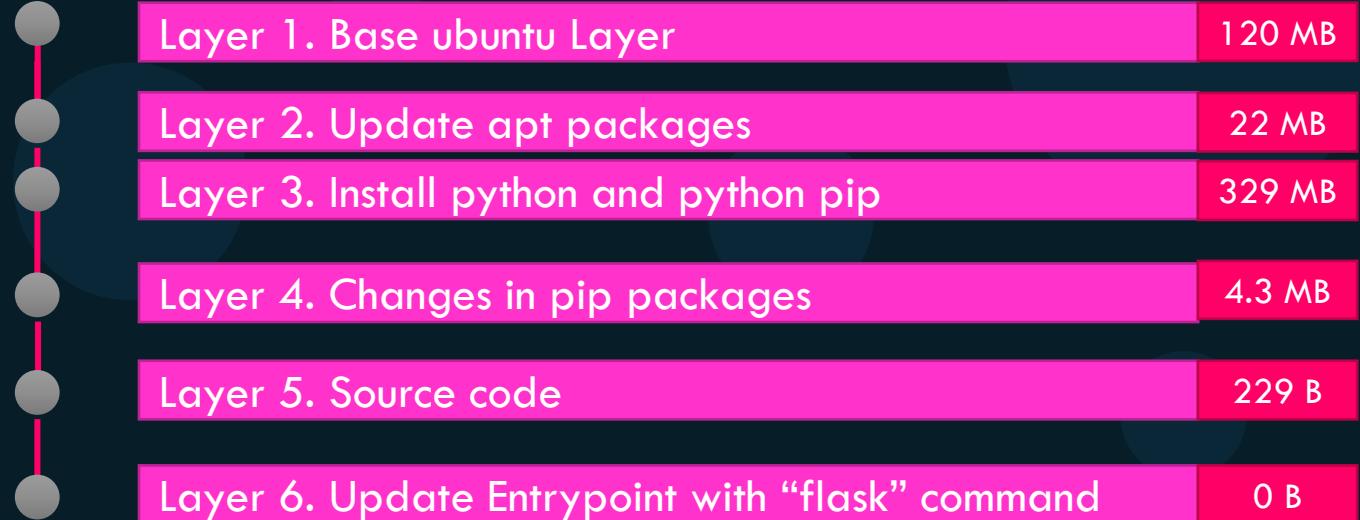
Build Cache



Build Cache

Dockerfile

```
FROM ubuntu  
  
RUN apt-get update  
RUN apt-get install -y python python3-pip  
  
RUN pip3 install flask  
  
COPY app.py /opt/source-code  
  
ENTRYPOINT flask run
```



Build Cache

cached	Layer 1. Base ubuntu Layer	120 MB
	Layer 2. Update apt packages	22 MB
	Layer 3. Install python and python pip	329 MB
	Layer 4. Changes in pip packages	4.3 MB
	Layer 5. Source code	229 B
	Layer 6. Update Entrypoint with “flask” command	0 B

▶ docker build .

```
Sending build context to Docker daemon 2.048kB
Step 1/6 : FROM ubuntu
--> bb0eaf4eee00
Step 2/6 : RUN apt-get update
--> Using cache
--> e09e593ec730
Step 3/6 : RUN apt-get install -y python python-pip
--> Running in e9944225690a
Reading package lists...
Building dependency tree...
Reading state information...
E: Unable to locate package python-pip
The command '/bin/sh -c apt-get install -y python python-pip' returned a non-zero code: 100
```



Build Cache

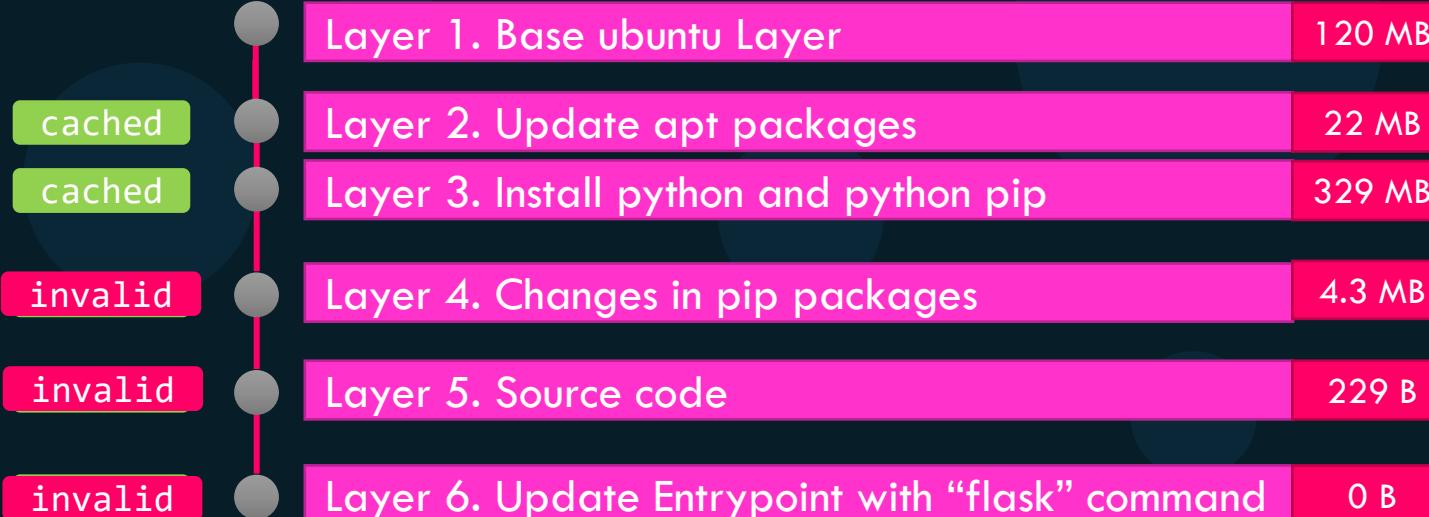
Dockerfile

```
FROM ubuntu

RUN apt-get update
RUN apt-get install -y python python3-pip

RUN pip3 install flask flask-mysql
COPY [app.py] /opt/source-code

ENTRYPOINT flask run
```



1. Compare instructions in Dockerfile
2. Compare checksums of files in ADD or COPY



Build Cache - Cache Busting

Dockerfile

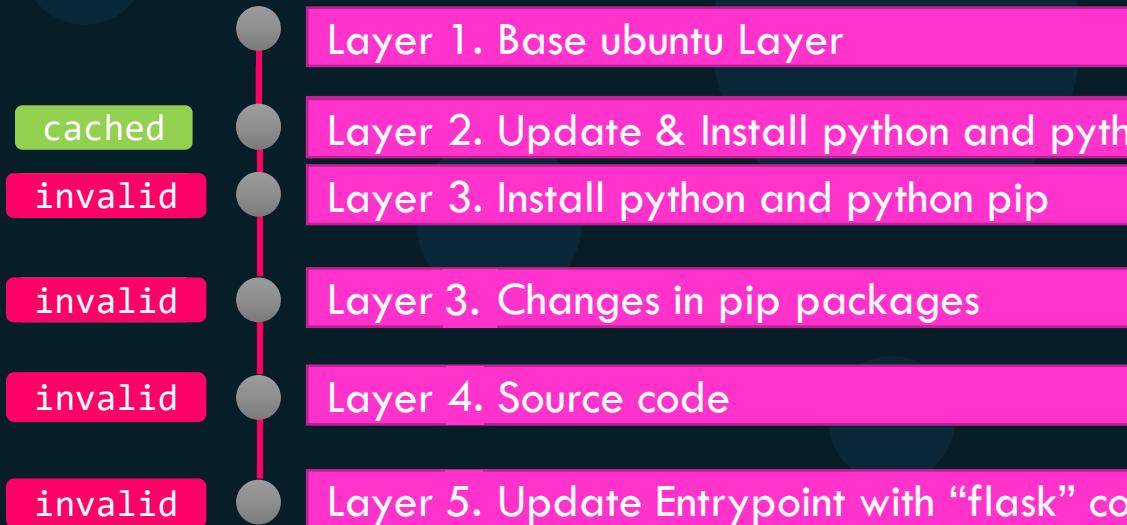
```
FROM ubuntu

RUN apt-get update && apt-get install -y \
    python python3-pip python-dev

RUN pip3 install flask flask-mysql

COPY app.py /opt/source-code

ENTRYPOINT flask run
```



Build Cache - Version Pinning

Dockerfile

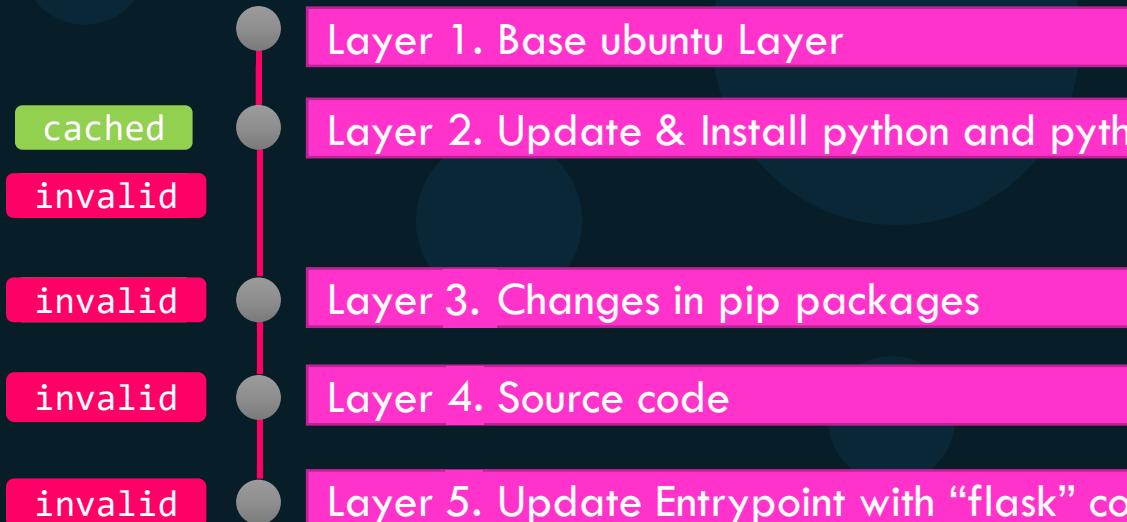
```
FROM ubuntu

RUN apt-get update && apt-get install -y \
    python \
    python-dev \
    python3-pip=20.0.2

RUN pip3 install flask flask-mysql

COPY app.py /opt/source-code

ENTRYPOINT flask run
```



Build Cache

Dockerfile

```
FROM ubuntu

RUN apt-get update && apt-get install -y \
    python \
    python-dev \
    python3-pip=20.0.2

RUN pip3 install flask flask-mysql

COPY app.py /opt/source-code

ENTRYPOINT flask run
```

cached

cached

invalid



Build Cache

Dockerfile

```
FROM ubuntu

COPY app.py /opt/source-code

RUN apt-get update && apt-get install -y \
    python \
    python-dev \
    python3-pip=20.0.2

RUN pip3 install flask flask-mysql

ENTRYPOINT flask run
```

invalid

invalid

invalid



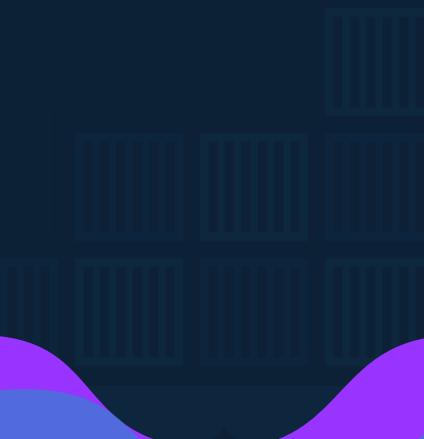
References

- https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#leverage-build-cache





COPY VS ADD



Difference between COPY and ADD

Dockerfile

```
FROM centos:7  
COPY /testdir /testdir
```

Dockerfile

```
FROM centos:7  
ADD /testdir /testdir
```

Dockerfile

```
FROM centos:7  
ADD app.tar.xz /testdir
```

Dockerfile

```
FROM centos:7  
ADD http://app.tar.xz /testdir  
RUN tar -xJf /testdir/app.tar.xz -C /tmp/app  
RUN make -C /tmp/app
```



Copy or ADD?

Dockerfile

```
FROM centos:7  
COPY /testdir /testdir
```

Dockerfile

```
FROM centos:7  
ADD /testdir /testdir
```

Dockerfile

```
FROM centos:7  
RUN curl http://app.tar.xz \  
| tar -xcJ /testdir/file.tar.xz \  
&& yarn build \  
&& rm /testdir/file.tar.xz
```

Dockerfile

```
FROM centos:7  
ADD http://app.tar.xz /testdir  
RUN tar -xJf /testdir/app.tar.xz -C /tmp/app  
RUN make -C /tmp/app
```





KODEKLOUD



Base Image

Base vs Parent Image

Parent

```
Dockerfile - My Custom Webapp  
-----  
FROM httpd  
  
COPY index.html /var/www/html/index.html
```

httpd (Parent)

My Custom WebApp



Base vs Parent Image

Parent

```
Dockerfile - httpd
FROM debian:buster-slim
ENV HTTPD_PREFIX /usr/local/apache2
ENV PATH $HTTPD_PREFIX/bin:$PATH
WORKDIR $HTTPD_PREFIX
<content trimmed>
```

debian

httpd (Parent)

My Custom WebApp

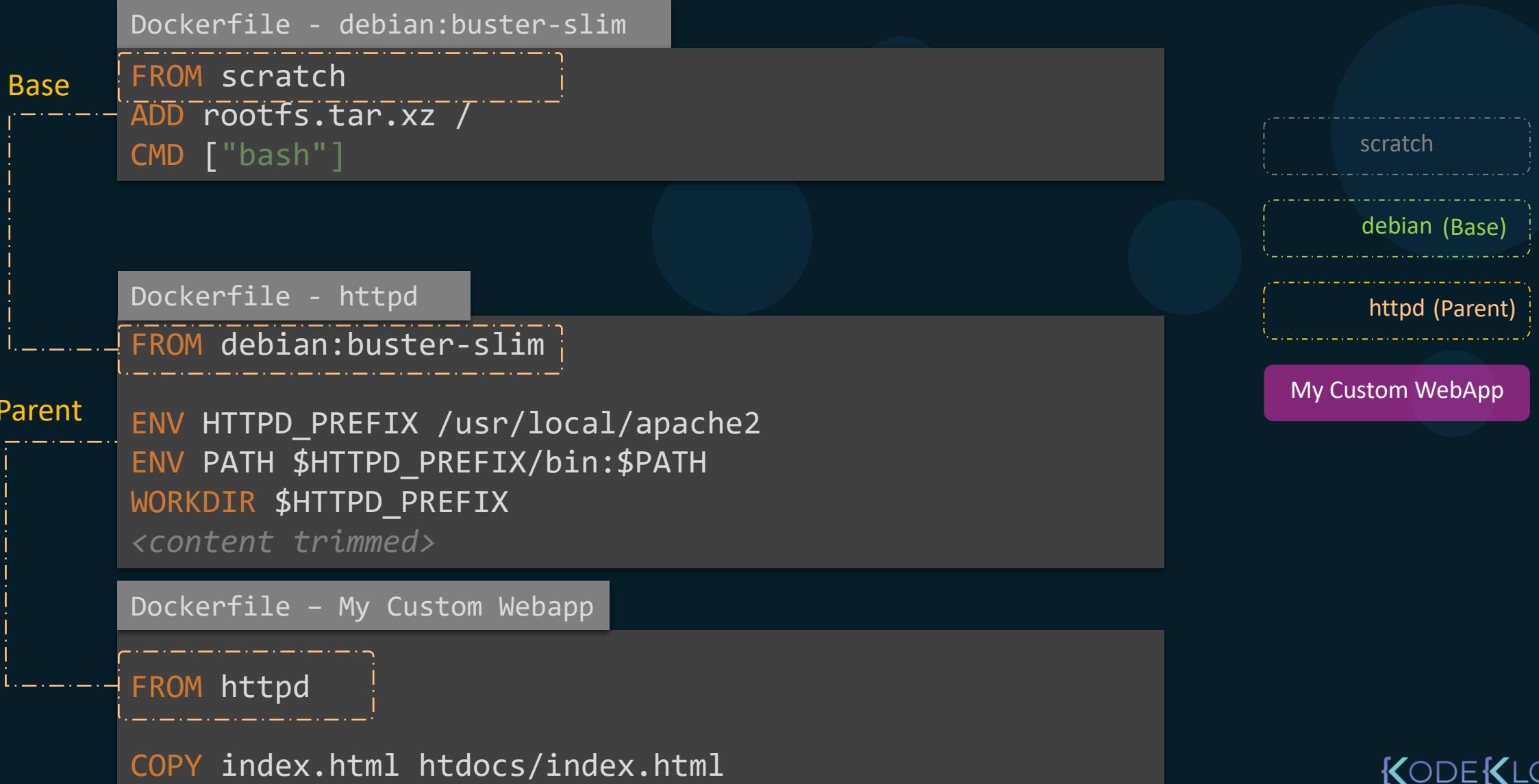
```
Dockerfile - My Custom Webapp
```

```
FROM httpd
```

```
COPY index.html htdocs/index.html
```



Base vs Parent Image



Base vs Parent Image

44 lines (40 sloc) | 2.62 KB

```
1 FROM scratch
2 ADD ubuntu-xenial-core-cloudimg-amd64-root.tar.gz /
3
```

108 lines (96 sloc) | 3.97 KB

```
1 FROM ubuntu:xenial
2
3 # add our user and group first to make sure their IDs get assigned correctly
4 RUN groupadd -r mongodb && useradd -r -g mongodb mongodb
5
6 RUN set -eux; \
7     apt-get update; \
8     apt-get install -y --no-install-recommends \
9         ca-certificates \
10        jq \
11        numactl \
12    ; \
13    if ! command -v ps > /dev/null; then \
14        apt-get install -y --no-install-recommends procps; \
15    fi
```

scratch

ubuntu (Base)

scratch

debian (Base)

httpd (Parent)

MongoDB

My Custom WebApp

Base vs Parent Image

```
Dockerfile - debian:buster-slim  
FROM scratch  
ADD rootfs.tar.xz /  
CMD ["bash"]
```



References

<https://docs.docker.com/develop/develop-images/baseimages/>





KODEKLOUD

Multi-Stage Builds





```
my-application
├── Dockerfile
├── LICENSE
├── README.md
├── package.json
└── app.js
```

```
public
tests
config
routes
services
db
core
dist
```

Development Server

1. Build

```
▶ npm run build
```

2. Containerize for Production

Dockerfile

```
FROM nginx
```

```
COPY dist /usr/share/nginx/html
```

```
CMD [ "nginx", "-g", "daemon off;" ]
```

```
▶ docker build -t my-app .
```



```
my-application
  Dockerfile.builder
  Dockerfile
  LICENSE
  README.md
  package.json
  app.js
  public
  tests
  config
  routes
  services
  db
  core
  dist
```

Development Server

1. Build

```
Dockerfile.builder
```

```
FROM node
COPY . .
RUN npm install
RUN npm run build
```

```
▶ docker build -t builder .
```

2. Containerize for Production

```
Dockerfile
```

```
FROM nginx
COPY [dist] /usr/share/nginx/html
CMD [ "nginx", "-g", "daemon off;" ]
```

```
▶ docker build -t my-app .
```

1. Build

```
Dockerfile.builder
FROM node
COPY . .
RUN npm install
RUN npm run build
```

▶ docker build -t builder .

3. Extract build from first image

```
copy-dist-from-builder.sh
docker container create --name builder builder
docker container cp builder:dist ./dist
docker container rm -f builder
```

3. Containerize for Production

```
Dockerfile
FROM nginx
COPY dist /usr/share/nginx/html
CMD [ "nginx", "-g", "daemon off;" ]
```

▶ docker build -t my-app .



Multi-stage builds

1. Build

```
Dockerfile
FROM node
COPY . .
RUN npm install
RUN npm run build
FROM nginx
COPY dist /usr/share/nginx/html
CMD [ "nginx", "-g", "daemon off;" ]
```

3. Containerize for Production

► docker build -t my-app .



Multi-stage builds

1. Build Stage 0
2. Stage 1
3. Extract build from first image

Dockerfile

```
FROM node AS builder
COPY . .
RUN npm install
RUN npm run build

FROM nginx
COPY --from=builder dist /usr/share/nginx/html
CMD [ "nginx", "-g", "daemon off;" ]
```

3. Containerize for Production

► docker build -t my-app .

► docker build --target builder -t my-app .



Multi-Stage Builds

- Optimize Dockerfiles and keeps them easy to read and maintain
- Helps keep size of images low
- Helps avoid having to maintain multiple Dockerfiles – Builder and Production
- No intermediate images

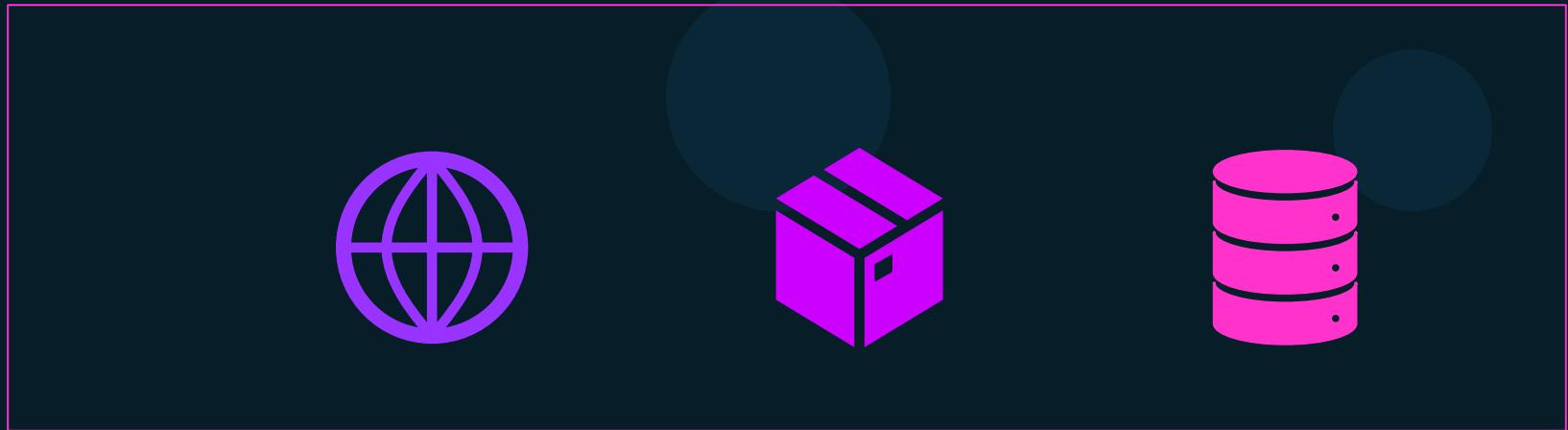




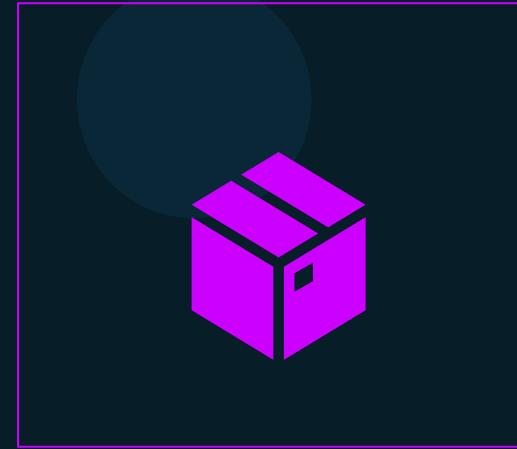
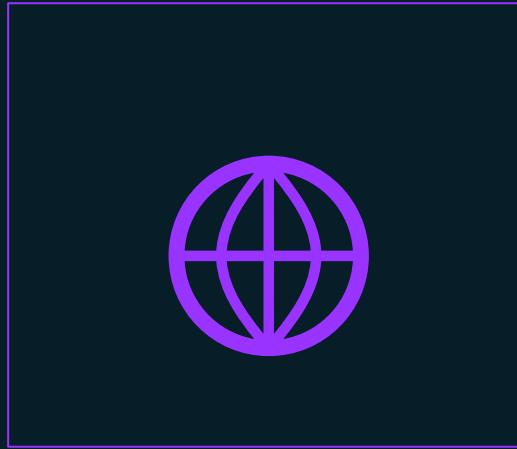
Best Practices



Modular



Modular



Persist State

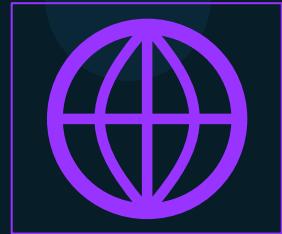


Persist State



Slim/Minimal Images

1. Create slim/minimal images
2. Find an official minimal image that exists
3. Only install necessary packages
4. Maintain different images for different environments:
 - Development – debug tools
 - Production - lean
5. Use multi-stage builds to create lean production ready images.
6. Avoid sending unwanted files to the build context



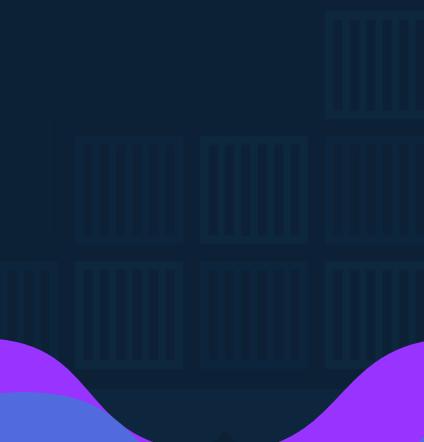
References

1. <https://docs.docker.com/develop/dev-best-practices/>
2. https://docs.docker.com/develop/develop-images/dockerfile_best-practices/





Networking



Network: List

```
▶ docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
599dcaf4e856	bridge	bridge	local
c817f1bca596	host	host	local
e6508d3404a3	none	null	local

```
▶ docker network inspect 599dcaf4e856
```

```
[  
 {  
   "Name": "bridge",  
   "Id": "599dcaf4e85684c8c3a111baa52b7530f097853b96485a8a3ffcd9088b20f0cb",  
   "Created": "2020-01-20T18:10:46.896056535Z",  
   "Scope": "local",  
   "Driver": "bridge",  
   "EnableIPv6": false,  
   "IPAM": {  
     "Driver": "default",  
     "Options": null,  
     "Config": [  
       {  
         "Subnet": "172.17.0.0/16",  
         "Gateway": "172.17.0.1"  
       }  
     ]  
   }  
 }
```

Custom Network

```
▶ docker network connect custom-net my-container
```

```
▶ docker network disconnect custom-net my-container
```

```
▶ docker network rm custom-net
```

```
▶ docker network prune
```

WARNING! This will remove all networks not used by at least one container.

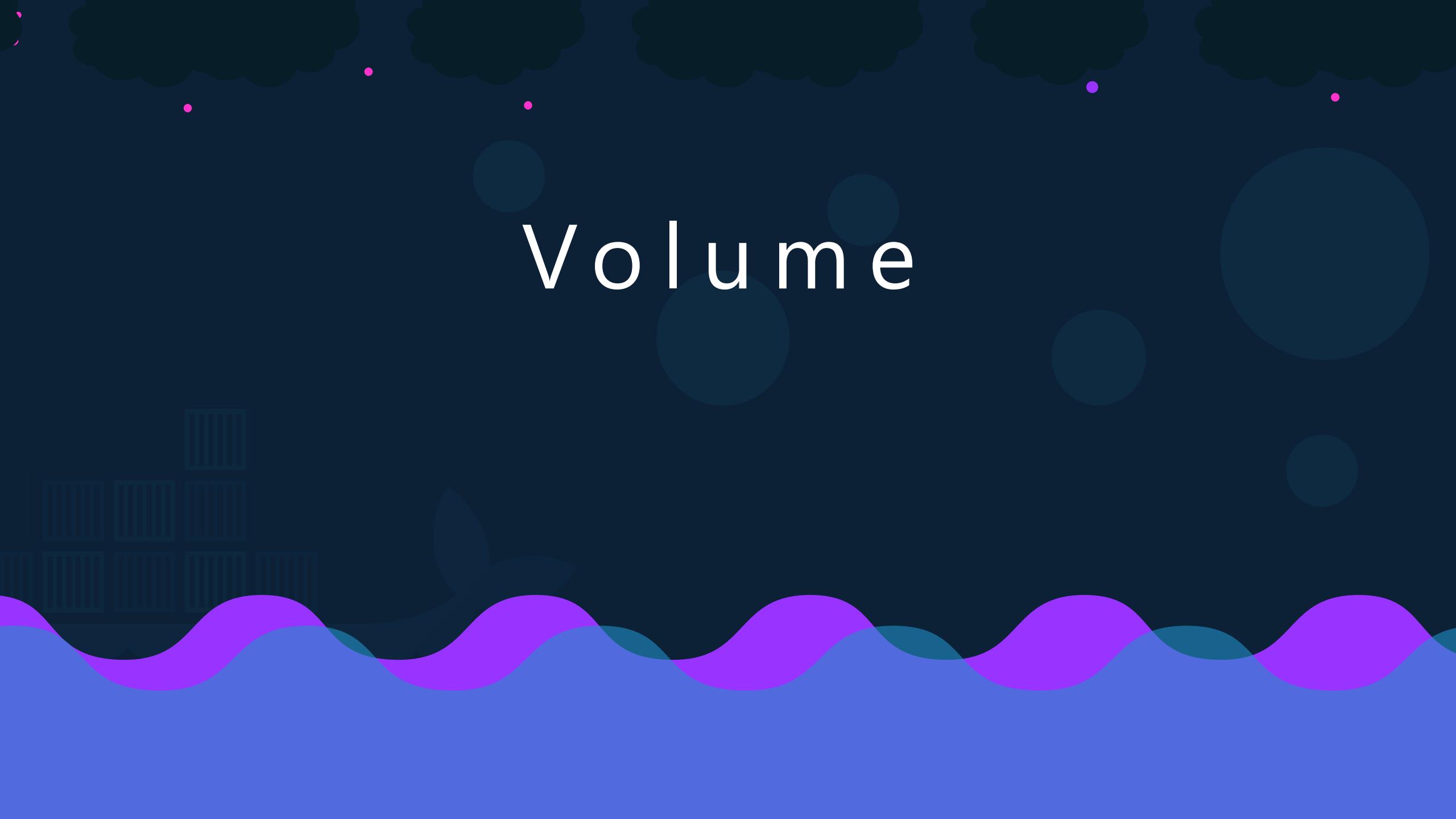
Are you sure you want to continue? [y/N] y

Deleted Networks:

custom-net







Volume



Volume Inspect

```
▶ docker volume inspect data_volume
[
  {
    "CreatedAt": "2020-01-20T19:52:34Z",
    "Driver": "local",
    "Labels": {},
    "Mountpoint": "/var/lib/docker/volumes/data_volume/_data",
    "Name": "data_volume",
    "Options": {},
    "Scope": "local"
  }
]
```



Volume Removal: rm and prune

```
▶ docker volume remove data_volume
```

```
Error response from daemon: remove data_volume: volume is in use -  
[2be4d91822964882504a31992aac9dd0b228c03f8739b1afe74984aae6409620]
```

```
▶ docker volume remove data_volume
```

```
data_volume
```

```
▶ docker volume prune
```

```
WARNING! This will remove all local volumes not used by at least one container.
```

```
Are you sure you want to continue? [y/N] y
```

```
Deleted Volumes:
```

```
data_vol3
```

```
data_vol1
```

```
data_vol2
```

```
Total reclaimed space: 12MB
```



ReadOnly Volume

```
▶ docker container inspect my-container
```

```
"Mounts": [  
    {  
        "Type": "volume",  
        "Name": "data_vol1",  
        "Source": "/var/lib/docker/volumes/data_vol1/_data",  
        "Destination": "/var/www/html/index.html",  
        "Driver": "local",  
        "Mode": "z",  
        "RW": true,  
        "Propagation": ""  
    }  
,
```

```
▶ docker container run --mount \  
  source=data_vol1,destination=/var/www/html/index.html,readonly httpd
```



References

<https://docs.docker.com/storage/>

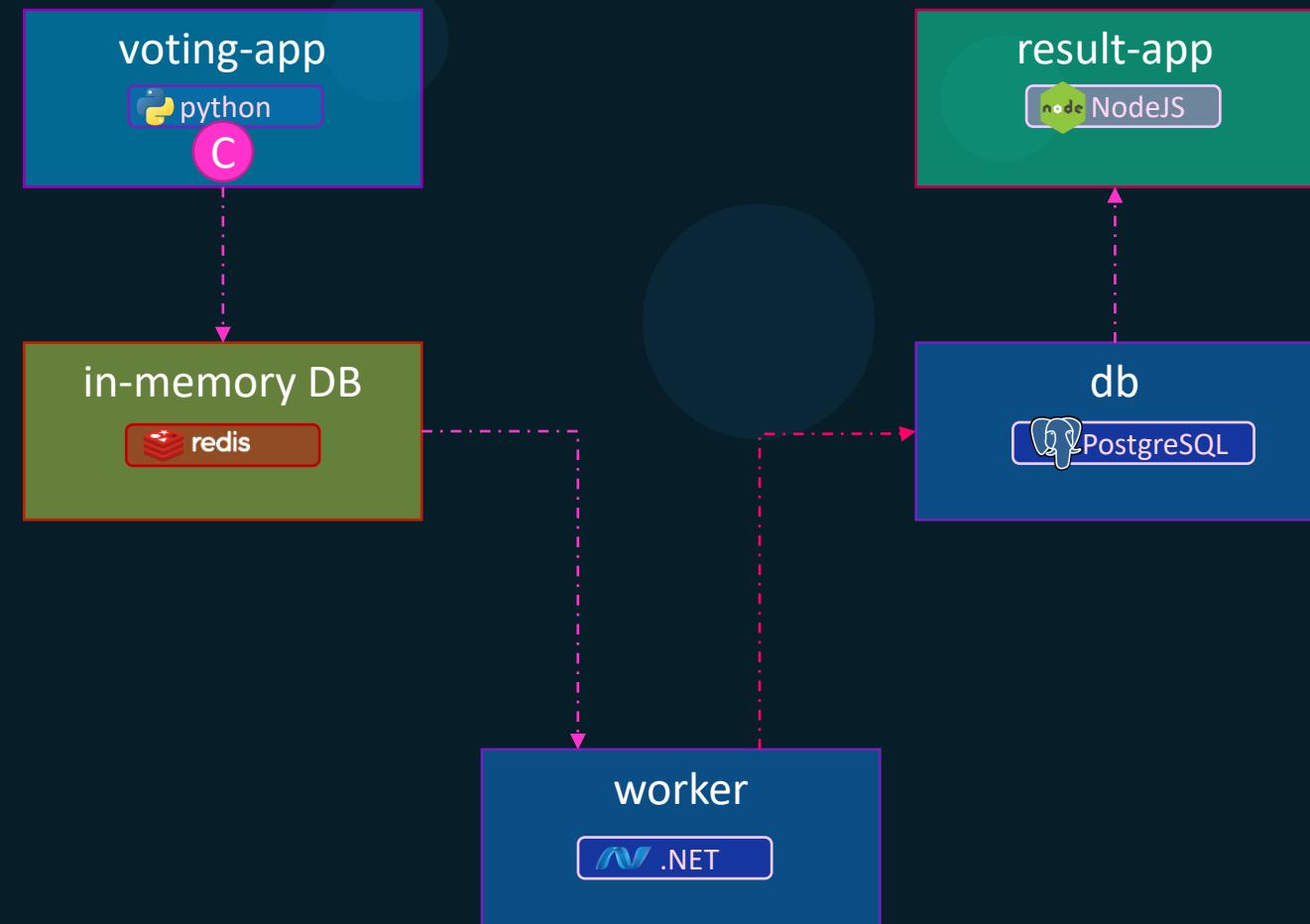




End to End Engine Demo



Sample application – voting application



Sample application – voting application

► Build and Pull Images

► Build a user-defined Network for your voting app

► Create containers inside the user-defined network

► Test your voting app





KODEKLOUD



d o c k e r

compose

Docker compose

```
docker container run -itd -name=web nodejs  
docker container run -itd -name=db mongodb  
docker container run -itd -name=messaging redis  
docker container run -itd -name=orchestration ansible
```

docker-compose.yml

```
services:  
  web:  
    image: "nodejs"  
  db:  
    image: "mongodb"  
  messaging:  
    image: "redis"  
  orchestration:  
    image: "ansible"
```

docker-compose up



Public Docker registry - dockerhub



Docker compose - versions

docker-compose.yml

```
version: "3.8"
services:
  web:
    image: httpd:alpine
    ports:
      - "80"
    networks:
      - appnet
    volumes:
      - appvol:/webfs
  networks:
    - appnet
  volumes:
    - appvol

  configs:

  secrets:
```

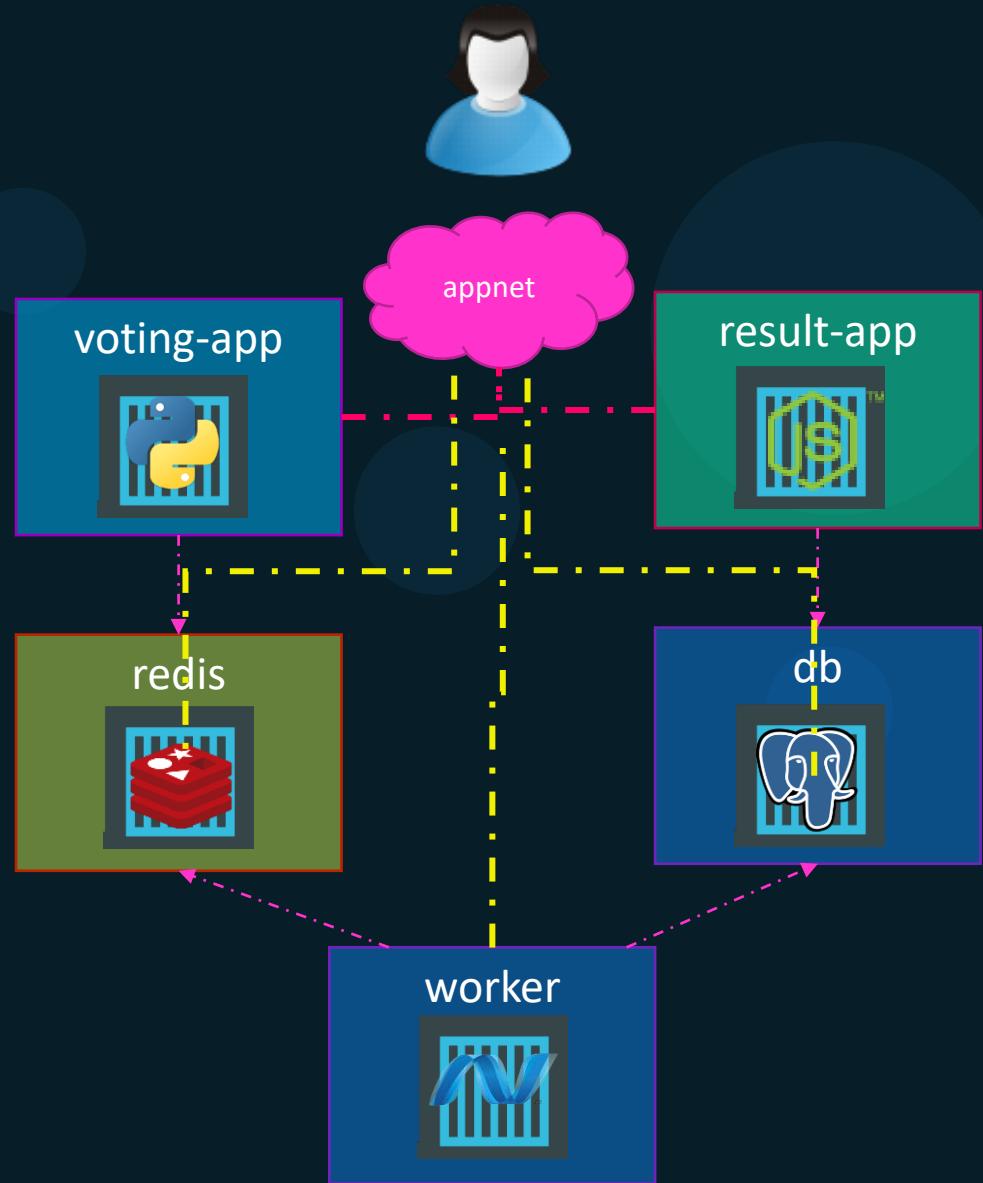
version: 3



Docker compose

docker-compose.yml

```
version: '3.8'
services:
  vote:
    image: yogeshraheja/vote:v1
    ports:
      - "81:80"
    networks:
      - appnet
  redis:
    image: yogeshraheja/redis:v1
    networks:
      - appnet
  db:
    image: yogeshraheja/db:v1
    networks:
      - appnet
  worker:
    image: yogeshraheja/worker:v1
    networks:
      - appnet
  result:
    image: yogeshraheja/result:v1
    ports:
      - "82:80"
    networks:
      - appnet
networks:
  appnet:
    driver: bridge
```



Compose Commands

▶ docker-compose up

▶ docker-compose up -d

▶ docker-compose ps

▶ docker-compose logs

▶ docker-compose stop

▶ docker-compose start



Compose Commands

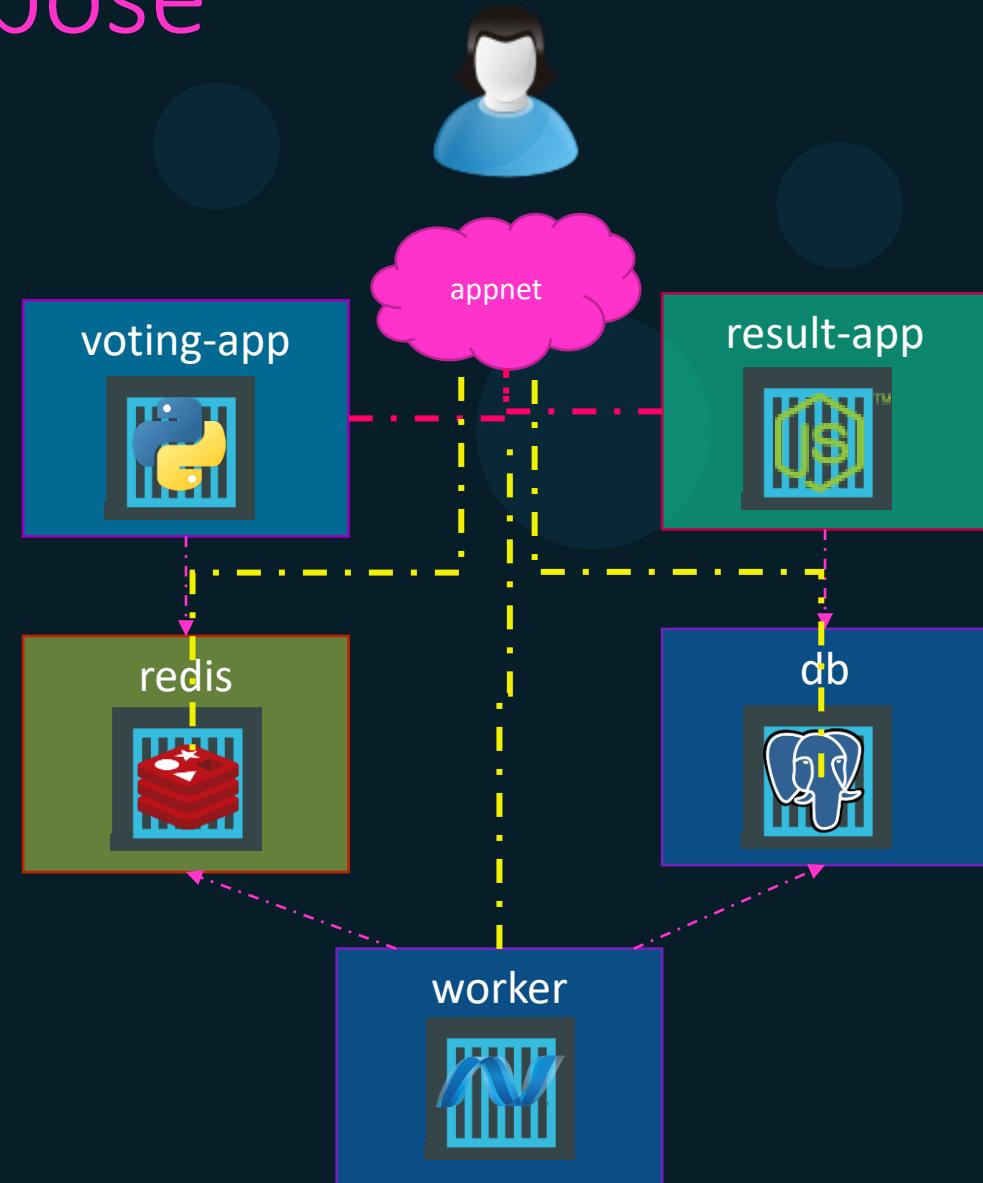
▶ docker-compose stop

▶ docker-compose rm

▶ docker-compose down



Docker compose





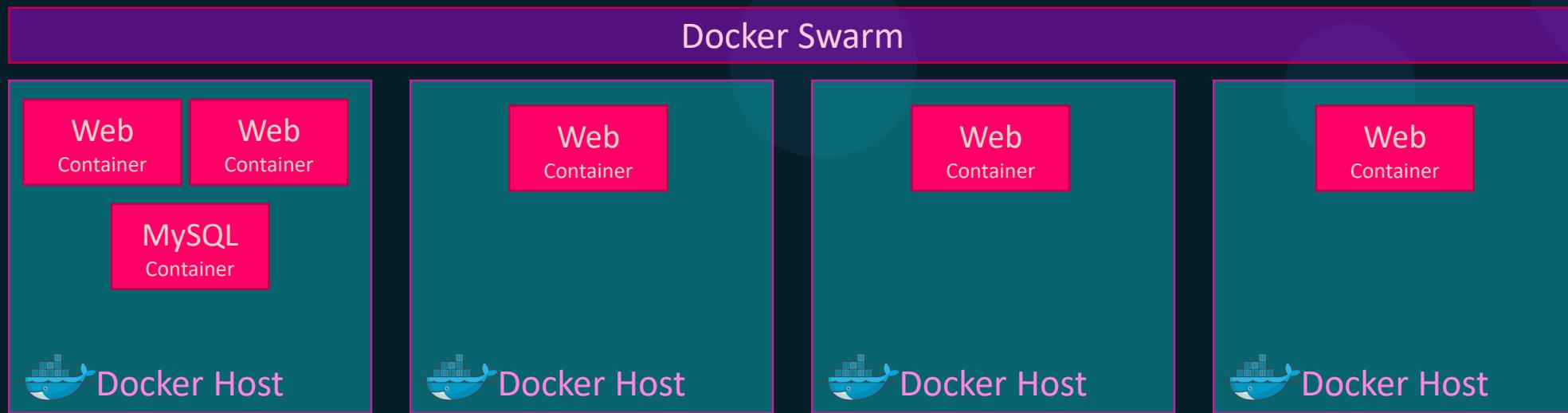
KODEKLOUD



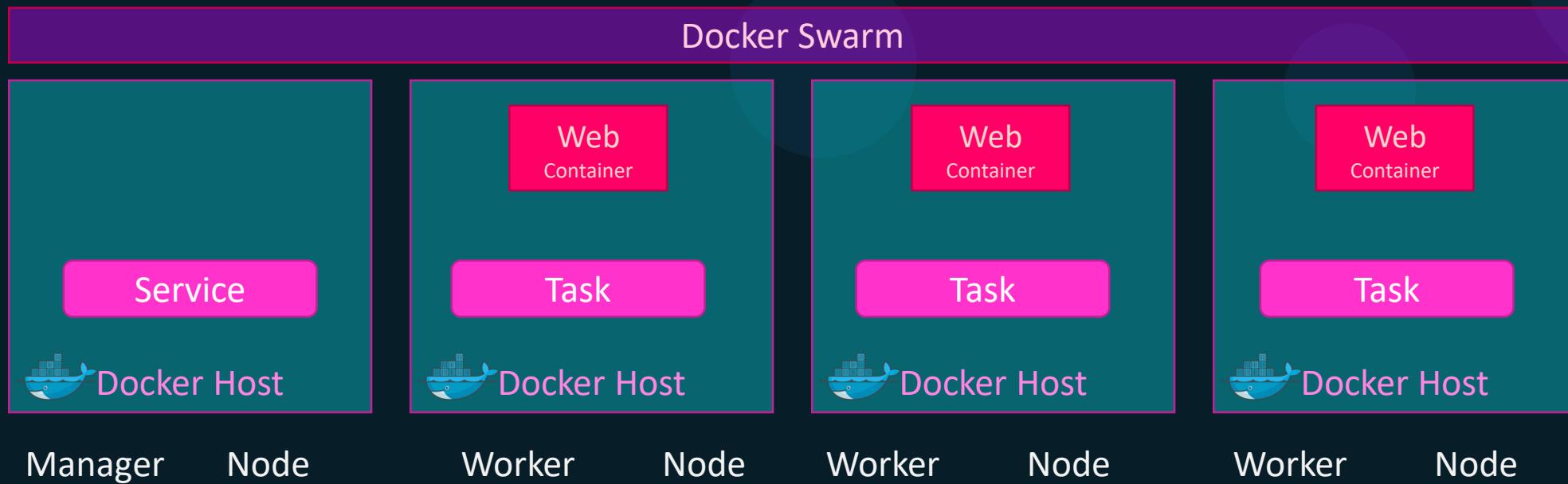
d o c k e r

swarm

Docker swarm

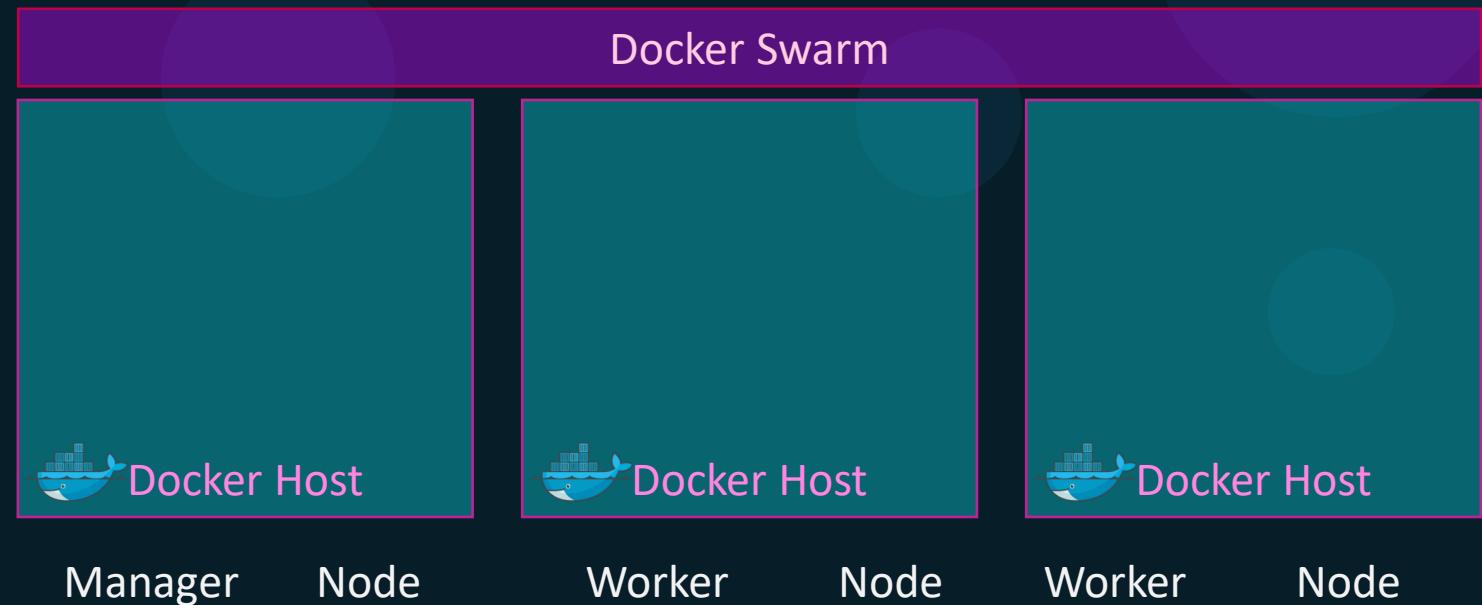


Docker swarm



Features

- Simplified Setup
- Declarative
- Scaling
- Rolling Updates
- Self Healing
- Security
- Load balancing
- Service Discovery

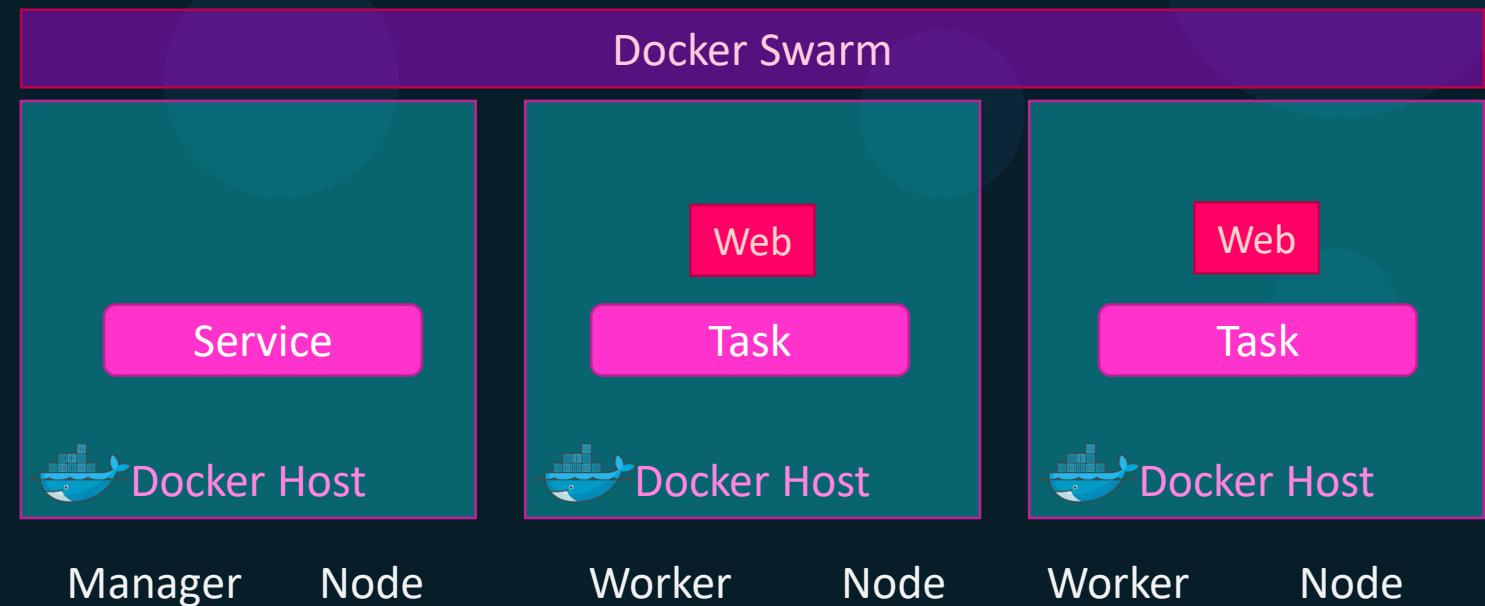


Features

- Simplified Setup
- Declarative
- Scaling
- Rolling Updates
- Self Healing

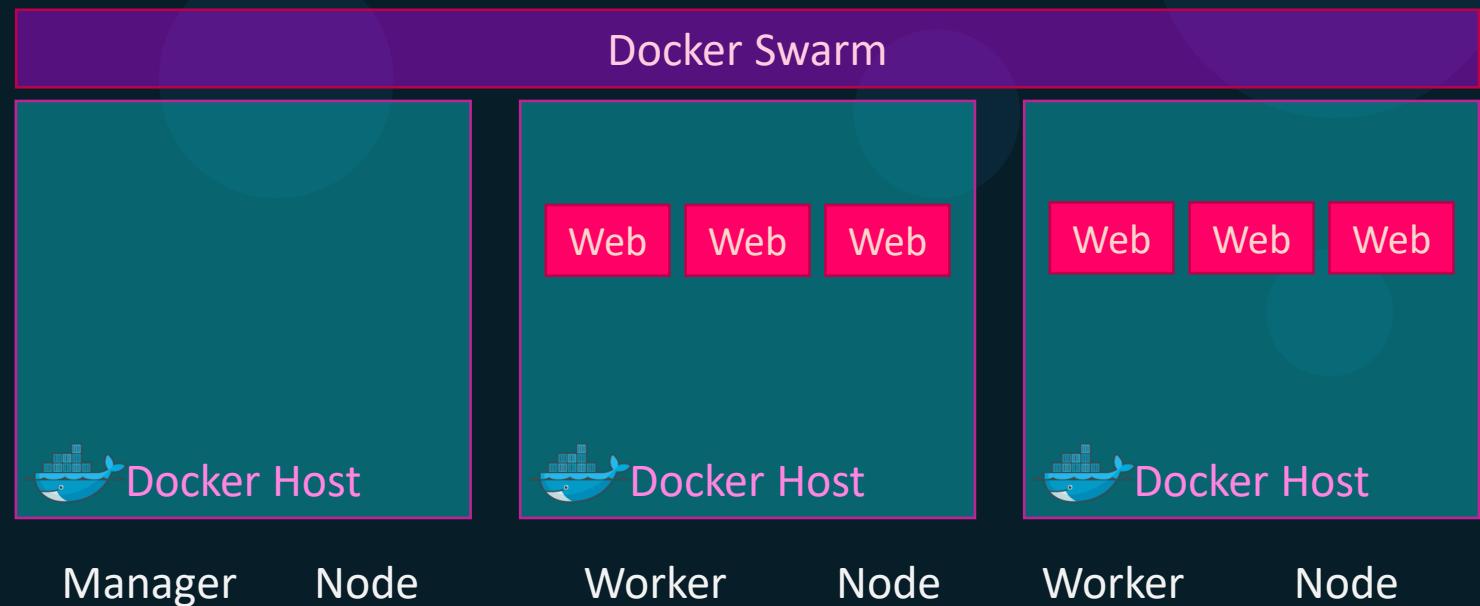
service-definition.yml

```
services:  
  web:  
    image: "simple-webapp"  
  database:  
    image: "mongodb"  
  messaging:  
    image: "redis:alpine"
```



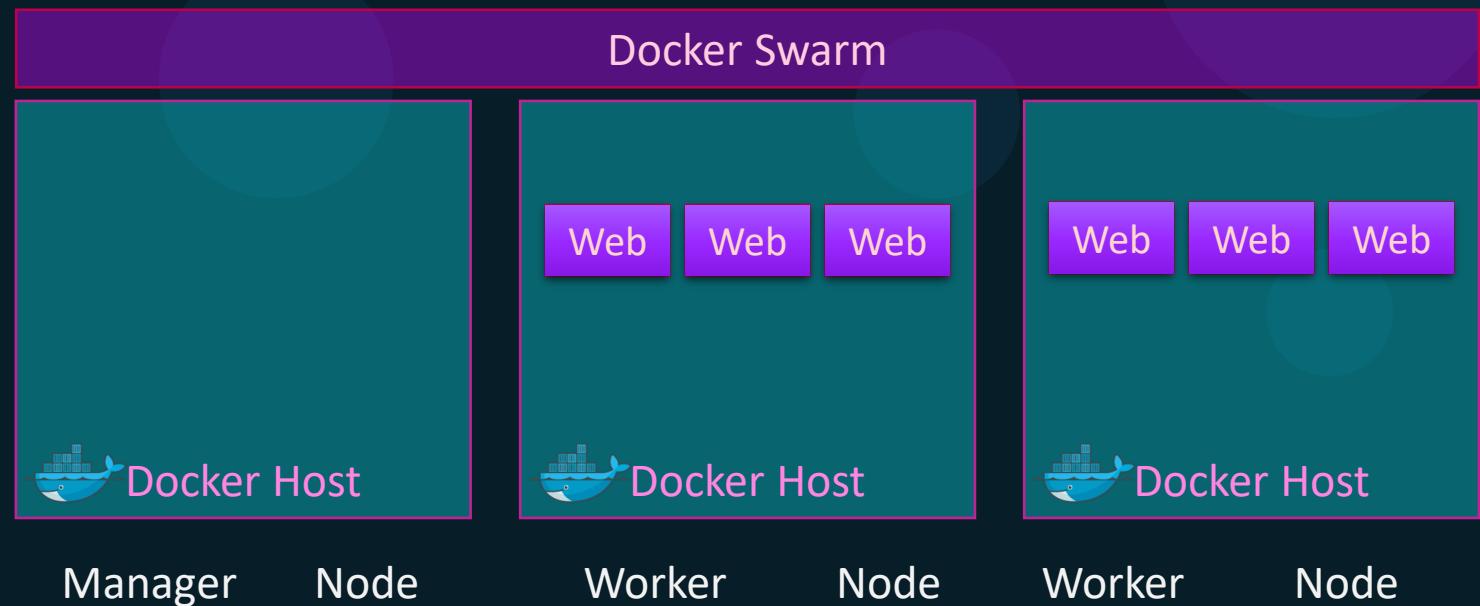
Features

- Simplified Setup
- Declarative
- **Scaling**
- Rolling Updates
- Self Healing
- Security
- Load balancing
- Service Discovery



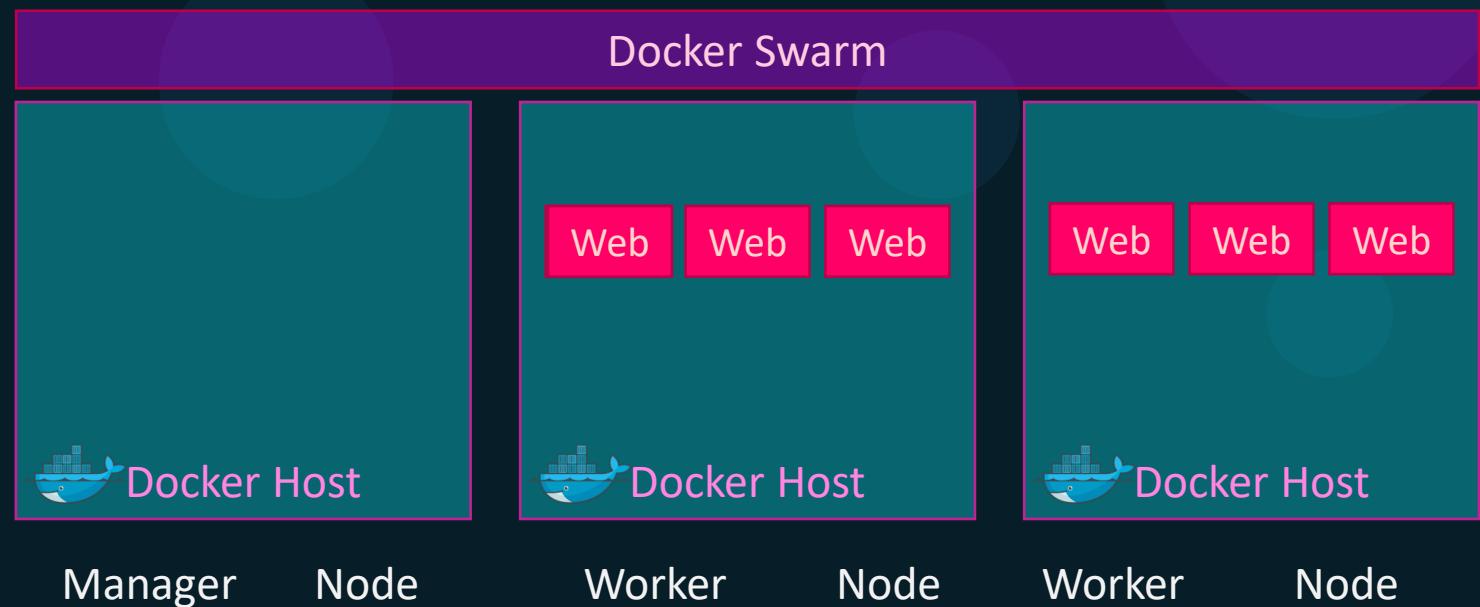
Features

- Simplified Setup
- Declarative
- Scaling
- Rolling Updates
- Self Healing
- Security
- Load balancing
- Service Discovery



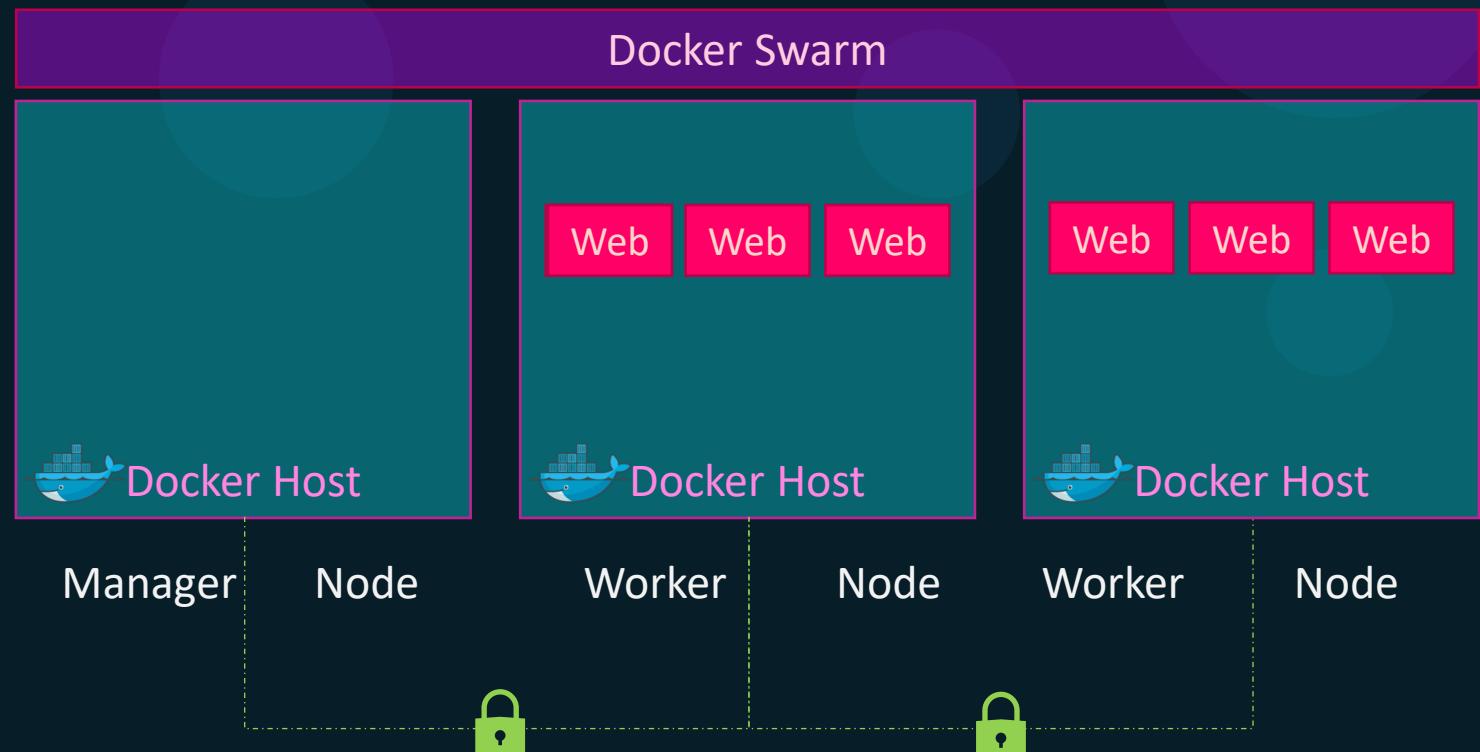
Features

- Simplified Setup
- Declarative
- Scaling
- Rolling Updates
- **Self Healing**
- Security
- Load balancing
- Service Discovery



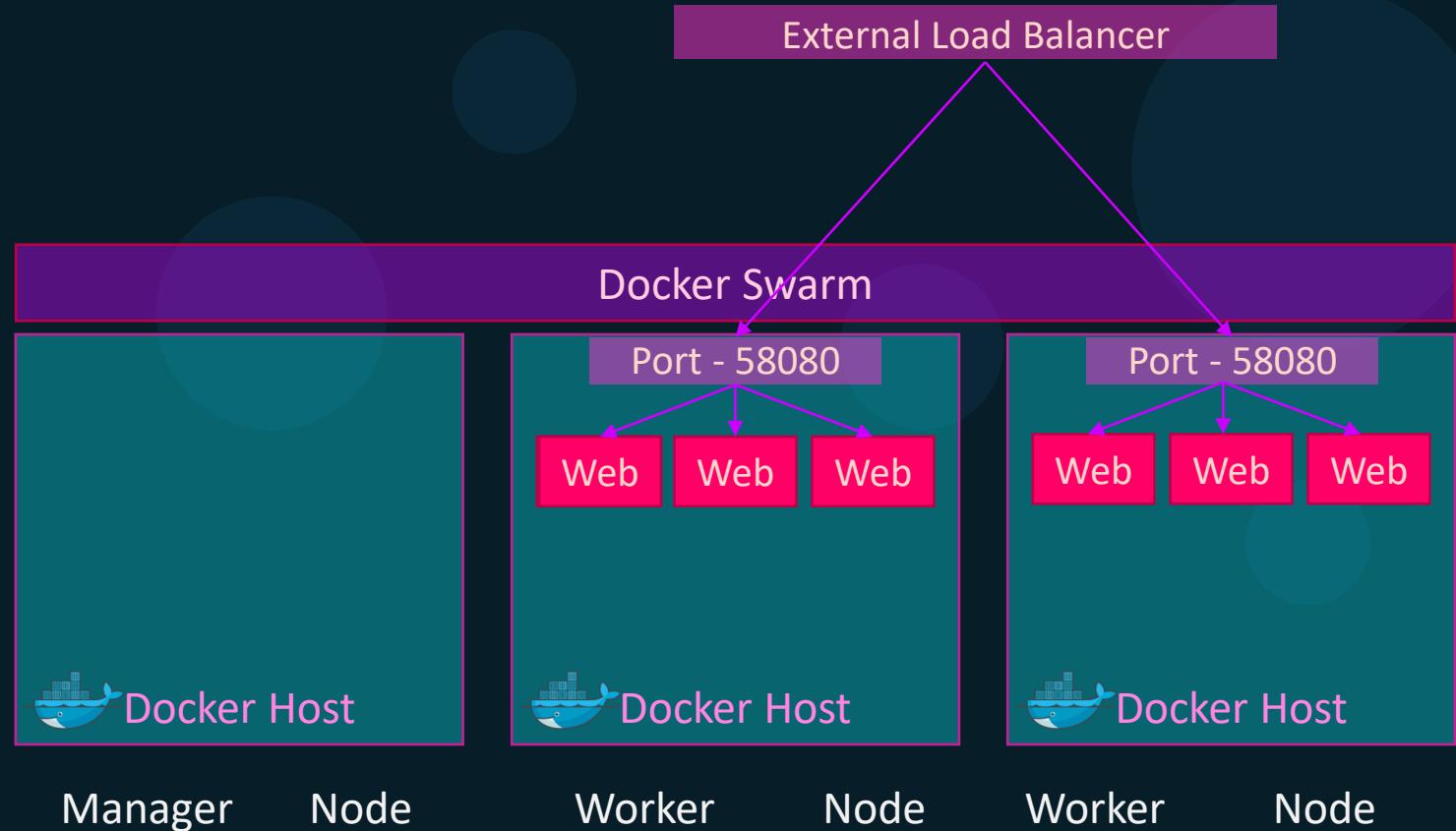
Features

- Simplified Setup
- Declarative
- Scaling
- Rolling Updates
- Self Healing
- Security
- Load balancing
- Service Discovery



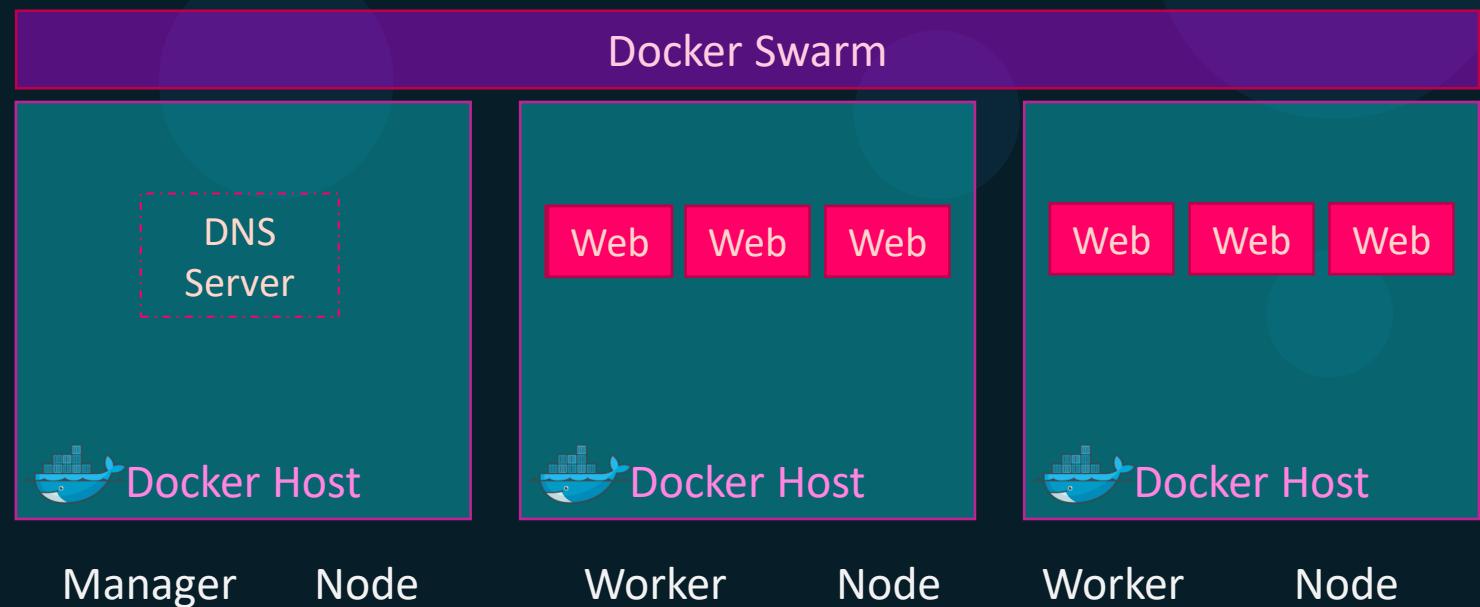
Features

- Simplified Setup
- Declarative
- Scaling
- Rolling Updates
- Self Healing
- Security
- Load balancing
- Service Discovery



Features

- Simplified Setup
- Declarative
- Scaling
- Rolling Updates
- Self Healing
- Security
- Load balancing
- **Service Discovery**





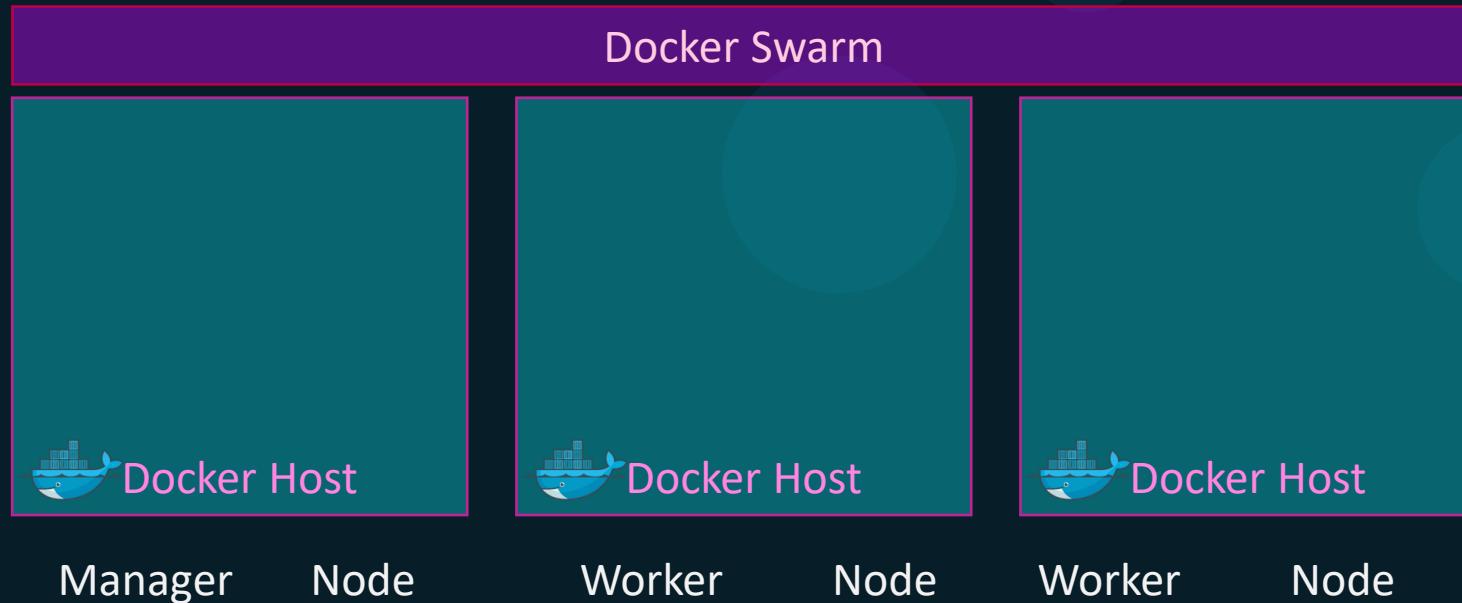
KODEKLOUD

Setup

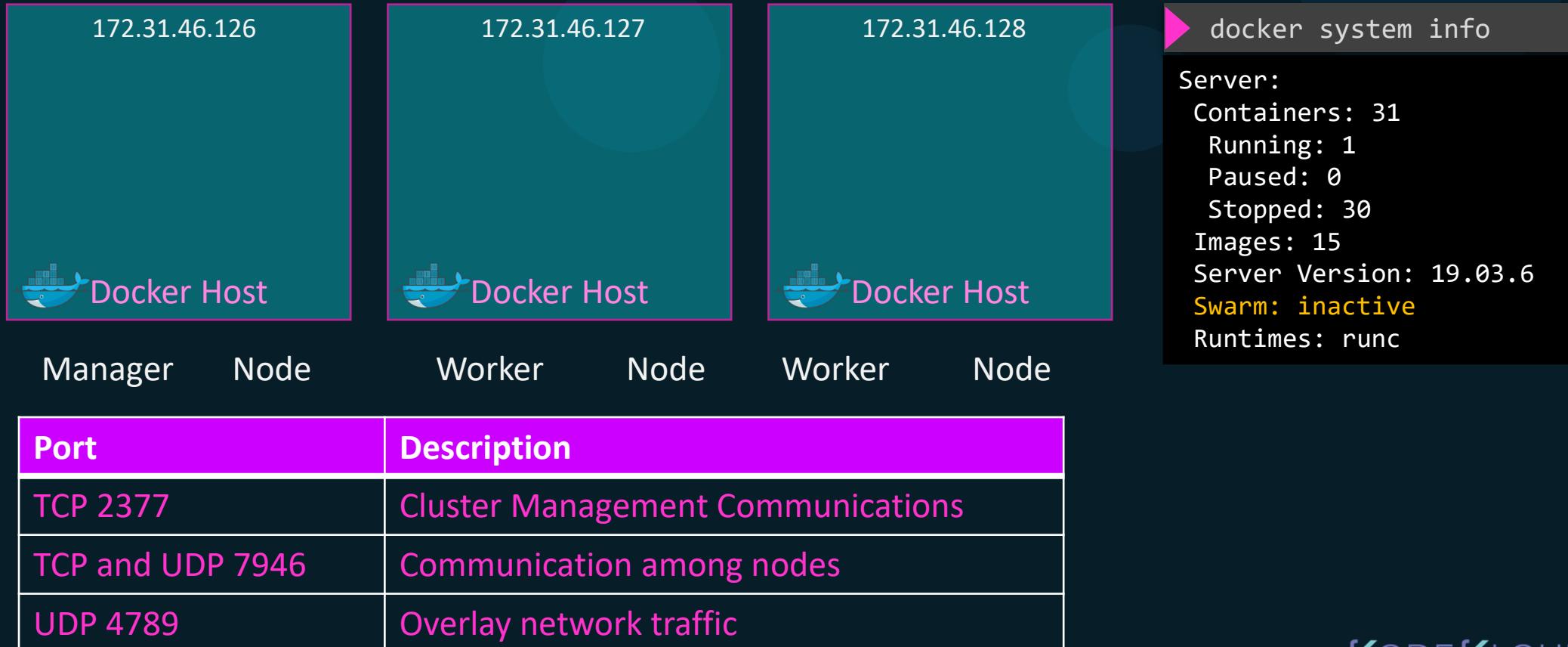
Docker Swarm



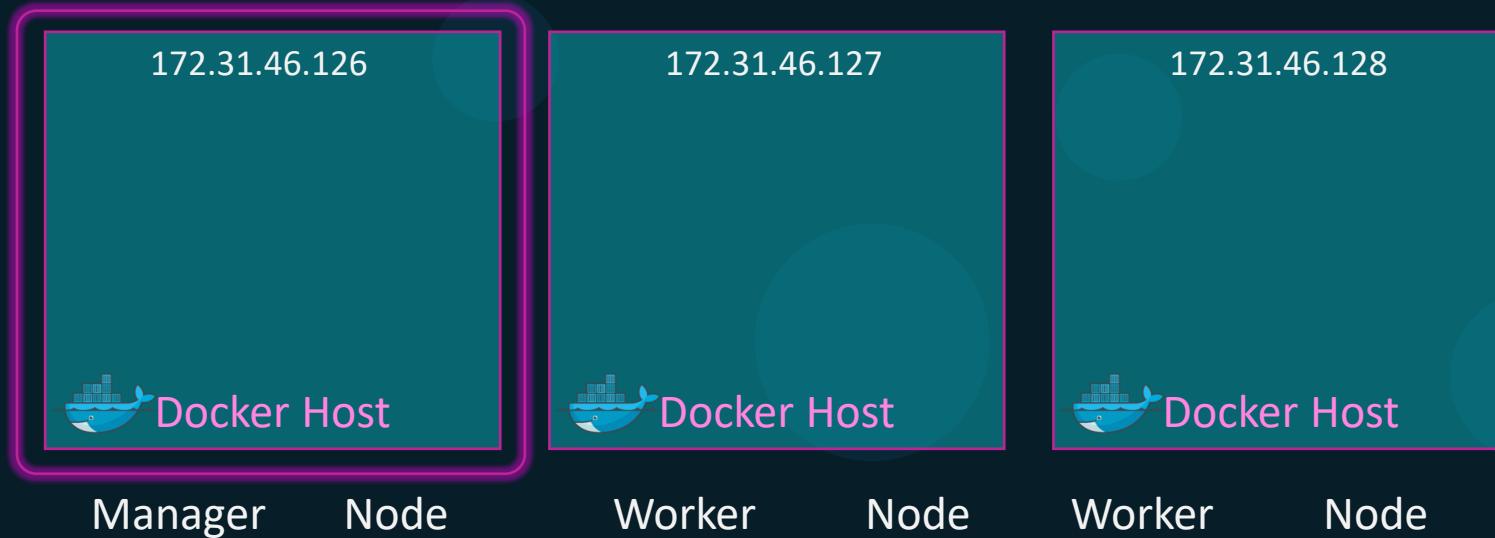
Setup swarm



Pre-Requisites



Cluster Setup



▶ docker swarm init

```
Swarm initialized: current node (91uxgq6i78j1h1u5v7moq7vgz) is now a manager.
```

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

```
docker swarm join --token SWMTKN-1-1m989y6yl10qhgyz4bqc8eks1wx13ks1vuzzi7q3tt12epcwn6-4cq5kbifs4wpkyq68n9ynxmnd  
172.31.46.126:2377
```

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

▶ docker system info | grep -i swarm

```
Swarm: active
```

Cluster Setup



```
▶ docker node ls
```

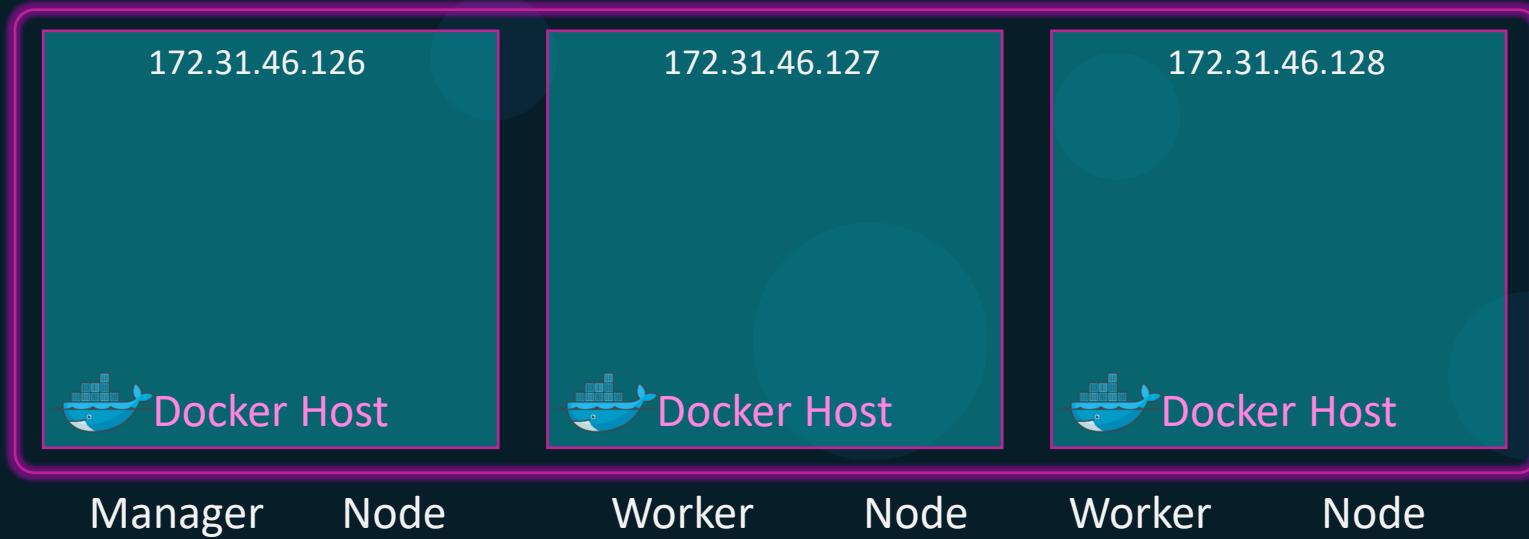
ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER	STATUS	ENGINE
91uxgq6i78j1h1u5v7moq7vgz *	manager1	Ready	Active		Leader	19.03.8
2lux7z6p96gc6vtx0h6a2wo2r	worker1	Ready	Active			19.03.8

```
▶ docker swarm join-token worker
```

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

```
docker swarm join --token SWMTKN-1-1m989y6yl10qhgyz4bqc8eks1wx13kslvuzzi7q3tt12epcwn6-4cq5kbifs4wpkyq68n9ynxmnd  
172.31.46.126:2377
```

Cluster Setup



```
▶ docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE
91uxgq6i78j1h1u5v7moq7vgz*	manager1	Ready	Active	Leader	19.03.8
2lux7z6p96gc6vtx0h6a2wo2r	worker1	Ready	Active		19.03.8
w0qr6k2ce03ojawmf1c26pvp3	worker2	Ready	Active		19.03.8

Active
Pause
Drain

Leader
Reachable
Unavailable



Cluster Setup

```
▶ docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE
91uxgq6i78j1h1u5v7moq7vgz *	manager1	Ready	Active	Leader	19.03.8
2lux7z6p96gc6vtx0h6a2wo2r	worker1	Ready	Active		19.03.8
w0qr6k2ce03ojawmf1c26pvp3	worker2	Ready	Active		19.03.8

Active
Pause
Drain

Leader
Reachable
Unavailable

```
▶ docker node inspect manager1 --pretty
```

```
ID: 91uxgq6i78j1h1u5v7moq7vgz
Hostname: manager1
Status:
  State: Ready
  Availability: Active
  Address: 172.31.46.126
Manager Status:
  Address: 172.31.46.126:2377
  Raft Status: Reachable
```



Operations

Docker Swarm



Promote a Worker to Manager

```
▶ docker node promote worker1
```

Node worker1 promoted to a manager in the swarm.

```
▶ docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE
91uxgq6i78j1h1u5v7moq7vgz *	manager1	Ready	Active	Leader	19.03.8
2lux7z6p96gc6vtx0h6a2wo2r	worker1	Ready	Active	Reachable	19.03.8
w0qr6k2ce03ojawmf1c26pvp3	worker2	Ready	Active		19.03.8

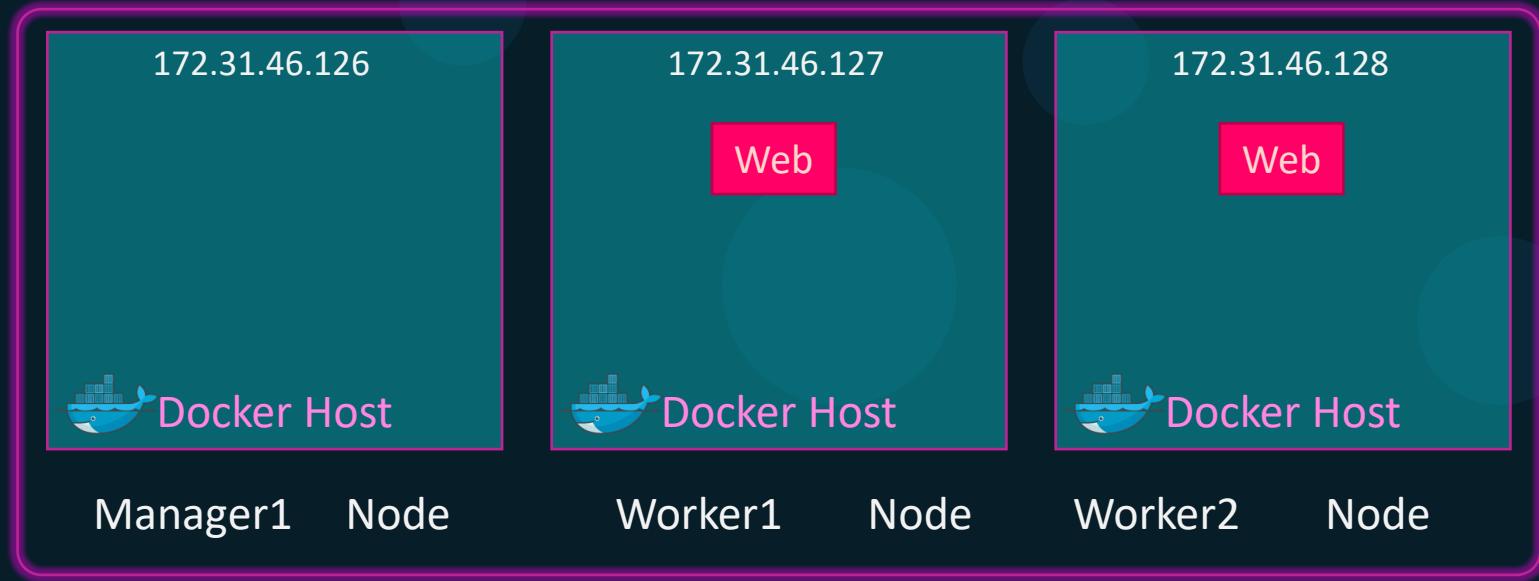
```
▶ docker node demote worker1
```

Manager worker1 demoted in the swarm.

```
▶ docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE
91uxgq6i78j1h1u5v7moq7vgz *	manager1	Ready	Active	Leader	19.03.8
2lux7z6p96gc6vtx0h6a2wo2r	worker1	Ready	Active		19.03.8
w0qr6k2ce03ojawmf1c26pvp3	worker2	Ready	Active		19.03.8

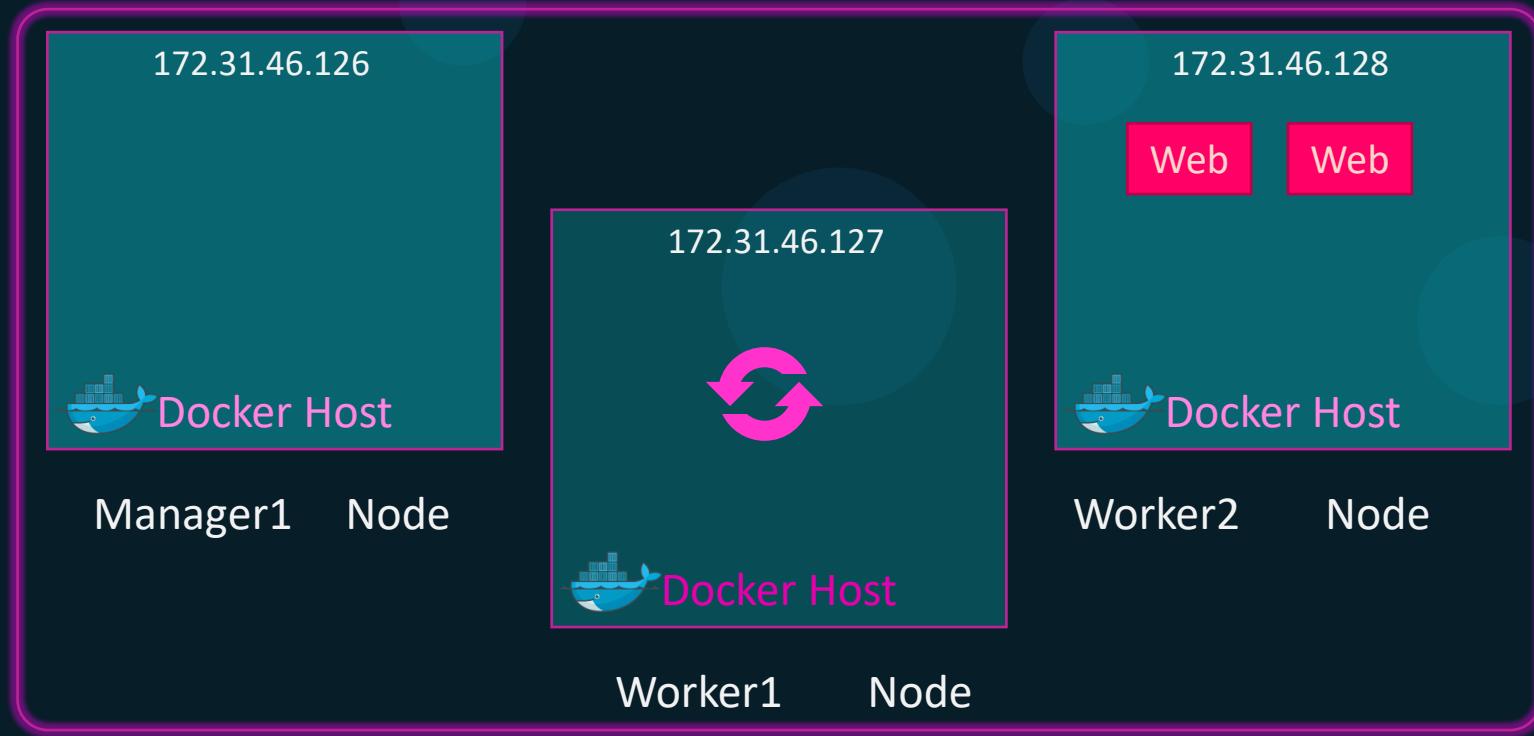
Draining A Node



```
▶ docker node update --availability drain worker1  
worker1
```



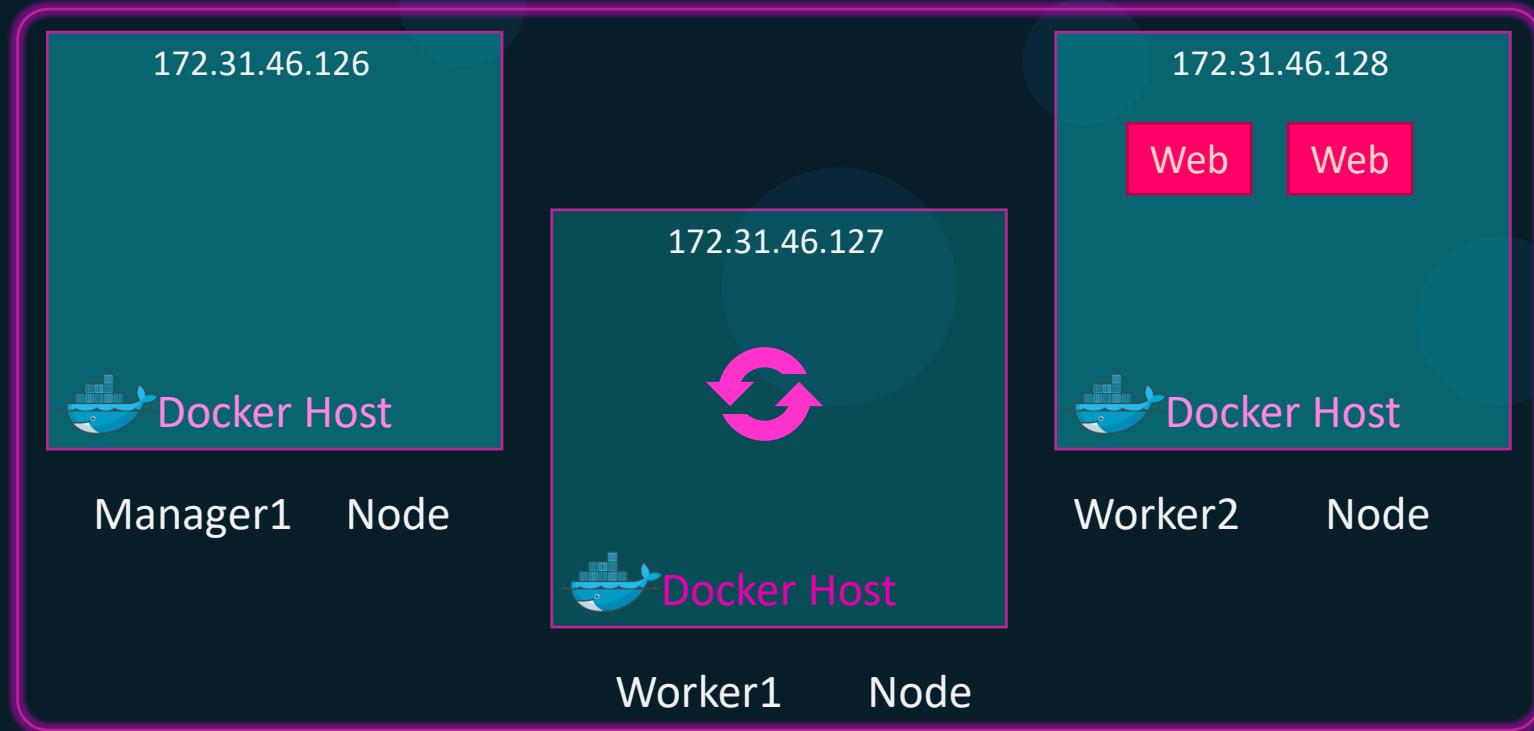
Draining A Node



```
▶ docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE
91uxgq6i78j1h1u5v7moq7vgz *	manager1	Ready	Active	Leader	19.03.8
21ux7z6p96gc6vtx0h6a2wo2r	worker1	Ready	Drain		19.03.8
w0qr6k2ce03ojawmflc26pvp3	worker2	Ready	Active		19.03.8

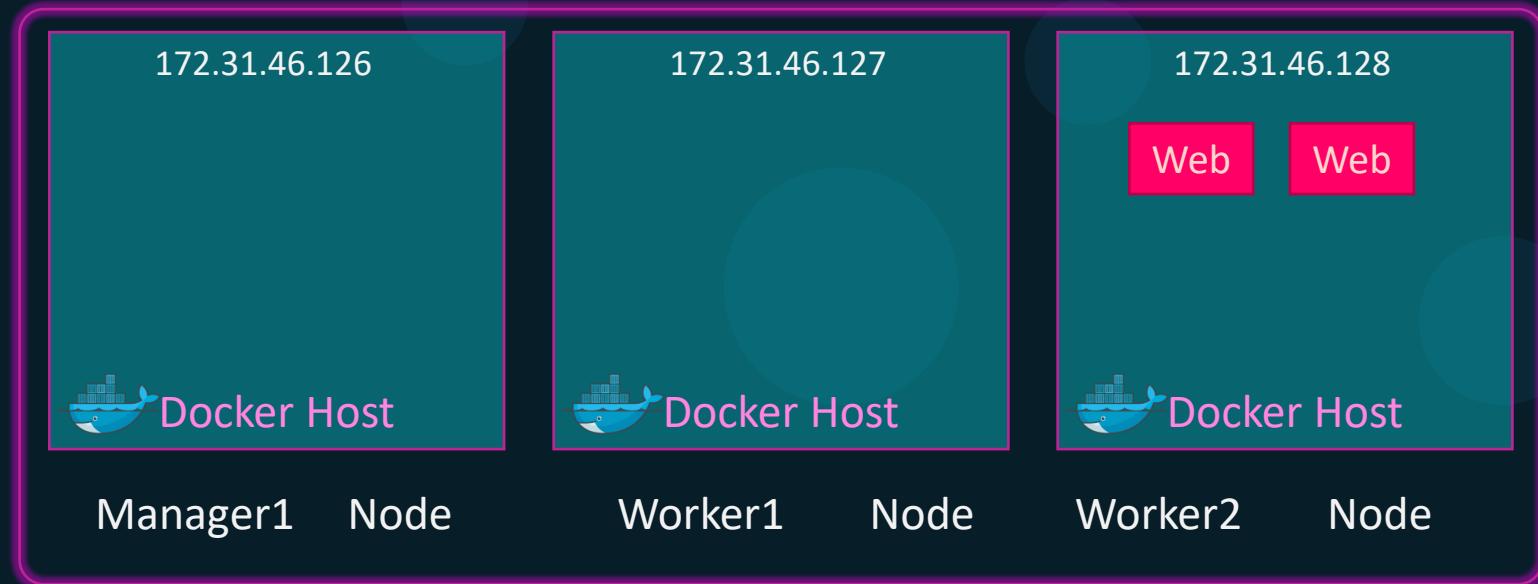
Draining A Node



```
▶ docker node update --availability active worker1  
worker1
```



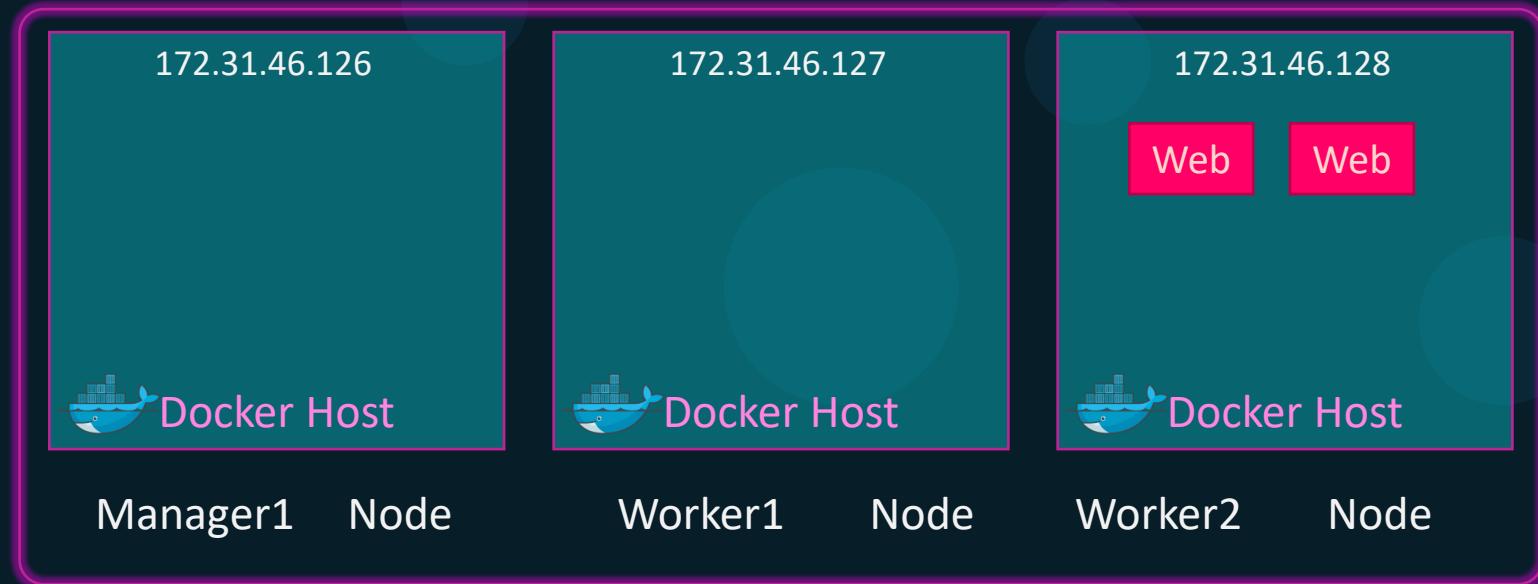
Draining A Node



```
▶ docker node update --availability active worker1  
worker1
```



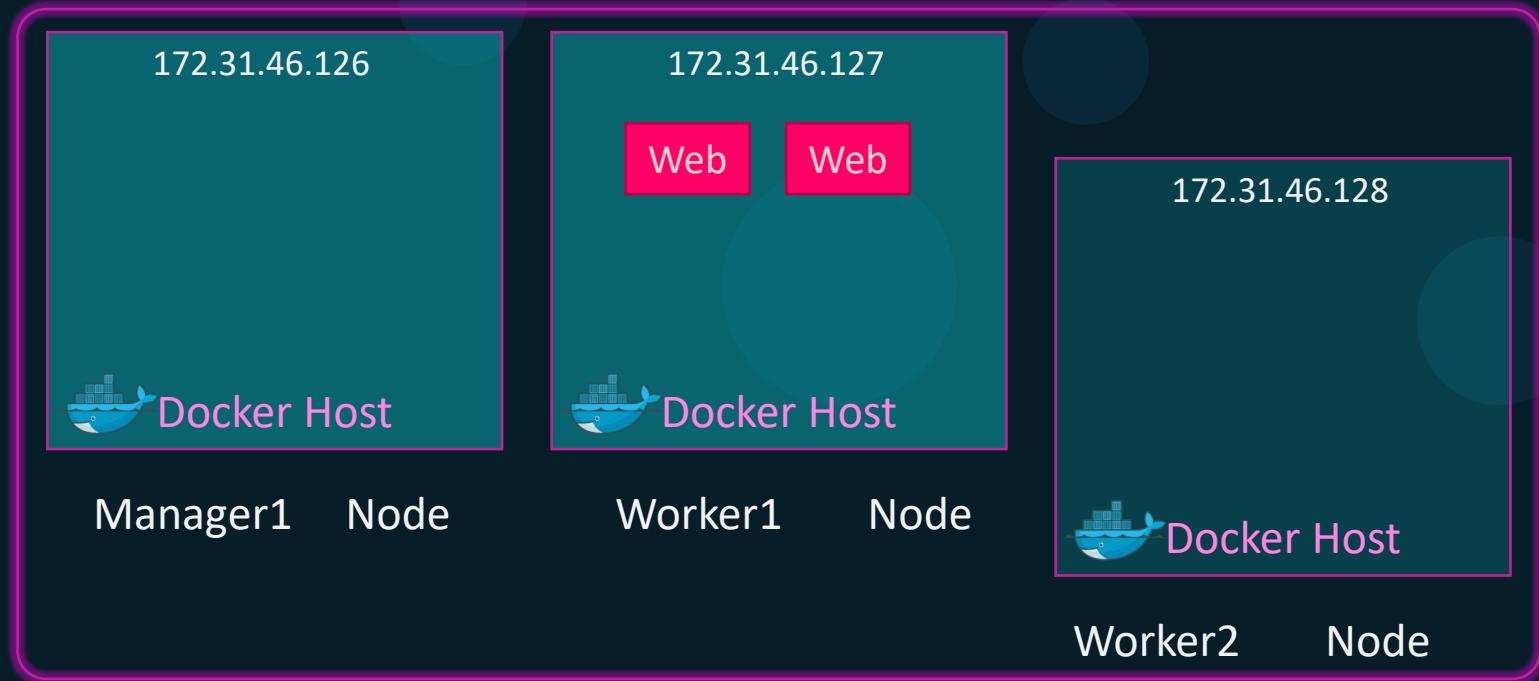
Deleting A Node



```
▶ docker node update --availability drain worker2  
worker2
```



Deleting A Node

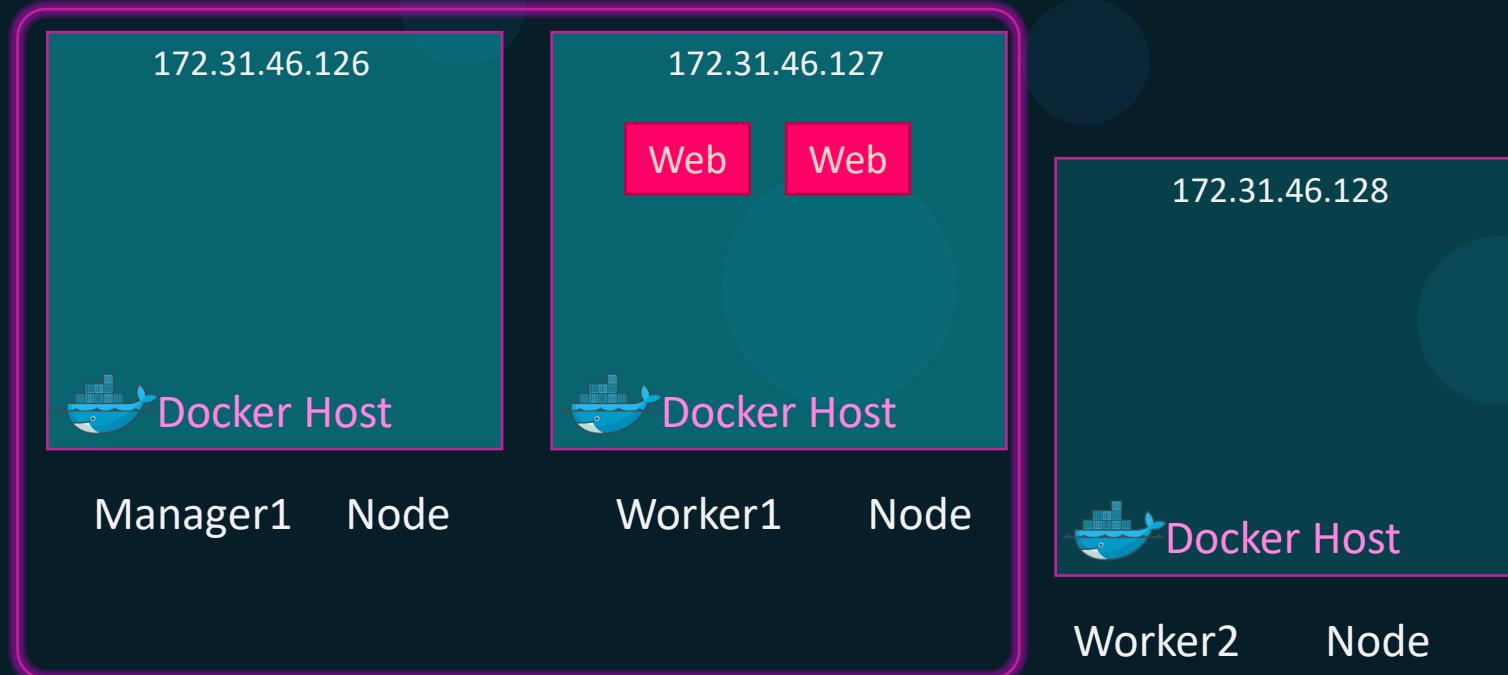


```
▶ docker node update --availability drain worker2  
worker2
```

```
▶ docker swarm leave
```



Deleting A Node



```
▶ docker node update --availability drain worker2  
worker2
```

```
▶ docker swarm leave  
Node left the swarm.
```



Talk about 12 Factor App





Manager Nodes

Docker Swarm



Manager nodes

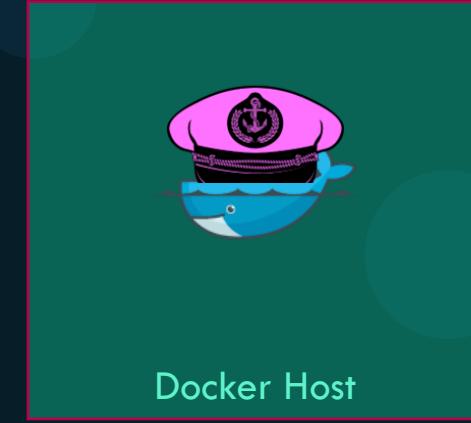
Swarm Manager



Swarm Manager



Swarm Manager



Worker



Worker



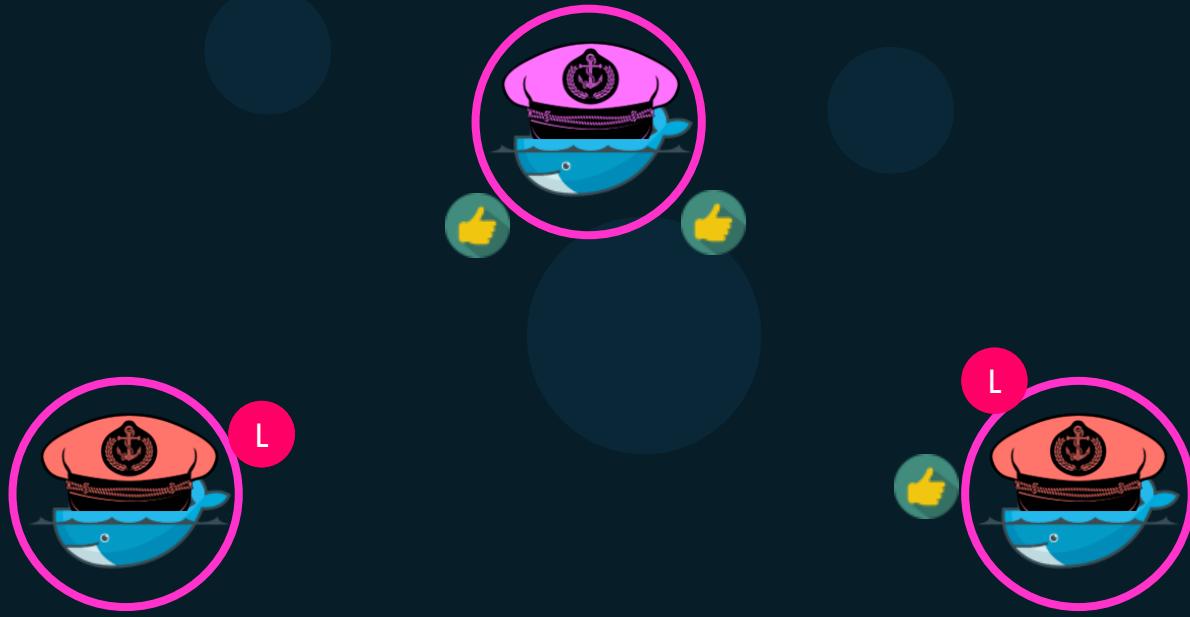
Worker



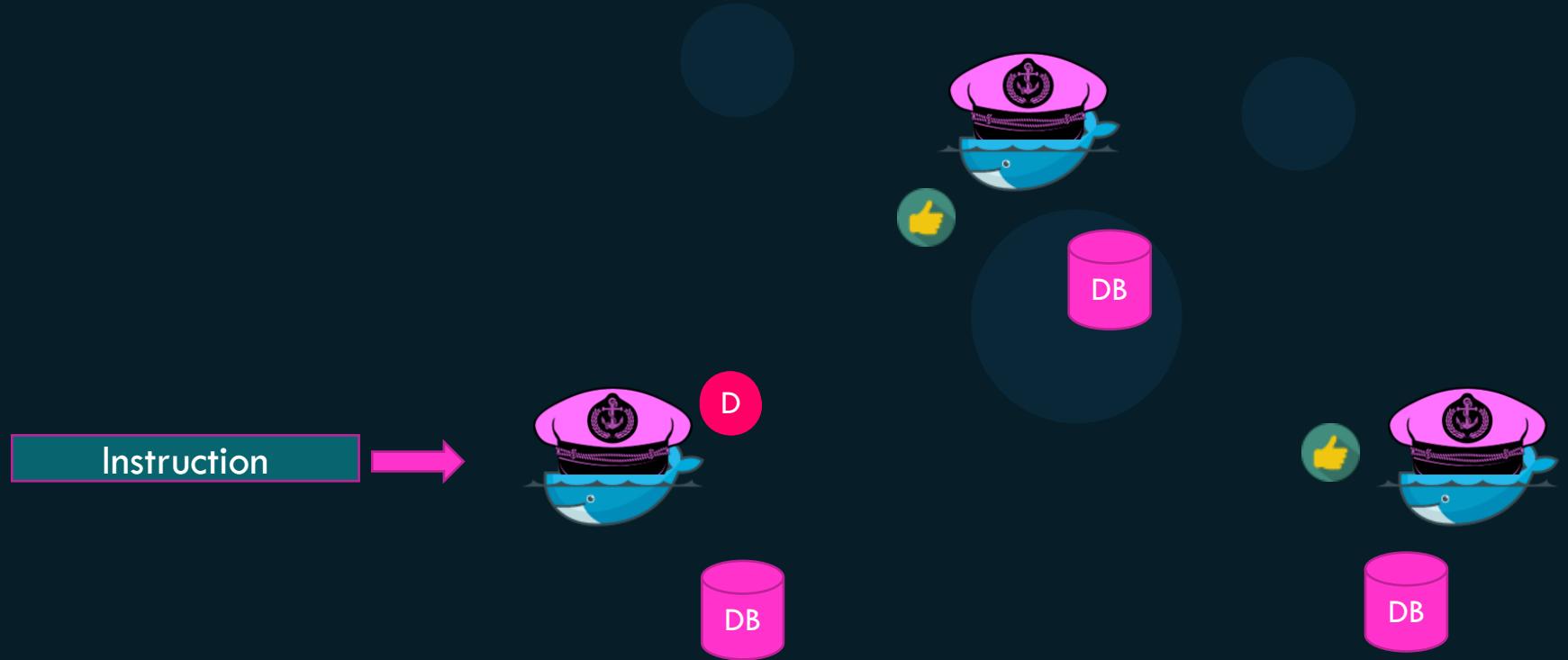
Worker



Distributed consensus - RAFT

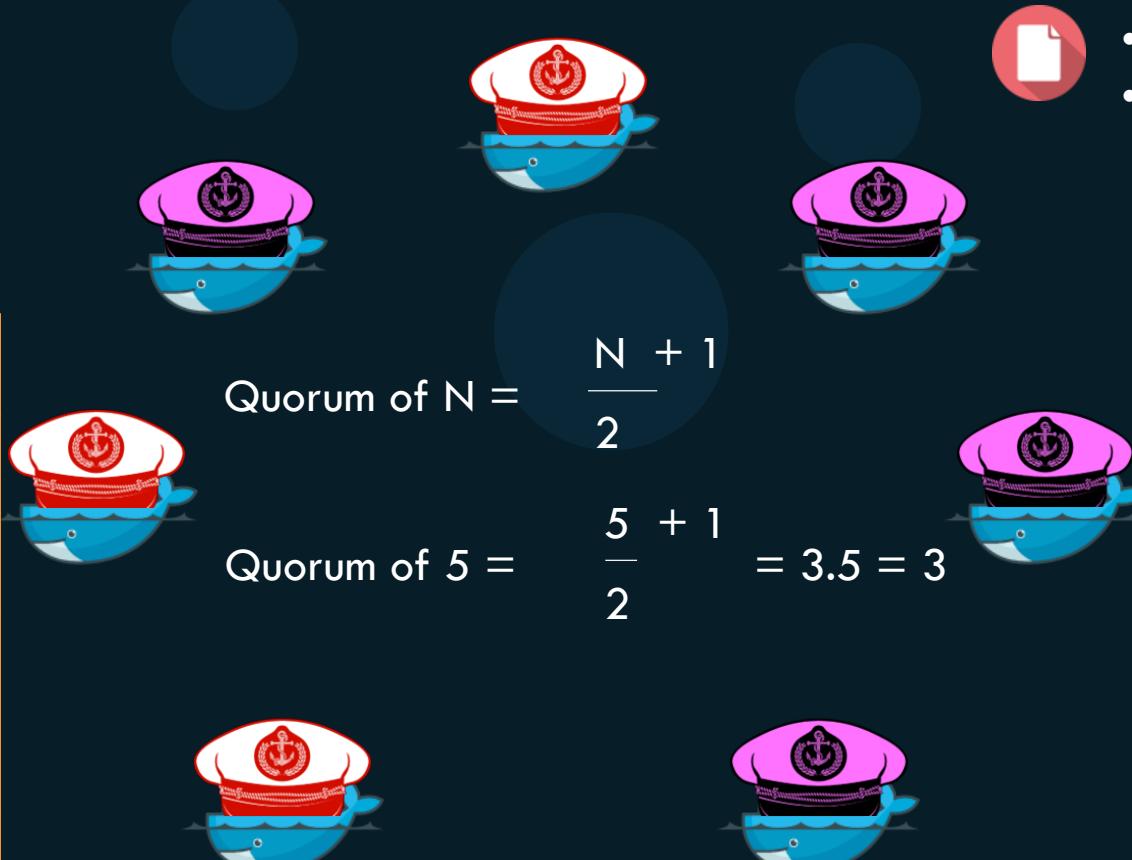


Distributed consensus - RAFT



How many Manager nodes?

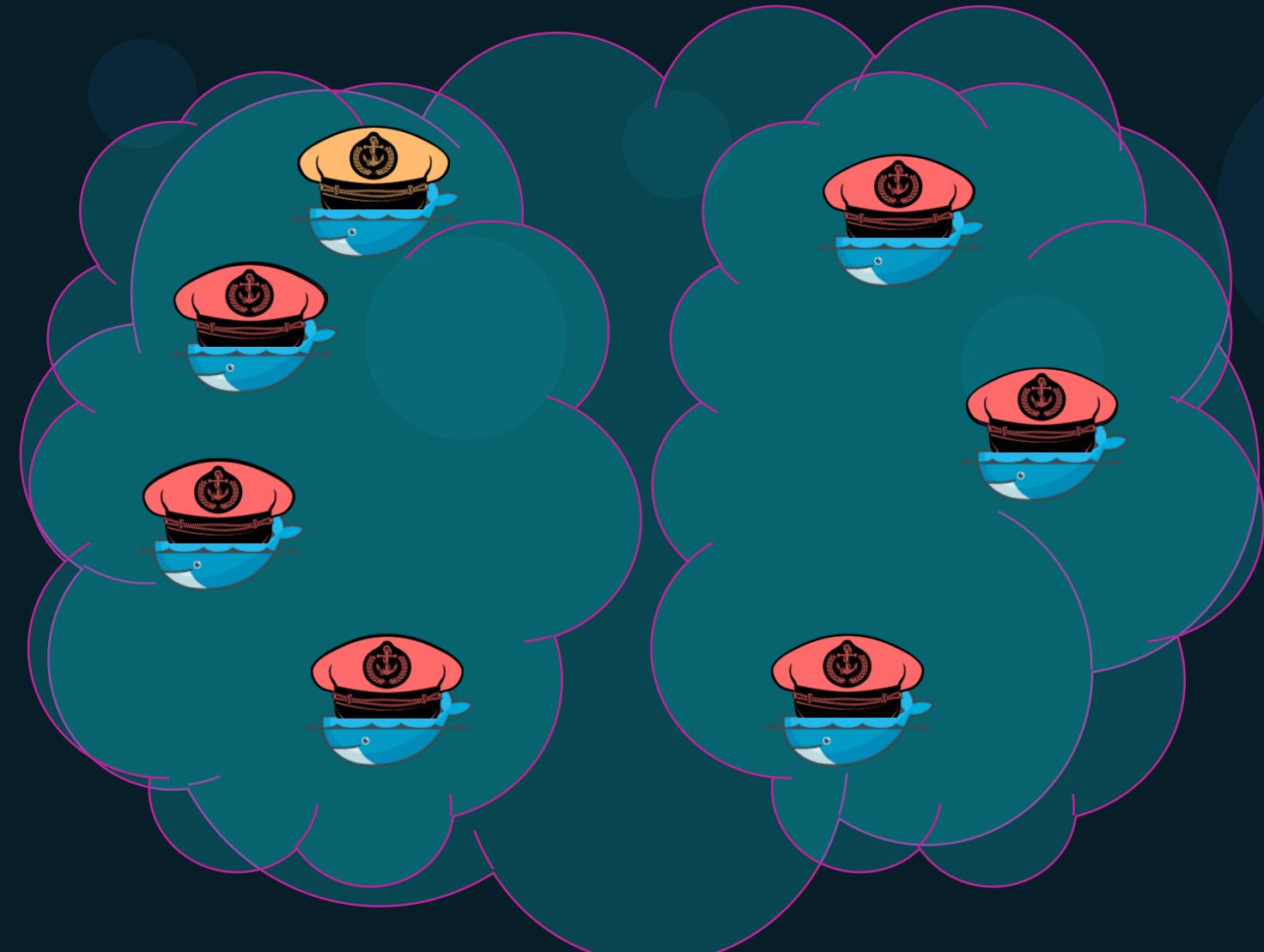
Managers	Majority	Fault Tolerance
1	1	0
2	2	0
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3


$$\text{Quorum of } N = \frac{N + 1}{2}$$
$$\text{Quorum of } 5 = \frac{5 + 1}{2} = 3.5 = 3$$
$$\text{Fault Tolerance of } N = \frac{N - 1}{2}$$

- Docker Recommends – 7 Managers
- No limit on Managers

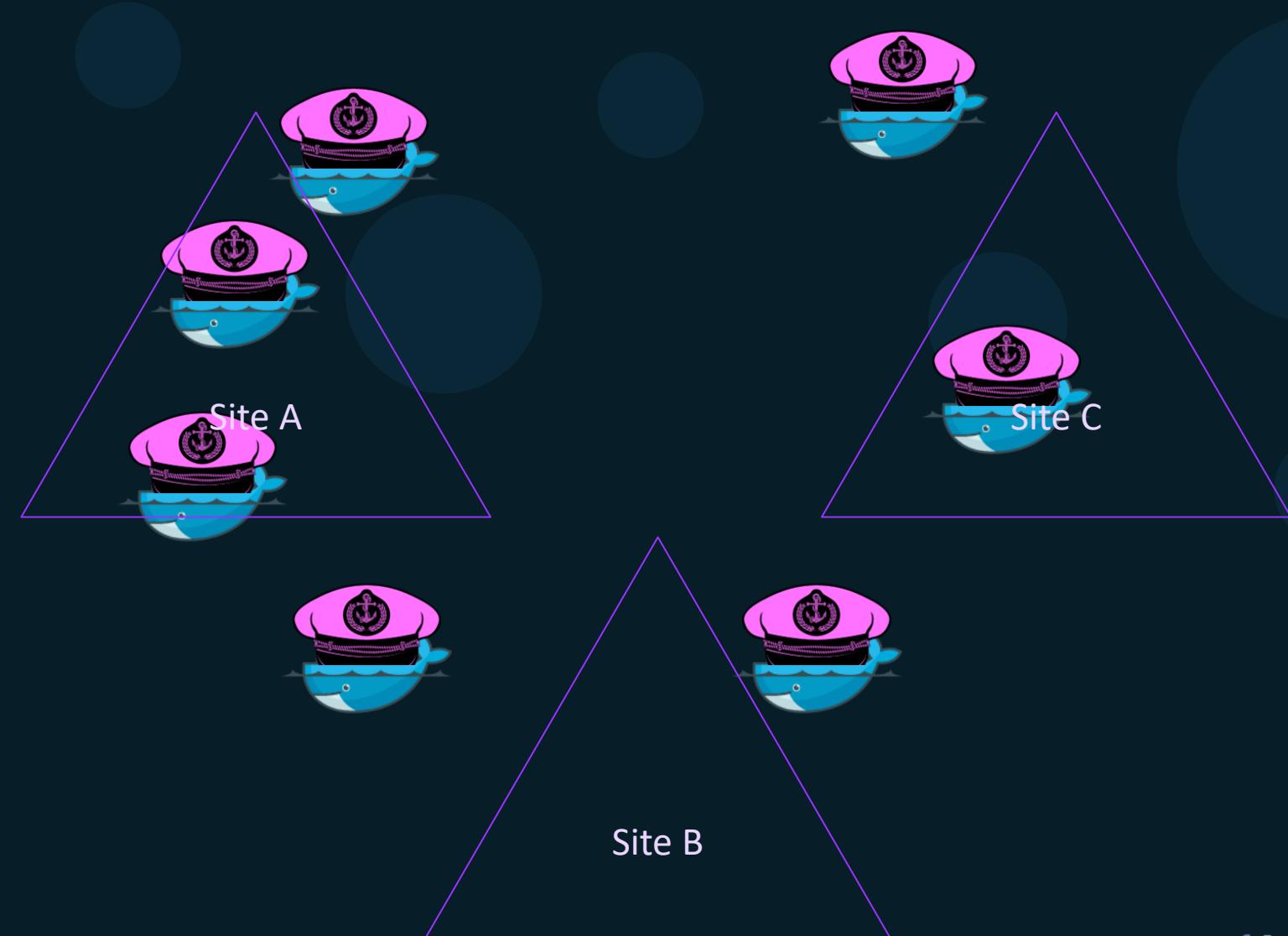
Odd or even?

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



Distributing 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



Distributing 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



Site B

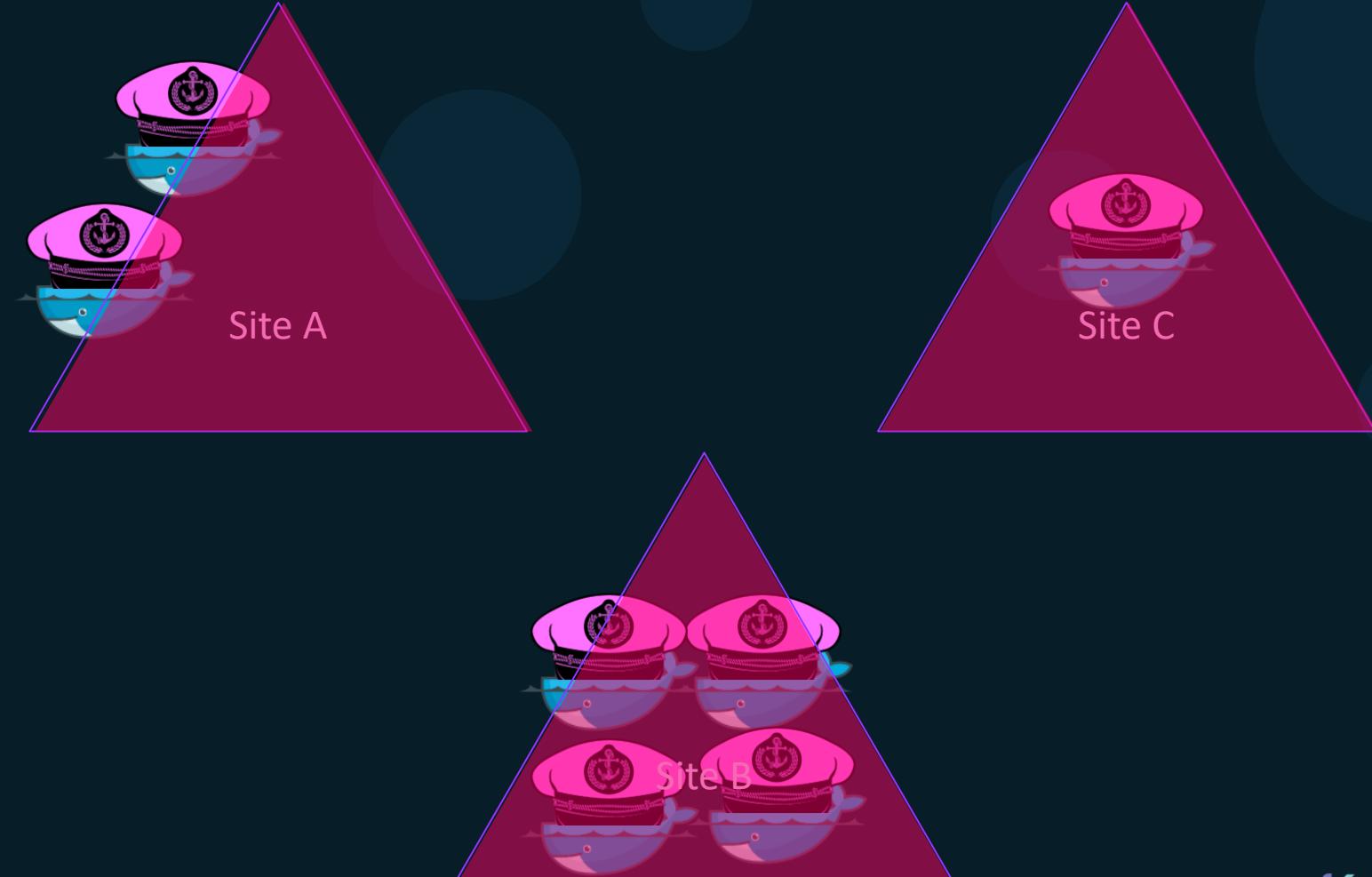


Site C



Distributing Managers

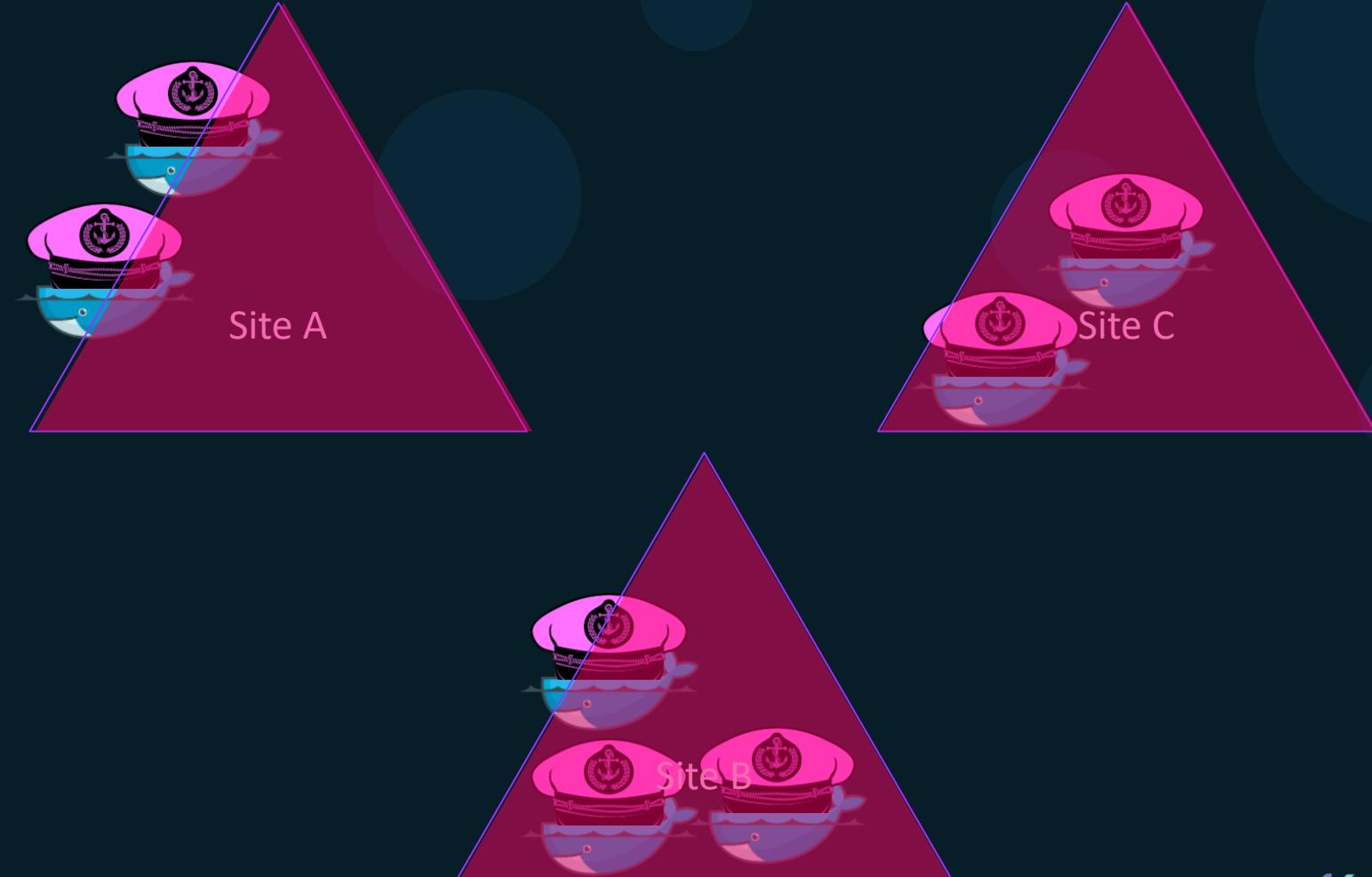
Managers	Majority	Fault Tolerance
1	1	0
2	2	0
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3



Distributing Managers

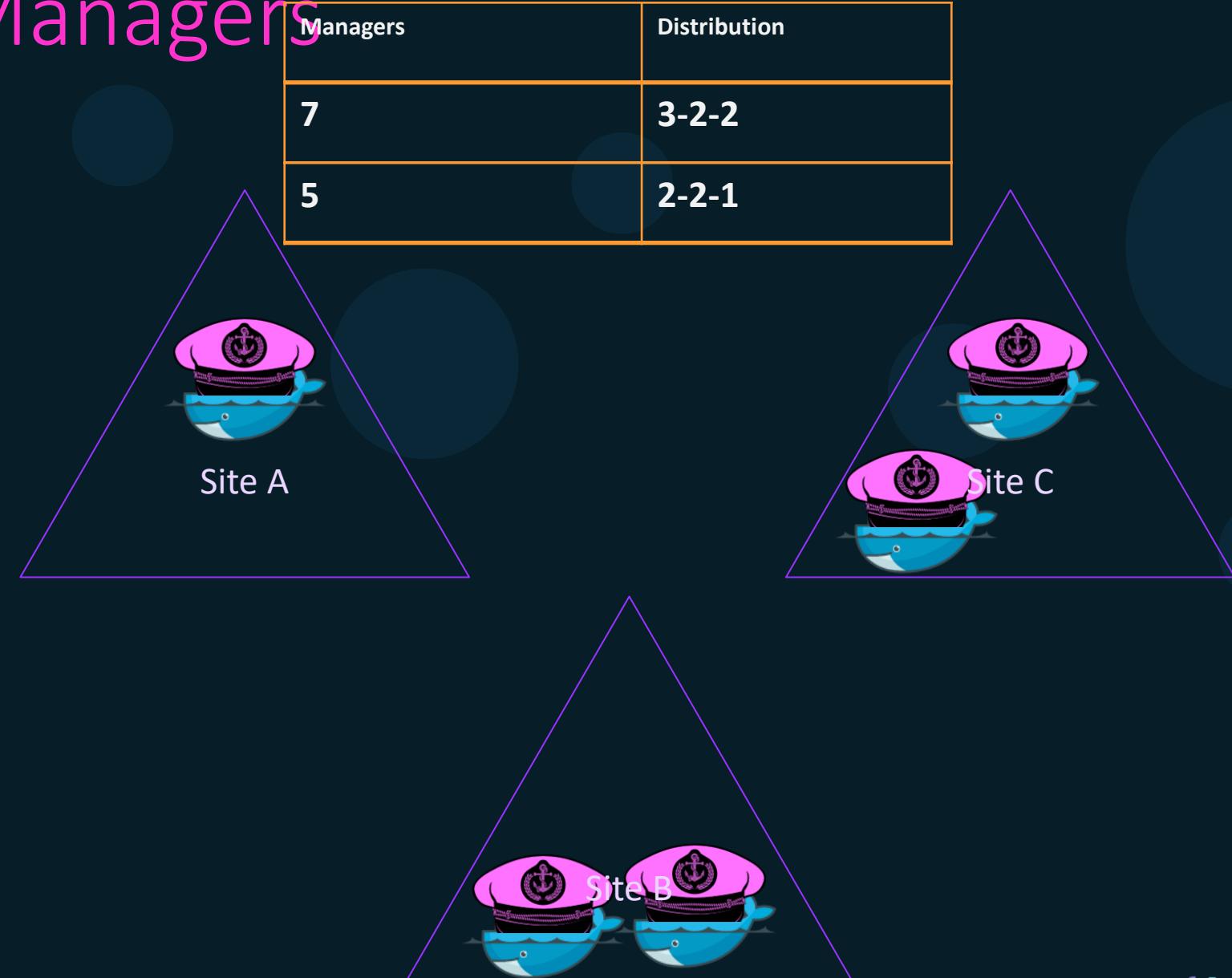
Managers	Distribution
7	3-2-2

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



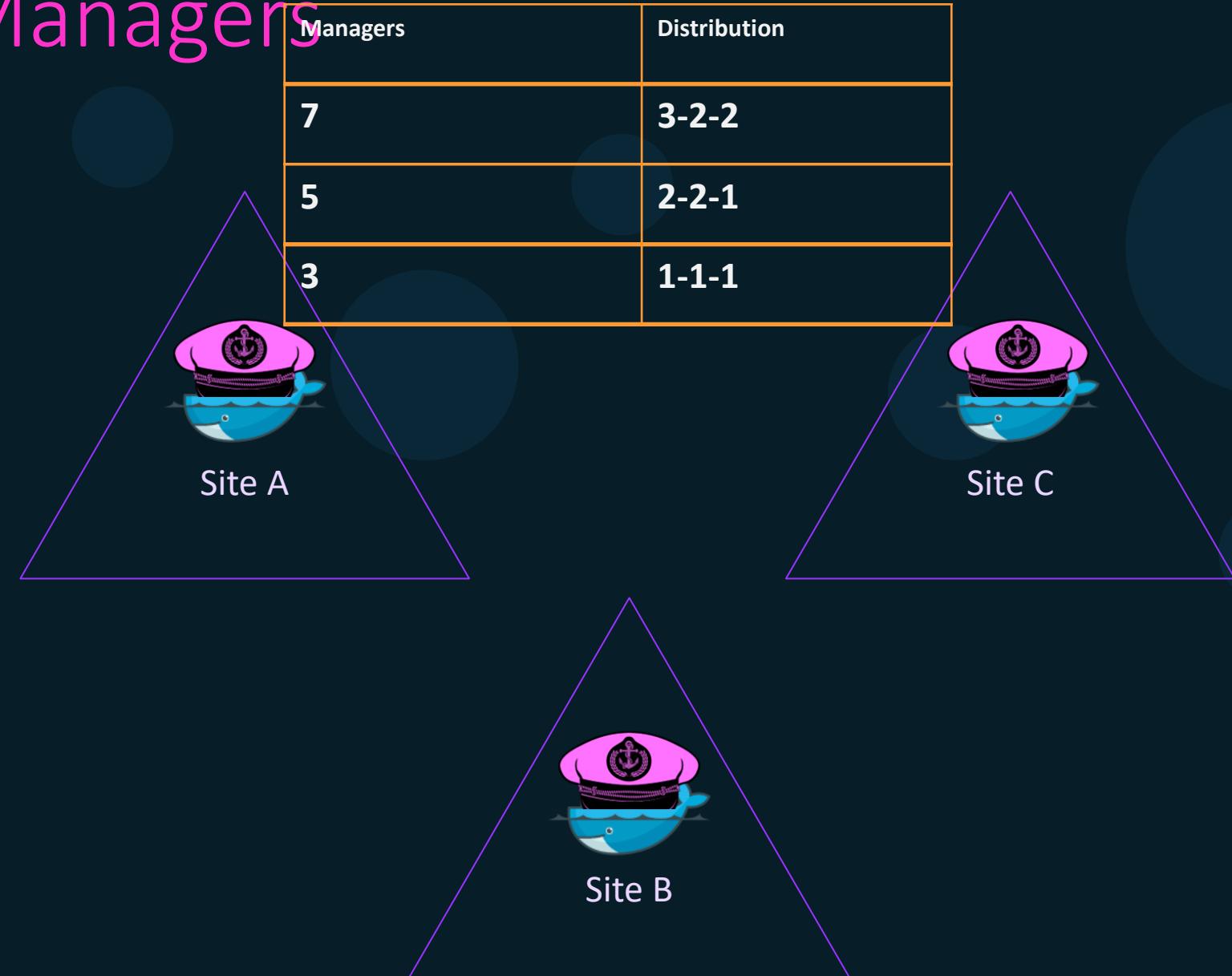
Distributing Managers

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

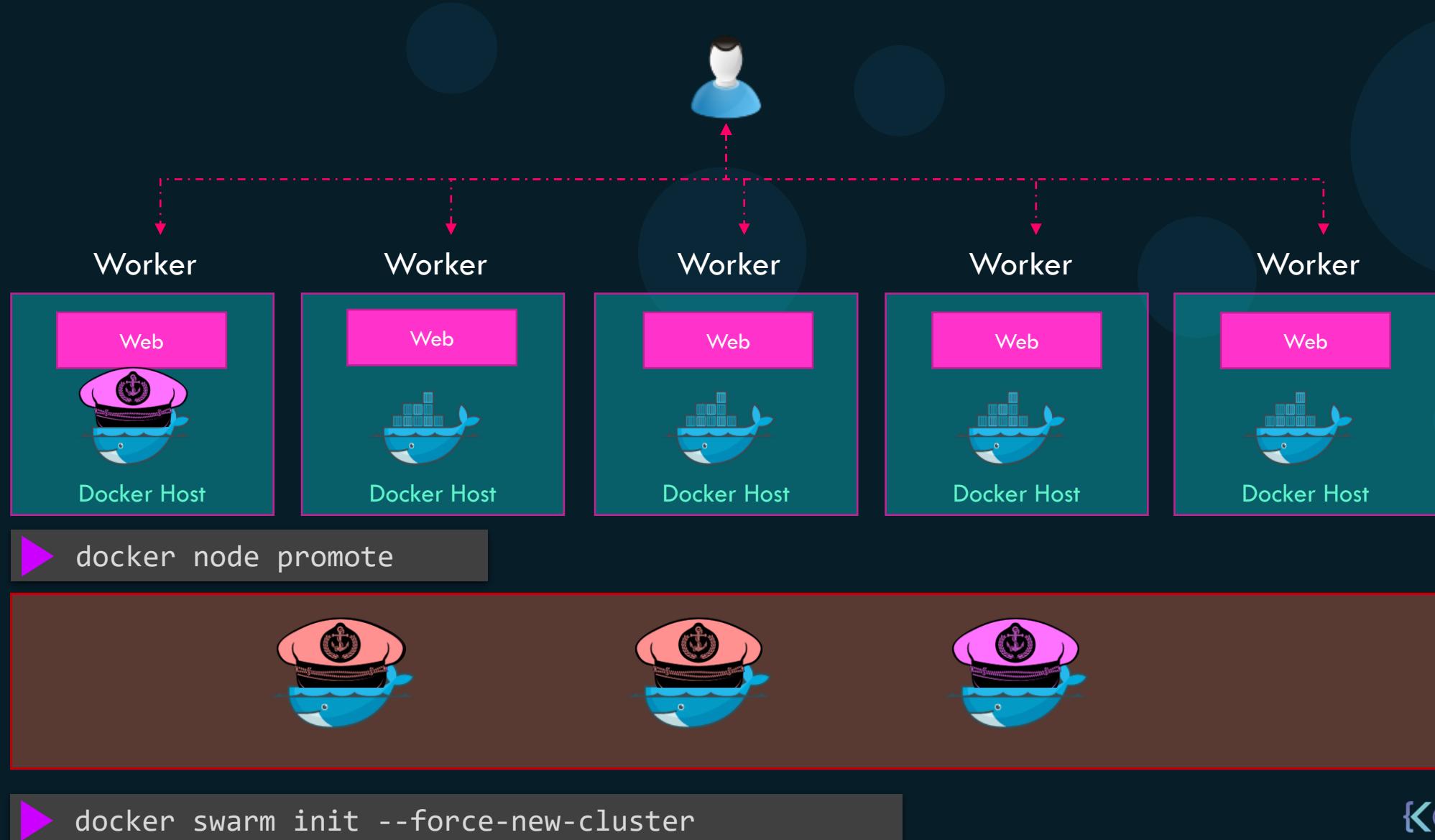


Distributing Managers

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



What happens when it fails?



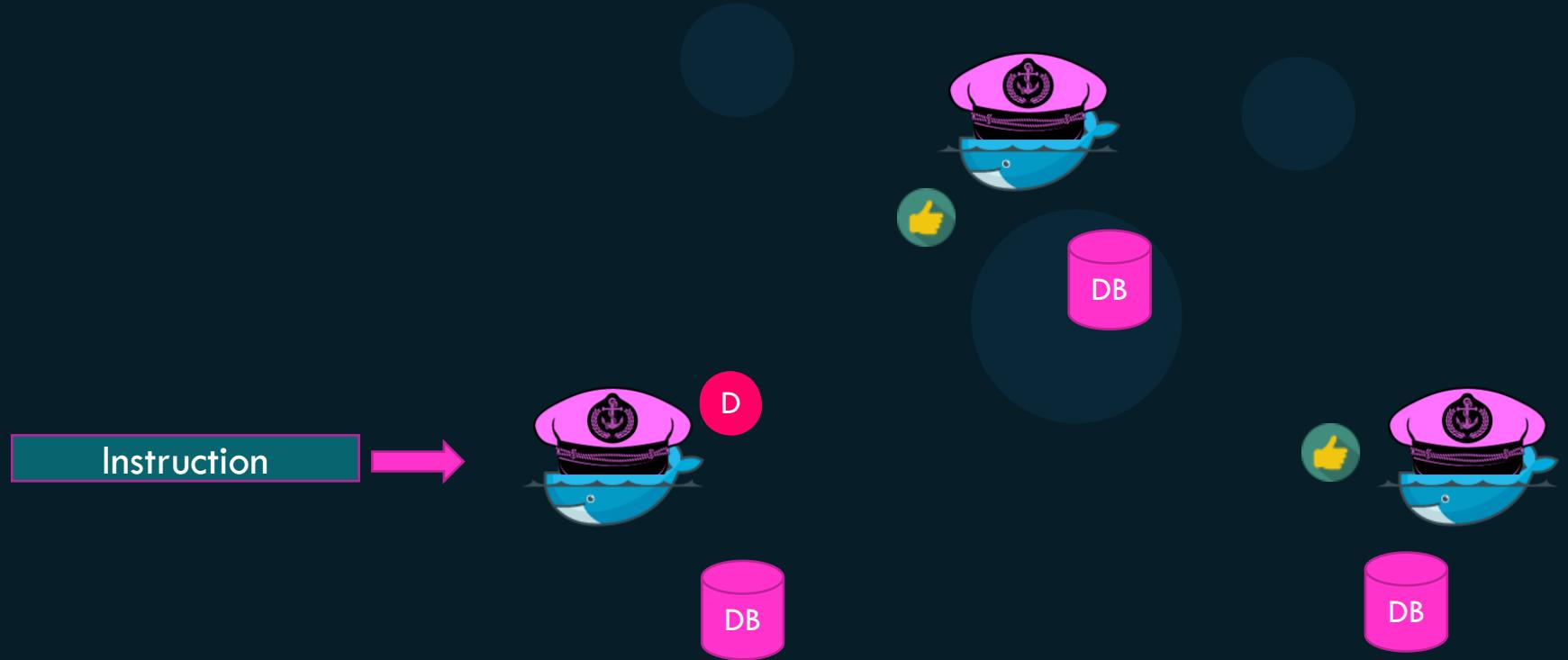


KODEKLOUD



Locking your
swarm cluster

Distributed consensus - RAFT



Lock your Swarm Cluster

```
▶ docker swarm init --autolock=true
```

```
▶ docker swarm update --autolock=true
```

Swarm updated.

To unlock a swarm manager after it restarts, run the `docker swarm unlock` command and provide the following key:

```
SWMKEY-1-7K9wg5n85QeC4Zh7rZ0vSV0b5MteDsUvpVhG/lQnb10
```

Please remember to store this key in a password manager, since without it you will not be able to restart the manager.



Unlock and Join back to Swarm Cluster

```
▶ docker node ls
```

```
Error response from daemon: Swarm is encrypted and needs to be unlocked before it can be used. Please use "docker swarm unlock" to unlock it.
```

```
▶ docker swarm unlock
```

```
Please enter unlock key: SWMKEY-1-7K9wg5n85QeC4Zh7rZ0vSV0b5MteDsUvpVhG/lQnbl0
```



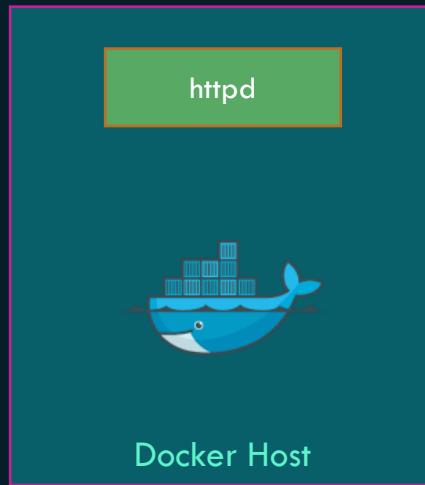


Swarm Services

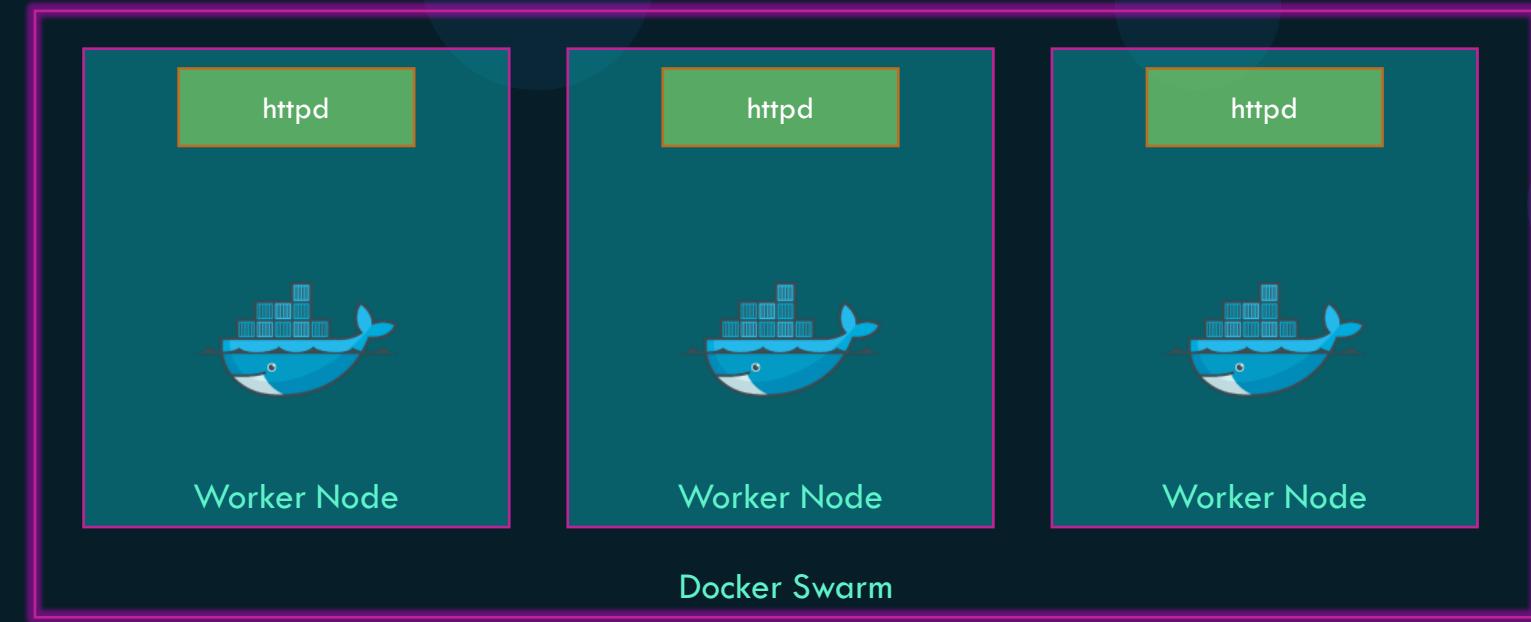


Docker service

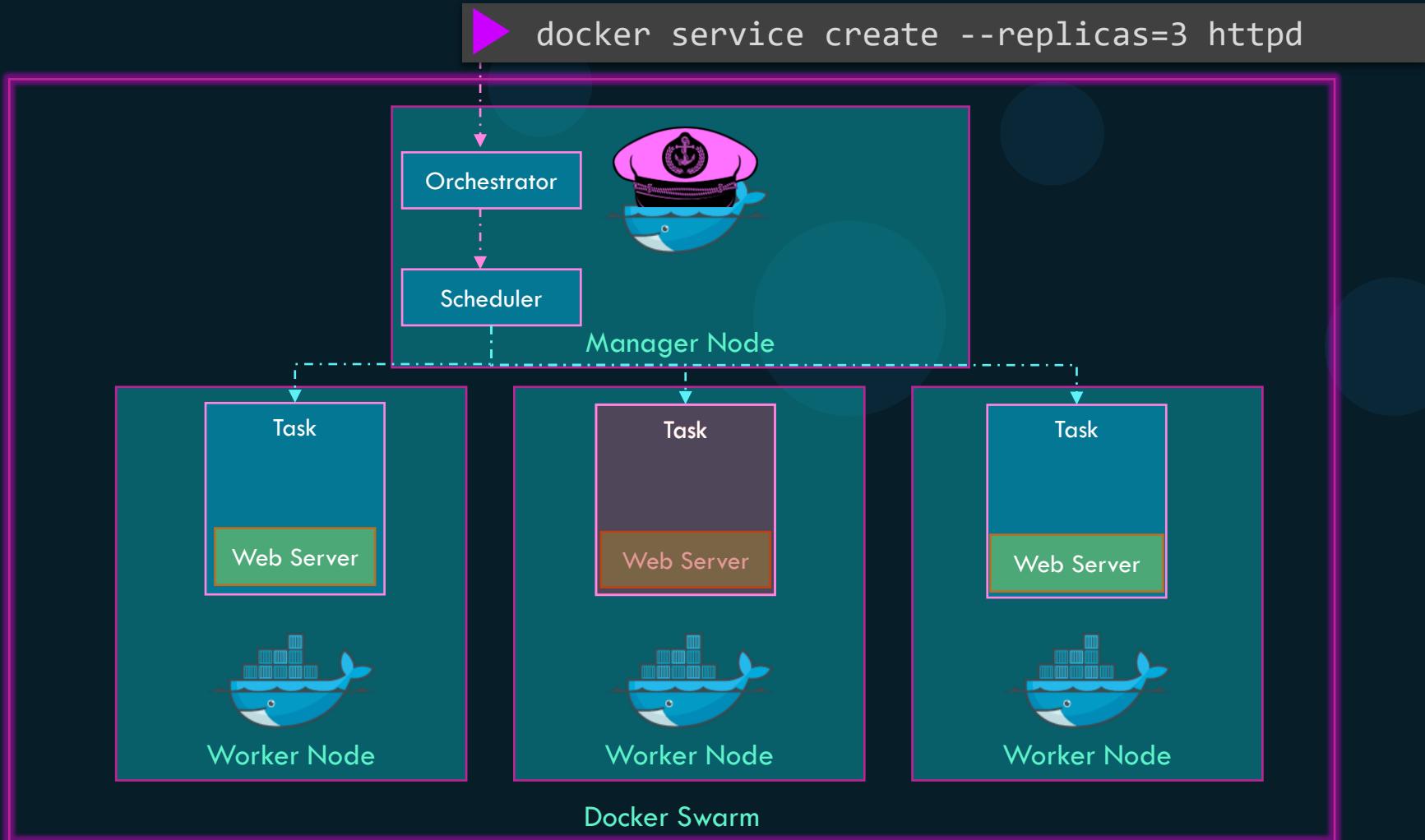
```
▶ docker run httpd
```



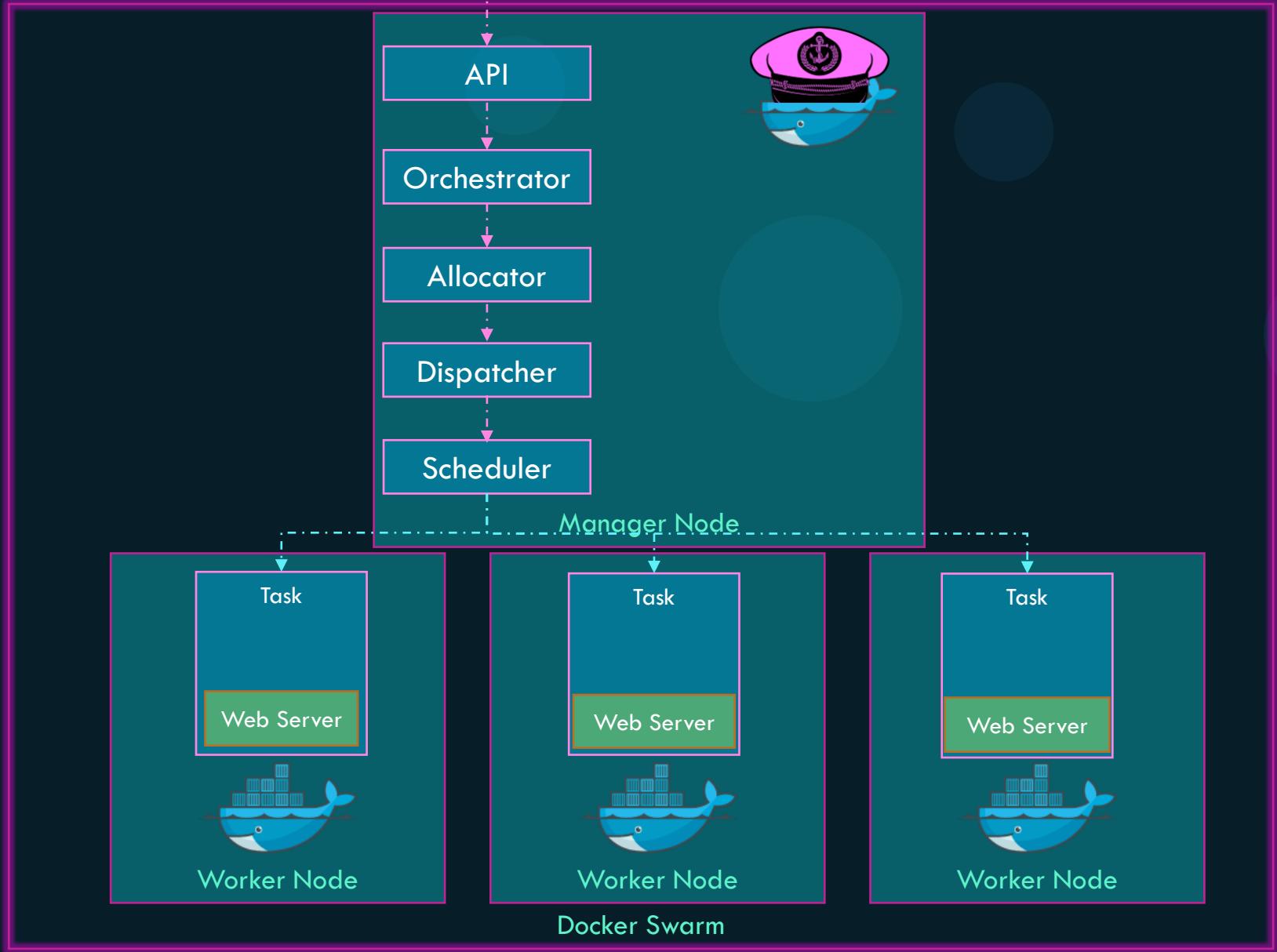
```
▶ docker service create --replicas=3 httpd
```



Tasks



Tasks



Service Creation

```
▶ docker service create --name=firstservice -p 80:80 httpd:alpine
```

```
3zhe91mns5vzi6dyyqhld177c
overall progress: 1 out of 1 tasks
1/1: running  [=====>]
verify: Service converged
```

```
▶ docker service ls
```

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
3zhe91mns5vz	firstservice	replicated	1/1	httpd:alpine	*:80->80/tcp

```
▶ docker service ps firstservice
```

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE PORTS
cfxpavgps2cy	firstservice.1	httpd:alpine	worker1	Running	Running 2 minutes ago



Service Inspect

```
▶ docker service inspect firstservice --pretty
```

```
ID:          3zhe91mns5vzi6dyyqhld177c
Name:        firstservice
Service Mode: Replicated
Replicas:    1
Placement:
UpdateConfig:
  Parallelism: 1
  On failure:  pause
  Monitoring Period: 5s
  Max failure ratio: 0
  Update order:  stop-first
RollbackConfig:
  Parallelism: 1
  On failure:  pause
  Monitoring Period: 5s
  Max failure ratio: 0
  Rollback order:  stop-first
ContainerSpec:
  Image:      httpd:alpine@sha256:30a98fa70cb11a4b388328c8512c5cd2528b3c0bd4c4f02def164f165cbb153e
  Init:       false
Resources:
Endpoint Mode: vip
Ports:
  PublishedPort = 80
  Protocol = tcp
  TargetPort = 80
  PublishMode = ingress
```



Service Logs

▶ docker service logs firstservice

```
firstservice.1.cfxpavgps2cy@worker1 | AH00557: httpd: apr_sockaddr_info_get() failed for  
06235d80b97e  
firstservice.1.cfxpavgps2cy@worker1 | AH00558: httpd: Could not reliably determine the  
server's fully qualified domain name, using 127.0.0.1. Set the 'ServerName' directive globally  
to suppress this message  
firstservice.1.cfxpavgps2cy@worker1 | AH00557: httpd: apr_sockaddr_info_get() failed for  
06235d80b97e  
firstservice.1.cfxpavgps2cy@worker1 | AH00558: httpd: Could not reliably determine the  
server's fully qualified domain name, using 127.0.0.1. Set the 'ServerName' directive globally  
to suppress this message  
firstservice.1.cfxpavgps2cy@worker1 | [Fri Apr 24 18:55:56.440200 2020] [mpm_event:notice]  
[pid 1:tid 139963811605832] AH00489: Apache/2.4.43 (Unix) configured -- resuming normal  
operations  
firstservice.1.cfxpavgps2cy@worker1 | [Fri Apr 24 18:55:56.440244 2020] [core:notice] [pid  
1:tid 139963811605832] AH00094: Command line: 'httpd -D FOREGROUND'  
firstservice.1.cfxpavgps2cy@worker1 | 10.0.0.7 - - [24/Apr/2020:18:56:10 +0000] "POST /cgi-  
bin/mainfunction.cgi HTTP/1.1" 400 226  
firstservice.1.cfxpavgps2cy@worker1 | 10.0.0.4 - - [24/Apr/2020:19:00:00 +0000] "POST /cgi-  
bin/mainfunction.cgi HTTP/1.1" 400 226
```



Delete a Service

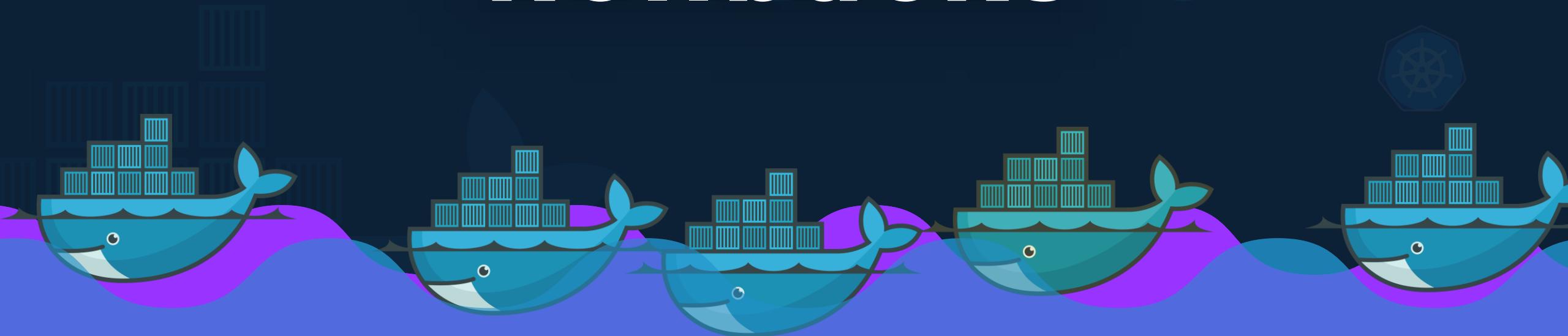
```
▶ docker service rm firstservice  
firstservice
```



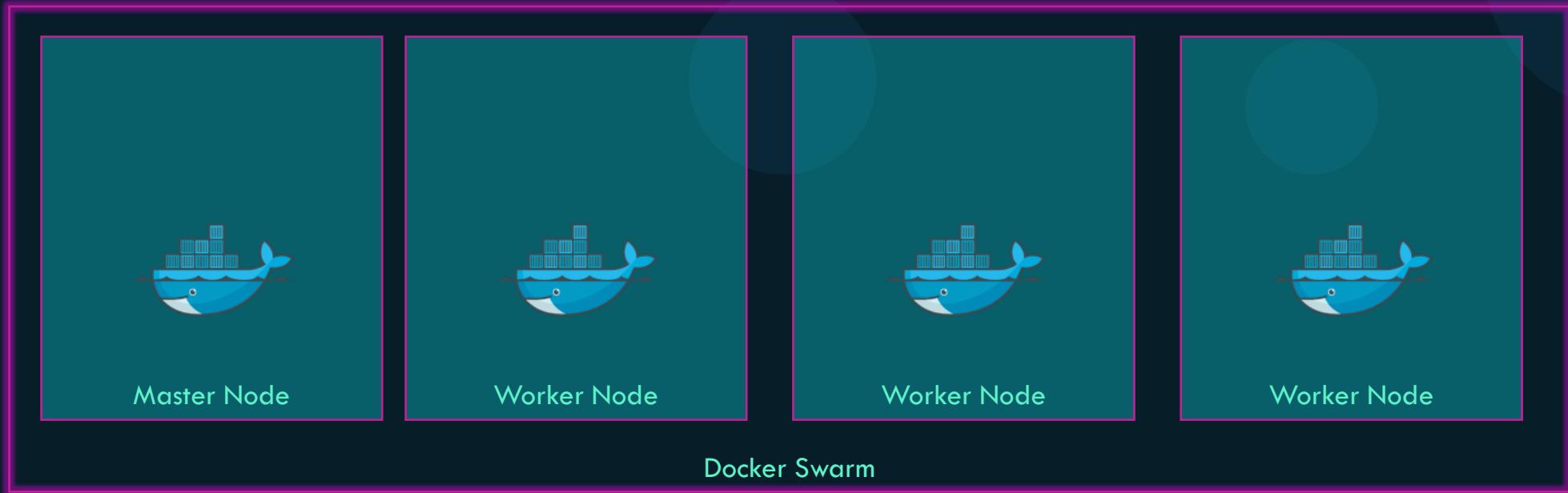


KODEKLOUD

Rolling Updates & Rollbacks

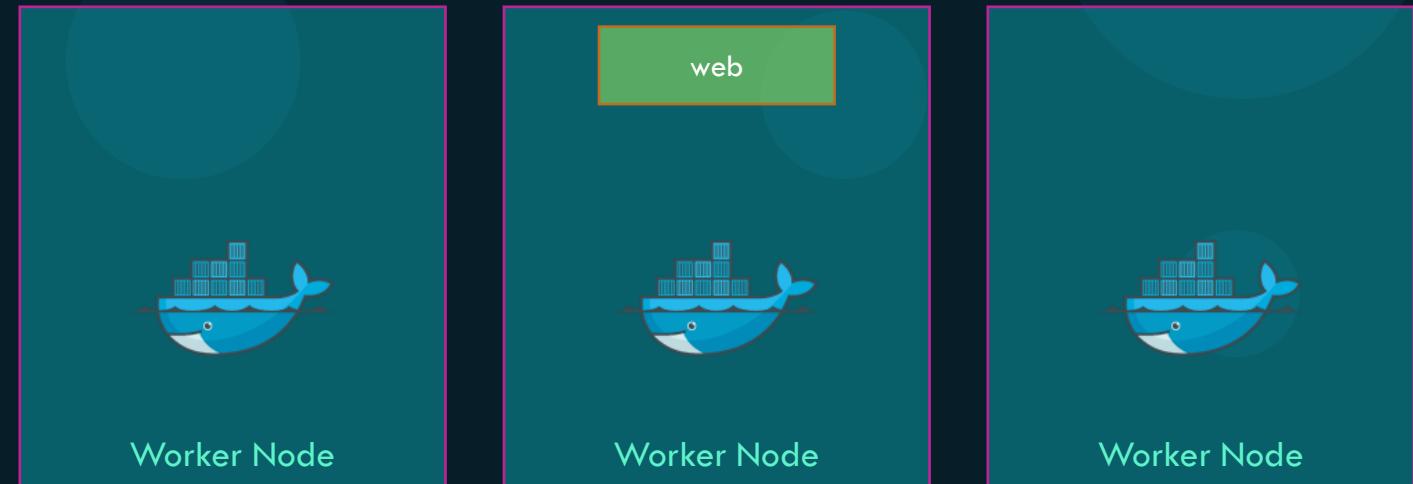


Docker Service



Docker Service

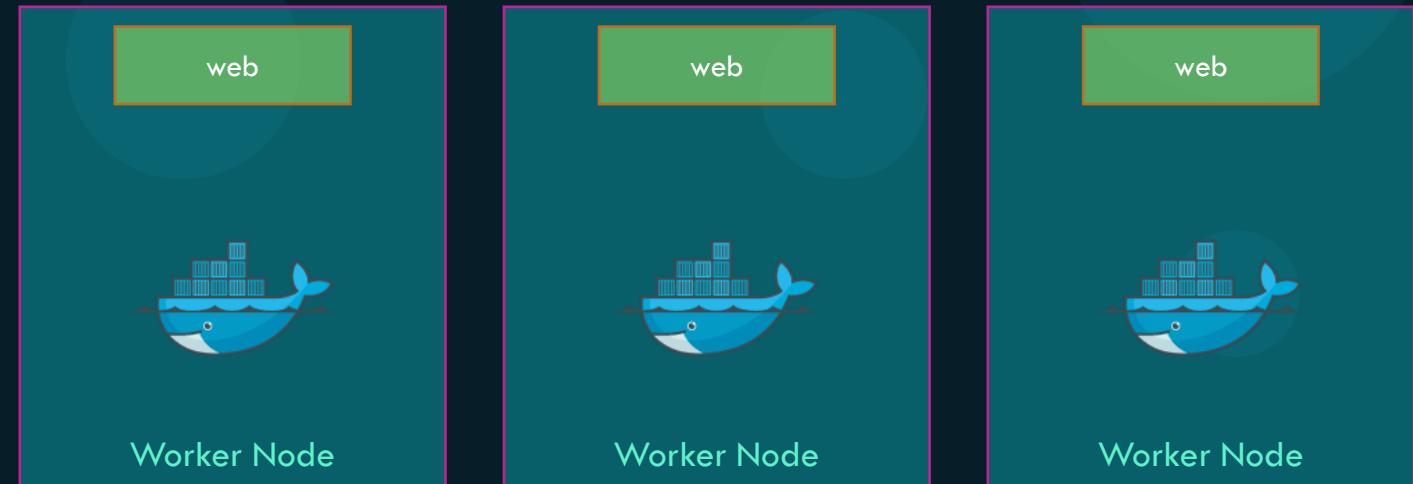
```
▶ docker service create -p 80:80 web
```



Docker Service – Scale up

```
▶ docker service create -p 80:80 web
```

```
▶ docker service update --replicas=3 -p 80:80 web
```



Docker Service – Scale up

```
▶ docker service create -p 80:80 web
```

```
▶ docker service update --replicas=3 -p 80:80 web
```



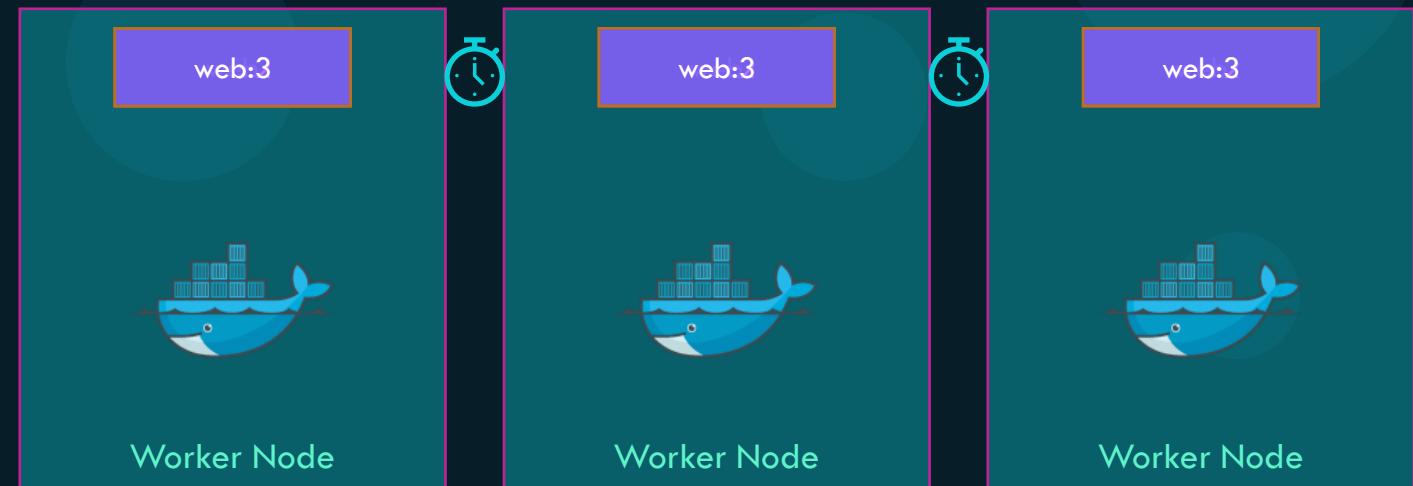
```
▶ docker service update --replicas=1 -p 80:80 web
```



Docker Service – Rolling Update

```
▶ docker service update -p 80:80 --image=web:2 web
```

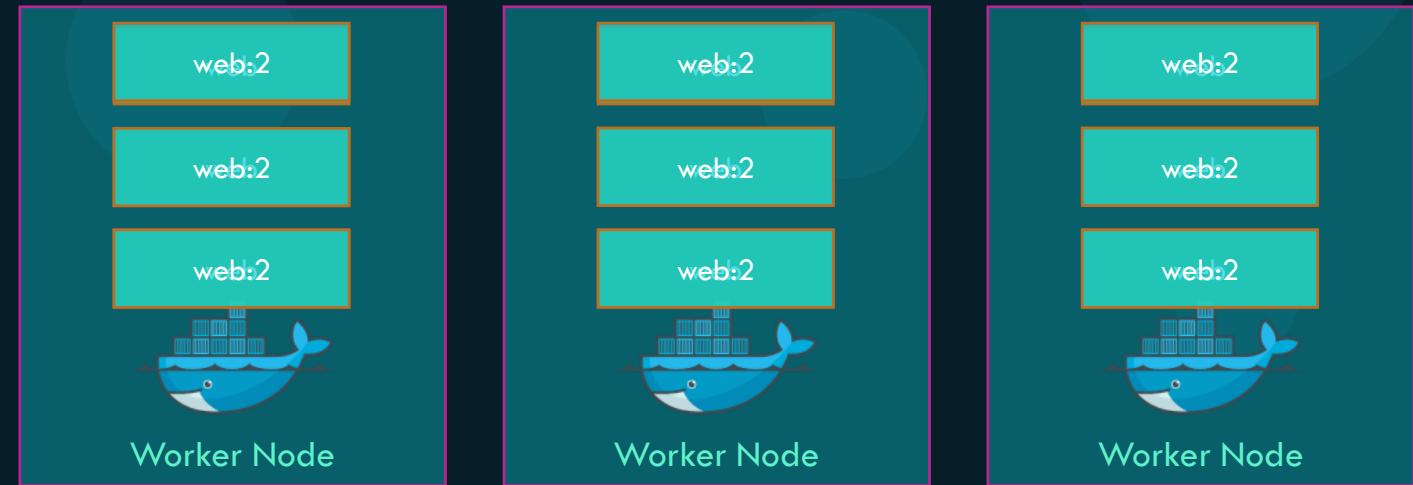
```
▶ docker service update -p 80:80 --update-delay 60s --image=web:3 web
```



Docker Service – Rolling Update

```
▶ docker service update -p 80:80 --image=web:2 web
```

```
▶ docker service update -p 80:80 --update-delay 60s --image=web:3 web
```



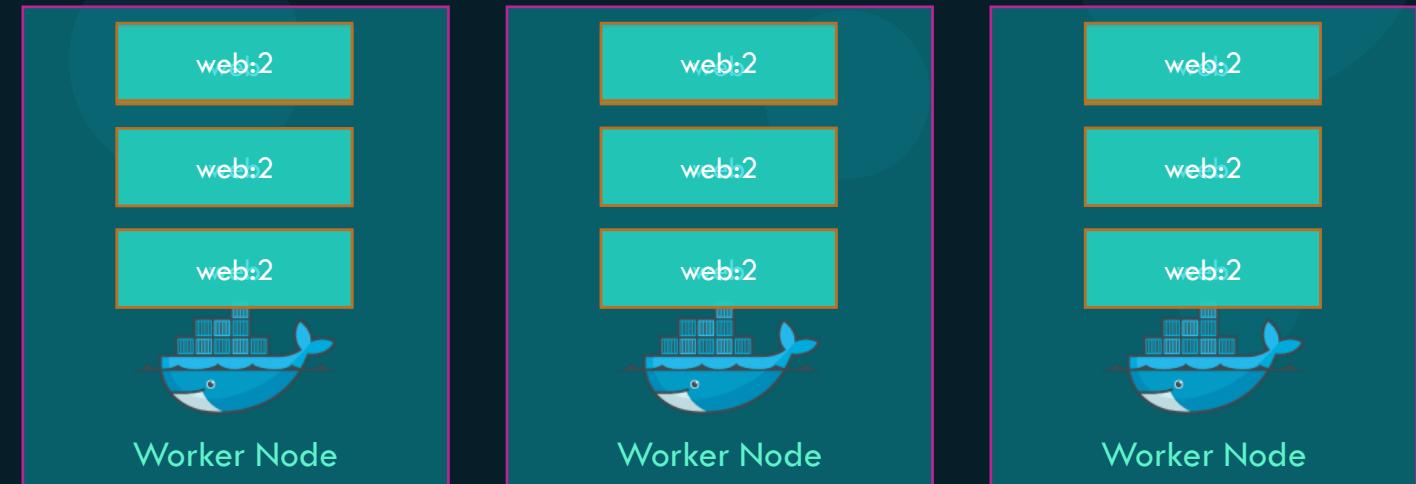
```
▶ docker service update -p 80:80 --update-parallelism 3 --image=web:2 web
```



Docker Service – Rolling Update

```
▶ docker service inspect web
```

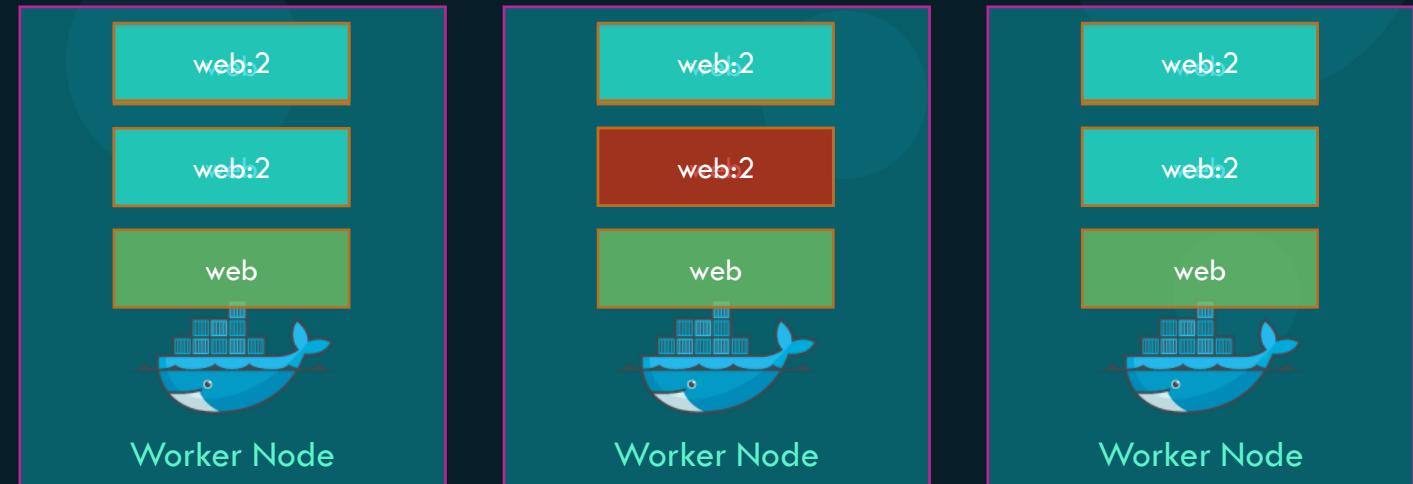
```
ID: y1k8vhoyqxulgthxrkph7xtug
Name: web
Service Mode: Replicated
Replicas: 5
Placement:
UpdateConfig:
  Parallelism: 3
  Delay: 60s
  On failure: pause
  Monitoring Period: 5s
  Max failure ratio: 0
  Update order: stop-first
RollbackConfig:
  Parallelism: 1
  On failure: pause
  Monitoring Period: 5s
  Max failure ratio: 0
  Rollback order: stop-first
ContainerSpec:
  Image: web:2...
  Init: false
Resources:
Endpoint Mode: vip
```



Docker Service – Rolling Update

```
▶ docker service inspect web
```

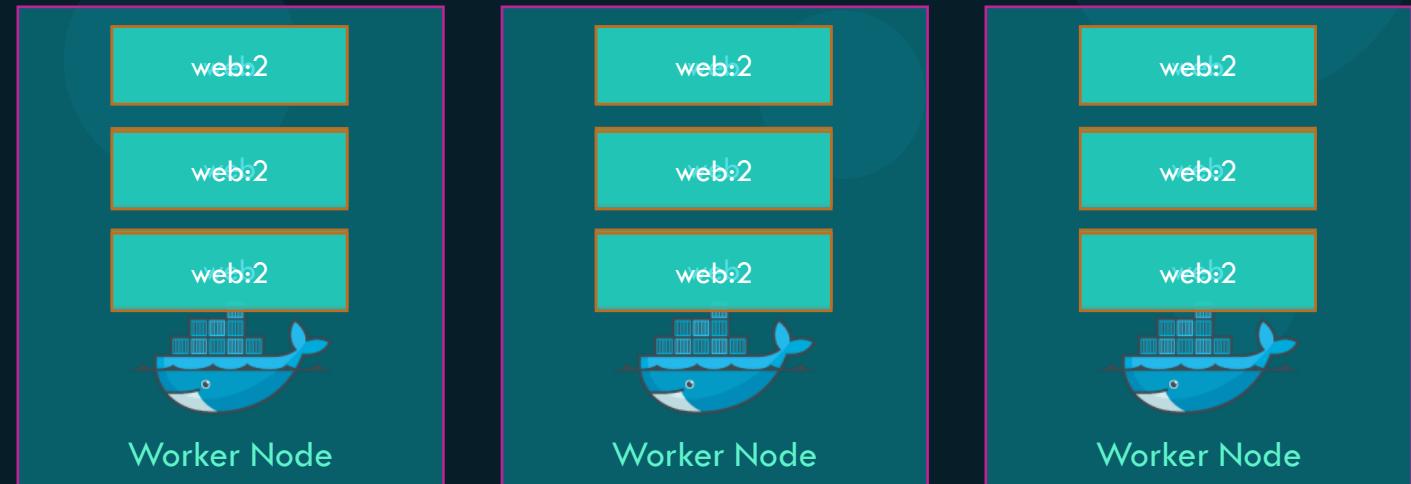
```
ID: y1k8vhoyqxulgthxrkph7xtug
Name: web
Service Mode: Replicated
Replicas: 5
Placement:
UpdateConfig:
  Parallelism: 3
  Delay: 60s
  On failure: pause
  Monitoring Period: 5s
  Max failure ratio: 0
  Update order: stop-first
RollbackConfig:
  Parallelism: 1
  On failure: pause
  Monitoring Period: 5s
  Max failure ratio: 0
  Rollback order: stop-first
ContainerSpec:
  Image: web:2...
  Init: false
Resources:
  Endpoint Mode: vip
```



```
▶ docker service update -p 80:80 \
  --update-failure-action pause|continue|rollback \
  --image=web:2 web
```

Docker Service – Rollback

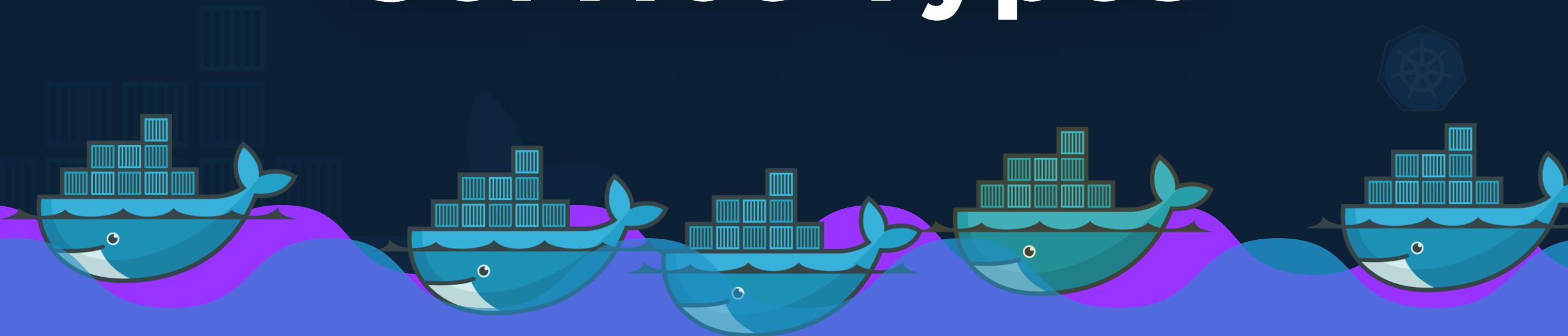
```
▶ docker service update --rollback web
```





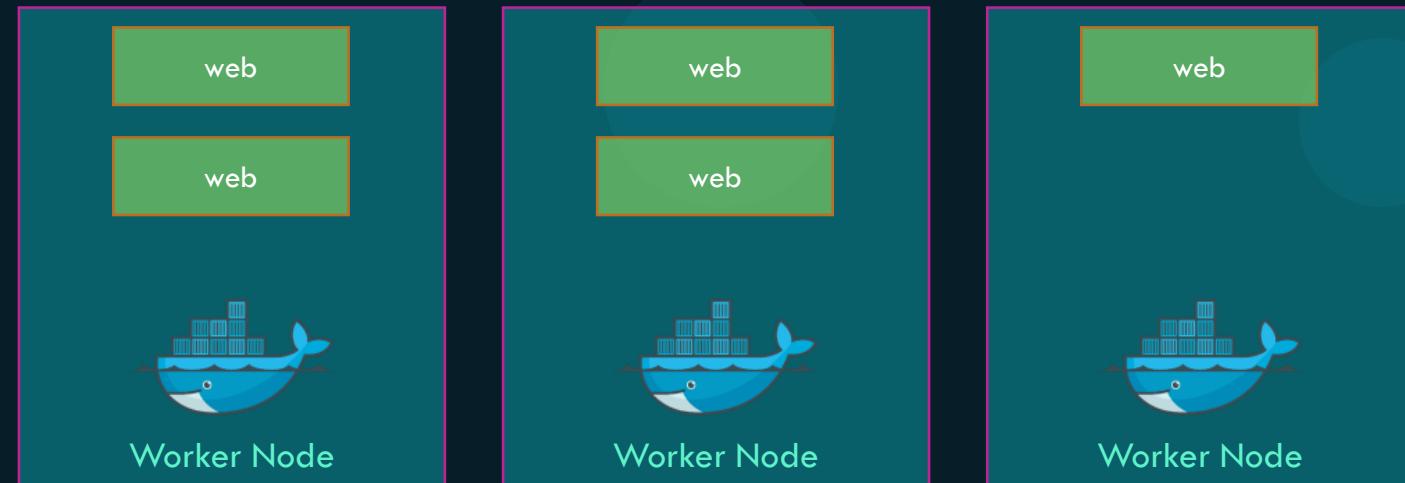
KODEKLOUD

Replicas vs Global Service Types



Global vs Replicated Services

```
▶ docker service create --replicas=5 web
```



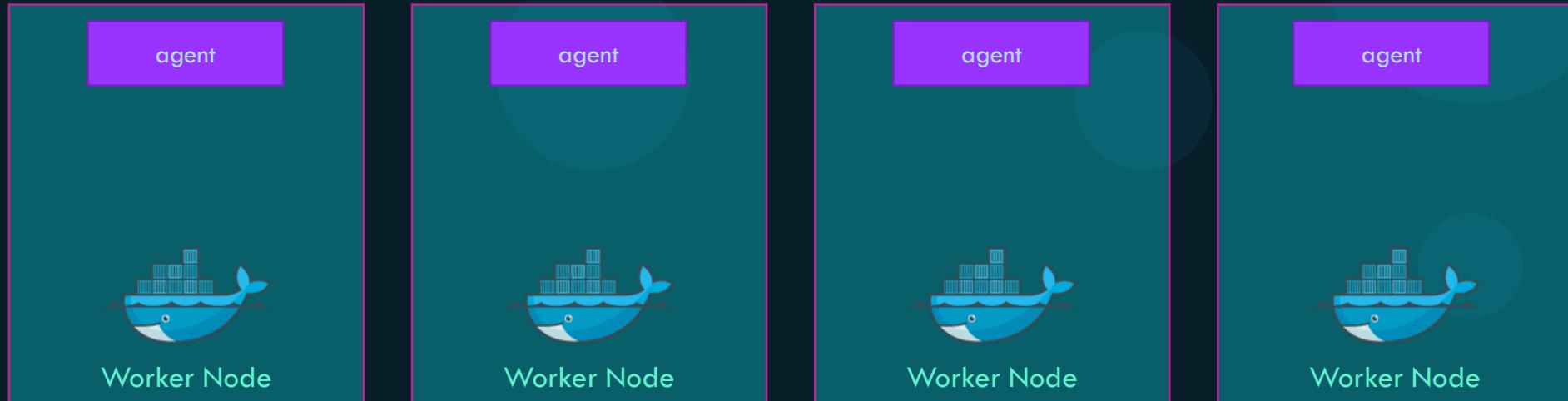
```
▶ docker service inspect web --pretty | grep -i "service mode"
```

```
Service Mode: Replicated
```



Global vs Replicated Services

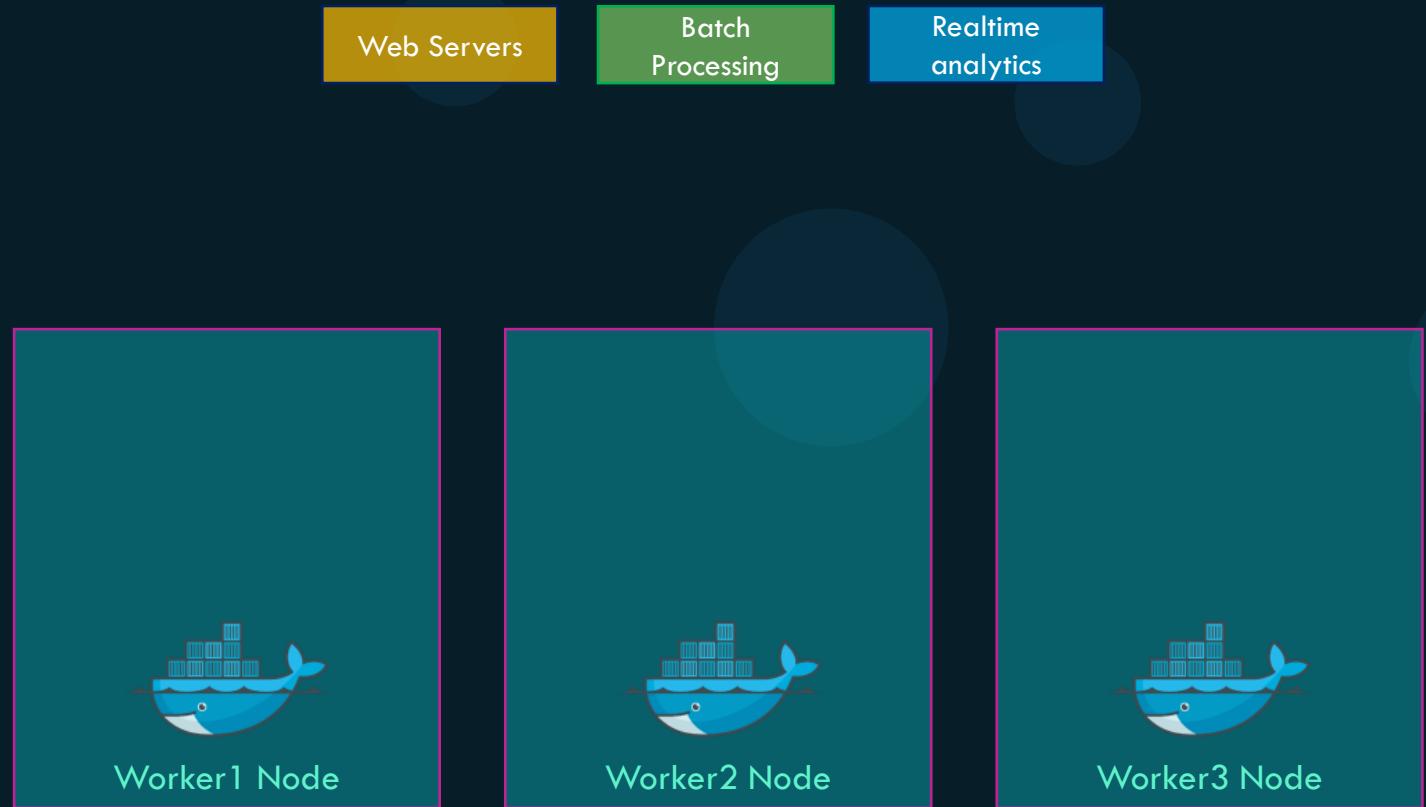
```
▶ docker service create --mode=global agent
```

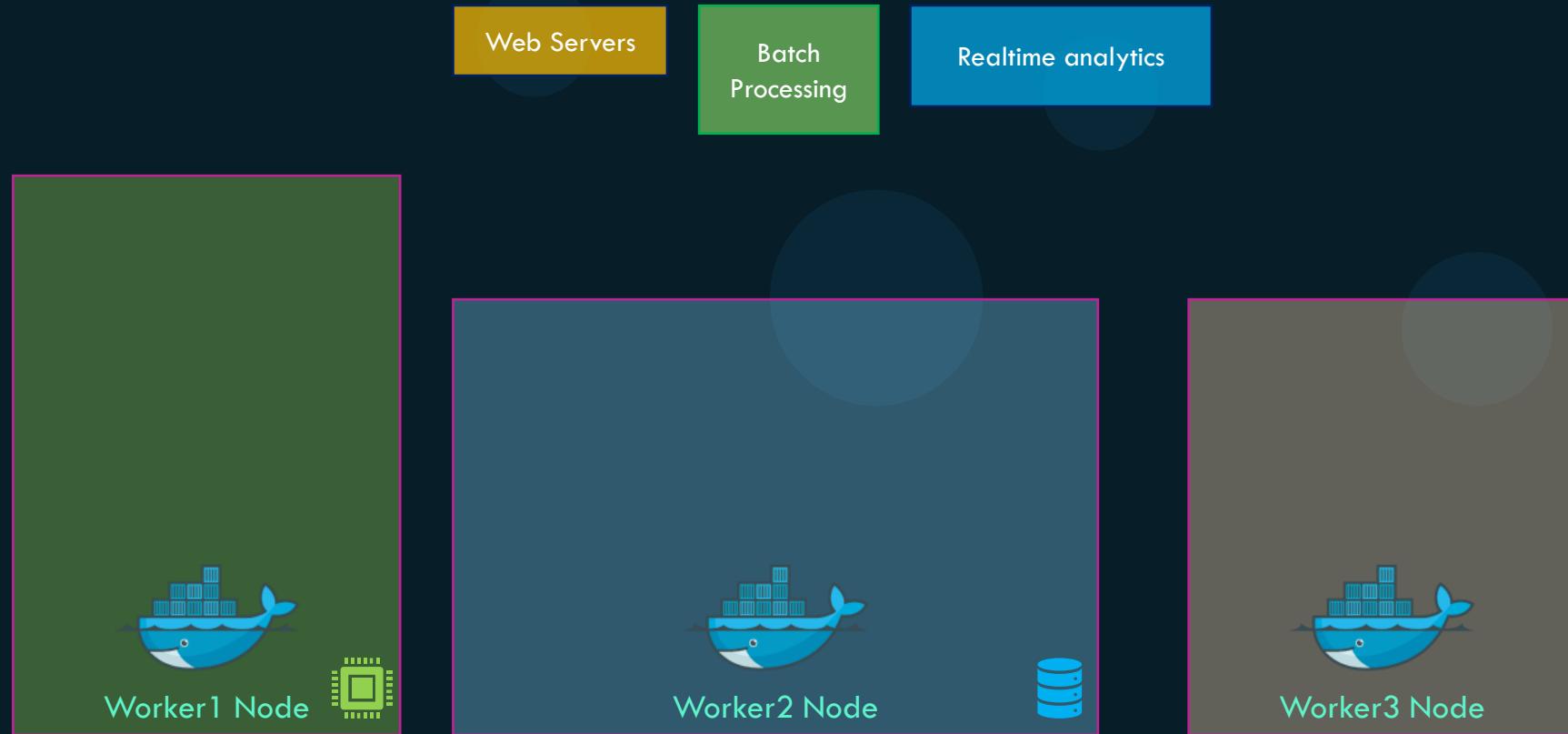


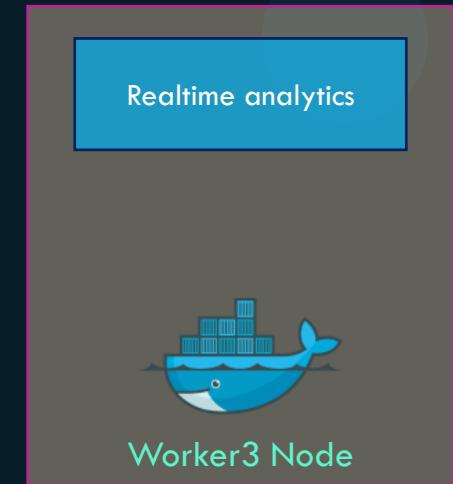
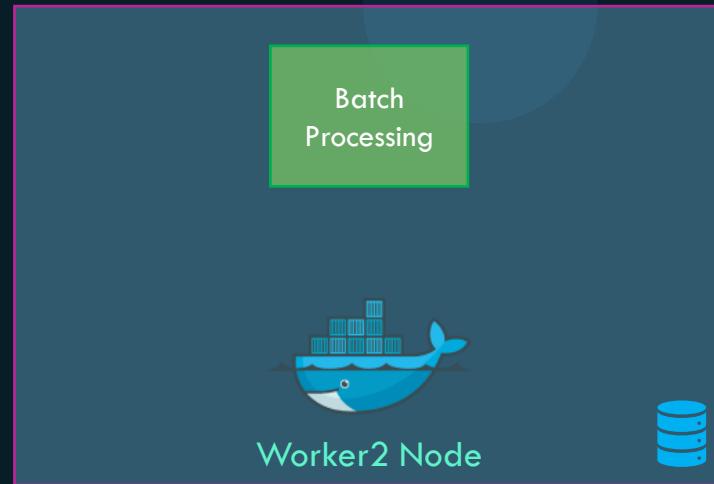


Placement Swarm Service

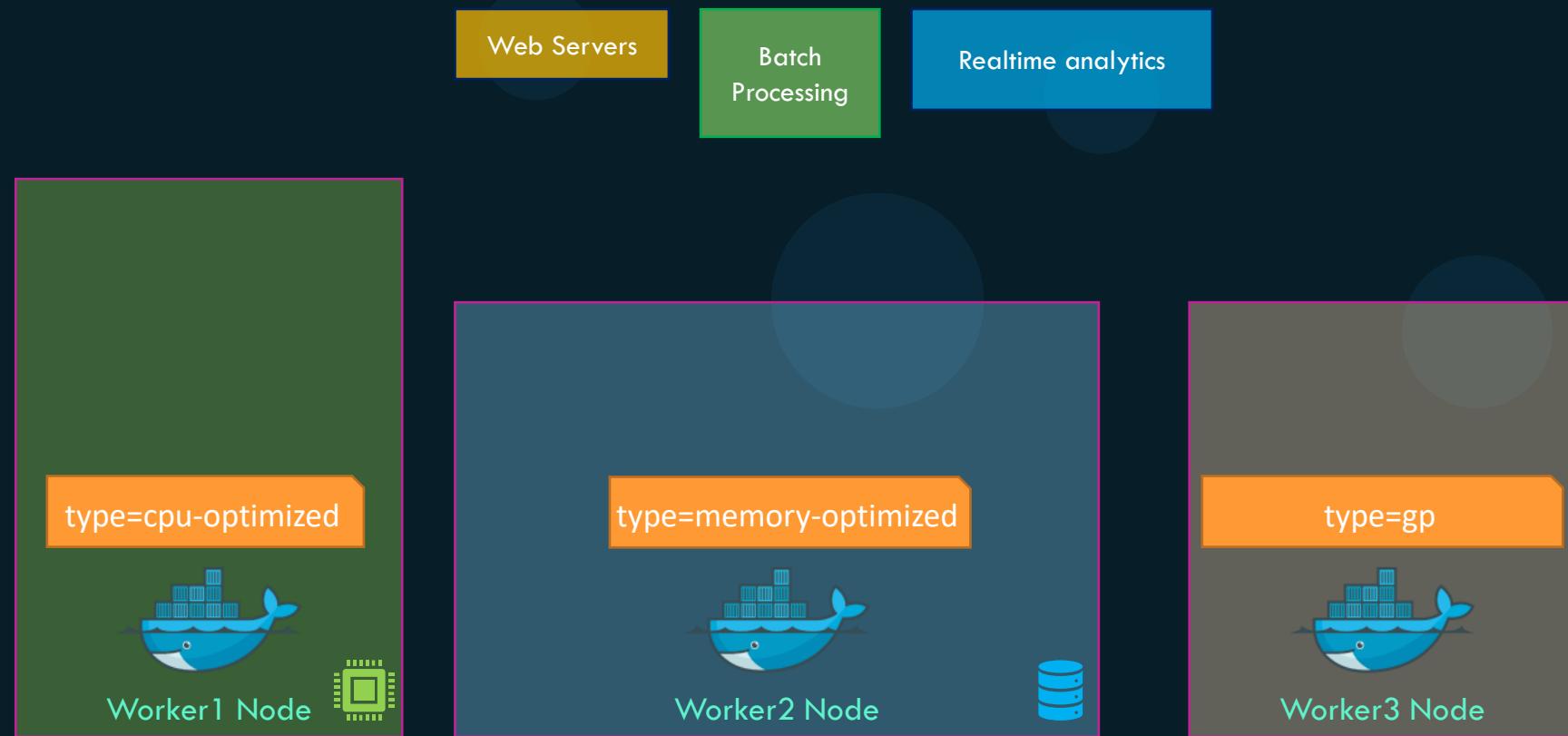








Labels & Constraints



```
▶ docker node update --label-add type=cpu-optimized worker1
```

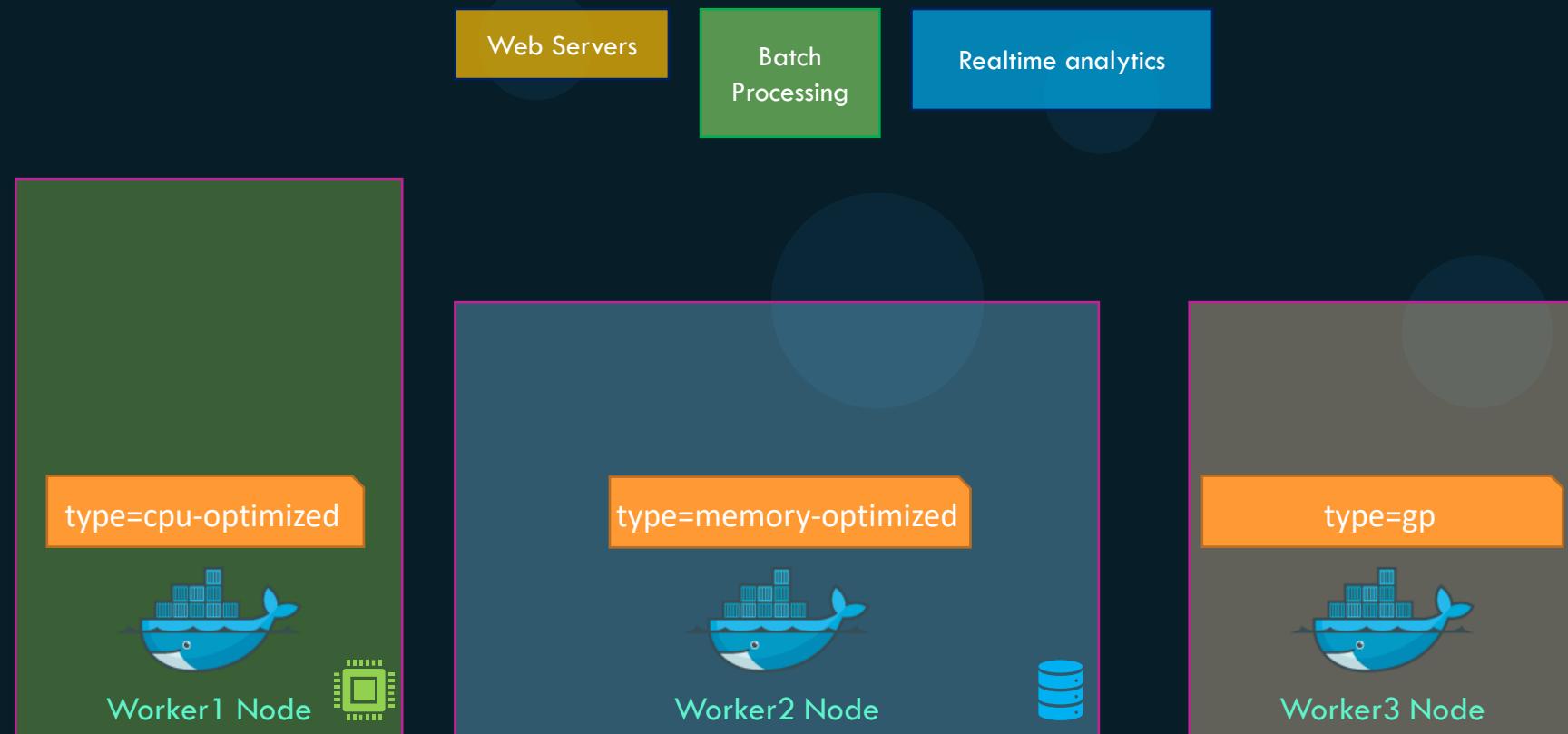
```
▶ docker node update --label-add type=memory-optimized worker2
```

```
▶ docker node update --label-add type=gp worker3
```

```
▶ docker node inspect worker1 --pretty
```

ID:	7t1vexyw8semg7z277mhliou
Labels:	- <code>type=cpu-optimized</code>
Hostname:	worker1
Joined at:	2020-04-24 11:21:42.05927
Status:	

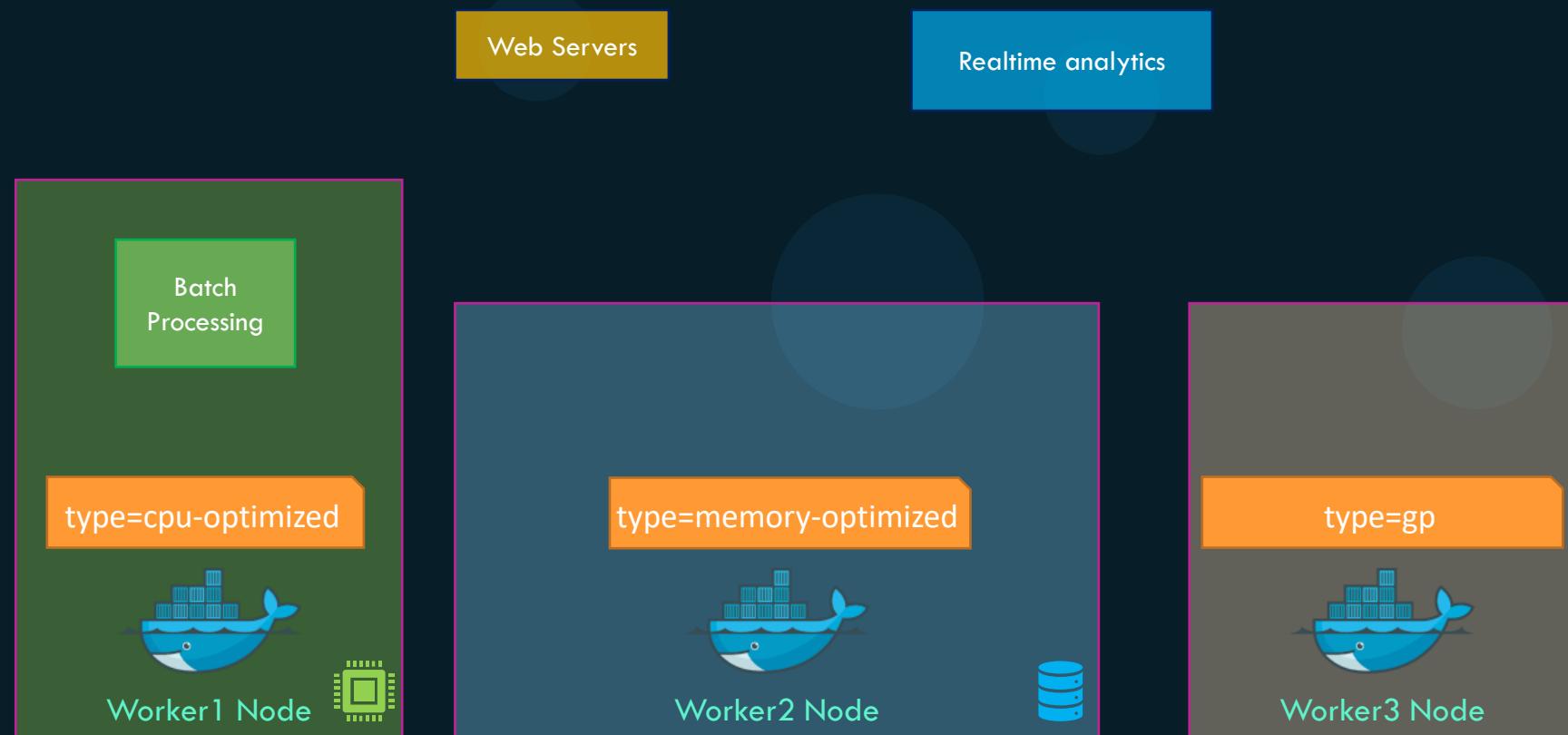
Labels & Constraints



```
▶ docker service create --constraint=node.labels.type==cpu-optimized batch-processing
```



Labels & Constraints

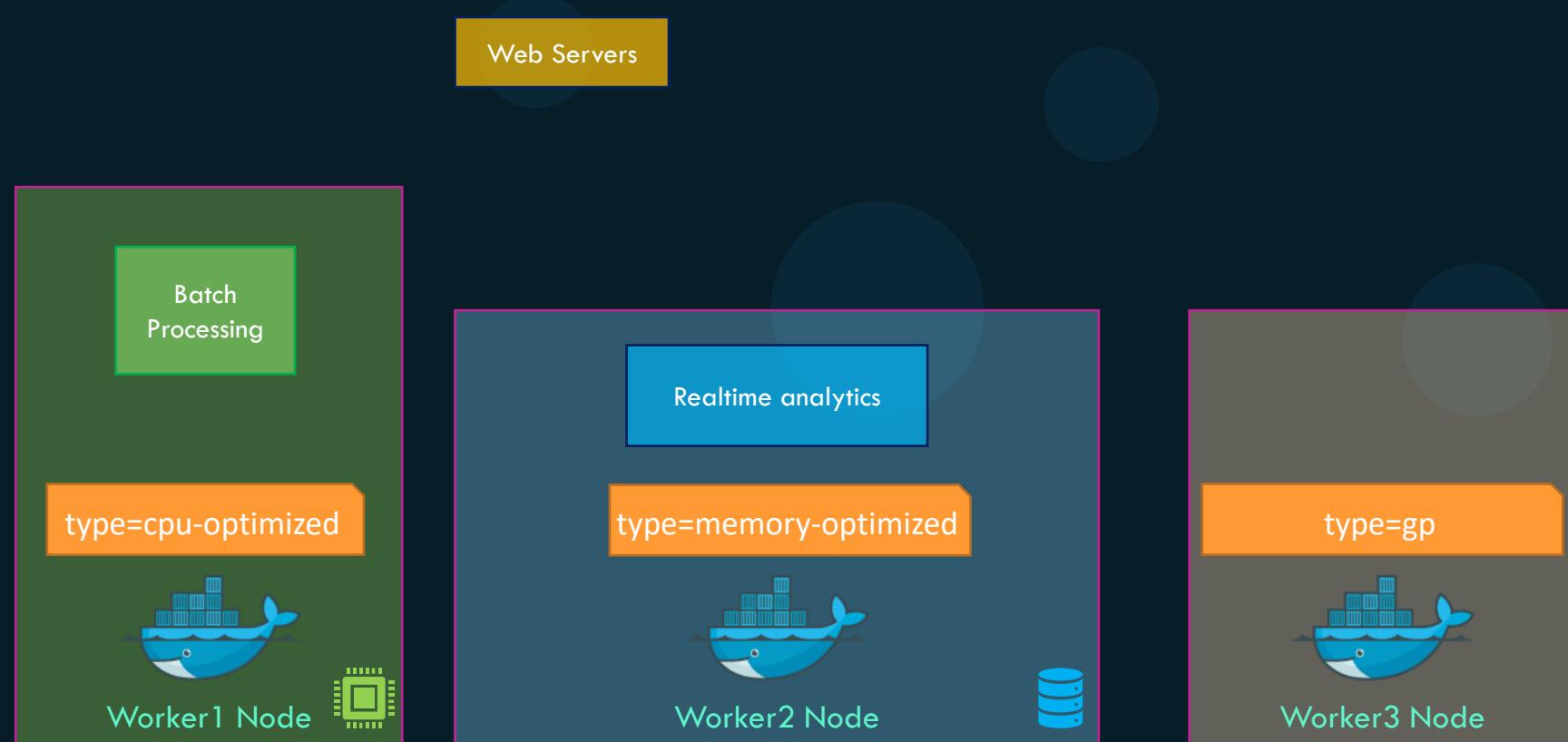


```
▶ docker service create --constraint=node.labels.type==cpu-optimized batch-processing
```

```
▶ docker service create --constraint=node.labels.type==memory-optimized realtime-analytics
```



Labels & Constraints



```
▶ docker service create --constraint=node.labels.type==cpu-optimized batch-processing
```

```
▶ docker service create --constraint=node.labels.type==memory-optimized realtime-analytics
```



Labels & Constraints

```
▶ docker service create --constraint=node.labels.type==cpu-optimized batch-processing
```

```
▶ docker service create --constraint=node.labels.type==memory-optimized realtime-analytics
```

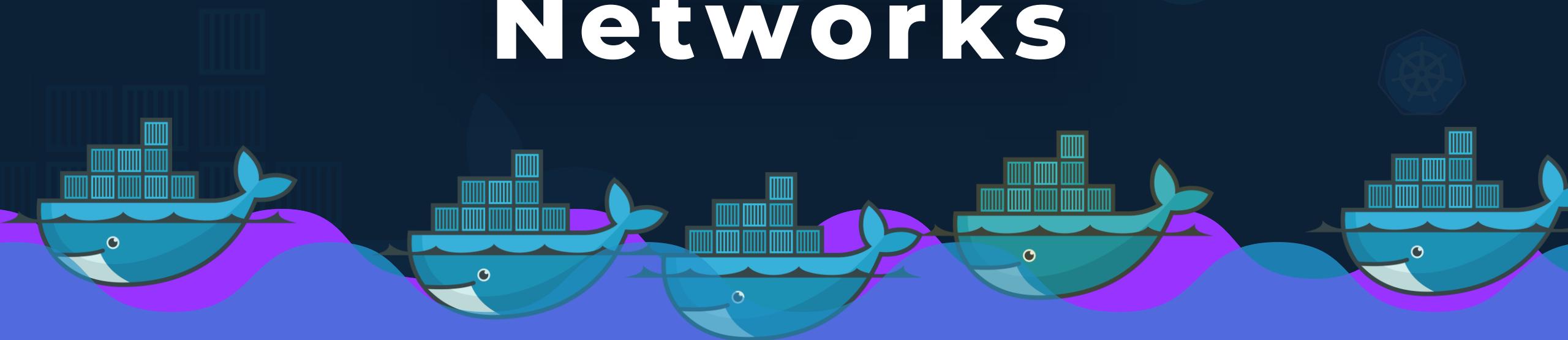
```
▶ docker service create --constraint=node.labels.type!=memory-optimized web
```

```
▶ docker service create --constraint=node.role==worker web
```





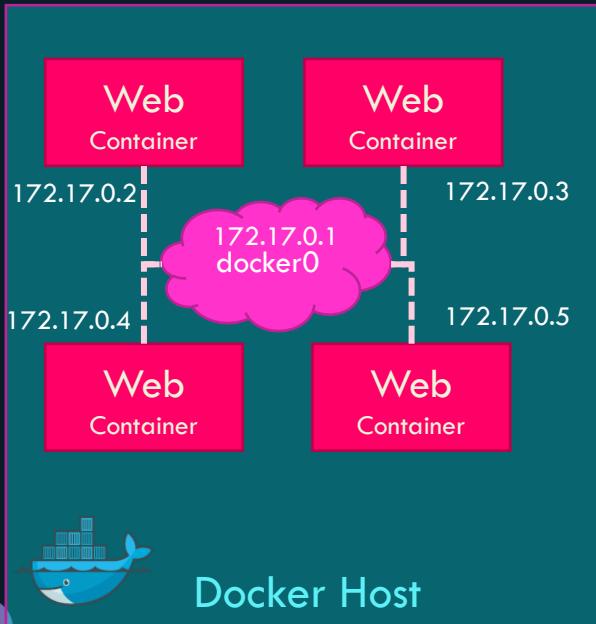
Docker Overlay Networks



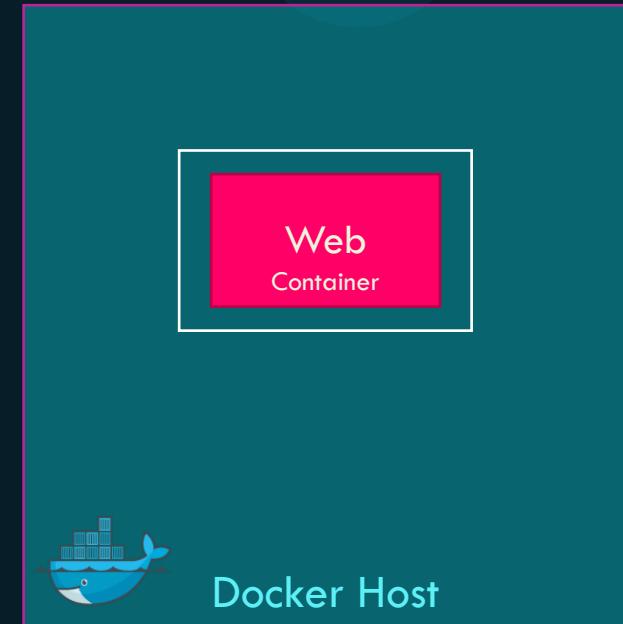
Default networks



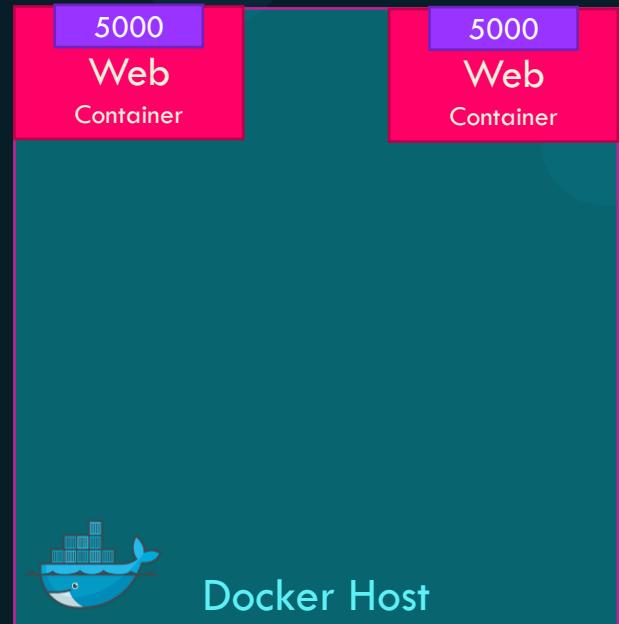
▶ docker run ubuntu



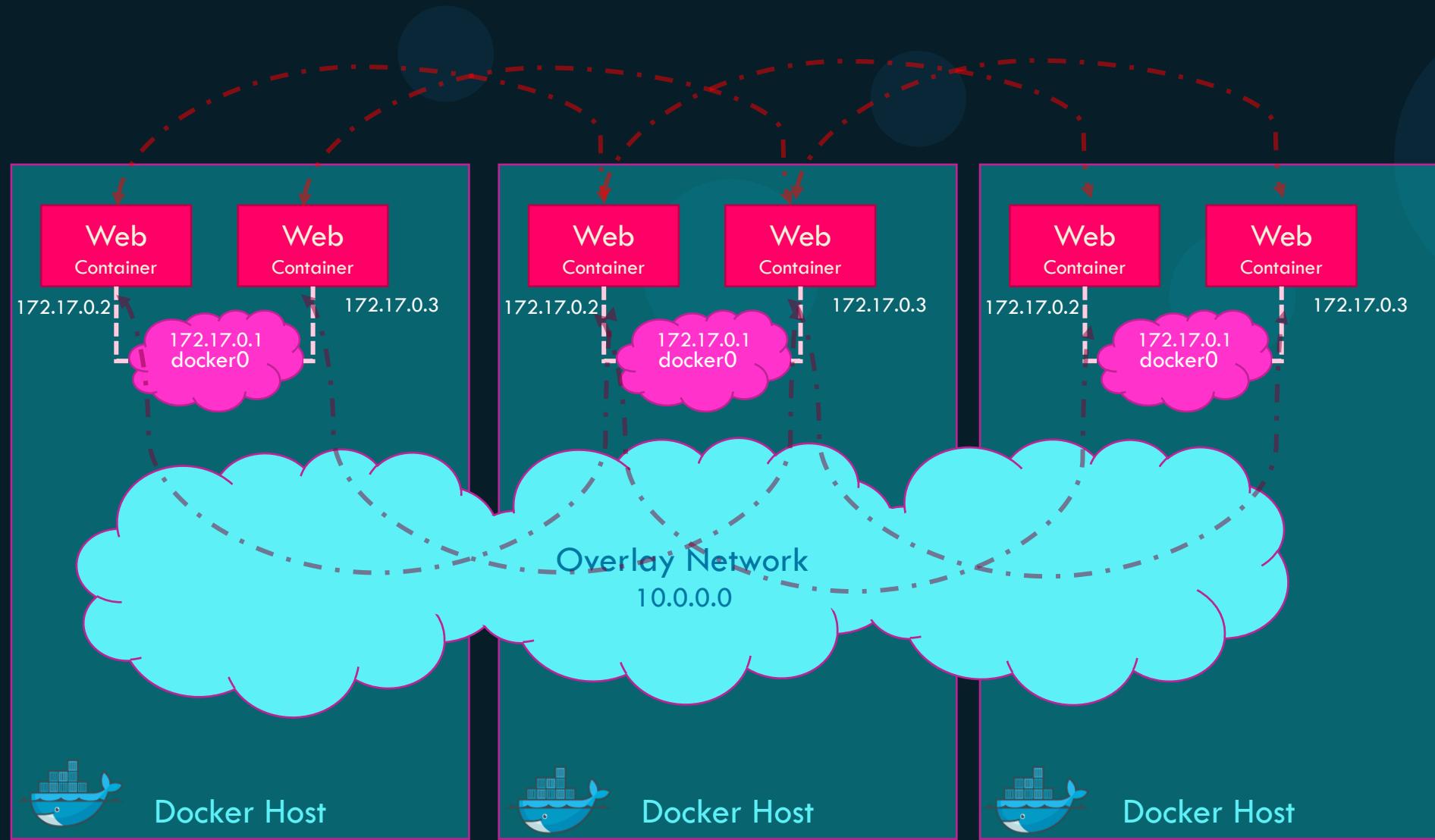
▶ docker run --network=none ubuntu



▶ docker run --network=host ubuntu



Overlay network



Ingress network

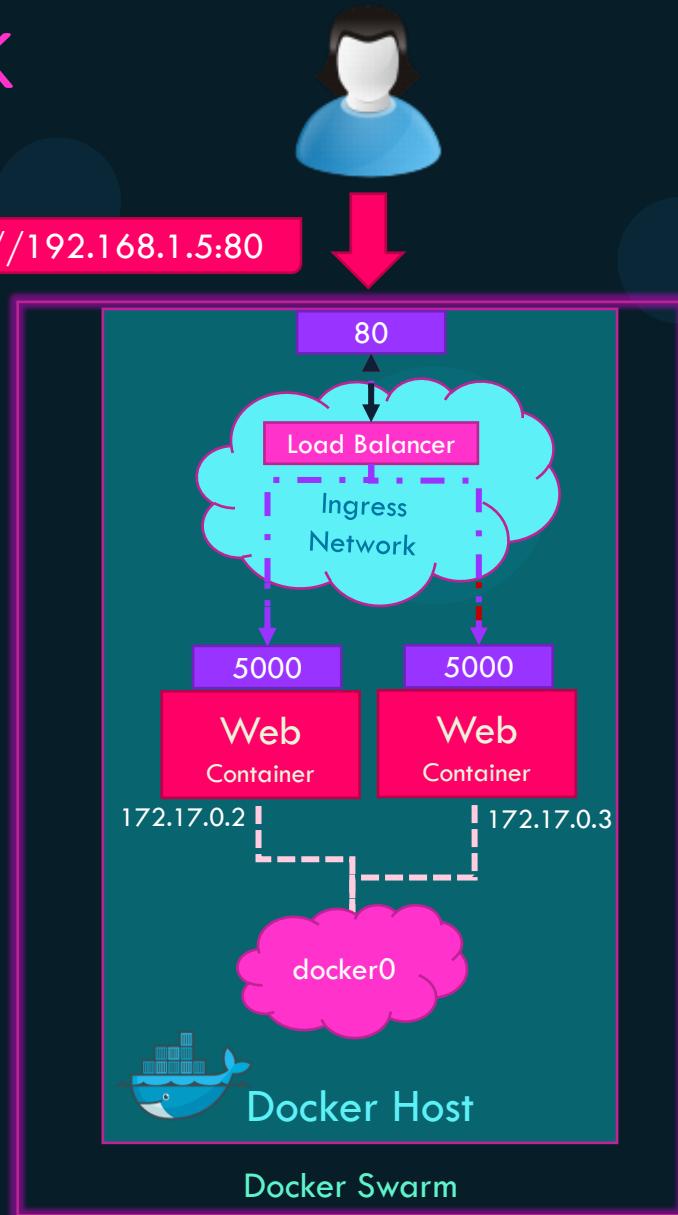
http://192.168.1.5:80

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

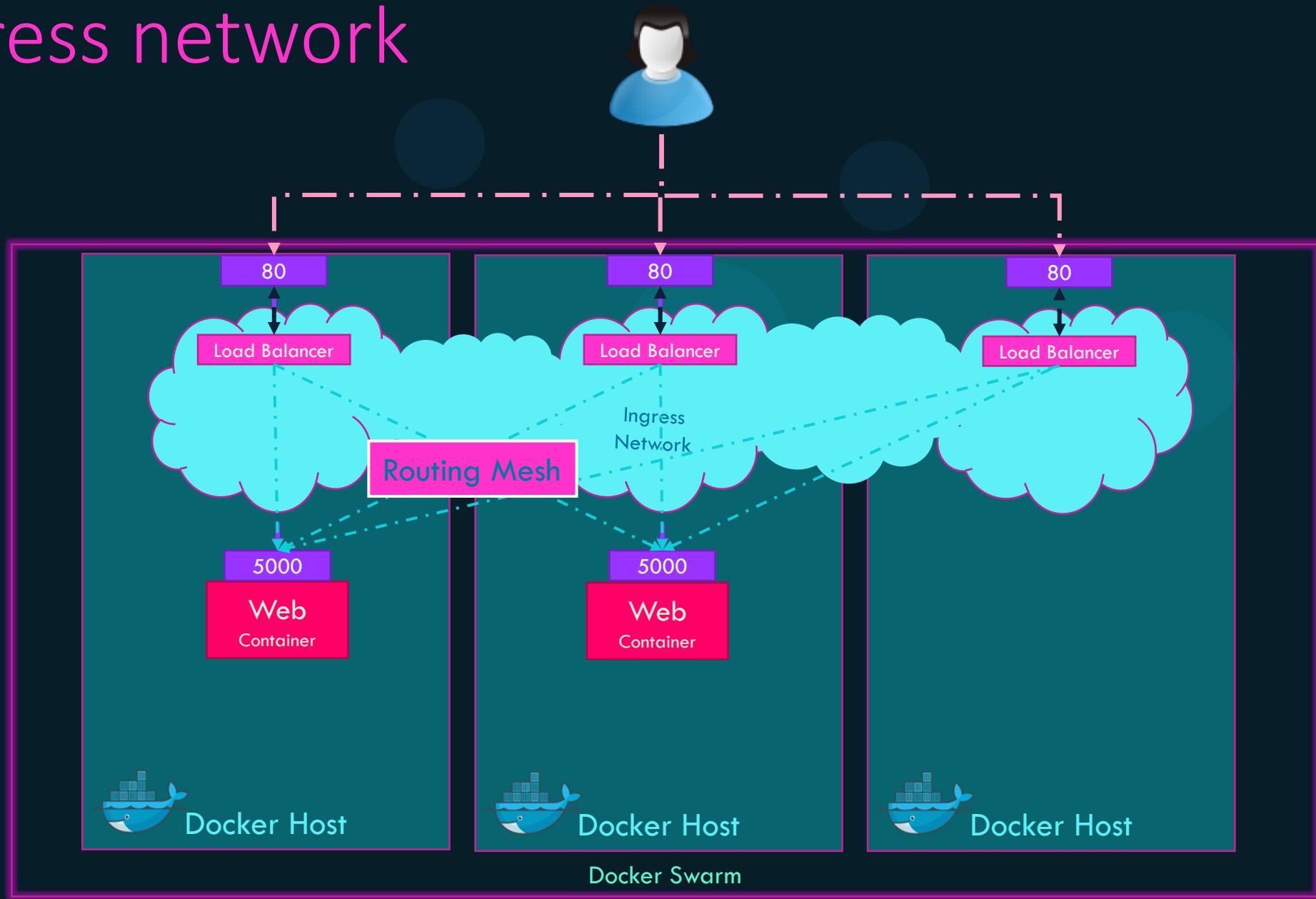
```
▶ docker service create \
  --replicas=2 \
  -p 80:5000 \
  my-web-server
```

```
▶ docker network ls
```

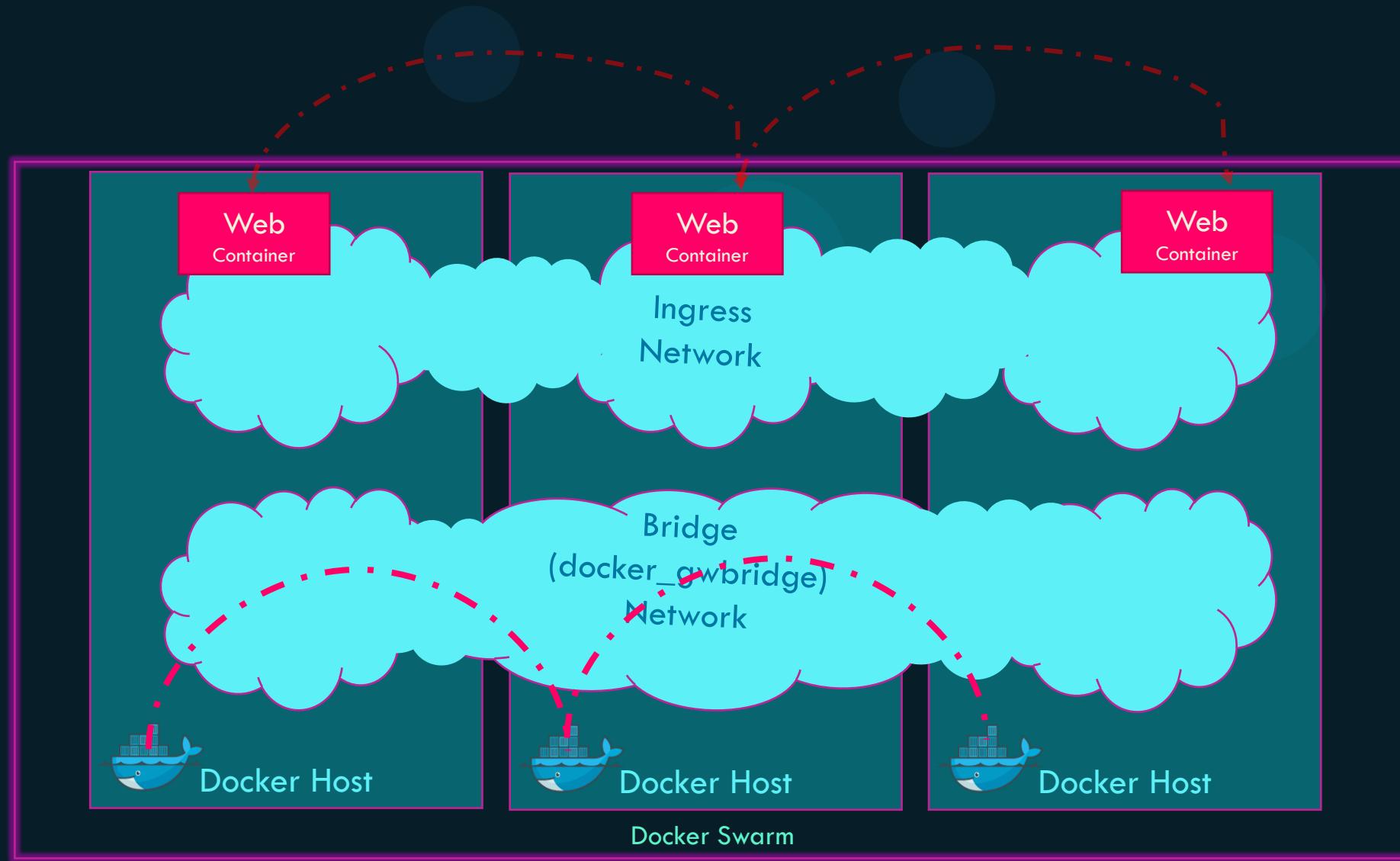
NETWORK ID	NAME	DRIVER
68abeebf1f2e	bridge	bridge
5bab4adc7d02	host	host
e43bd489dd57	none	null
mevcdb5b40zz	ingress	overlay



Ingress network



Default Networks



Overlay Network

```
▶ docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
68abeffb1f2e	bridge	bridge	local
5bab4adc7d02	host	host	local
e43bd489dd57	none	null	local
mevcdb5b40zz	ingress	overlay	swarm
c8fb2c361202	docker_gwbridge	bridge	local

```
▶ docker network create --driver overlay my-overlay-network
```

```
▶ docker network create --driver overlay --subnet 10.15.0.0/16 my-overlay-network
```

```
▶ docker network create --driver overlay --attachable my-overlay-network
```

```
▶ docker network create --driver overlay --opt encrypted my-overlay-network
```

```
▶ docker service create --network my-overlay-network my-web-service
```



Overlay Network Deletion

```
▶ docker network rm my-overlay-network  
my-overlay-network
```

```
▶ docker network prune
```



Ports

Port	Description
TCP 2377	Cluster Management Communications
TCP and UDP 7946	Communication among nodes/Container Network Discovery
UDP 4789	Overlay network traffic



Publishing Ports

```
▶ docker service create -p 80:5000 my-web-server
```

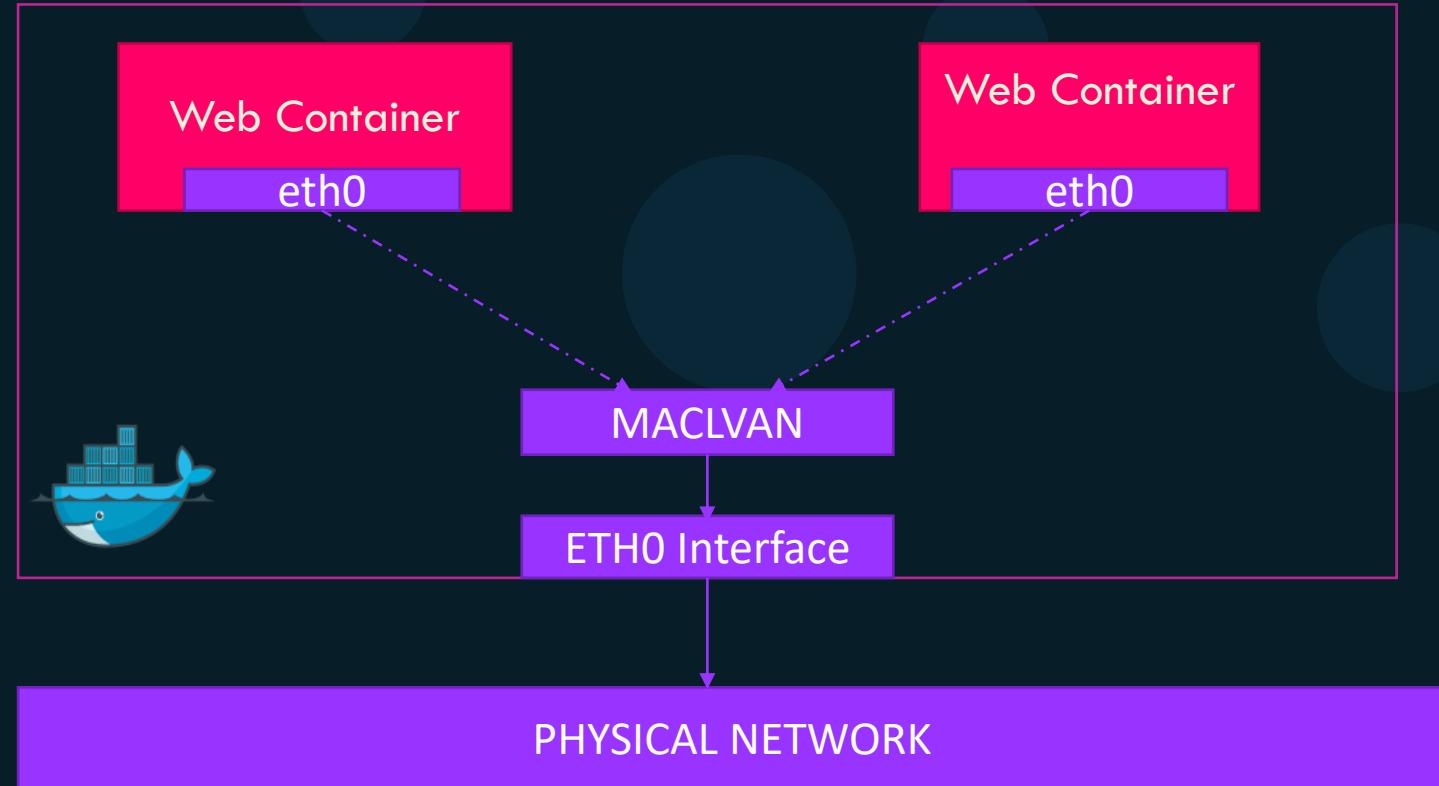
```
▶ docker service create --publish published=80,target=5000 my-web-server
```

```
▶ docker service create -p 80:5000/udp my-web-server
```

```
▶ docker service create --publish published=80,target=5000,protocol=udp my-web-server
```



Default MACVLAN networks



```
▶ docker network create --driver mcvlan --o parent=eth0 my-overlay-network
```

bridge	Traffic goes through a physical device on the host
802.1q trunk bridge	Traffic goes through 802.1q sub-interface. Allows control over routing and filtering at a more granular level

Summary

Type	Use Case
None	To disable all network. This is not available for swarm services
Host	To remove network isolation. Container uses host's network.
Bridge	For multiple containers to communicate on the same docker host.
Overlay Networks	For multiple containers to communicate on different docker hosts.
Macvlan	Legacy applications that need containers to look like physical hosts on network with unique MAC Address. Used for multiple containers to communicate across different docker hosts. L3 Bridge
IPVLAN	Used for multiple containers to communicate across different docker hosts. L2 Bridge.



References

- <https://docs.docker.com/network/overlay/>
- <https://docs.docker.com/engine/swarm/ingress/>



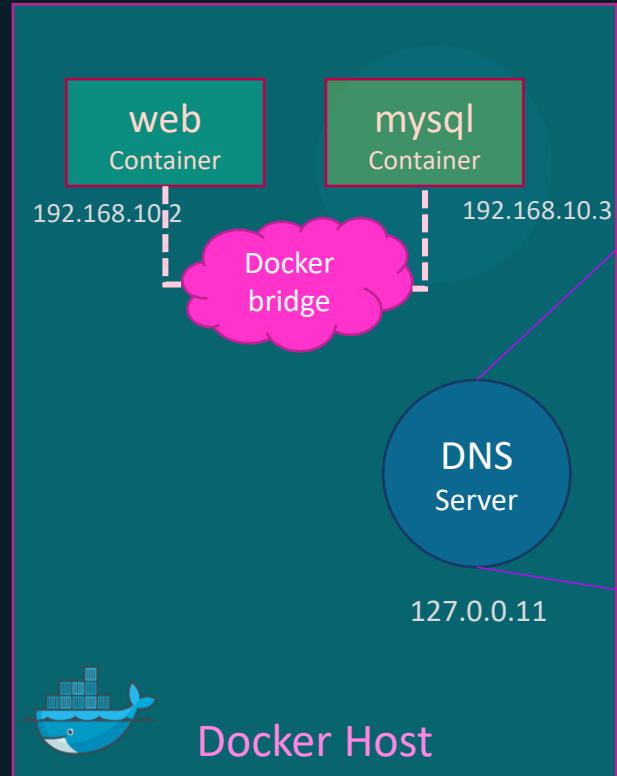


Service Discovery

Docker Swarm

Service Discovery - DNS

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



Host	IP
web	192.168.10.2
mysql	192.168.10.2

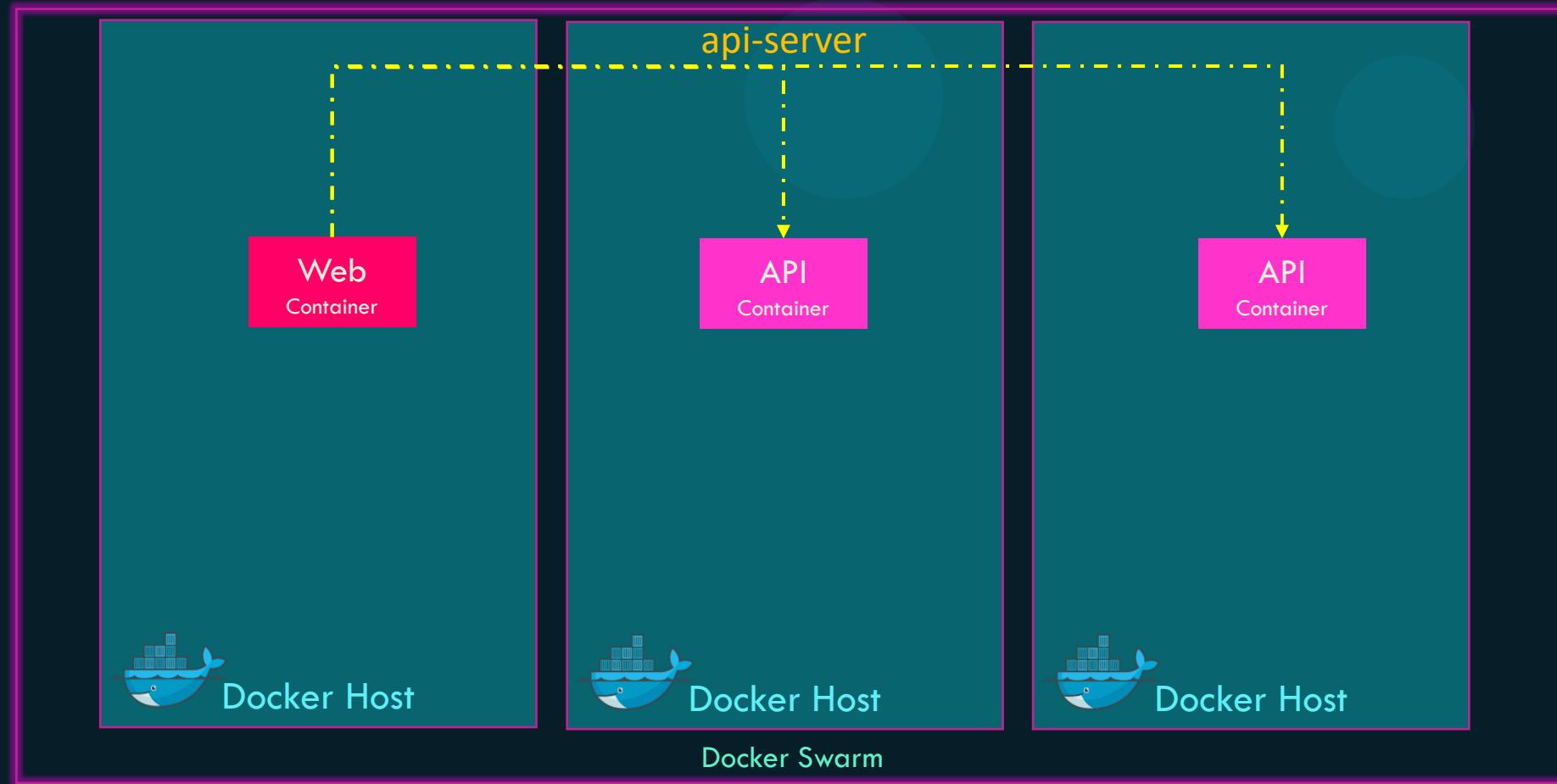
```
▶ docker exec -it web cat /etc/resolv.conf
```

```
search ec2.internal
nameserver 127.0.0.11
options ndots:0
```

Service Discovery - DNS

```
▶ docker service create --name=api-server --replicas=2 api-server
```

```
▶ docker service create --name=web web
```

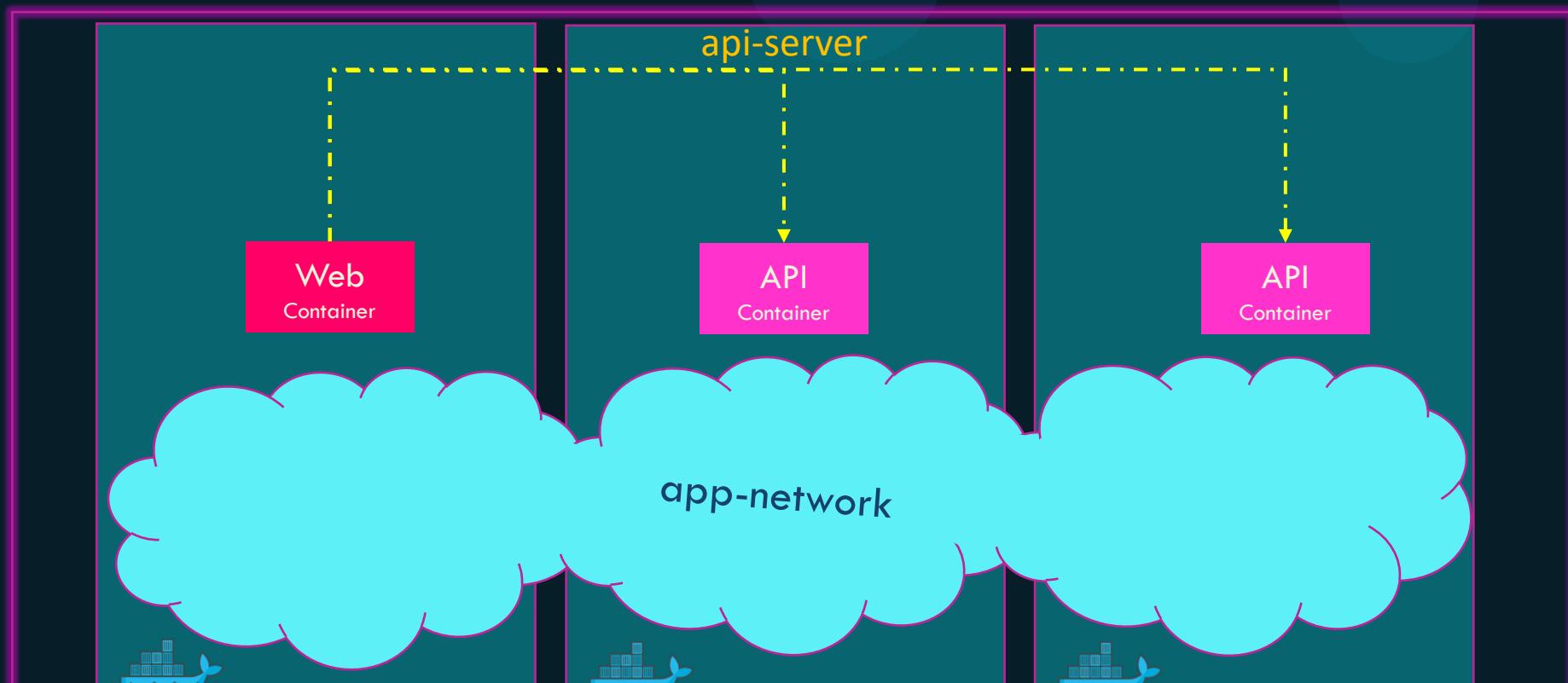


Service Discovery - DNS

```
▶ docker network create --driver=overlay app-network
```

```
▶ docker service create --name=api-server --replicas=2 api-server
```

```
▶ docker service create --name=web web
```

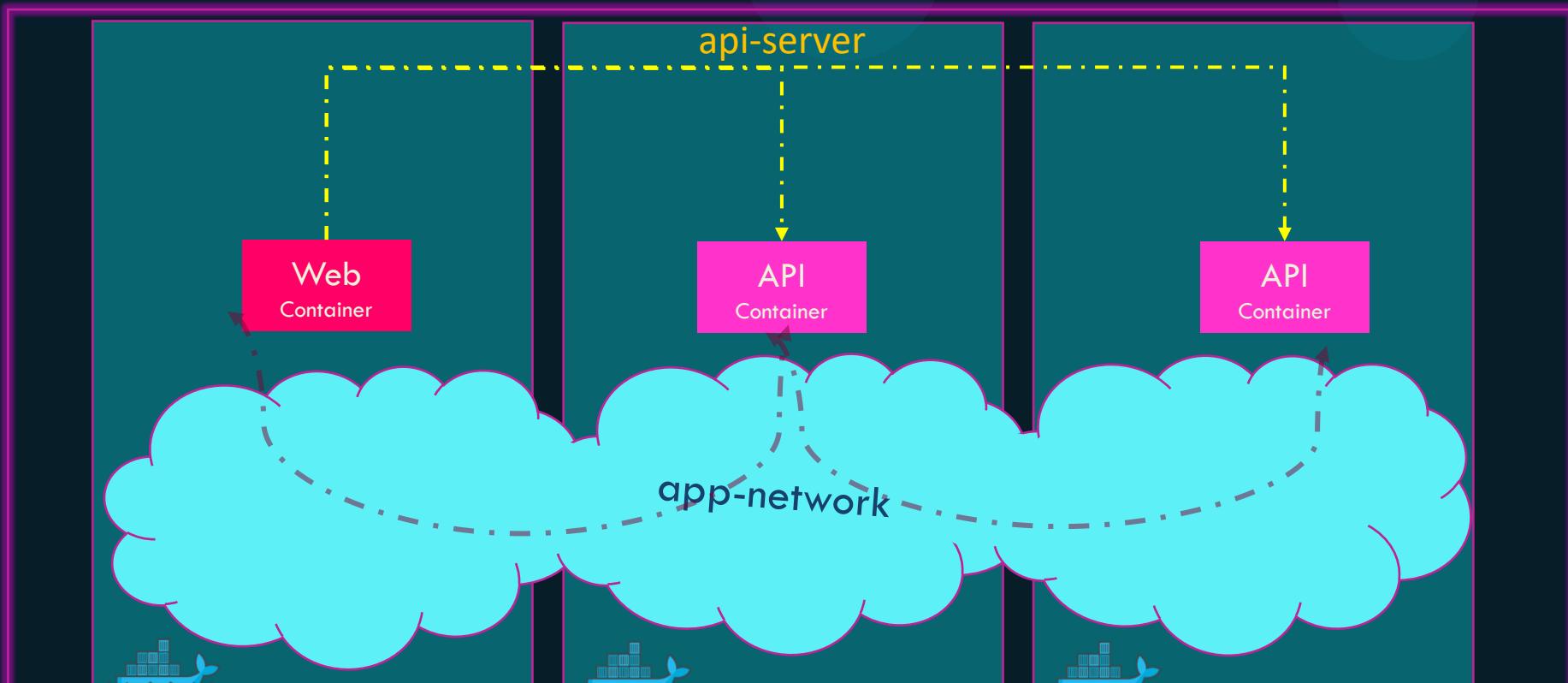


Service Discovery - DNS

```
▶ docker network create --driver=overlay app-network
```

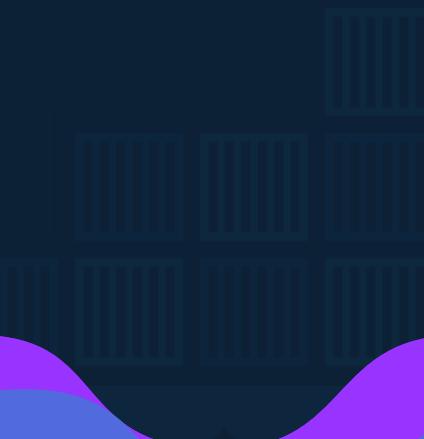
```
▶ docker service create --name=api-server --replicas=2 --network=app-network api-server
```

```
▶ docker service create --name=web --network=app-network web
```



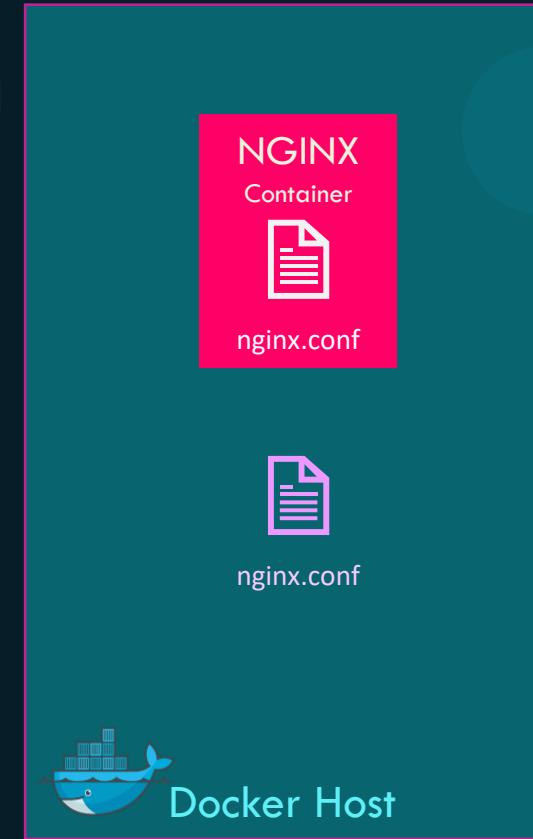


Docker Config



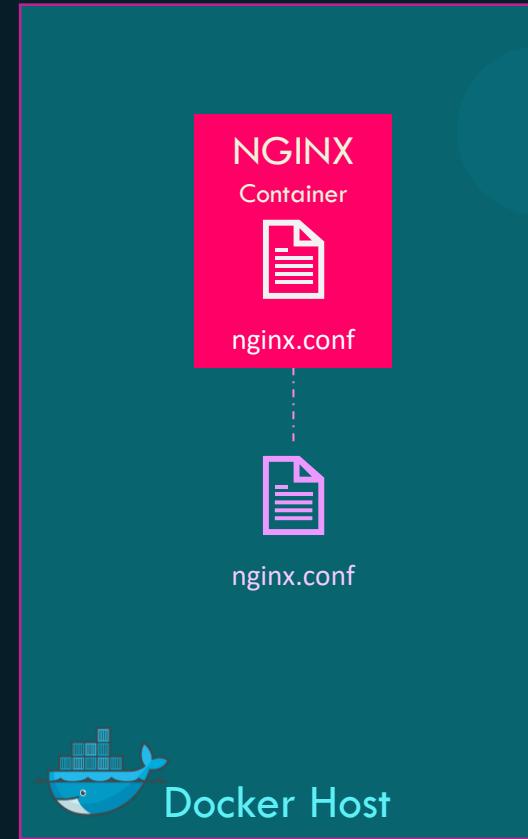
Docker Volume

```
▶ docker run nginx
```



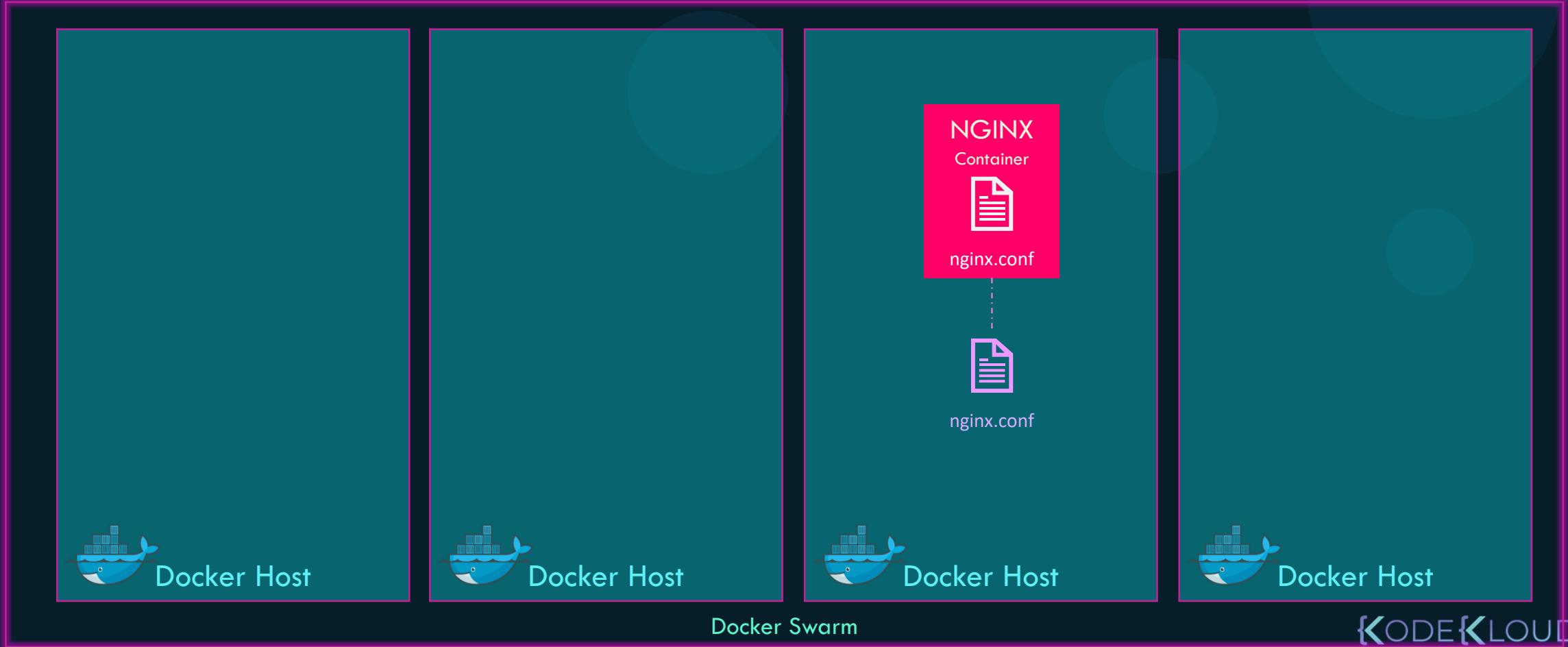
Docker Volume

```
▶ docker run -v /tmp/nginx.conf:/etc/nginx/nginx.conf nginx
```



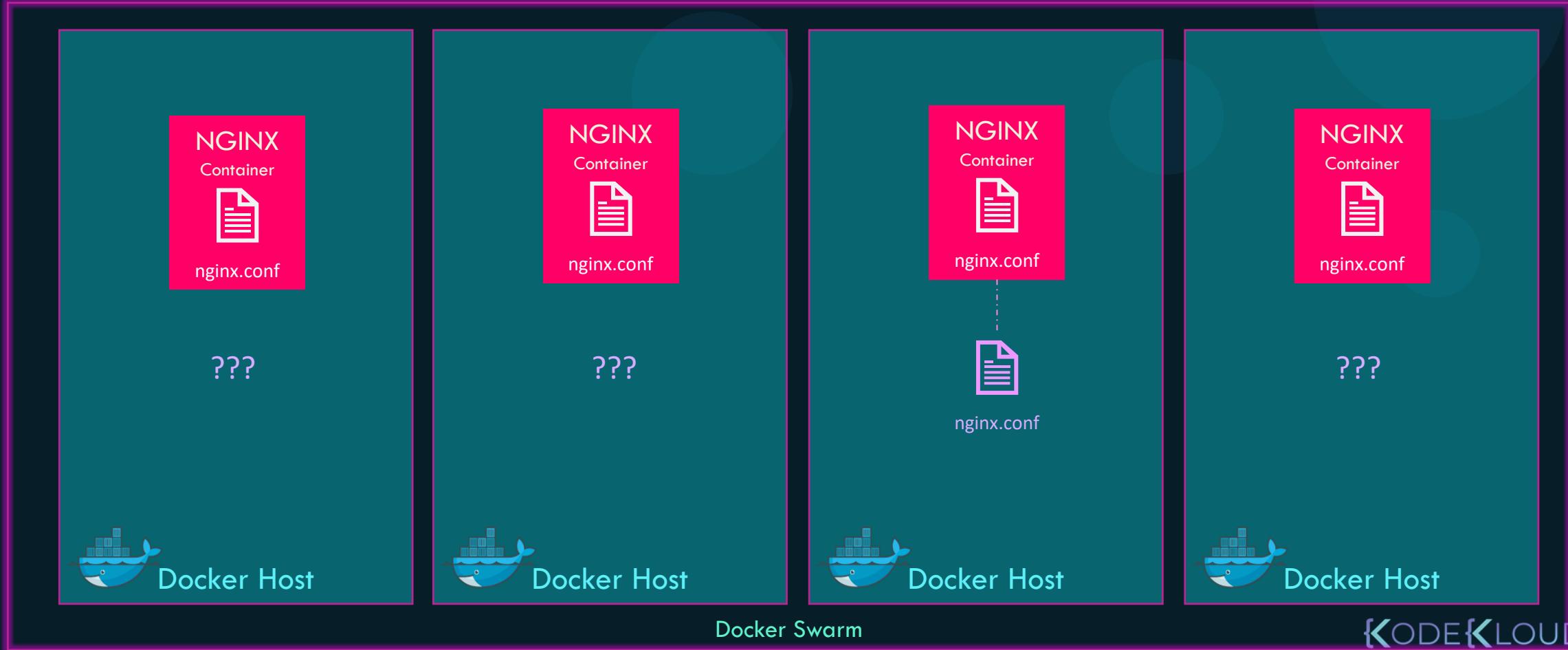
Docker Volume

```
▶ docker run -v /tmp/nginx.conf:/etc/nginx/nginx.conf nginx
```



Docker Volume

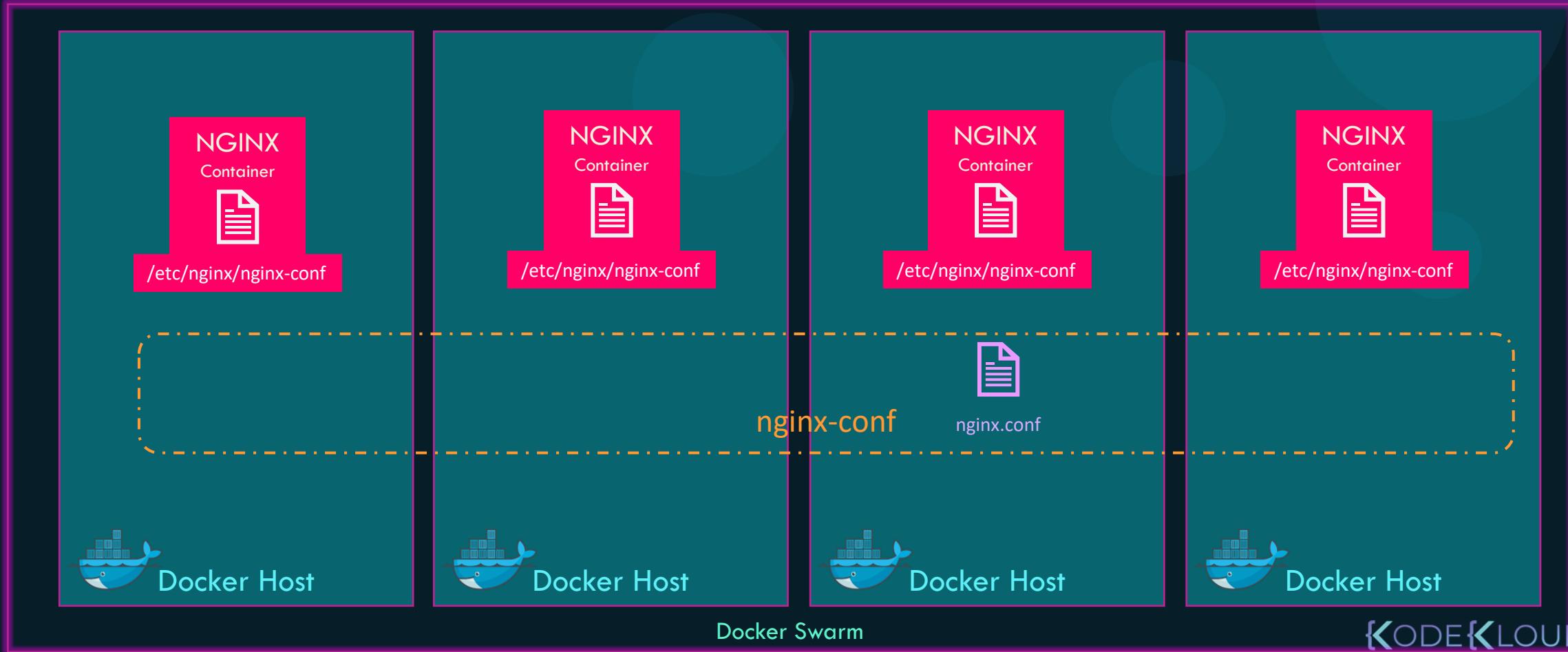
```
▶ docker service create --replicas=4 -v /tmp/nginx.conf:/etc/nginx/nginx.conf nginx
```



Docker Configs

```
▶ docker config create nginx-conf /tmp/nginx.conf
```

```
▶ docker service create --replicas=4 --config src=nginx-conf,target="/etc/nginx/nginx.conf" nginx
```



Docker Configs

```
▶ docker config create nginx-conf /tmp/nginx.conf
```

```
▶ docker service create --replicas=4 --config src=nginx-conf,target="/etc/nginx/nginx.conf" nginx
```

```
▶ docker service update --config-rm nginx-conf nginx
```

```
▶ docker config rm nginx-conf
```

```
▶ docker config create nginx-conf-new /tmp/nginx-new.conf
```

```
▶ docker service update --config-rm nginx-conf --config-add nginx-conf-new nginx
```





KODEKLOUD



Stack

Docker Swarm

Docker Compose

▶ docker run simple-webapp

▶ docker run mongodb

▶ docker run redis:alpine

docker-compose.yml

```
services:  
  web:  
    image: "simple-webapp"  
  database:  
    image: "mongodb"  
  messaging:  
    image: "redis:alpine"
```

▶ docker-compose up



Docker Compose

```
▶ docker run simple-webapp
```

```
▶ docker run mongodb
```

```
▶ docker run redis:alpine
```

```
▶ docker service create simple-webapp
```

```
▶ docker service create mongodb
```

```
▶ docker service create redis
```

docker-compose.yml

```
services:  
  web:  
    image: "simple-webapp"  
  database:  
    image: "mongodb"  
  messaging:  
    image: "redis:alpine"
```

docker-compose.yml

```
services:  
  web:  
    image: "simple-webapp"  
  database:  
    image: "mongodb"  
  messaging:  
    image: "redis:alpine"
```

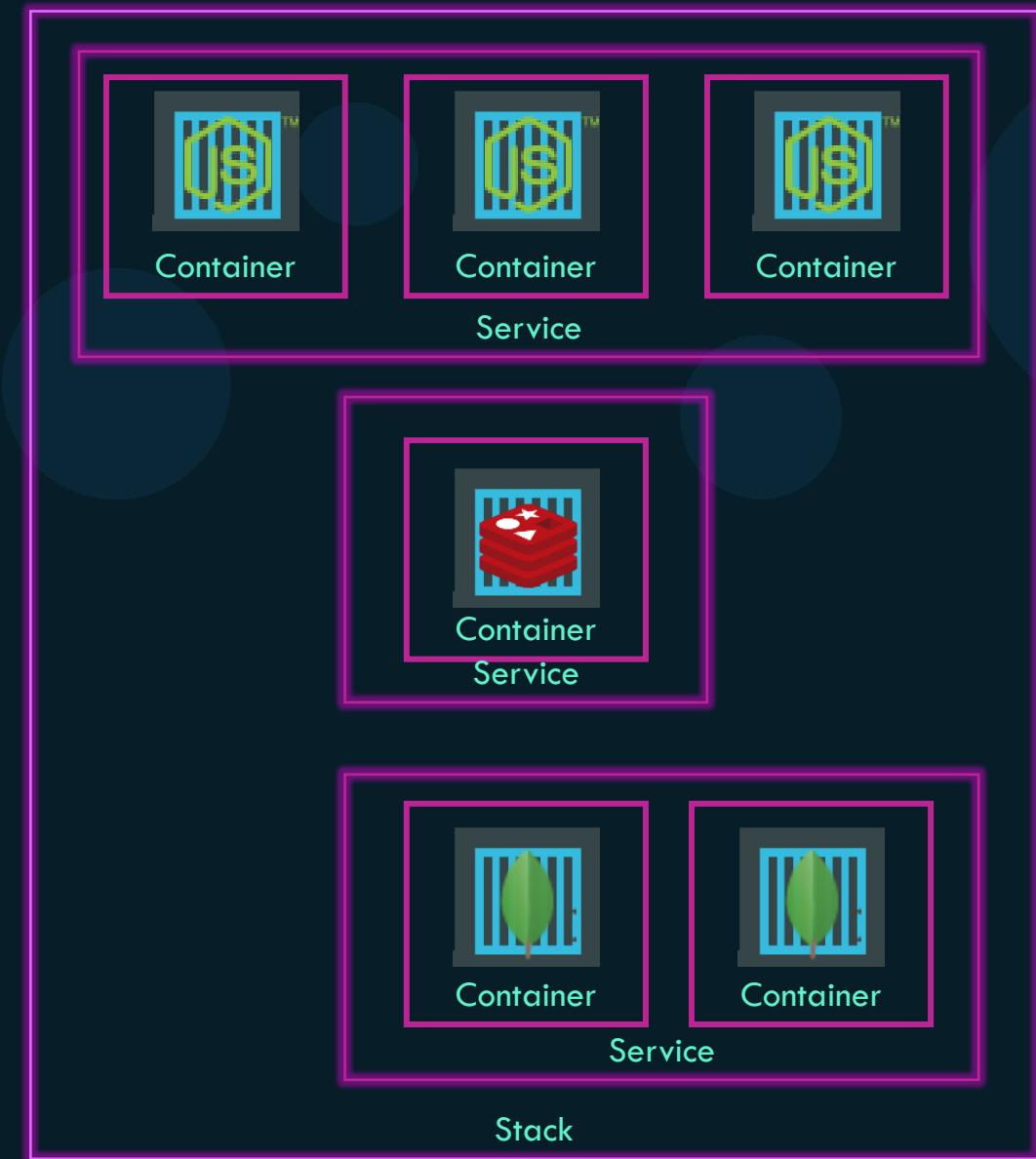
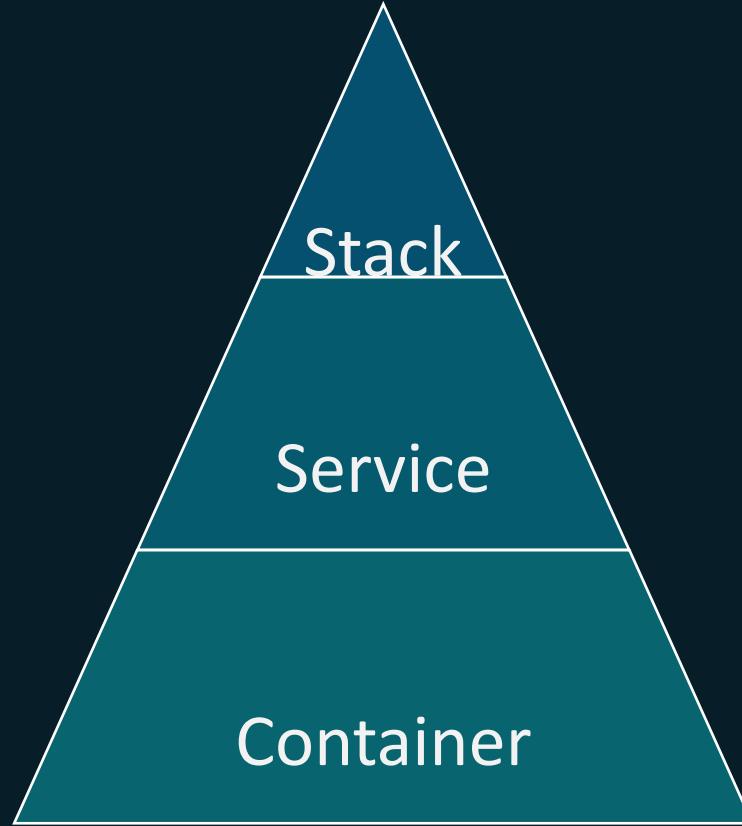


```
▶ docker-compose up
```

```
▶ docker stack deploy --compose-file docker-compose.yml
```

UD

STACK



Docker Compose

docker-compose.yml

```
version: 3
services:
  redis:
    image: redis

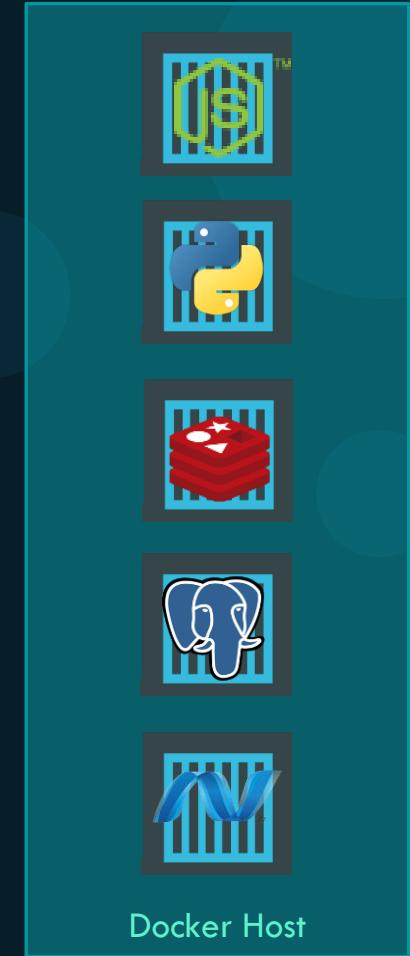
  db:
    image: postgres:9.4

  vote:
    image: voting-app

  result:
    image: result

  worker:
    image: worker
```

▶ docker-compose up



Docker Compose

docker-compose.yml

```
version: 3
services:
  redis:
    image: redis

  db:
    image: postgres:9.4

  vote:
    image: voting-app

  result:
    image: result

  worker:
    image: worker
```

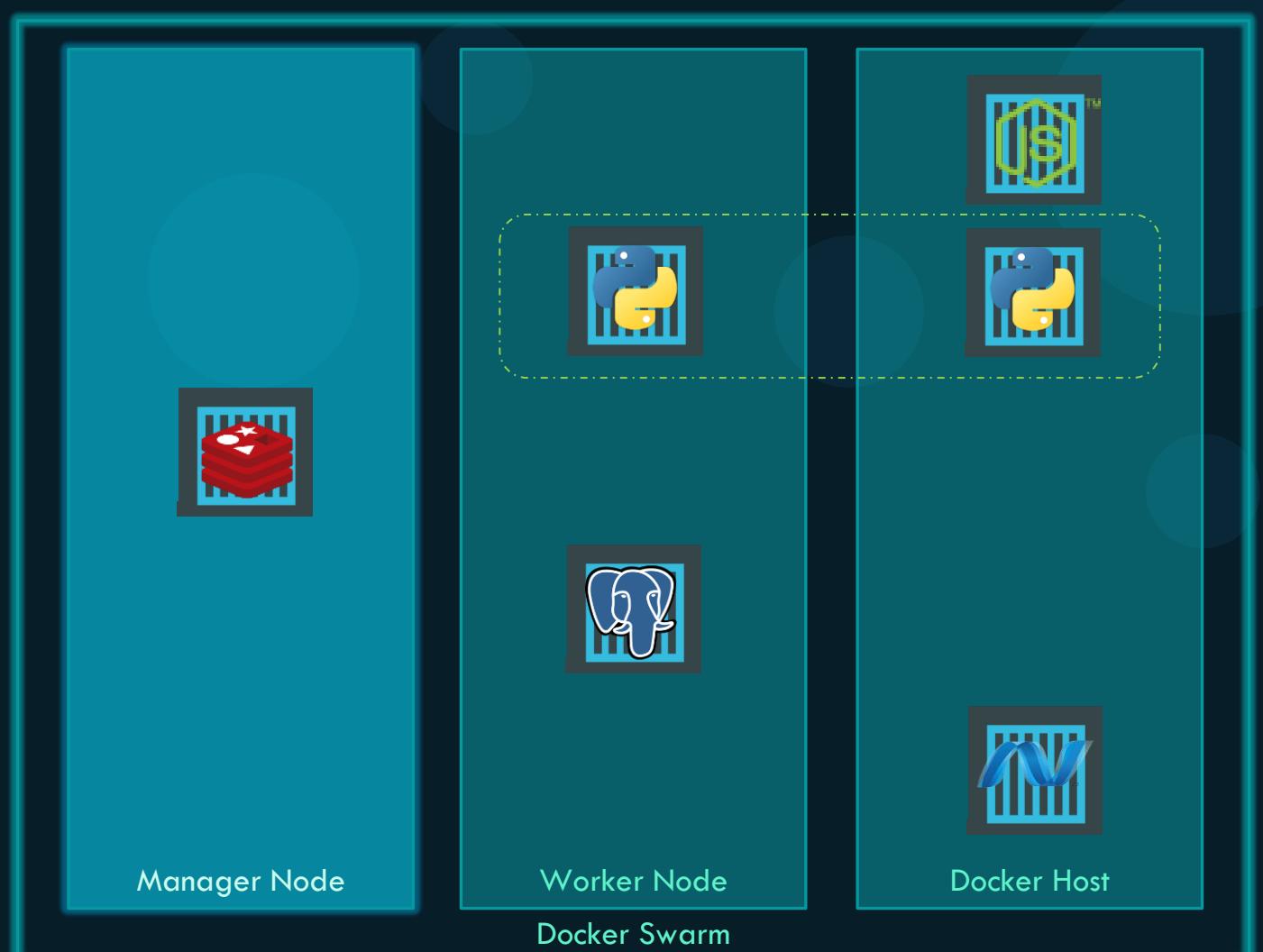
► docker-compose up



Docker Compose

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
```



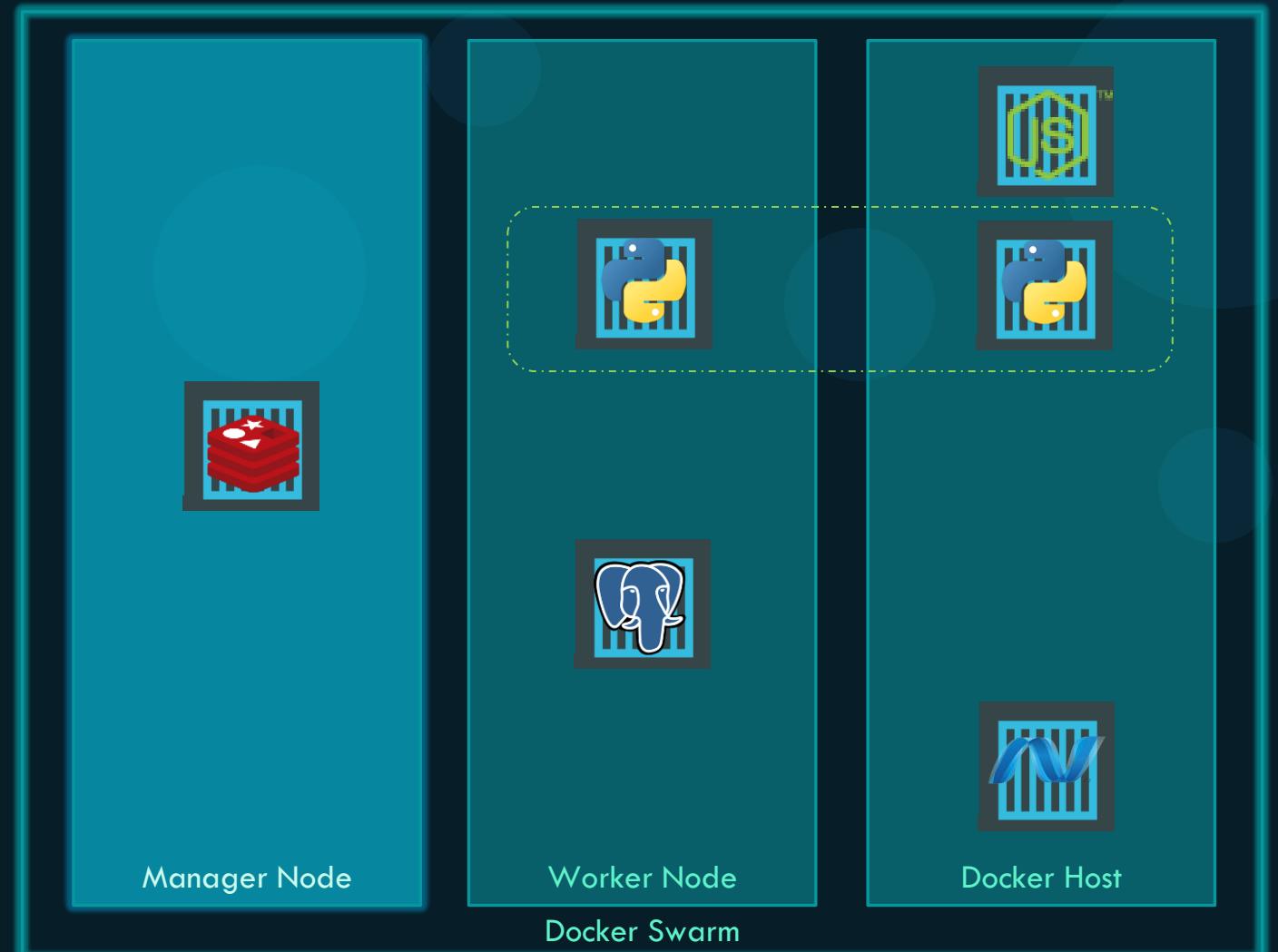
 docker stack deploy --compose-file docker-compose.yml

KODEKLOUD

Docker Compose

docker-compose.yml

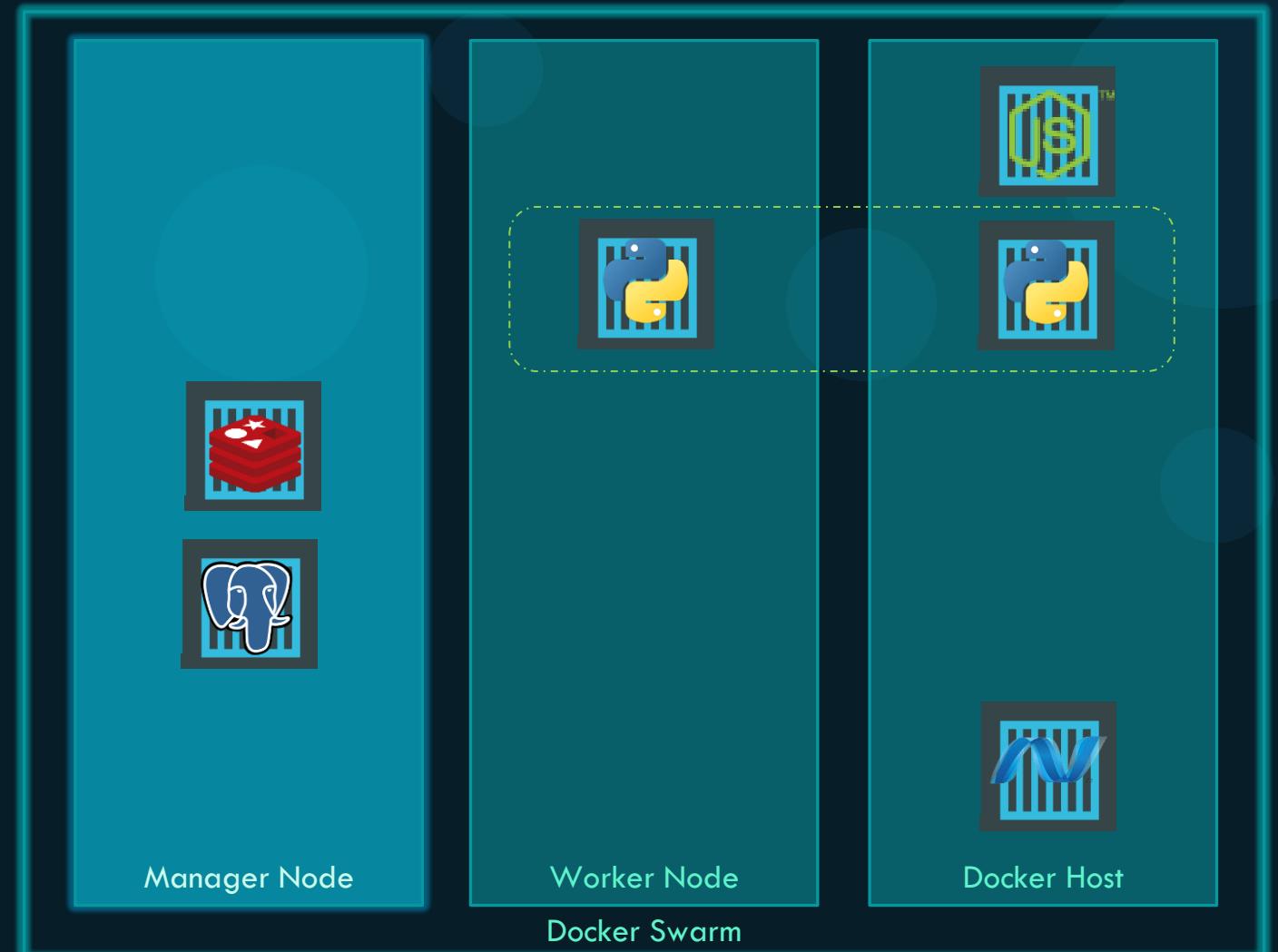
```
version: 3
services:
  redis:
    image: redis
    deploy:
      replicas: 1
  db:
    image: postgres:9.4
    deploy:
      replicas: 1
      placement:
        constraints:
          - node.role == manager
  vote:
    image: voting-app
    deploy:
      replicas: 2
  result:
    image: result
    deploy:
      replicas: 1
  worker:
    image: worker
```



Docker Compose

docker-compose.yml

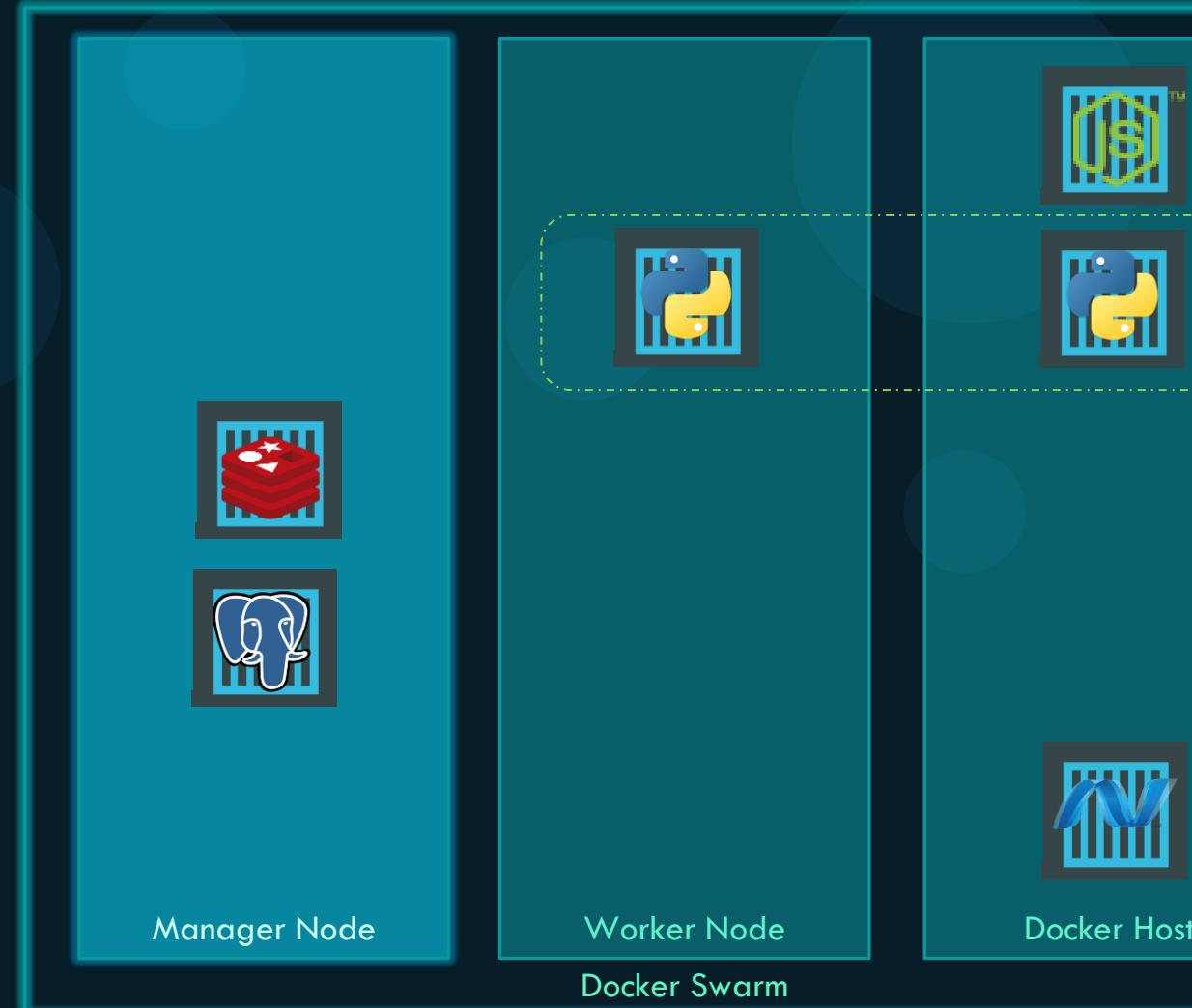
```
version: 3
services:
  redis:
    image: redis
    deploy:
      replicas: 1
  db:
    image: postgres:9.4
    deploy:
      replicas: 1
      placement:
        constraints:
          - node.role == manager
  vote:
    image: voting-app
    deploy:
      replicas: 2
      resources:
        limits:
          cpus: 0.01
          memory: 50M
```



Docker Compose

docker-compose.yml

```
version: 3
services:
  redis:
    image: redis
    deploy:
      replicas: 1
  db:
    image: postgres:9.4
    deploy:
      replicas: 1
      placement:
        constraints:
          - node.role == manager
  vote:
    image: voting-app
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost"]
      interval: 1m30s
      timeout: 10s
      retries: 3
      start_period: 40s
    deploy:
      replicas: 2
      resources:
```



Stack Commands

▶ docker stack deploy

▶ docker stack ls

▶ docker stack services

▶ docker stack ps

▶ docker stack rm





Curriculum

Docker Engine

Docker Swarm

Kubernetes

Docker Enterprise

- Kubernetes Architecture
- PODs
- ReplicaSets
- Deployments
- Services
- Commands & Arguments
- Environment Variables
- ConfigMaps
- Secrets
- Readiness Probes
- Liveness Probes
- Network Policies
- Volume driver plugins
- Volumes in Kubernetes
- PVs, PVCs, Storage Classes



Kubernetes Essentials



Kubernetes

Please add videos for earlier courses for:

- K8S overview
- POD
- RS
- Deployments
- Services

NOTE: The demo for Voting App using Kubernetes object has been already created and uploaded on drive.

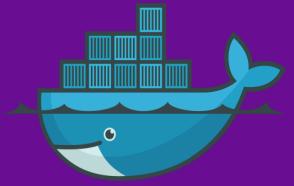




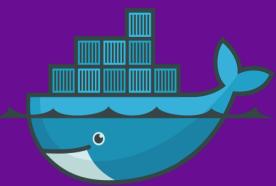
Docker Security



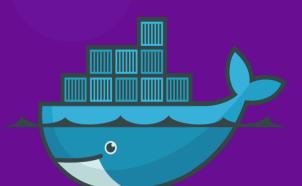
Security



Namespaces



Cgroups



Kernel
Capabilities

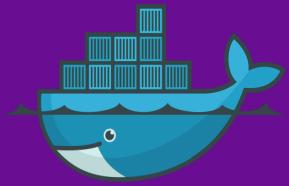


Other Kernel
Security

- Application Armor
- SeLinux
- GRSEC
- Seccomp



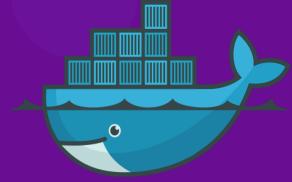
Security



Secure Docker
Swarm



Encrypted Overlay
Network



Docker Content
Trust and Signed
Image



RBAC

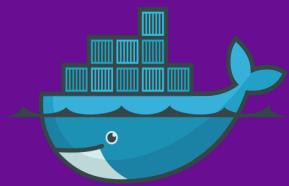


Image Scanning





Securing the Daemon

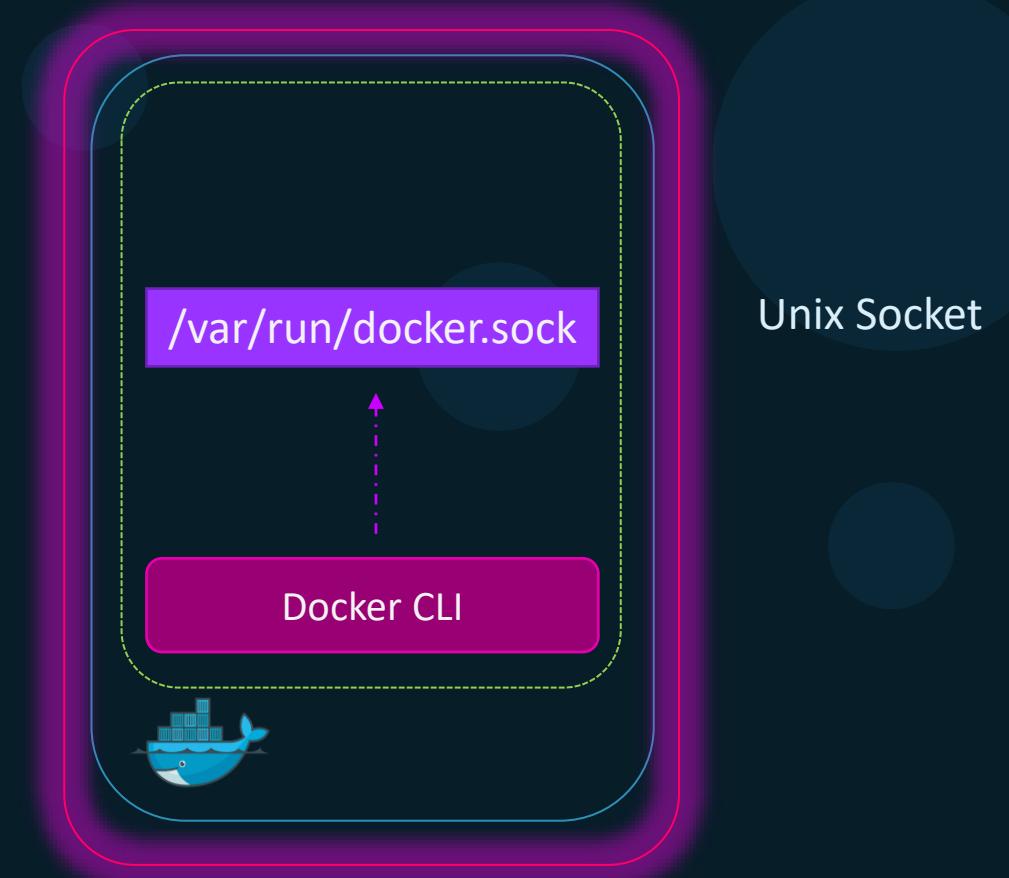


Secure Docker Server

- Delete existing containers hosting applications
- Delete volumes storing data
- Run containers to run their applications (bit coin mining)
- Gain root access to the host system by running a privileged container, which we will see in a bit.
- Target the other systems in the network and network itself

```
/etc/docker/daemon.json
```

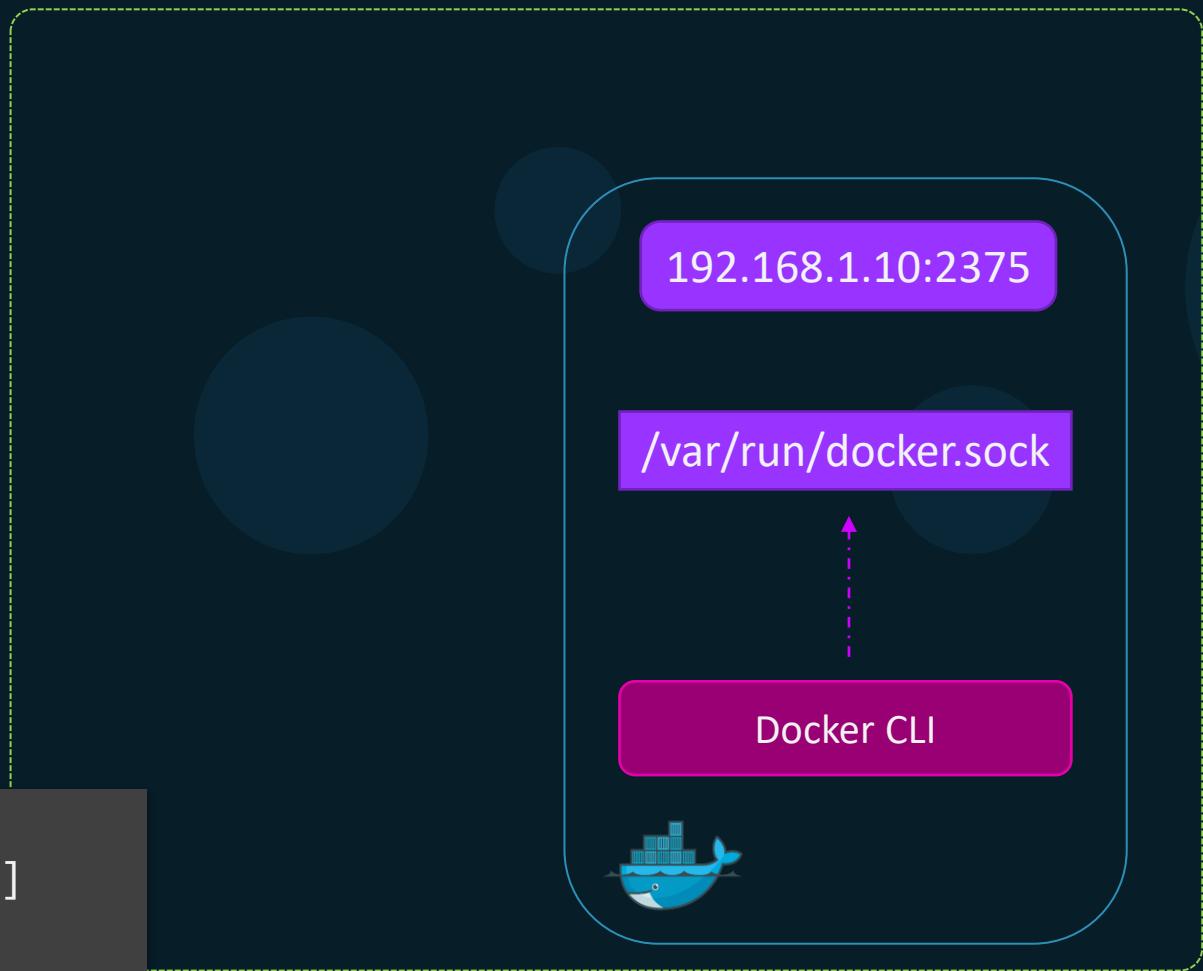
```
{  
  "hosts": ["tcp://192.168.1.10:2375"]}
```



- Disable Password based authentication
- Enable SSH key based authentication
- Determine users who needs access to the server

TLS Encryption

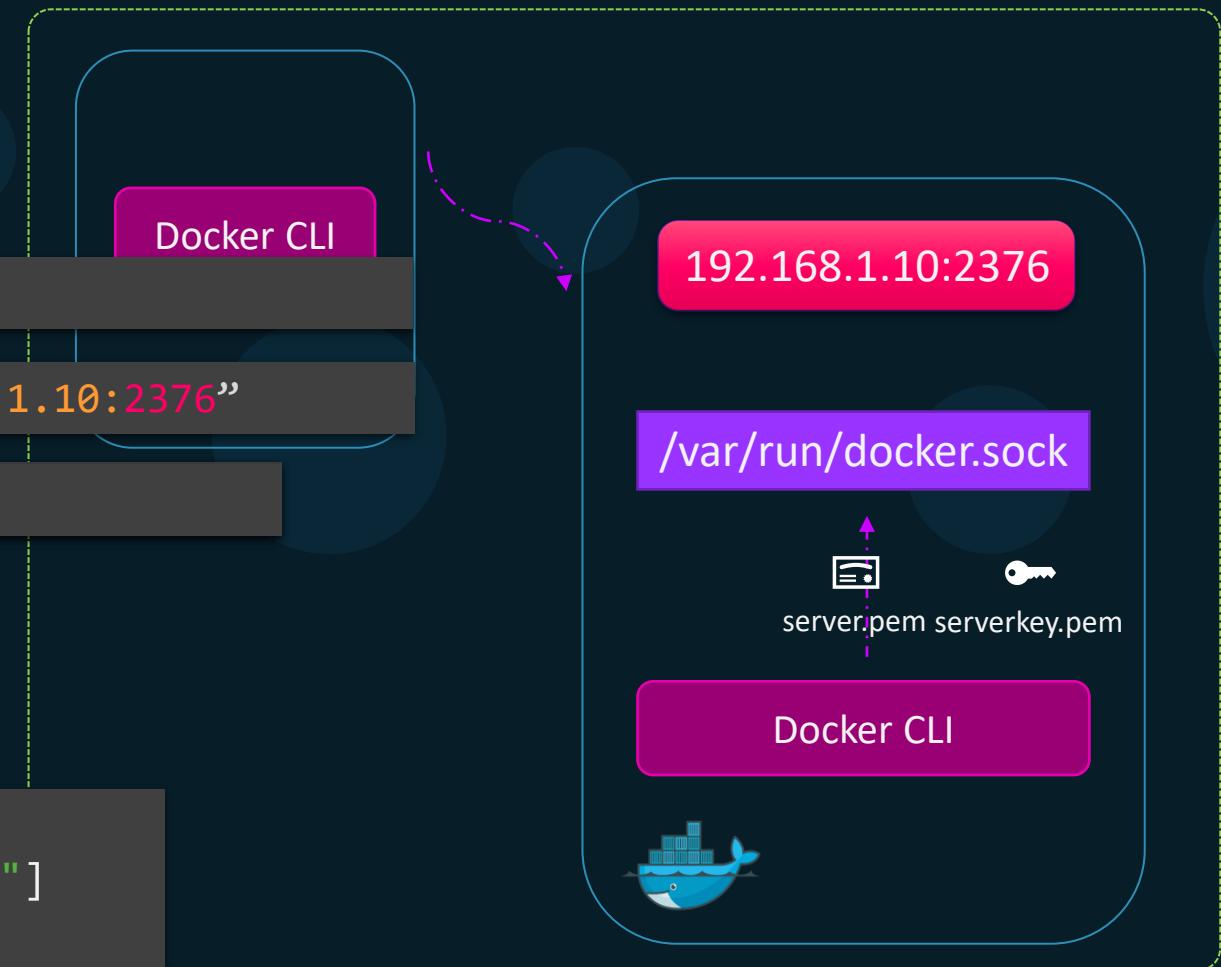
```
/etc/docker/daemon.json
{
  "hosts": ["tcp://192.168.1.10:2375"]
}
```



TLS Encryption

```
▶ export DOCKER_TLS=true  
▶ export DOCKER_HOST="tcp://192.168.1.10:2376"  
▶ docker --tls ps
```

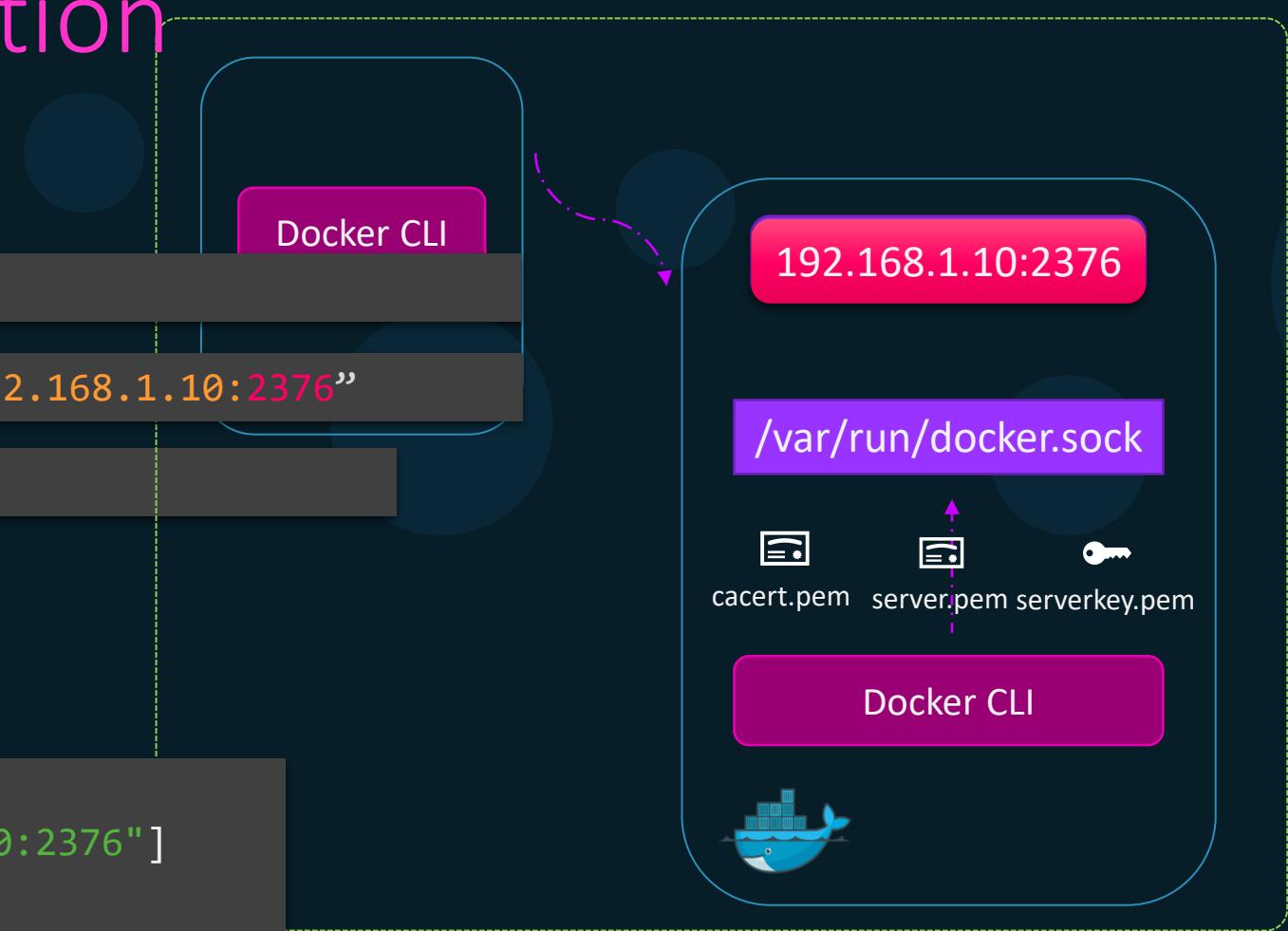
```
/etc/docker/daemon.json  
{  
  "hosts": ["tcp://192.168.1.10:2376"]  
  
  "tls": true,  
  "tlscert": "/var/docker/server.pem",  
  "tlskey": "/var/docker/serverkey.pem"  
}
```



TLS Authentication

```
▶ export DOCKER_TLS=true  
▶ export DOCKER_HOST="tcp://192.168.1.10:2376"  
▶ docker ps
```

```
/etc/docker/daemon.json  
{  
  "hosts": ["tcp://192.168.1.10:2376"]  
  
  "tls": true,  
  "tlscert": "/var/docker/server.pem",  
  "tlskey": "/var/docker/serverkey.pem",  
  "tlsverify": true,  
  "tlscacert": "/var/docker/caserver.pem"  
}
```



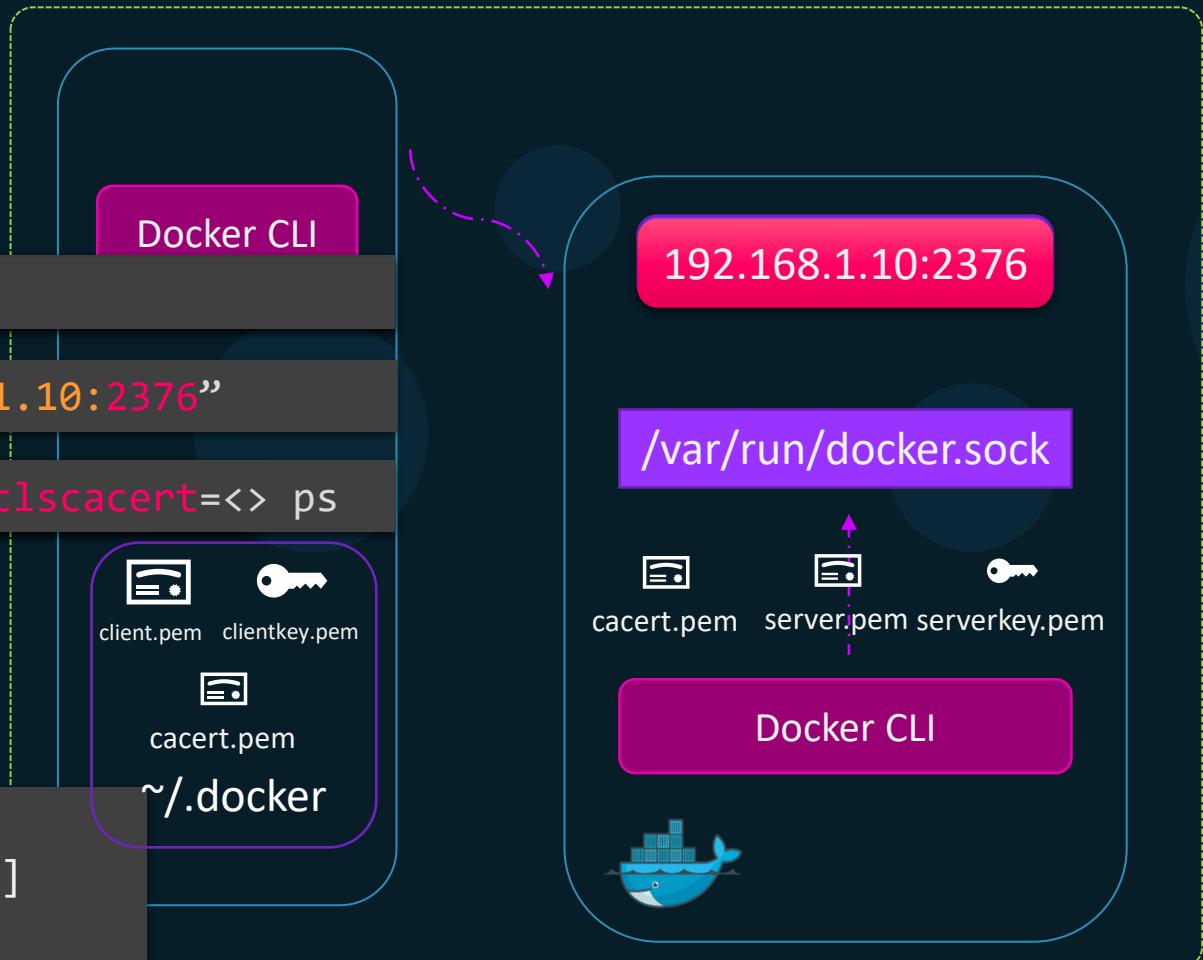
client.pem
 clientkey.pem

cacert.pem
 CA Server

Authentication

```
▶ export DOCKER_TLS_VERIFY=true  
▶ export DOCKER_HOST="tcp://192.168.1.10:2376"  
▶ docker --tlscert=<> --tlskey=<> --tlsCACert=<> ps
```

```
/etc/docker/daemon.json  
{  
  "hosts": ["tcp://192.168.1.10:2376"]  
  
  "tls": true,  
  "tlscert": "/var/docker/server.pem",  
  "tlskey": "/var/docker/serverkey.pem",  
  "tlsverify": true,  
  "tlsCACert": "/var/docker/caserver.pem"  
}
```



Summary

```
/etc/docker/daemon.json
```

```
{  
  "hosts": ["tcp://192.168.1.10:2376"]  
  "tls": true,  
  "tlscert": "/var/docker/server.pem",  
  "tlskey": "/var/docker/serverkey.pem"  
}
```

```
▶ docker --tls ps
```

Without Authentication

```
/etc/docker/daemon.json
```

```
{  
  "hosts": ["tcp://192.168.1.10:2376"]  
  "tlscert": "/var/docker/server.pem",  
  "tlskey": "/var/docker/serverkey.pem",  
  "tlsverify": true,  
  "tlscacert": "/var/docker/caserver.pem"  
}
```

```
▶ docker --tlsverify  
  --tlscert=<> --tlskey=<>  
  --tlscacert=<> ps
```

With Authentication



References

<https://docs.docker.com/engine/security/https/>



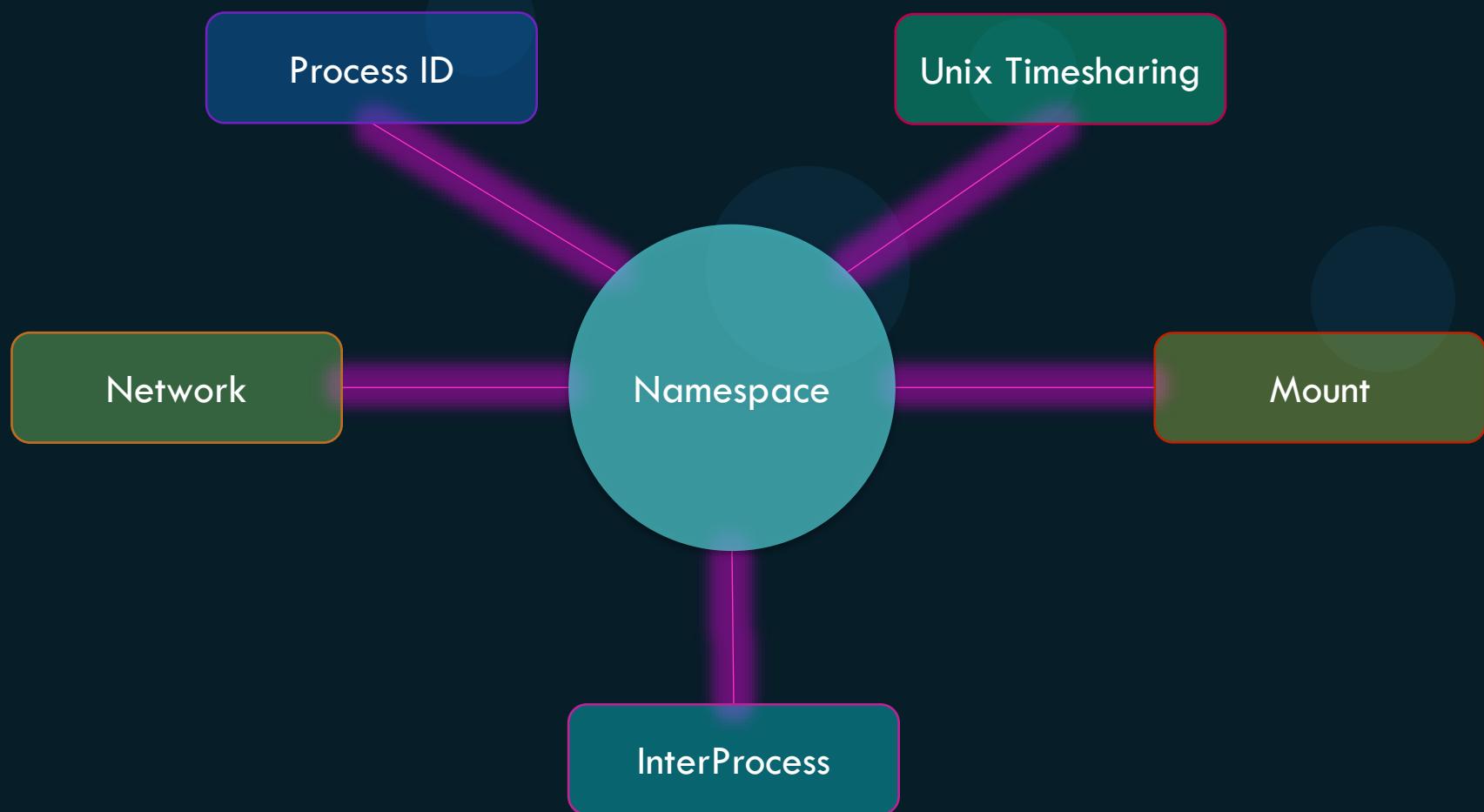


KODEKLOUD

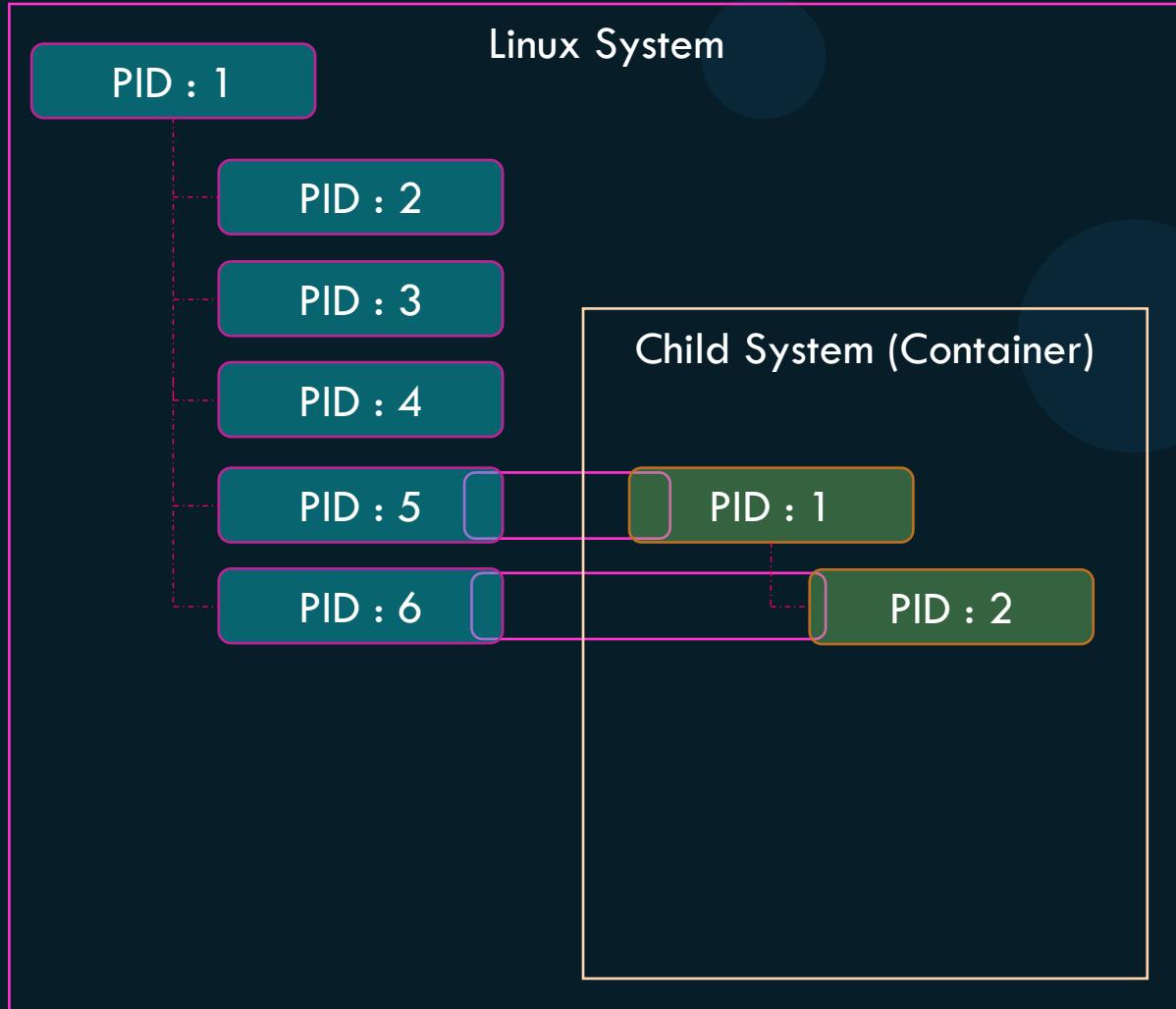
Namespaces



Containerization



Namespace - PID



(On the container)

▶ ps aux

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.0	4528	828	?	Ss	03:06	0:00	nginx

(On the host)

▶ ps aux

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
project	3720	0.1	0.1	95500	4916	?	R	06:06	0:00	sshd: project@
project	3725	0.0	0.1	95196	4132	?	S	06:06	0:00	sshd: project@
project	3727	0.2	0.1	21352	5340	pts/0	Ss	06:06	0:00	-bash
root	3802	0.0	0.0	8924	3616	?	S1	06:06	0:00	docker-contain
shim	-namespace	m								
root	3816	1.0	0.0	4528	828	?	Ss	06:06	0:00	nginx



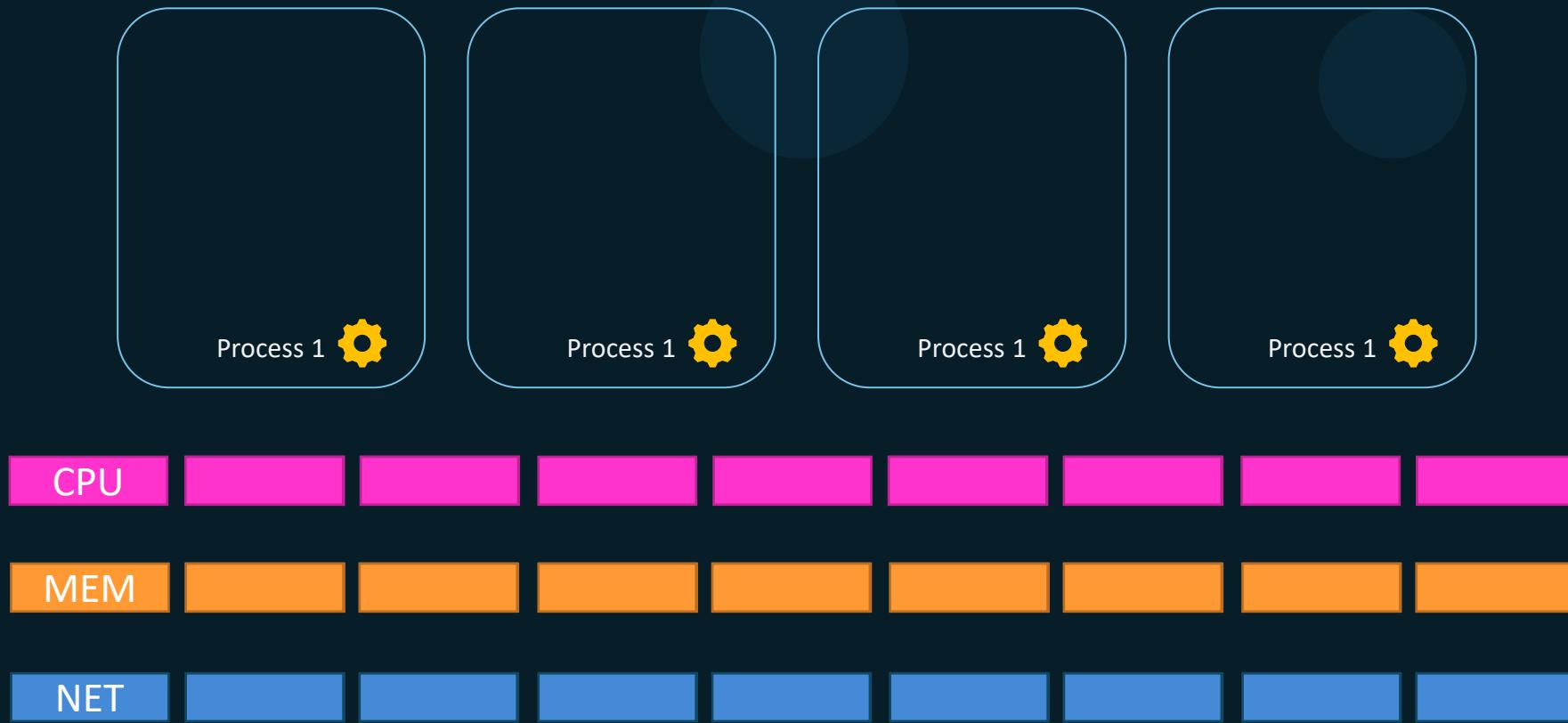


KODEKLOUD

cGroups



CGroups

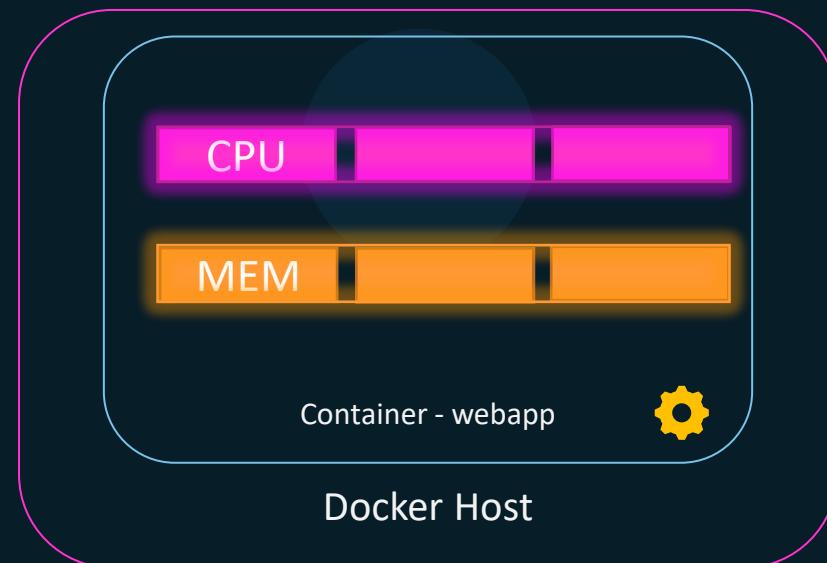




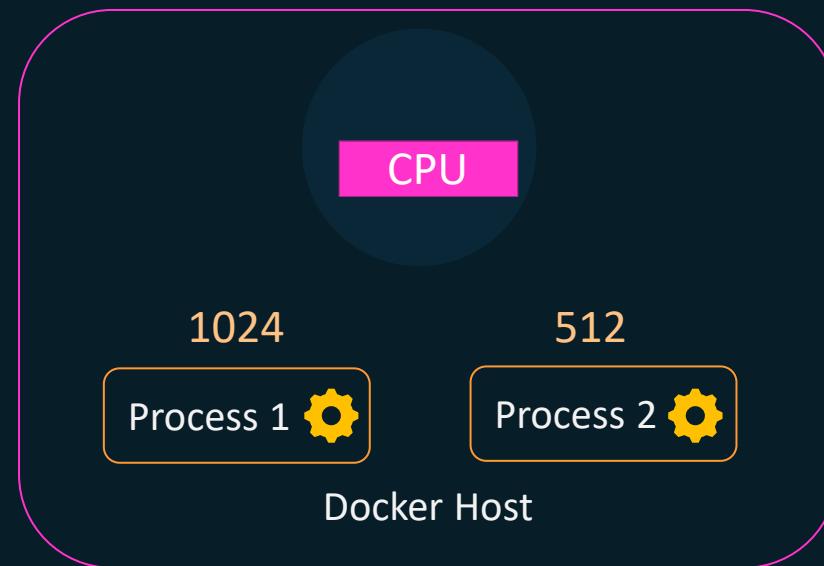
KODEKLOUD

Resource Constraints

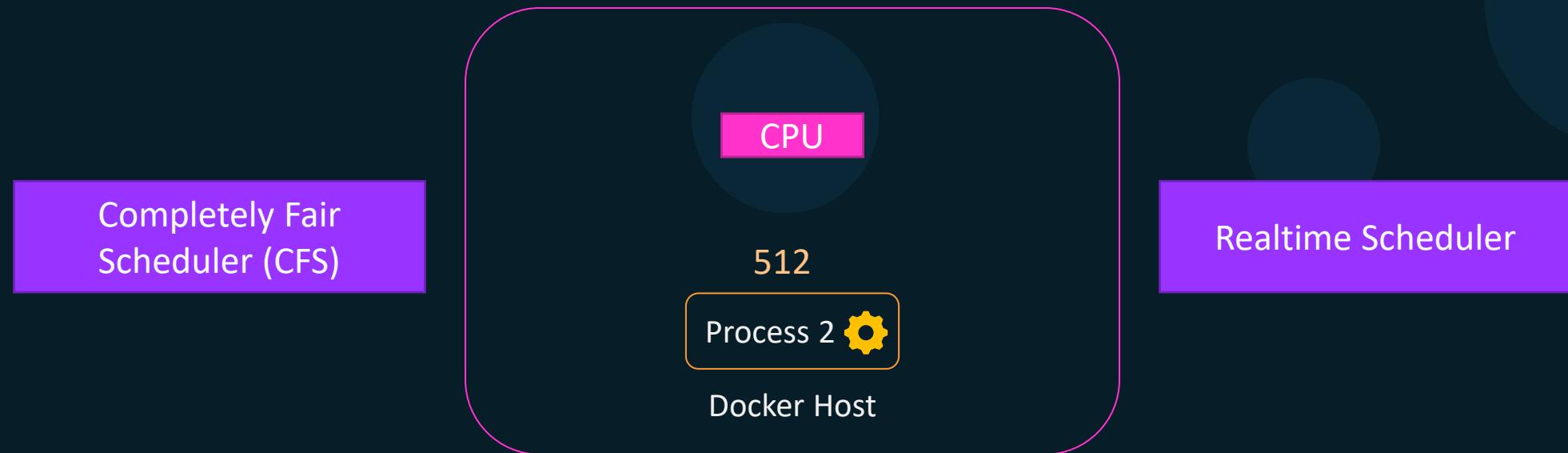
Container Memory – Limit and Reservations



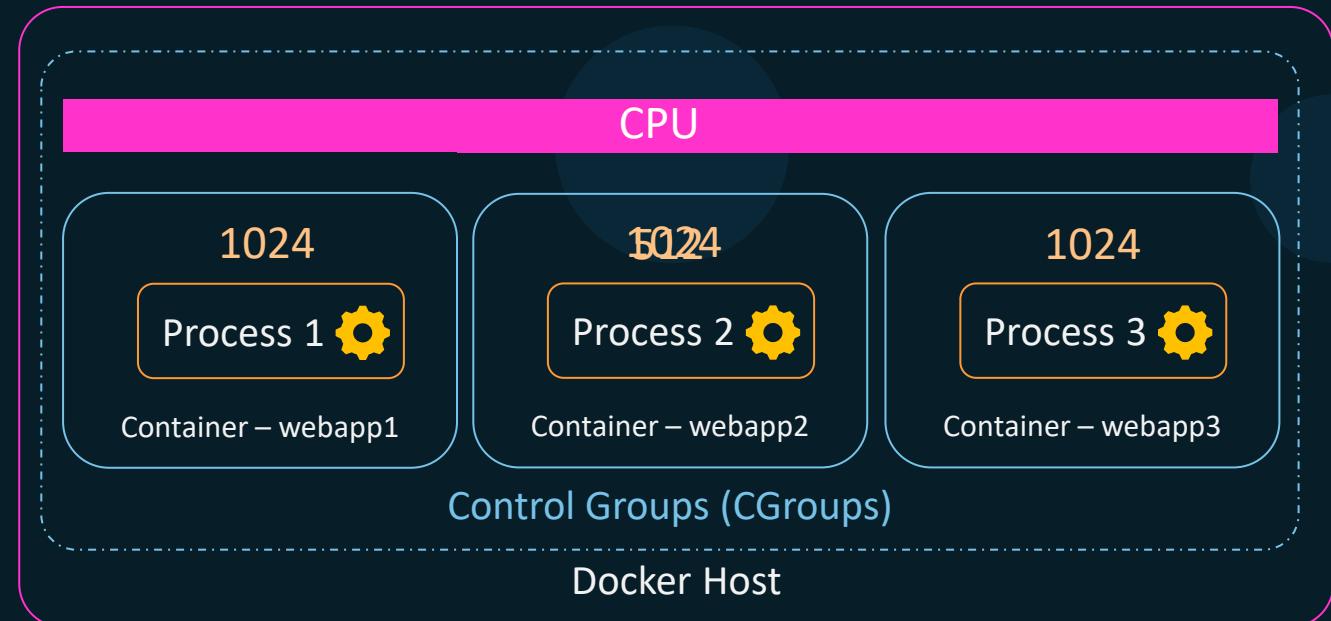
Linux – CPU Sharing



Linux – CPU Sharing



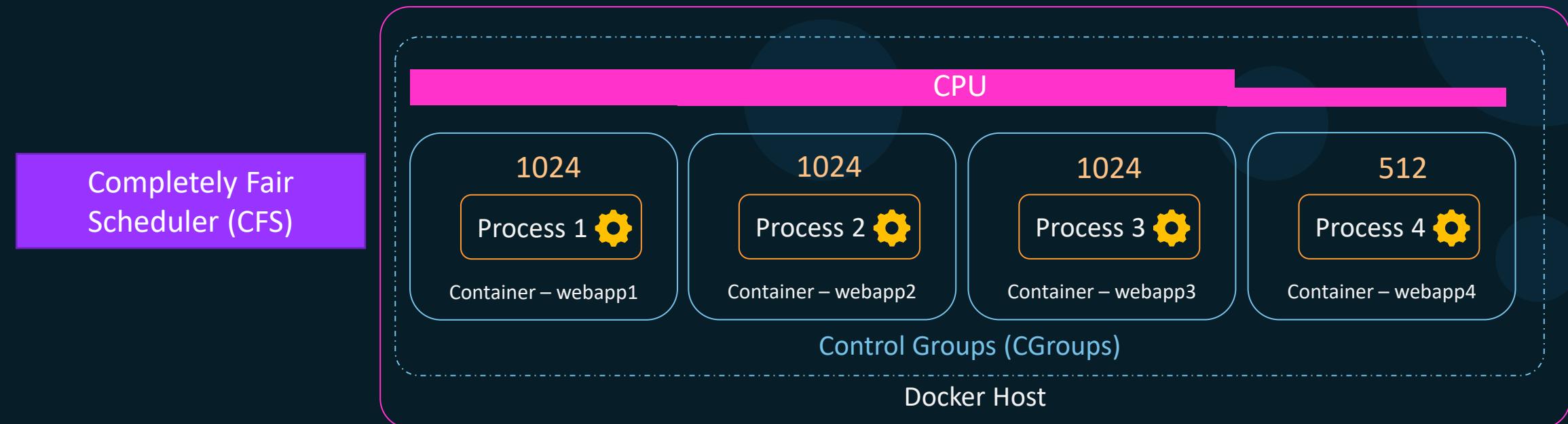
Containers – CPU Shares



```
▶ docker container run --cpu-shares=512 webapp4
```



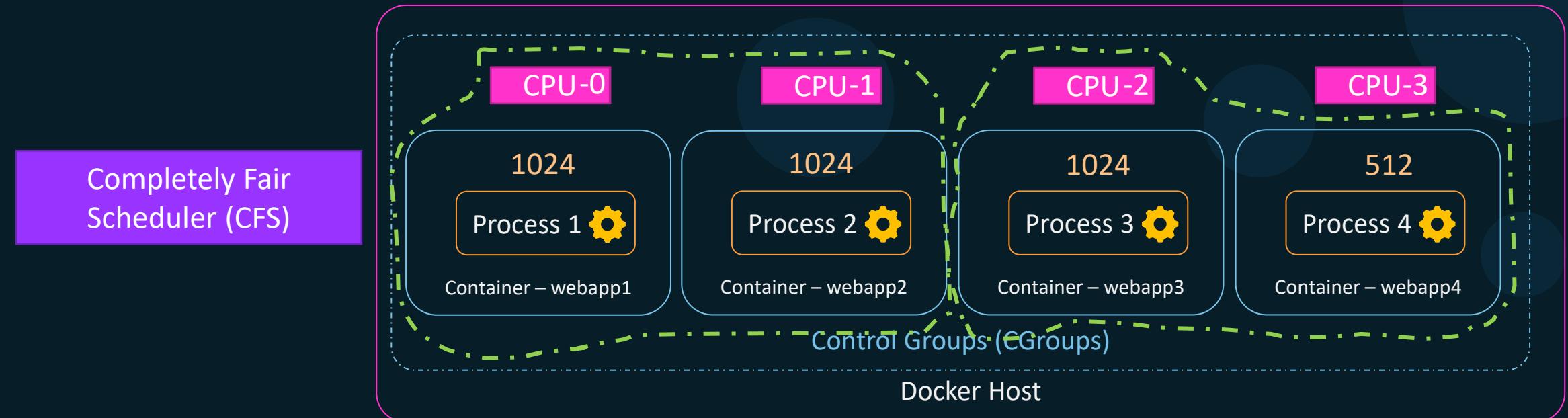
Containers – CPU Sets



```
▶ docker container run --cpu-shares=512 webapp4
```



Containers – CPU Sets



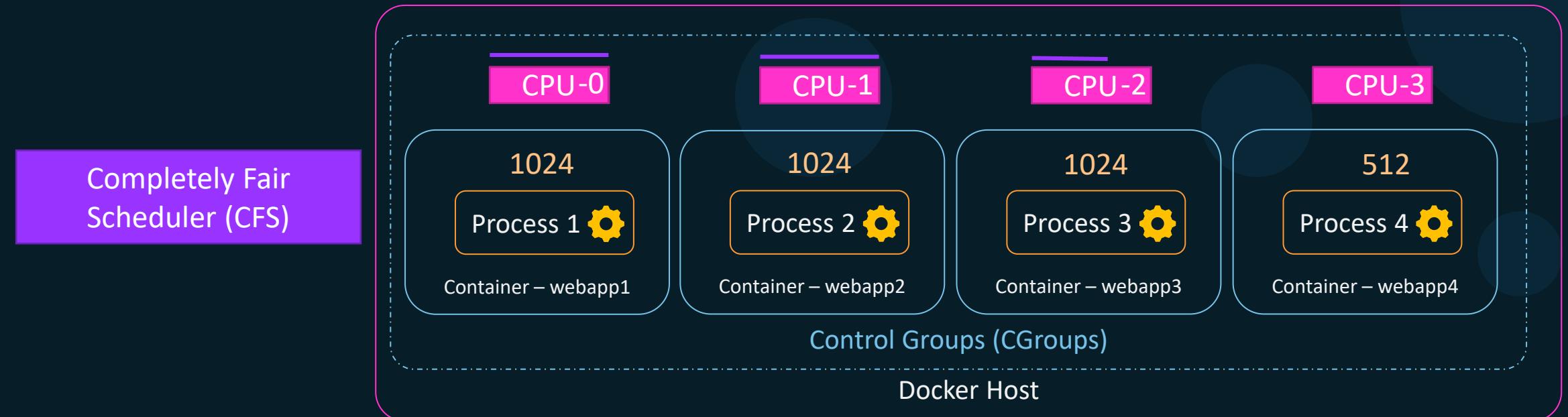
```
▶ docker container run --cpuset-cpus=0-1 webapp1
```

```
▶ docker container run --cpuset-cpus=2 webapp3
```

```
▶ docker container run --cpuset-cpus=0-1 webapp2
```

```
▶ docker container run --cpuset-cpus=2 webapp4
```

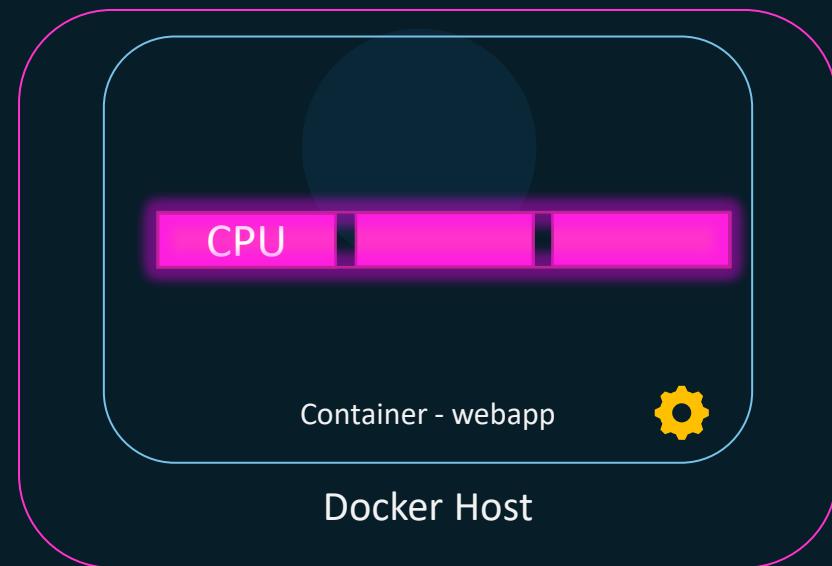
Containers – CPU Count



```
▶ docker container run --cpus=2.5 webapp4
```

```
▶ docker container update --cpus=0.5 webapp4
```

Containers – CPU Sharing

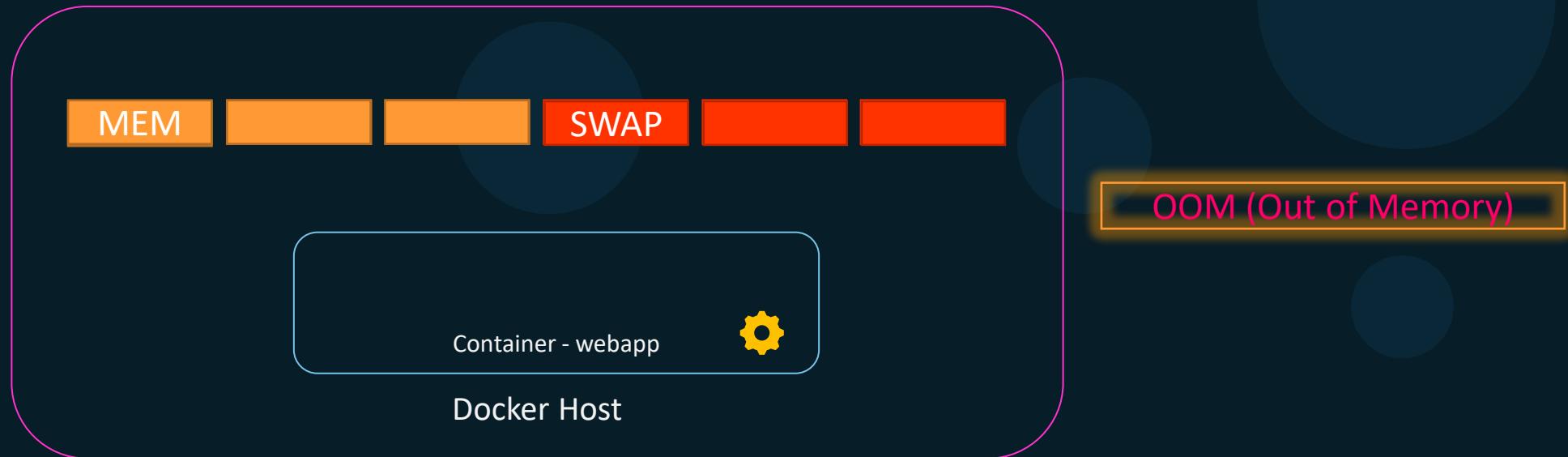




KODEKLOUD

Resource Constraints - Memory

Linux – Memory



```
▶ docker container run --memory=512m webapp
```

```
▶ docker container run --memory=512m --memory-swap=512m webapp
```

Swap Space = $512\text{m} - 512\text{m} = 0\text{m}$

```
▶ docker container run --memory=512m --memory-swap=768m webapp
```

Swap Space = $768\text{m} - 512\text{m} = 256\text{m}$

References

<https://www.cyberark.com/resources/threat-research-blog/the-route-to-root-container-escape-using-kernel-exploitation>





Curriculum

Docker Engine

Docker Swarm

Kubernetes

Docker Enterprise

- Docker EE Introduction
- Docker Enterprise Engine Setup
- Universal Control Plane Setup
- Node Addition in UCP cluster
- Docker Trusted Registry Setup
- Deployment in Docker EE
- Docker EE UCP Client Bundle
- RBAC
- UCP Setting for LDAP integration
- Docker EE

- Docker Trusted Registry
- Image Scanning
- Image Promotions
- Garbage Collection
- Docker Content Trust and Image Signing
- Docker Trusted Registry

- Backup & Disaster Recovery



Docker Enterprise



Docker EE



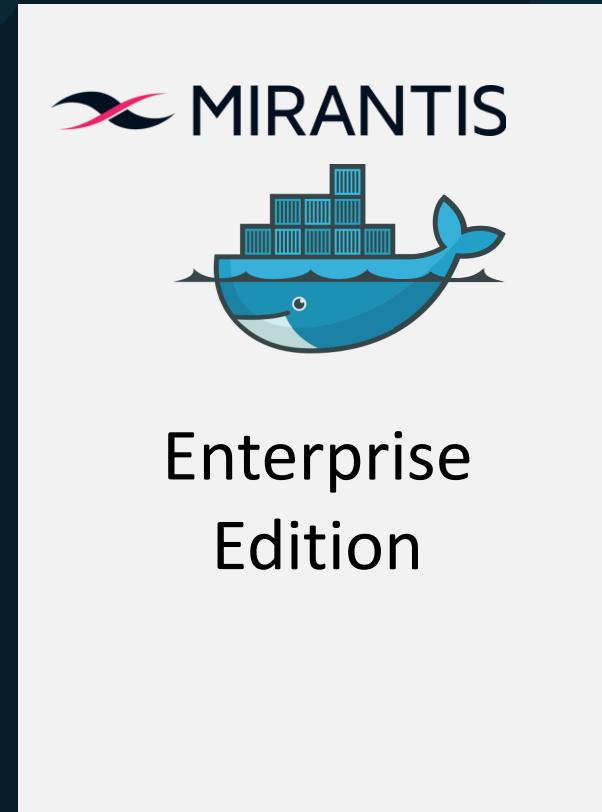
Community
Edition



Enterprise
Edition



Docker EE



Docker EE



Enterprise Edition

Security & Access Control

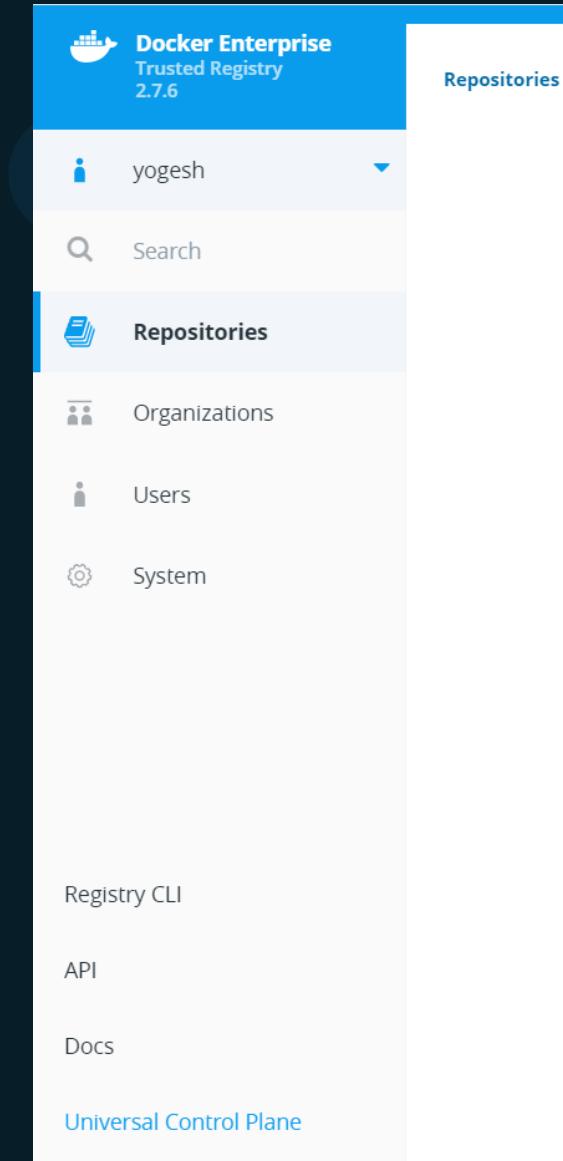
Trusted Registry

Universal Control Plane

Kubernetes Service

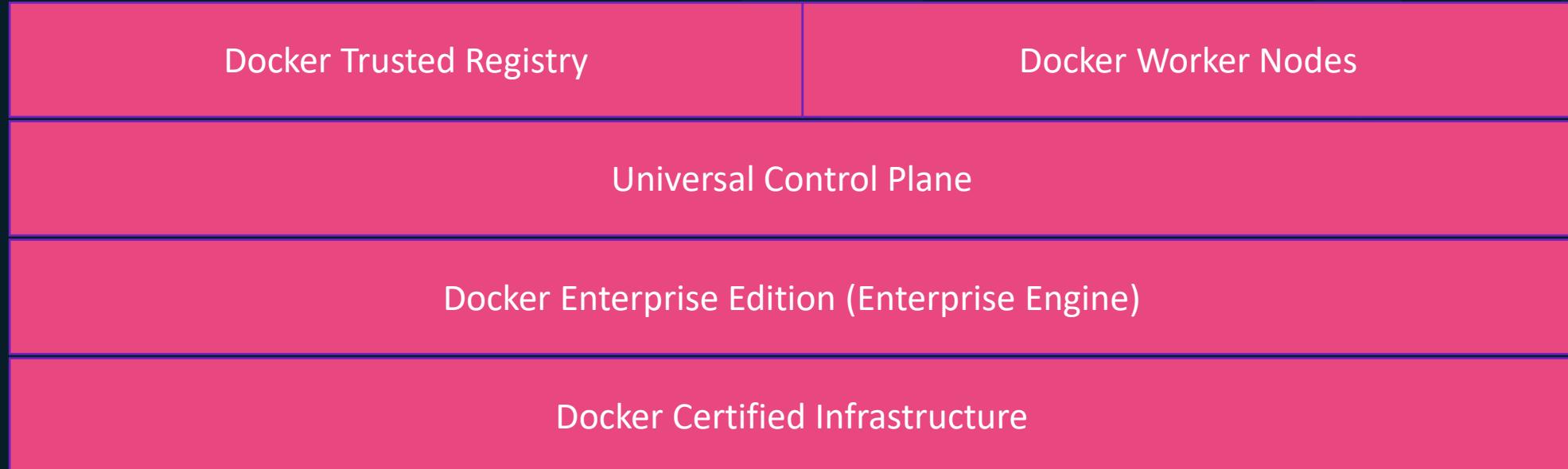
Docker Swarm Service

Docker Engine - Enterprise



A screenshot of the Docker Enterprise Trusted Registry interface. The top navigation bar is blue with the text "Docker Enterprise Trusted Registry 2.7.6". The left sidebar has a "Repositories" section selected, showing options like "yogesh", "Search", "Repositories", "Organizations", "Users", and "System". The main content area shows "Registry CLI", "API", "Docs", and "Universal Control Plane".

Docker EE



Pre-Requisites

- Linux Kernel Version 3.10 or higher for Managers
- Static IP and Persistent Host Name
- Network Connectivity Between all Servers
- Time Sync (NTP)
- User namespaces should not be configured on any node
(Currently not supported)
- Docker Engine - Enterprise



UCP - Minimum Requirements

- 8 GB of RAM for manager nodes (16GB)
- 4 GB of RAM for worker nodes
- 2 vCPUs for manager nodes (4 vCPUs)
- 10 GB of free disk space for the /var partition for manager nodes (25-100GB)
- 500 MB of free disk space for the /var partition for worker nodes



DTR - Minimum Requirements

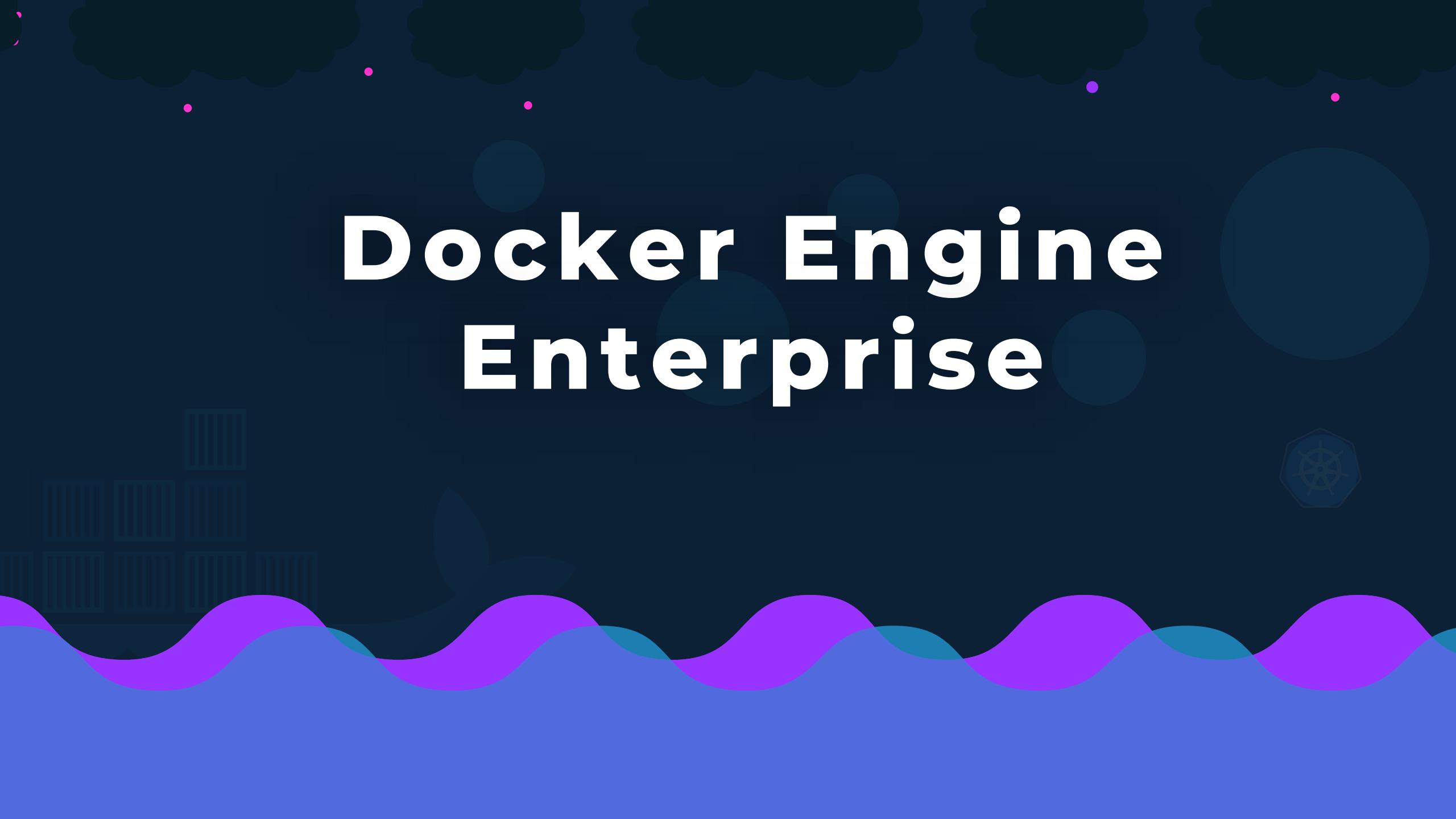
- 16 GB of RAM
- 2 vCPUs (4 vCPUs)
- 10 GB of free disk space (100GB)
- Port 80 and 443





KODEKLOUD

Docker Engine Enterprise



Docker Enterprise Engine Setup



Docker Enterprise Trial

By [Docker](#)

The best way to try Docker on any infrastructure. Includes entitlement to Docker Enterprise and Docker Datacenter (Universal Control Plane, Docker Trusted Registry, and Docker Security Scanner).

Edition

Docker Certified

Linux

Windows

x86-64

IBM Z



[Get Docker Enterprise Trial](#)

Includes Docker Enterprise and Docker Datacenter (UCP, DTR, and DSS) trial. Business Day or Business Critical support is not included with your trial but can be purchased as part of a Docker Enterprise subscription.

[Contact Sales](#) for additional nodes.

[Start 1 Month Trial](#)



KODEKLOUD

Docker Enterprise Engine Setup

▶ docker version

Client: Docker Engine - Enterprise

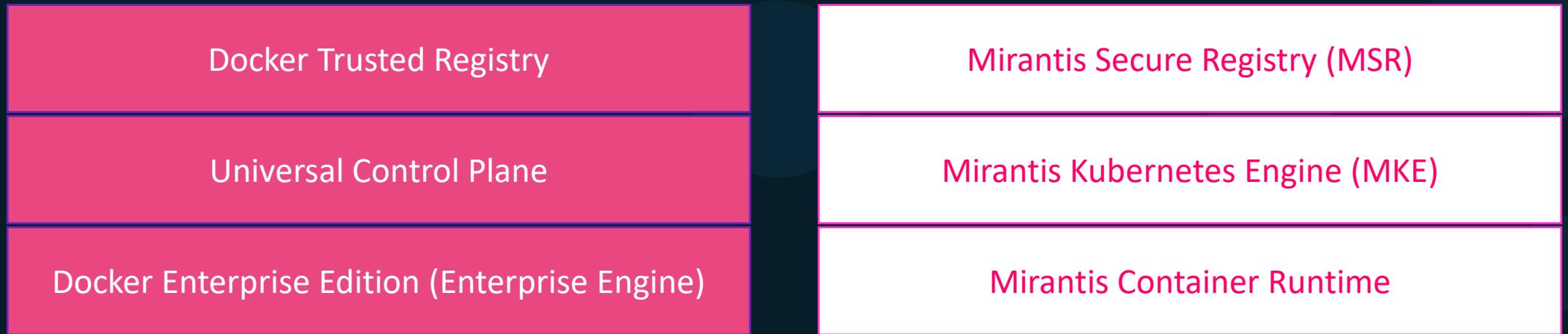
```
Version:          19.03.5
API version:     1.40
Go version:      go1.12.12
Git commit:      2ee0c57608
Built:           Wed Nov 13 07:36:57 2019
OS/Arch:         linux/amd64
Experimental:    false
```

Server: Docker Engine - Enterprise

```
Engine:
  Version:          19.03.5
  API version:     1.40 (minimum version 1.12)
  Go version:      go1.12.12
  Git commit:      2ee0c57608
  Built:           Wed Nov 13 07:35:23 2019
  OS/Arch:         linux/amd64
  Experimental:    false
containerd:
  Version:          1.2.10
  GitCommit:        b34a5c8af56e510852c35414db4c1f4fa6172339
runc:
  Version:          1.0.0-rc8+dev
  GitCommit:        3e425f80a8c931f88e6d94a8c831b9d5aa481657
docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
```



Note!



Note!

The screenshot shows the Docker Docs website with a dark blue header. The header includes the "docker docs" logo, a search bar with the placeholder "Search the docs", and navigation links for "Guides" and "Product n...". The main content area has a light gray background. On the left, there's a sidebar with sections like "Release notes", "Superseded products and tools", "Docker Desktop Enterprise", "Docker Enterprise" (which is expanded to show "Docker Enterprise", "Install or upgrade", "Licensing", "Get support", "Docker Cluster", "Docker Engine - Enterprise" (expanded to show "Overview", "Release notes", "Linux"), and "Docker user guide". The main content area features a large title "Docker Engine" with a subtitle "Estimated reading time: 1 minute". Below this, there's a paragraph about Docker Engine being an open source client-server application, followed by a bulleted list of its components: "A server with a long-running", "APIs which specify interface", and "A command line interface (CLI)". There's also a note about the CLI using Docker APIs to control Docker applications, mentioning Docker networks, and volumes. A link to "Docker Architecture" is provided for more details.

The screenshot shows the Mirantis website with a white background. At the top, there's a header with the Mirantis logo and the URL "docs.mirantis.com". The main content area features a large title "MIRANTIS" with a red swoosh icon. To the right, there's a breadcrumb navigation path: "HOME / Docker Enterprise products / Miran...". Below this, there's a section titled "Docker Engine - Enterprise is" with a note: "The product formerly known as Docker (MCR)." In the center, there's a list of products under the heading "Mirantis Container Runtime": "Docker Enterprise", "Mirantis Container Runtime" (which is expanded to show "Install Mirantis Container Runtime on Linux distros", "Install Mirantis Container Runtime on Windows Servers"), "Mirantis Kubernetes Engine", "Mirantis Secure Registry", "Cluster", and "Get support". To the right, there's another section titled "Mirantis Container" with a note: "Mirantis Container Runtime is a client-ser..." followed by a bulleted list: "A server which is a type of long-run..." and "A REST API which specifies interface...".



Universal Control Plane



UCP

Not secure | 52.90.239.129/manage/dashboard

Docker Enterprise
Universal Control Plane
v3.2.6

yogesh

Dashboard

Access Control

Shared Resources

Kubernetes

Swarm

Docs

Kubernetes API Docs

Live API

MANAGER NODES

Ready	Errors	0
Warnings	0	Pending

WORKER NODES

Ready	Errors	0
Warnings	0	Pending

LAST 6 HOURS

1 MANAGER NODE

Max CPU **12.51%** Max Memory **13.26%** Max Used Disk **20.57%**

20%
15%
10%

11:00 AM 12:00 PM 01:00 PM 02:00 PM 03:00 PM 04:00 PM

0 WORKER NODES

No Data

Max CPU - Max Memory - Max Used Disk -

SWARM

Services

Active	0
Errors	0

Updating

KUBERNETES

default

Pods

Running	0
Errors	0

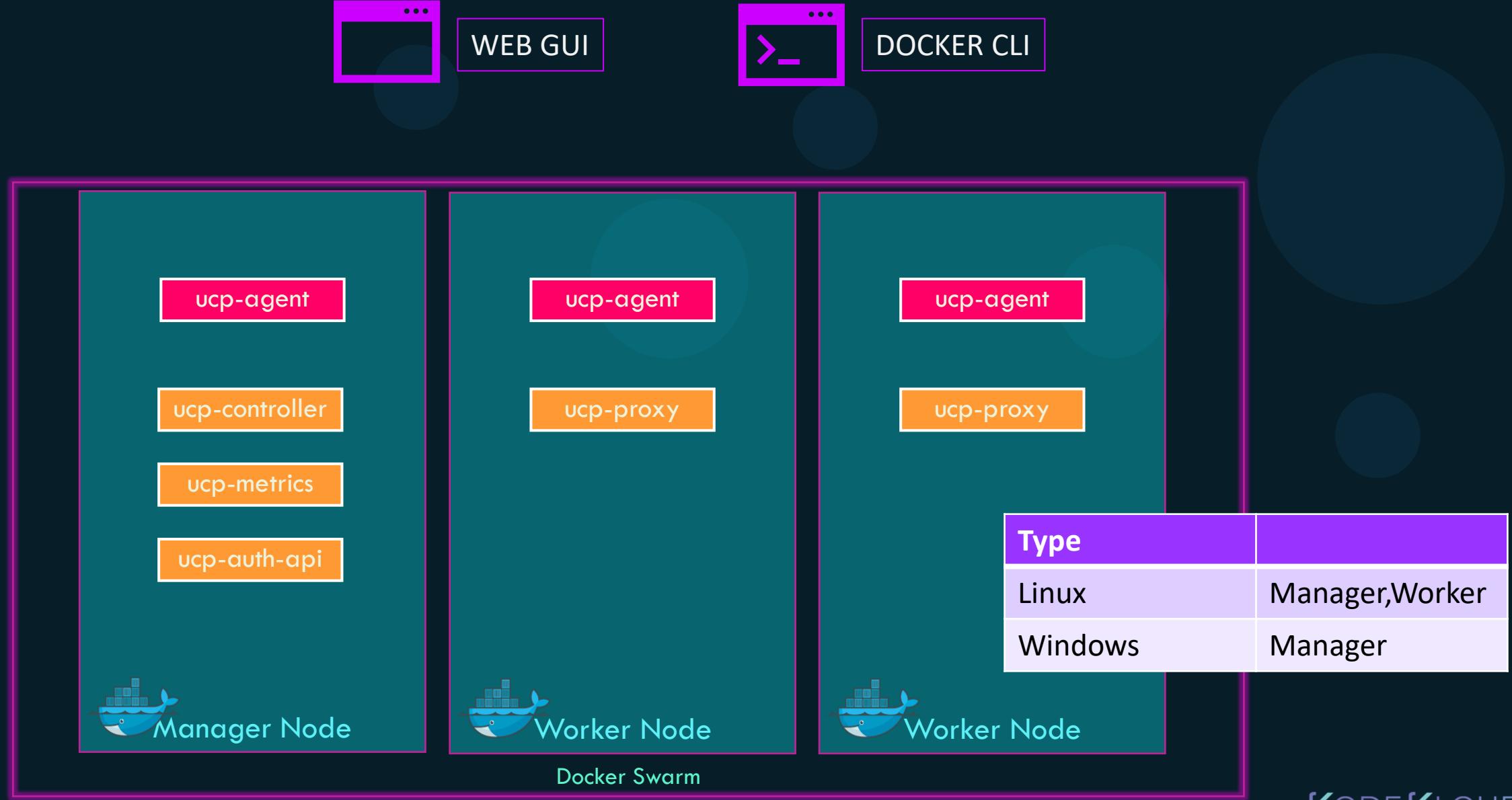
Pending

Controllers

None

The screenshot displays the Docker Enterprise Universal Control Plane (UCP) dashboard. On the left, a sidebar shows navigation links for Dashboard, Access Control, Shared Resources, Kubernetes, Swarm, Docs, Kubernetes API Docs, and Live API. The main content area is titled 'MANAGER NODES' and shows one node named 'yogesh' with 1 Ready node, 0 Errors, and 0 Warnings. It also shows 0 Worker nodes. A chart titled '1 MANAGER NODE' tracks resource usage over the last 6 hours, with CPU at 12.51%, Memory at 13.26%, and Disk at 20.57%. Below this, a section for '0 WORKER NODES' indicates 'No Data'. To the right, there are sections for 'SWARM' (Services, Active 0, Errors 0, Updating 0) and 'KUBERNETES' (default namespace, Pods (Running 0, Errors 0, Pending 0), Controllers (None)). The top of the page includes a header bar with back, forward, and refresh buttons, and a status message 'Not secure | 52.90.239.129/manage/dashboard'.

UCP



UCP Setup

Make sure Docker EE is up and running

Run a container with the `docker/ucp` image

Set the Admin Username and Password for UCP Console

Login into the Browser

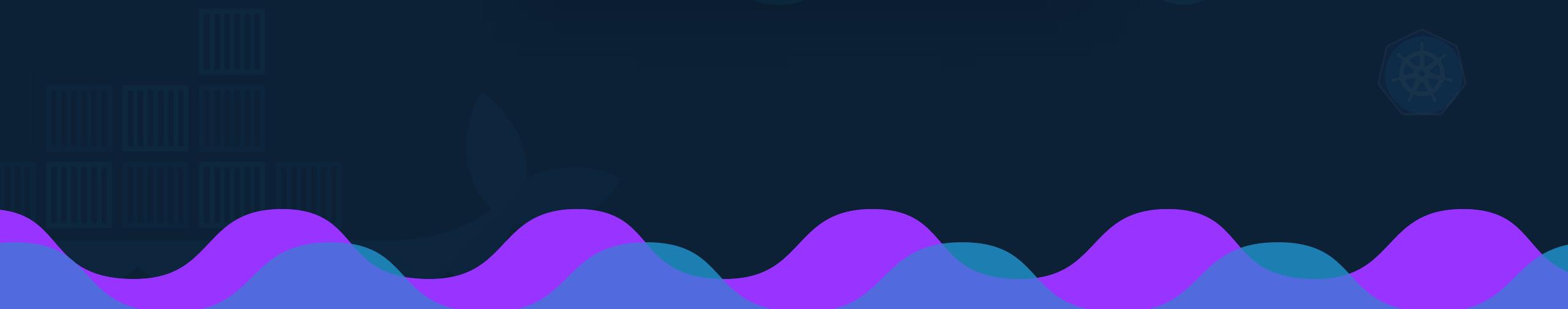
Download and Provide the Docker EE License

Add more Managers and Workers as per requirement

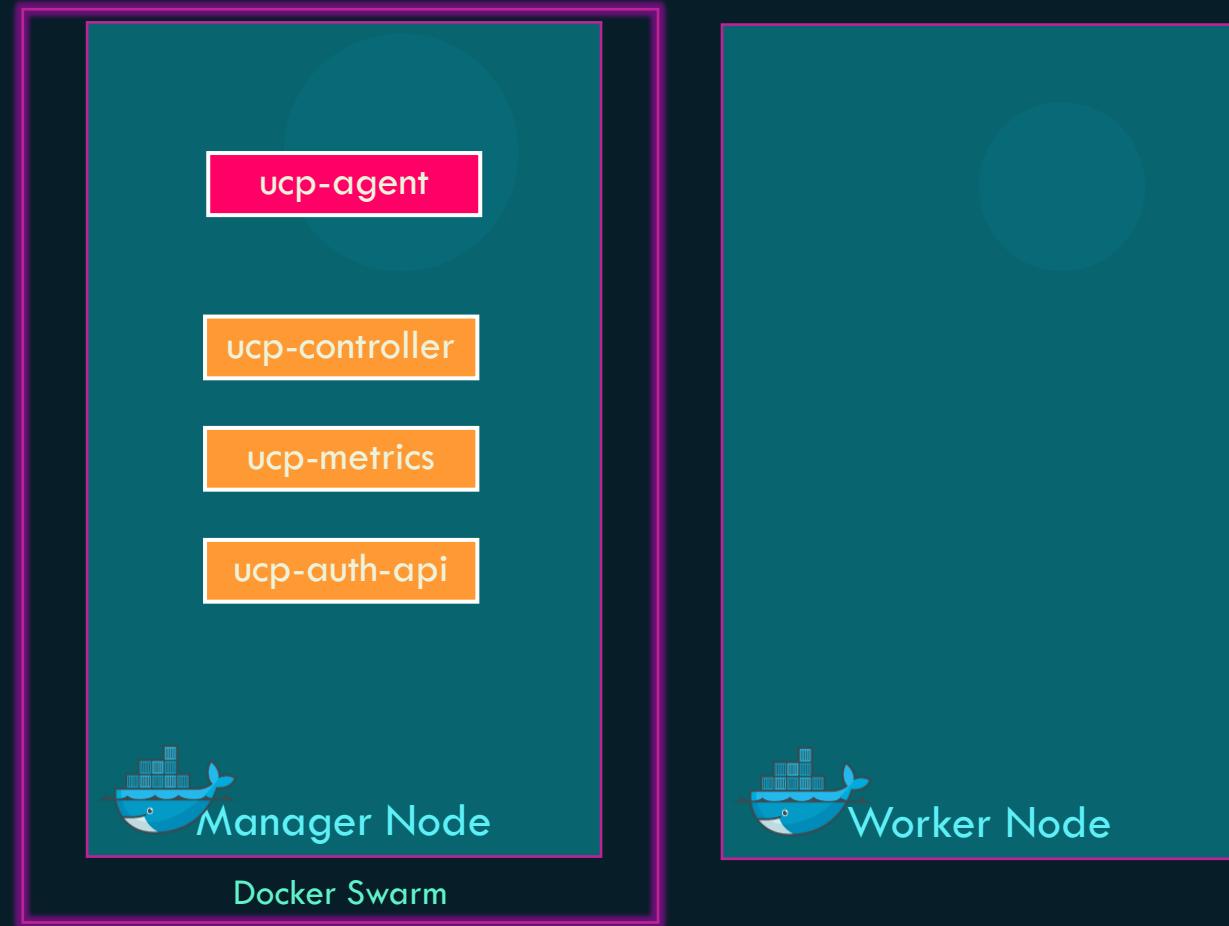




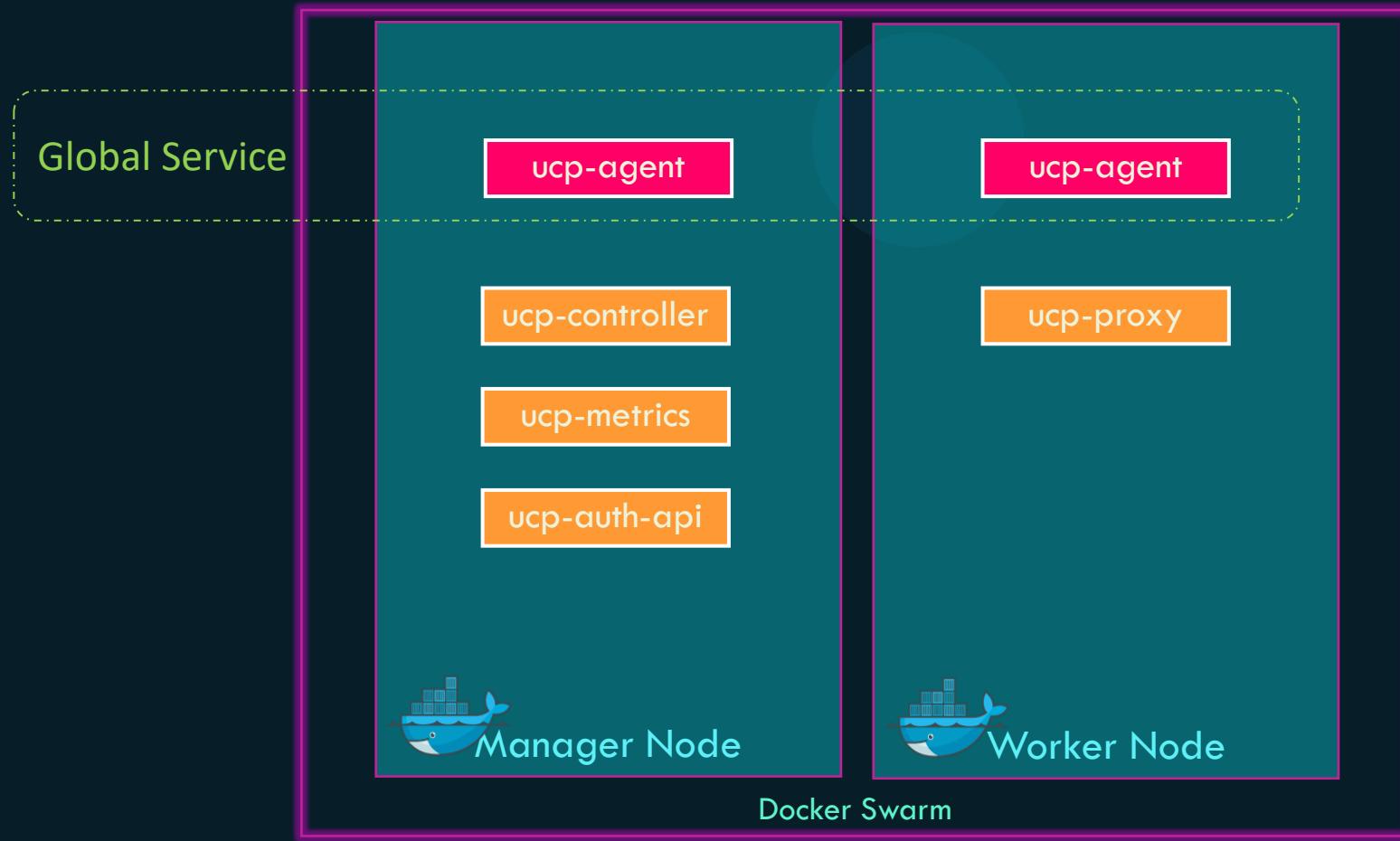
Worker Node Addition



Worker Node Addition



Worker Node Addition





KODEKLOUD

Docker Trusted Registry

Docker Registry

```
▶ docker pull ubuntu
```

```
▶ docker push ubuntu
```

```
▶ docker pull gcr.io/organization/ubuntu
```

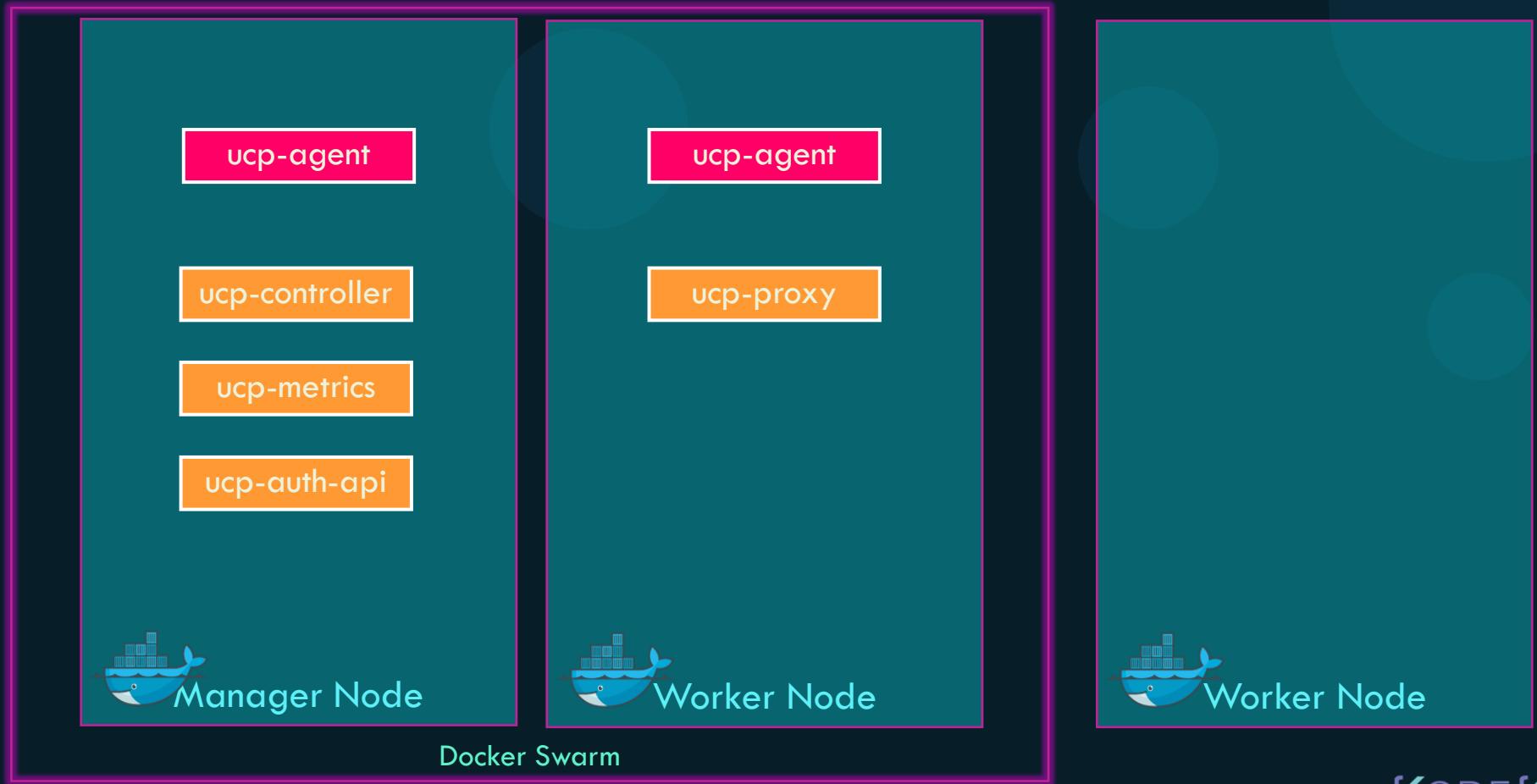


Docker Trusted Registry (DTR)

The screenshot shows the Docker Trusted Registry (DTR) interface. The top navigation bar includes the Docker logo, 'Docker Enterprise Trusted Registry 2.7.6', and a dropdown menu for 'yogesh'. Below the navigation is a sidebar with links: 'Search', 'Repositories' (which is selected and highlighted in blue), 'Organizations', 'Users', and 'System'. At the bottom of the sidebar are links for 'Registry CLI', 'API', 'Docs', and 'Universal Control Plane'. The main content area is titled 'Repositories' and displays a message: 'You have **no repositories**... make one now!'. It features a large blue whale icon with a stack of shipping containers on its back. A dashed arrow points upwards from the whale icon towards a blue button labeled 'New repository'.



Worker Node Addition



Worker Node Addition

Swarm
Certificates
Layer 7 Routing
Cluster Configuration
Authentication & Authorization
Logs
Audit Logs
License
Backup
Docker Trusted Registry
Docker Content Trust
Usage
Scheduler
Upgrade

Admin Settings

This UCP cluster does not yet have a Docker Trusted Registry installed.

DTR EXTERNAL URL ?
optional

UCP NODE ?
dtrnode x ▾

Assign a DTR replica ID ?

Disable TLS verification for UCP ?

Use a PEM-encoded TLS CA certificate for UCP ?

Run the following command against UCP using a client bundle to install DTR:
`docker run -it --rm docker/dtr install --ucp-node dtrnode --ucp-username yogeshraheja --ucp-url https://34.227.66.146`

[Learn How To Install DTR](#)

Docker Swarm

Worker Node

dtr-ol

dtr-*

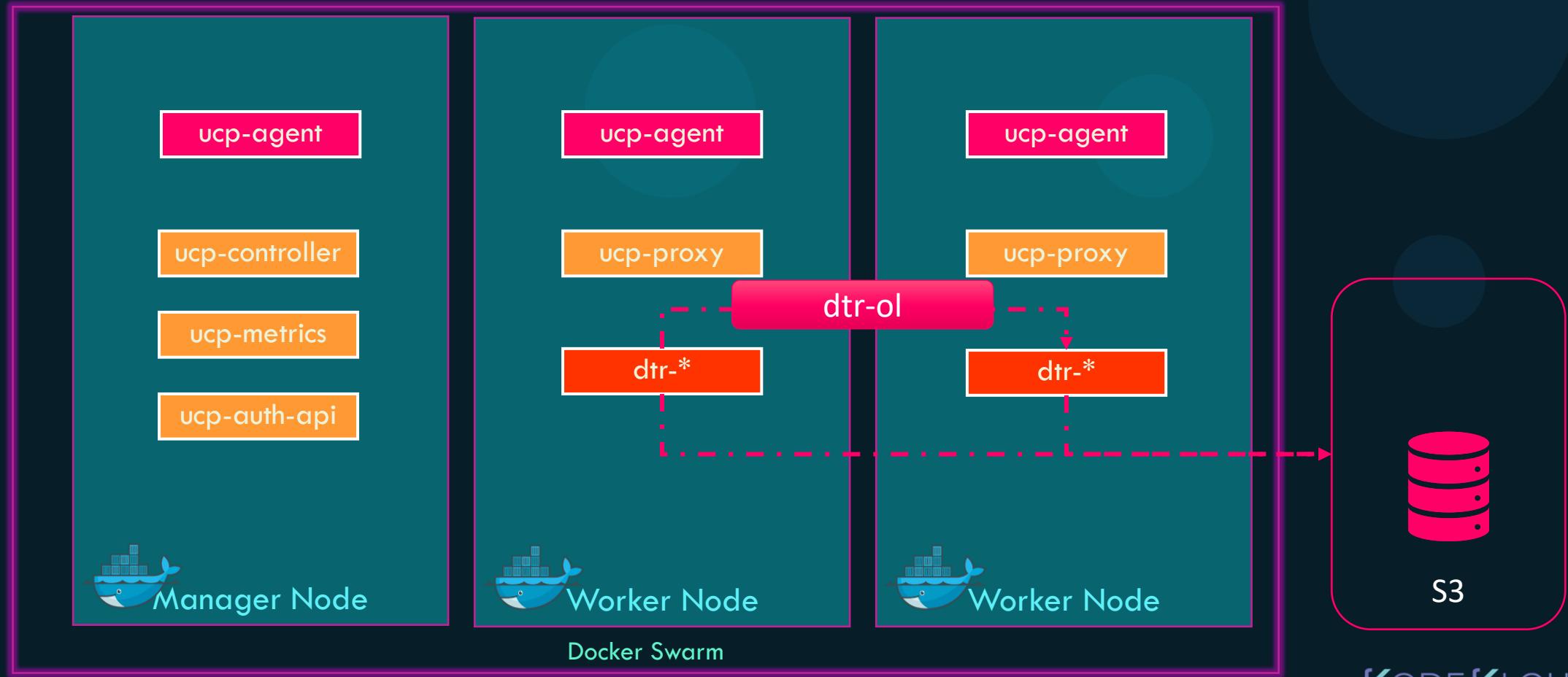
dtr-*

ucp-proxy

ucp-agent



Worker Node Addition



DTR Console

Docker Enterprise
Trusted Registry
2.7.6

yogesh

Search

Repositories

Organizations

Users

System

Registry CLI

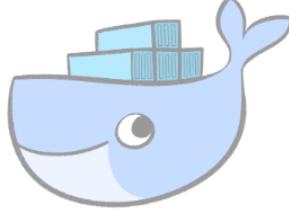
API

Docs

Universal Control Plane

Repositories

New repository



You have **no repositories...** make one now!





KODEKLOUD

Deploying Workload on Docker EE



Deploy and Test Workload on UCP Cluster



WEB GUI



DOCKER CLI



Deploy and Test Workload on UCP Cluster



WEB GUI

The screenshot shows the Docker Enterprise Universal Control Plane (UCP) web interface. The top navigation bar displays the logo and text "Docker Enterprise Universal Control Plane v3.2.6". On the left, a sidebar menu includes "yogeshraheja" (Details, Collection), "Dashboard" (Scheduling), "Access Control" (Network), "Shared Resources" (Environment), "Kubernetes" (Resources), and "Swarm" (selected, showing Services, Volumes, Networks, Secrets, Configurations). A "Create Service" button is located in the top right. The main content area is titled "Create Service" and contains sections for "Mounts" (Add Volume +, Add Bind Mount +, Add Tmpfs Mount +), "Reservations" (Nano CPU Shares, Memory (MB)), and "Limits" (Nano CPU Shares, Memory (MB)).



Deploy and Test Workload on UCP Cluster

The screenshot shows the Docker Enterprise Universal Control Plane (UCP) web interface. The top navigation bar displays "Docker Enterprise Universal Control Plane v3.2.6". The left sidebar menu includes "yogeshraheja", "Dashboard", "Access Control", "Shared Resources", and a expanded "Kubernetes" section with sub-options: "+ Create", "Namespaces" (with "default" selected), "Service Accounts", "Controllers", "Services", "Ingress", "Pods" (which is currently selected), "Configurations", "Storage", and "Swarm". The main content area is titled "Pod(s)" and contains a search bar. There are no visible pods listed.





UCP Client Bundles



Deploy and Test Workload on UCP Cluster



WEB GUI



DOCKER CLI



Deploy and Test Workload on UCP Cluster

The screenshot shows the KodeKloud interface. On the left, there's a sidebar with the following items: Client Bundles (highlighted in blue), Default Collection, All Roles, My Grants, and Security. The main area has a header "Profile". A dropdown menu is open under "Client Bundles" with two options: "Generate Client Bundle" (with a mouse cursor icon) and "Add Existing Client Bundle".

DOCKER_HOST=x.x.x.x

DOCKER_CERT_PATH=/tmp/client.crt

► docker ps





KODEKLOUD

Role Based Access Control

RBAC

Who can do what operations on which resources?



Create User

Details

Newly added users will automatically be added to the "docker-datacenter" organization; this organization does not have any privileges. These users will have restricted control to their own private collections.

Username**Password** **Full Name** Is a Docker Enterprise admin

RBAC Roles

Who can do what operations on which resources?

Create Role

Details Operations

All Node operations ▾

SECRET OPERATIONS

All Secret operations ▾

SERVICE OPERATIONS

All Service operations ^

Service Create

Service Create parameters ▾

Service Delete

Service Logs

Service Update

None

View Only

Restricted Control

Scheduler

Full Control



RBAC – Resource Sets

Create Collection: /Shared

Details

Label Constraints



Collections and Namespaces

Docker Enterprise enables controlling access to swarm resources by using collections and Kuber namespaces. Access to collections and namespaces goes through a directory structure that arran permissions, administrators create grants against directory branches

Details

Collection Name

Who can do what operations on which resources?



KODEKLOUD

Create Grant

A grant defines who (subject) has a specific access (role) to a resource set (Swarm collection).

1. Subject

Select Subject Type

USERS ORGANIZATIONS

Organization

Select...



Team(Optional)

Select...



Next

Notes

- Access Control High Level Steps:
 - Configure Subjects – Users, teams, organizations, service accounts
 - Configure custom roles – permissions per type of resource
 - Configure resource sets – Swarm Collections or Kubernetes Namespaces
 - Create Grants – Subjects + Roles + Resource Sets
- Best practice is to configure a team with the right privileges and add/remove users to it during organizational changes
- Create Users:
 - Create local users from UCP Console
 - Integrate UCP with LDAP/AD





KODEKLOUD

Docker Trusted Registry



Image Addressing Convention

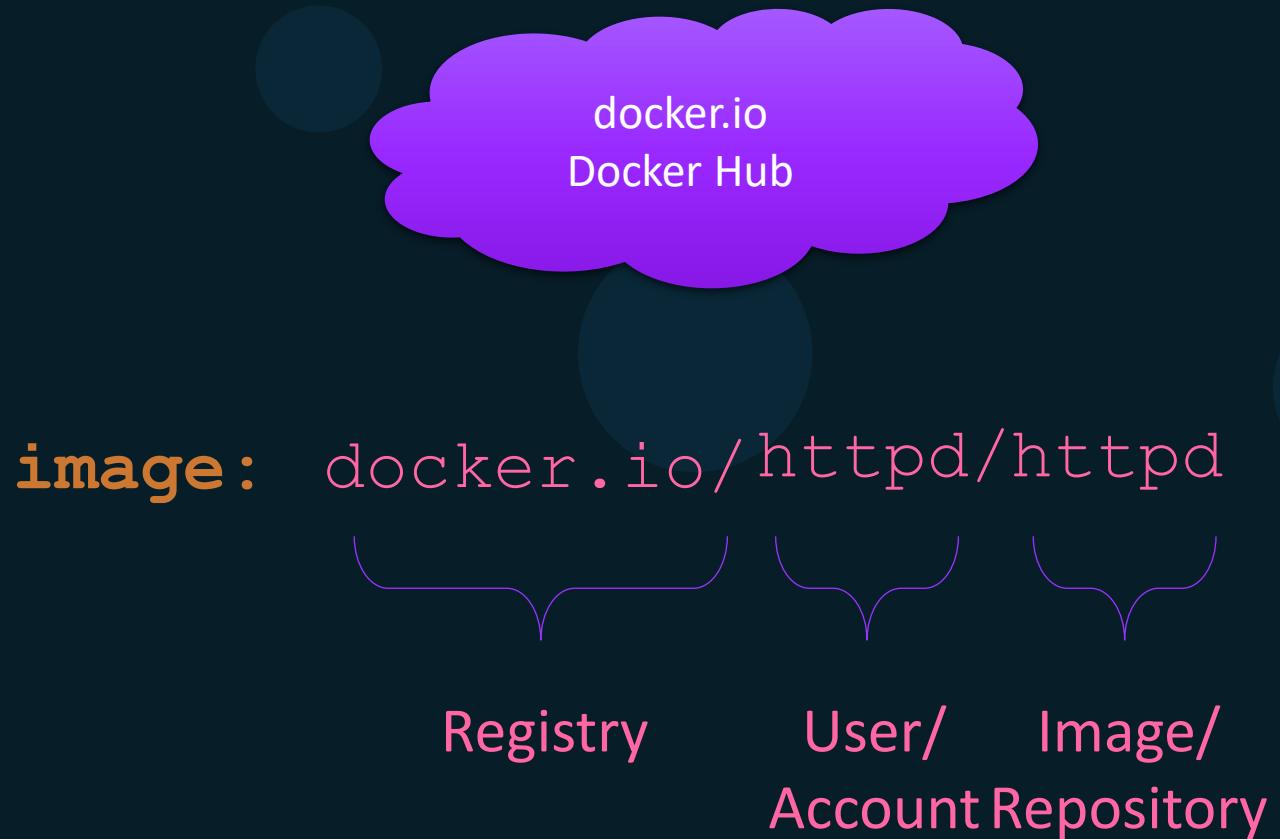


Image Addressing Convention

registry.company.org
Docker Trusted Registry

image: docker.io/httpd/httpd

Registry User/ Image/
 Account Repository



Image Addressing Convention

54.145.234.153
registry.company.org
Docker Trusted Registry

image: registry.company.org/httpd/httpd

Registry

User/ Image/
Account Repository

```
▶ docker build . -t 54.145.234.153/company/webapp
```

```
▶ docker tag httpd/httpd -t 54.145.234.153/httpd/httpd
```



Create new Repository



Create new Repository

Repositories

New Repository

Repository

yogeshraheja  / kodekloud

Visibility

 **Public**
Visible to everyone

Private
Hide this repository

Description (optional)

 ▾ Show advanced settings

LOUD

Push Image

```
▶ docker build . -t 54.145.234.153/yogeshraheja/kodekloud
```

```
▶ docker push 54.145.234.153/yogeshraheja/kodekloud
```

Repositories

New Repository

Repository

yogeshraheja



/ kodekloud

Description (optional)

▼ Show advanced settings



View Repositories

The screenshot shows the Docker Enterprise Trusted Registry 2.7.6 interface. The top navigation bar displays the product name and version. On the left, a sidebar menu includes options for 'yogeshraheja' (with a dropdown arrow), 'Search', 'Repositories' (which is selected and highlighted in blue), 'Organizations', 'Users', and 'System'. The main content area is titled 'Repositories' and features a 'Filter by' dropdown set to 'All namespaces'. Below this, a section titled 'Repository' lists 'yogeshraheja / kodekloud'. Navigation arrows are present below the repository list.

Docker Enterprise
Trusted Registry
2.7.6

yogeshraheja

Search

Repositories

Organizations

Users

System

Repositories

Filter by All namespaces

Repository

yogeshraheja / kodekloud

< >

Pull Image

The screenshot shows the Docker Enterprise Trusted Registry interface. The left sidebar has a blue header with the Docker logo and "Docker Enterprise Trusted Registry 2.7.6". Below it are links for "yogeshraheja", "Search", "Repositories" (which is selected and highlighted in blue), "Organizations", "Users", and "System". The main content area shows the "Info" tab for the repository "yogeshraheja / kodekloud". The repository name is displayed prominently. Below it is a "README" section with the message "README is empty for this repository". To the right of the README is an "Edit" button with a blue border. Further right is a "Docker Pull Command" section containing the command "docker pull 54.145.234.153/yogeshraheja/kodekloud". At the bottom right of the main content area is a "Your Permission" section showing "Admin".

Docker Enterprise
Trusted Registry
2.7.6

yogeshraheja

Search

Repositories

Organizations

Users

System

Repositories / yogeshraheja / kodekloud / Info

yogeshraheja / kodekloud

Info Tags Webhooks Promotions Pruning Mirrors Settings Activity

README

README is empty for this repository

Edit

Docker Pull Command

```
docker pull 54.145.234.153/yogeshraheja/kodekloud
```

Your Permission

Admin

```
▶ docker pull 54.145.234.153/yogeshraheja/kodekloud
```



Create new repository on Push



Docker Trusted Registry



DTR Security

Repositories

New Repository

Repository

yogeshraheja x ▾ / kodekloud

Visibility

Public Visible to everyone

Private Hide this repository

Description (optional)

▼ Show advanced settings

[Cancel](#) Create



DTR Users

Create User

Details

Newly added users will automatically be added to the "docker-datacenter" organization; this organization does not have any privileges. These users will have restricted control to their own private collections.

Username

Password

Full Name

Is a Docker Enterprise admin



DTR Users

The screenshot shows the Docker Trusted Registry (DTR) user interface. On the left, there is a sidebar with navigation links: 'Repositories' (blue), 'Organizations' (grey), 'Users' (blue, currently selected), and 'Settings' (grey). The main header bar has the DTR logo, a search bar with placeholder 'Search', and a user dropdown for 'admin'. The page title is 'Users'. A search bar at the top of the user list allows searching by username. A green button labeled 'New user' is located on the right side of the search bar. The user list table has columns: 'USERNAME', 'FULL NAME', and 'ORGANIZATIONS'. The data is as follows:

USERNAME	FULL NAME	ORGANIZATIONS
admin	No name	orca, docker, docker-datacenter, shark, whale
anna.jenkins	Anna Jenkins	shark
dave.lauper	Dave Lauper	shark
jamie.andrews	Jamie Andrews	orca
lynda.johnson	Lynda Johnson	
paul.newton	Paul Newton	orca

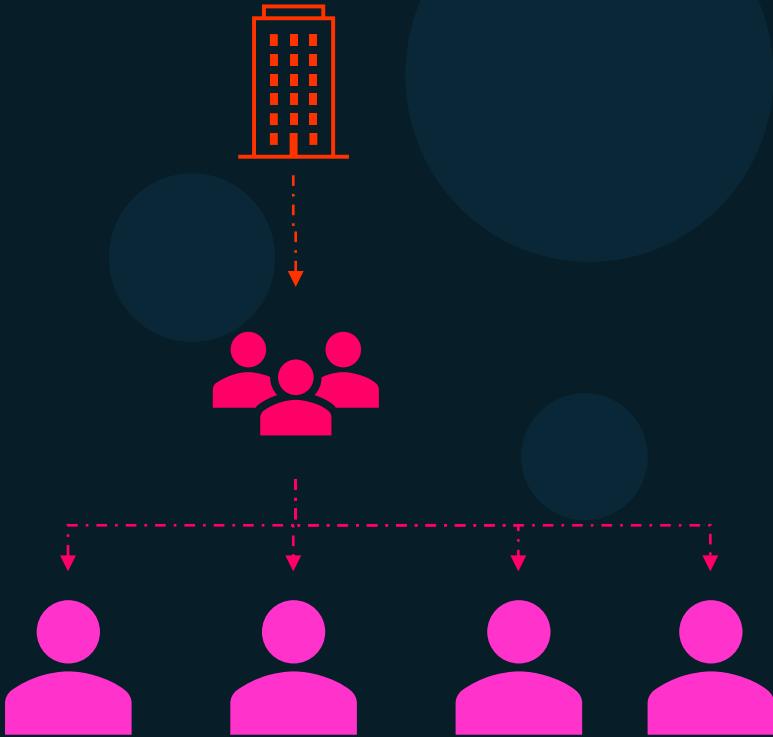
At the bottom, there are 'Previous' and 'Next' buttons, and a 'Items per page' dropdown set to 10, with options 10, 25, 50, and 100.

DTR Organizations & Teams

The screenshot shows the Docker Trusted Registry (DTR) interface. On the left, there's a sidebar with icons for Repositories, Organizations, Users, and Settings. The main area is titled "Organizations > whale". It displays a list of organization members. The "MEMBERS" tab is selected, showing one member: "admin" (username) with "No name" (full name). Below this, there's a "TEAMS" section with a note: "Create a team to give users more repository permissions." A hand cursor icon is hovering over a green plus sign button next to the "TEAMS" heading.

USERNAME	FULL NAME
admin	No name

Previous Next



DTR Team Permissions

The screenshot shows a user interface for managing a repository. At the top, there's a search bar and a user dropdown labeled 'admin'. Below the header, there are tabs for 'REPOSITORIES' and 'SETTINGS'. A green button labeled 'Add repository' is visible. In the main area, a 'repository' card for 'java' is shown. The card has a 'New' button, a path input field containing '/ java', and a 'PERMISSIONS' section. The 'PERMISSIONS' section includes a dropdown menu with options: 'Read-only' (selected), 'Read-write' (highlighted with a cursor icon), and 'Admin'. Below this, there's a 'PERMISSIONS' button with 'Read-only' and a close icon, and a 'View Details' link. The bottom of the card features a blue play button icon.

Repository operation	read	read-write	admin
View/ browse	x	x	x
Pull	x	x	x
Push		x	x
Start a scan		x	x
Delete tags		x	x
Edit description			x
Set public or private			x
Manage user access			x
Delete repository			x



DTR Image Scanning



Image Scanning

Docker Enterprise
Trusted Registry
2.7.6

yogeshraheja

System / Security

General Storage Security Garbage collection Job Logs

Image Scanning

Check for vulnerabilities in your repositories' images.
[Learn more](#)

Enable Scanning

Image Scanning Method

Security scanning requires installing a security database in DTR.

Select a method for installation and updates.

Online
Automatically syncs

Offline
Manually upload a file

Last sync: May 08, 2020 @ 2:49 AM
CVE Database version: 1055

Sync Database now

Image Scanning

yogeshraheja / kodekloud

Info Tags Webhooks Promotions Pruning Mirrors Settings Activity

	Image	Type	ID	Size	Signed	Last Pushed	Vulnerabilities	
<input type="checkbox"/>	v2	linux amd64	ed220d72fc7c	25.97 MB	Not signed	22 seconds ago by yogeshraheja	Start a scan	View details
<input type="checkbox"/>	v1	linux amd64	39eda93d1586	2.81 MB	Not signed	4 minutes ago by yogeshraheja	Critical 0 major 0 minor 0	View details



Image Scanning

yogeshraheja / kodekloud

Info

Tags

Webhooks

Promotions

Pruning

Mirrors

Settings

Activity

General

Visibility

 **Public**

Visible to everyone

 **Private**

Hide this repository

Immutability

 **On**

Tags are immutable

 **Off**

Tags can be overwritten

Description

Save

🛡 Image Scanning

Check for vulnerabilities in your images.

[Learn more](#) 

Scan on push

 **On push**

Images are scanned on push but also can be scanned manually

 **Manual**

Image scans must be manually initiated

Scan Report

yogeshraheja / kodekloud: v2

linux / amd64 ed220d72fc7c 24.77 MB

Pushed 42 seconds ago by [yogeshraheja](#)

3 critical 25 major 4 minor All layers already scanned

Delete

Promote

Scan

[Layers](#) Components

- 1 ADD file:6edc55fb54ec9fc3658c8f5176a70e792103a5161544
- 2 CMD ["/bin/sh"]
- 3 ENV NODE_VERSION=8.9.4
- 4 addgroup -g 1000 node && adduser -u 1000 -G node -s /bin/sh
- 5 ENV YARN_VERSION=1.3.2
- 6 apk add --no-cache --virtual .build-deps-yarn curl gnupg tar &&
- 7 CMD ["node"]

- ADD file:6edc55fb54ec9fc3658c8f5176a70e792103a516154442f94fed8e0290e4960e in / 1.9 MB

Components (9)

Vulnerabilities

musl@1.1.16-r14

1 Critical

busybox@1.26.2-r9

3 Major

libressl@2.5.5-r2

1 Minor

apk-tools@2.7.5-r0

pax-utils@1.2.2-r0

tre@1.1.16-r14

Summary

- Detects vulnerabilities in OS packages and libraries within images and version in which it was introduced
- Recommends fixed version
- Data about vulnerabilities are pulled either from a universal database known as the US national vulnerability database or it can also be configured manually by uploading a file.
- Scanning can be manually triggered or automatically when an image is pushed
- The scan report reports Critical, Major or Minor categories along with the count in each
- To fix vulnerabilities check application level dependencies, upgrade packages and rebuild docker image

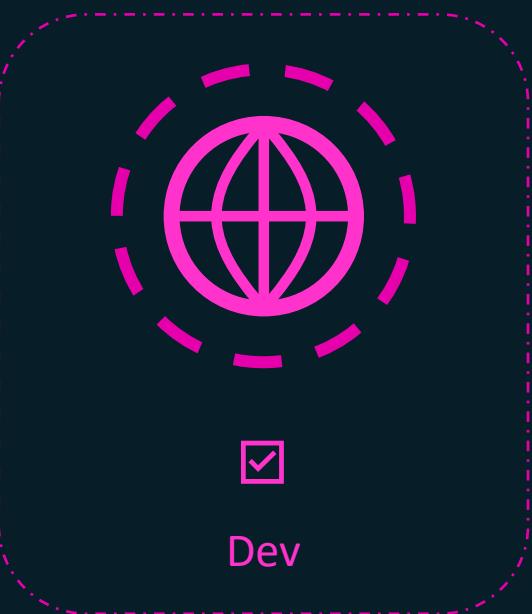




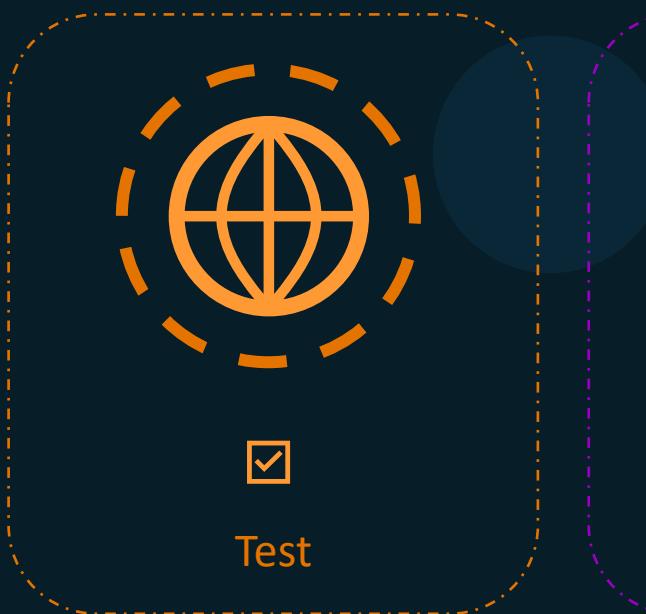
DTR Image Promotion



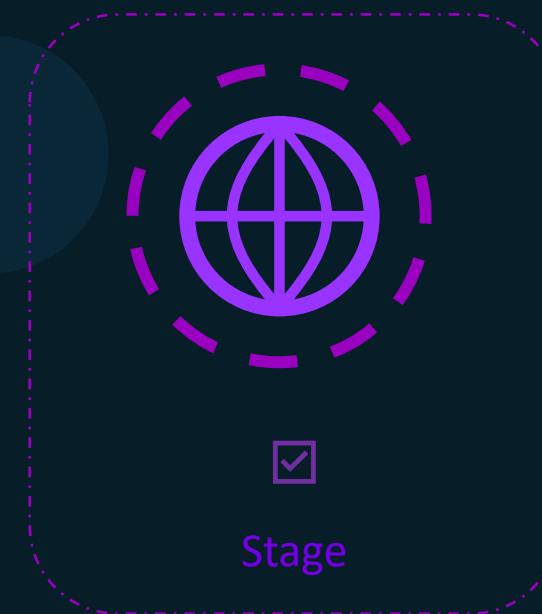
Development Pipeline



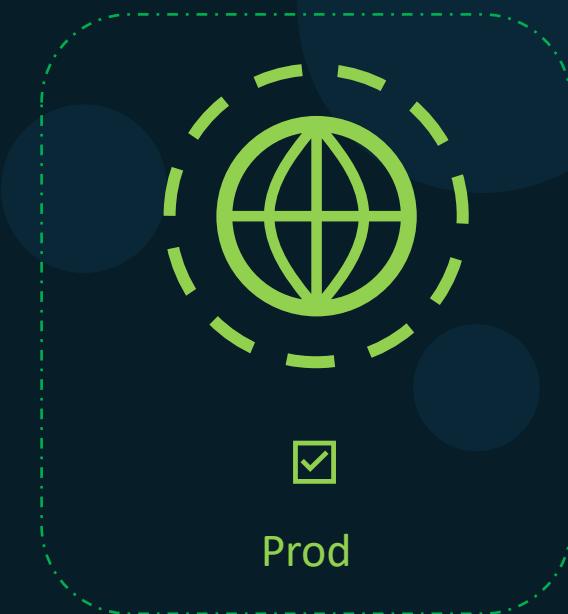
registry.company.org/dev/app



registry.company.org/test/app



registry.company.org/stage/app



registry.company.org/prod/app



Image Promotion

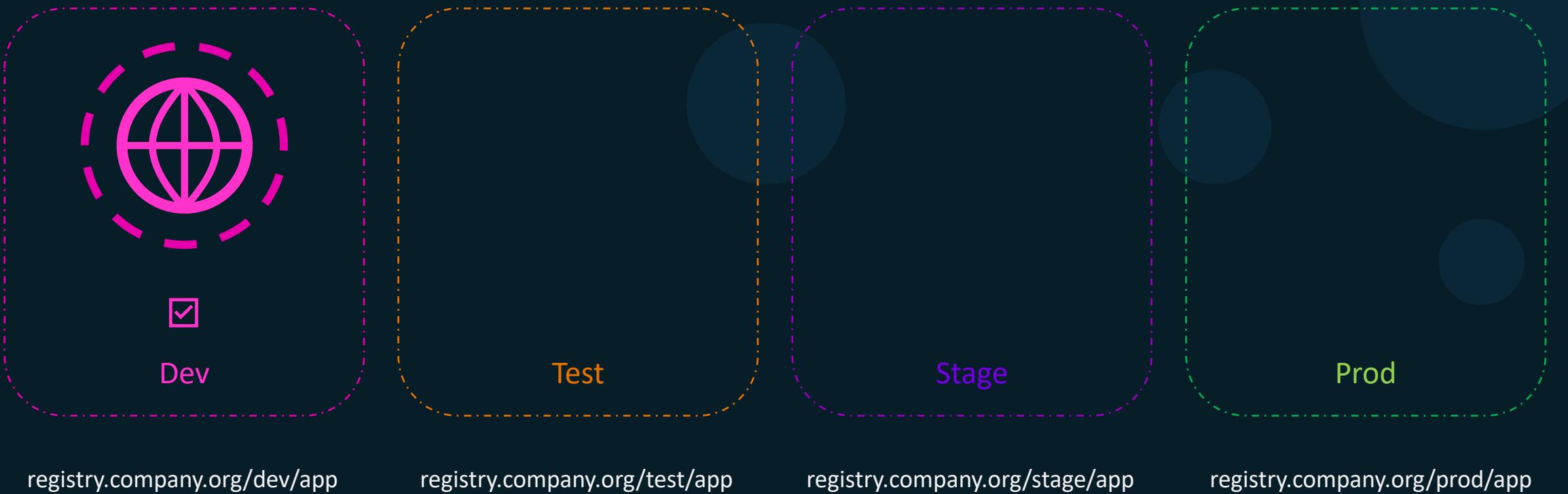


Image Promotion

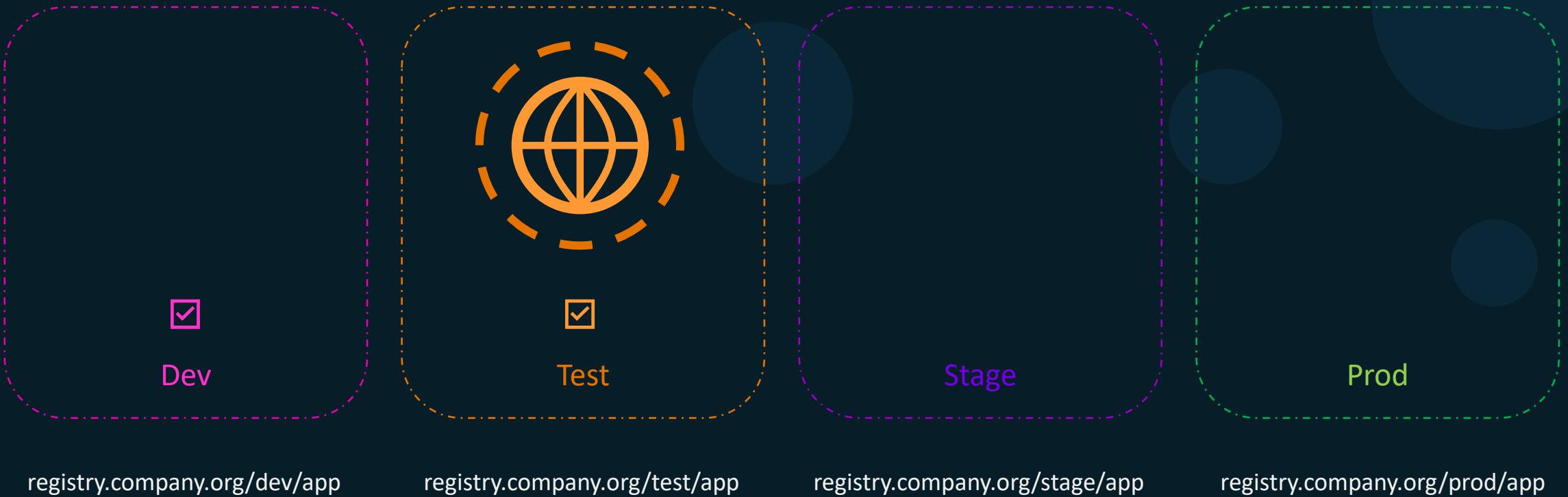


Image Promotion

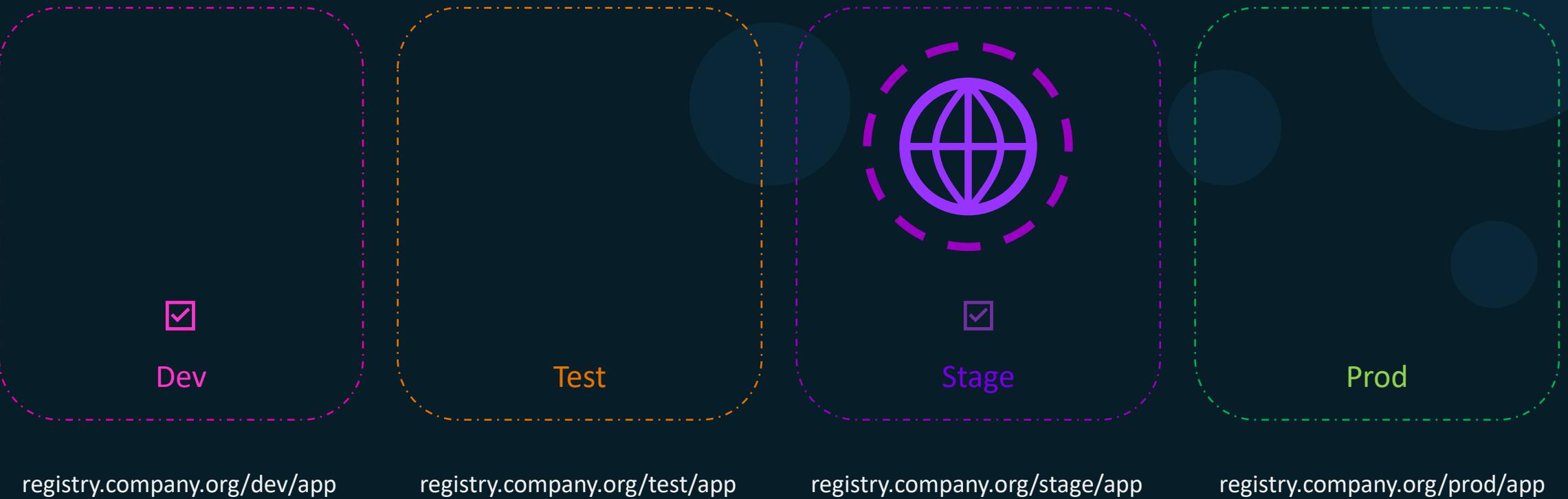


Image Promotion

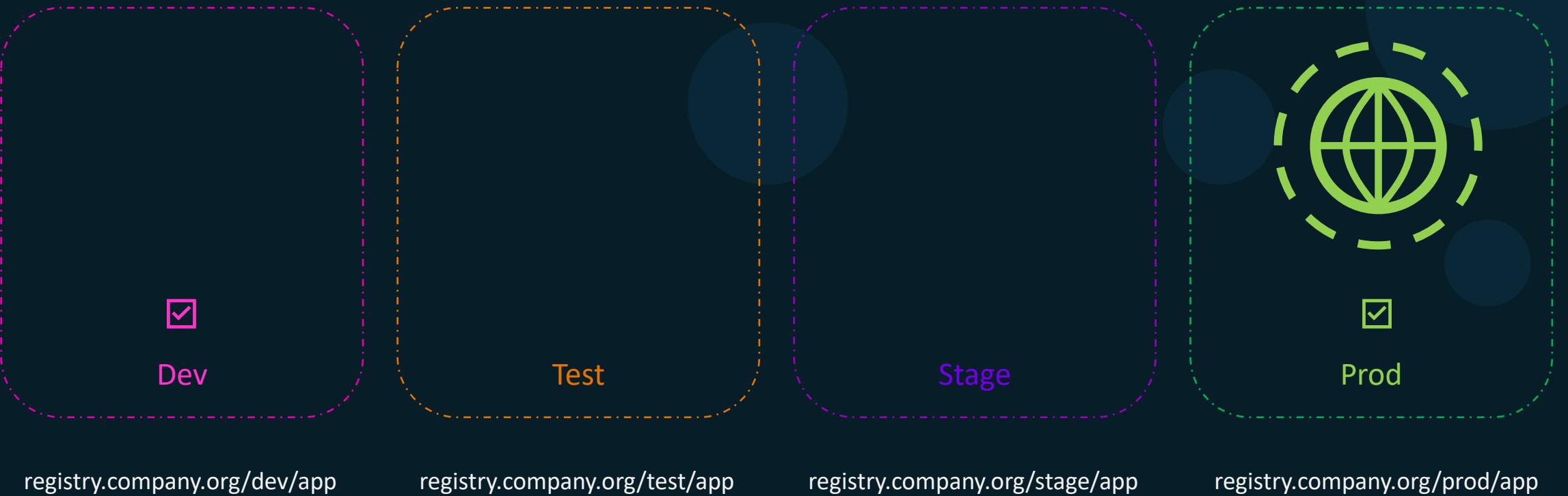


Image Promotion

Repositories / yogeshraheja / devimages / Promotions / New

yogeshraheja / devimages

Info Tags Webhooks **Promotions** Pruning Mirrors Settings Activity

Tag name

equals

starts with

ends with

contains

one of

not one of

I

[Cancel](#) [Add](#)

 TARGET REPOSITORY

namespace / name



KODEKLOUD

DTR Garbage Collection



DTR Operations

 **Docker Enterprise**
Trusted Registry
2.7.6

System / GC

yogeshraheja ▾

General Storage Security **Garbage collection** Job Logs

 Remove Untagged Images
Run garbage collection on your storage backend to remove deleted tags and images. [Learn more](#)

Delete images

 Until done
This may take a while

 For 1 minutes

 Never
Disable garbage collection

Save

Notes

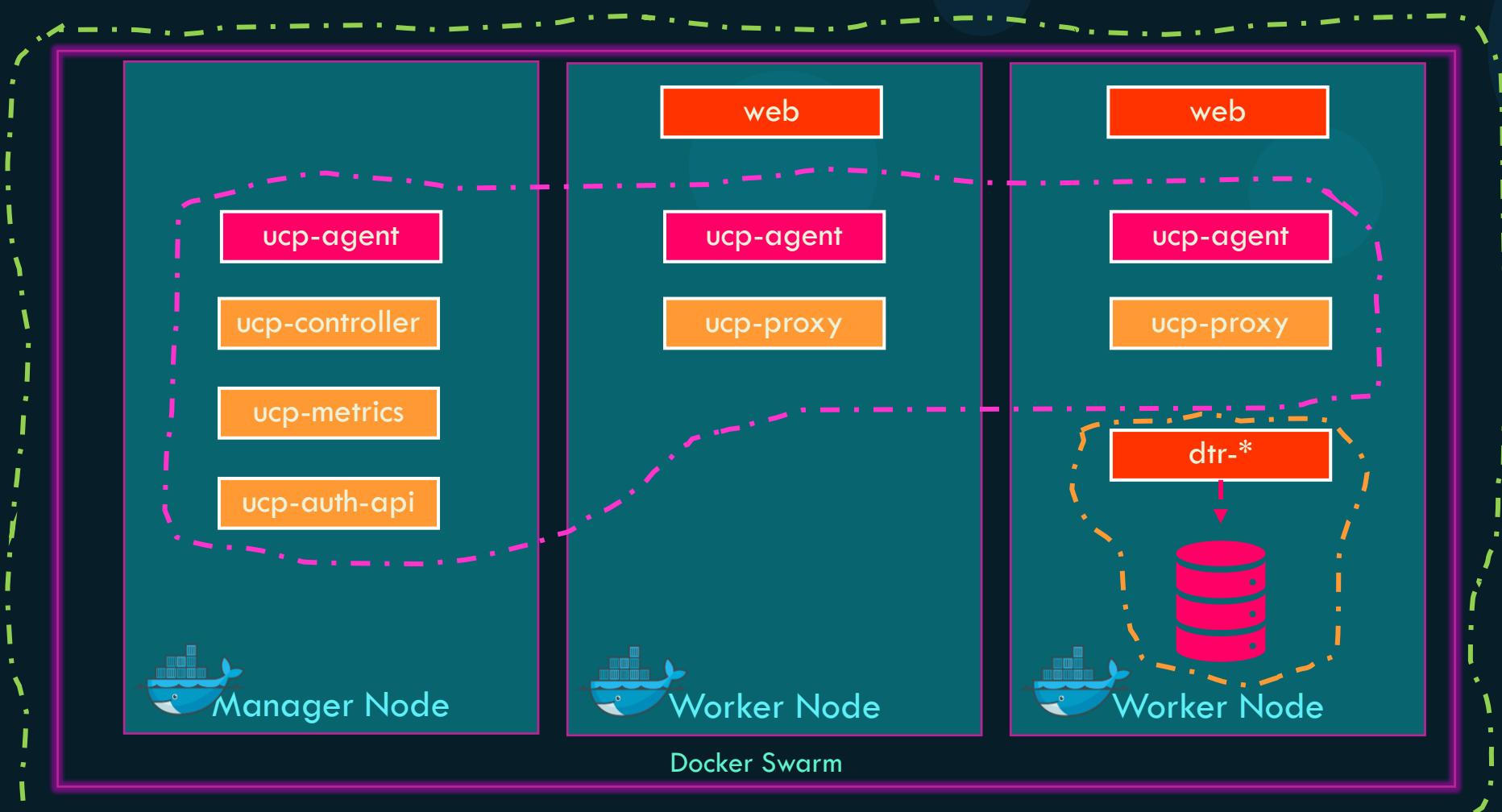
- Deleting image does not delete image layers
- Does not free up space
- For this we must schedule garbage collection
- During Garbage Collection:
 - DTR becomes read-only. Images can be pulled, but pushes are not allowed
 - DTR identifies and marks all unused image layers
 - DTR deletes the marked image layers.
- Garbage collection is a CPU intensive process
- Must be scheduled outside of business peak hours
- May be configured to run
 - Until done
 - For X minutes
 - Never





Disaster Recovery Docker Swarm

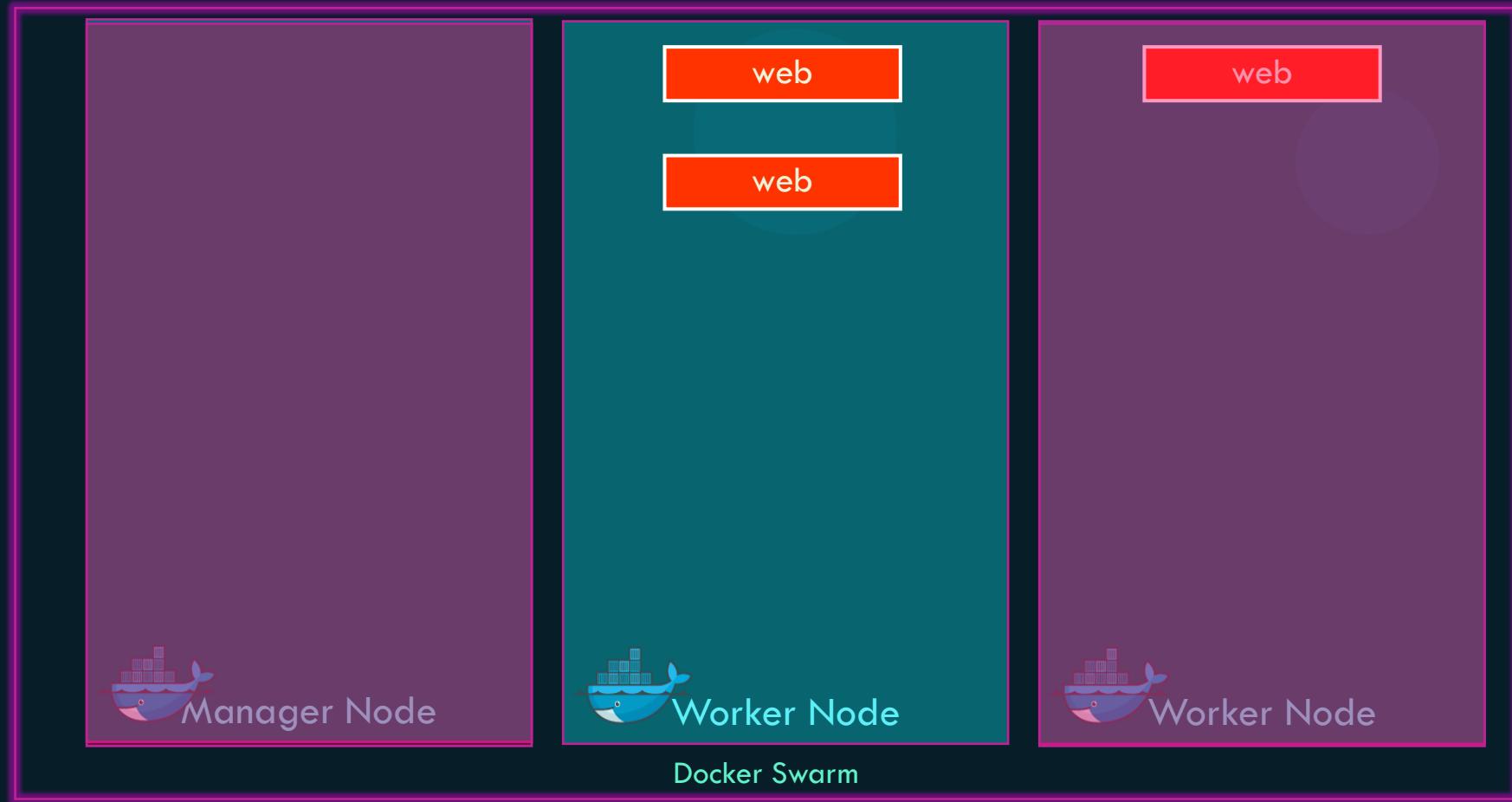
Backup and Restoration



Docker Swarm - Recovery

$$\text{Quorum of } 1 = \frac{1 + 1}{2} = 1.5 = 1$$

▶ docker service update --force web

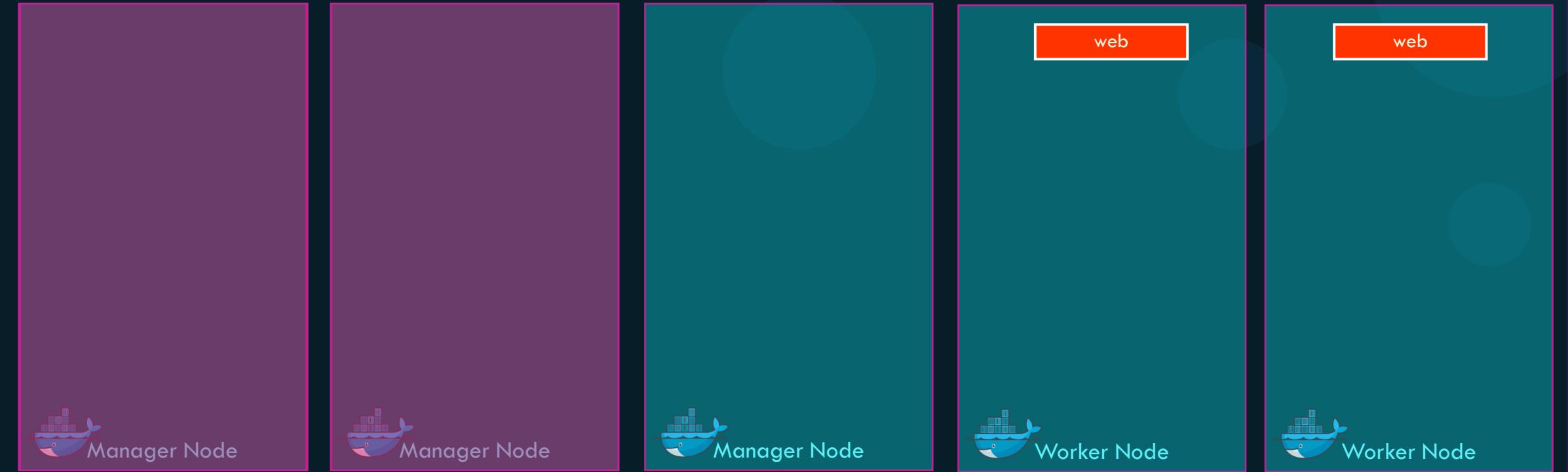


Docker Swarm - Recovery

$$\text{Quorum of 3} = \frac{3 + 1}{2} = 2.5 = 2$$

► docker node promote

► docker swarm init --force-new-cluster

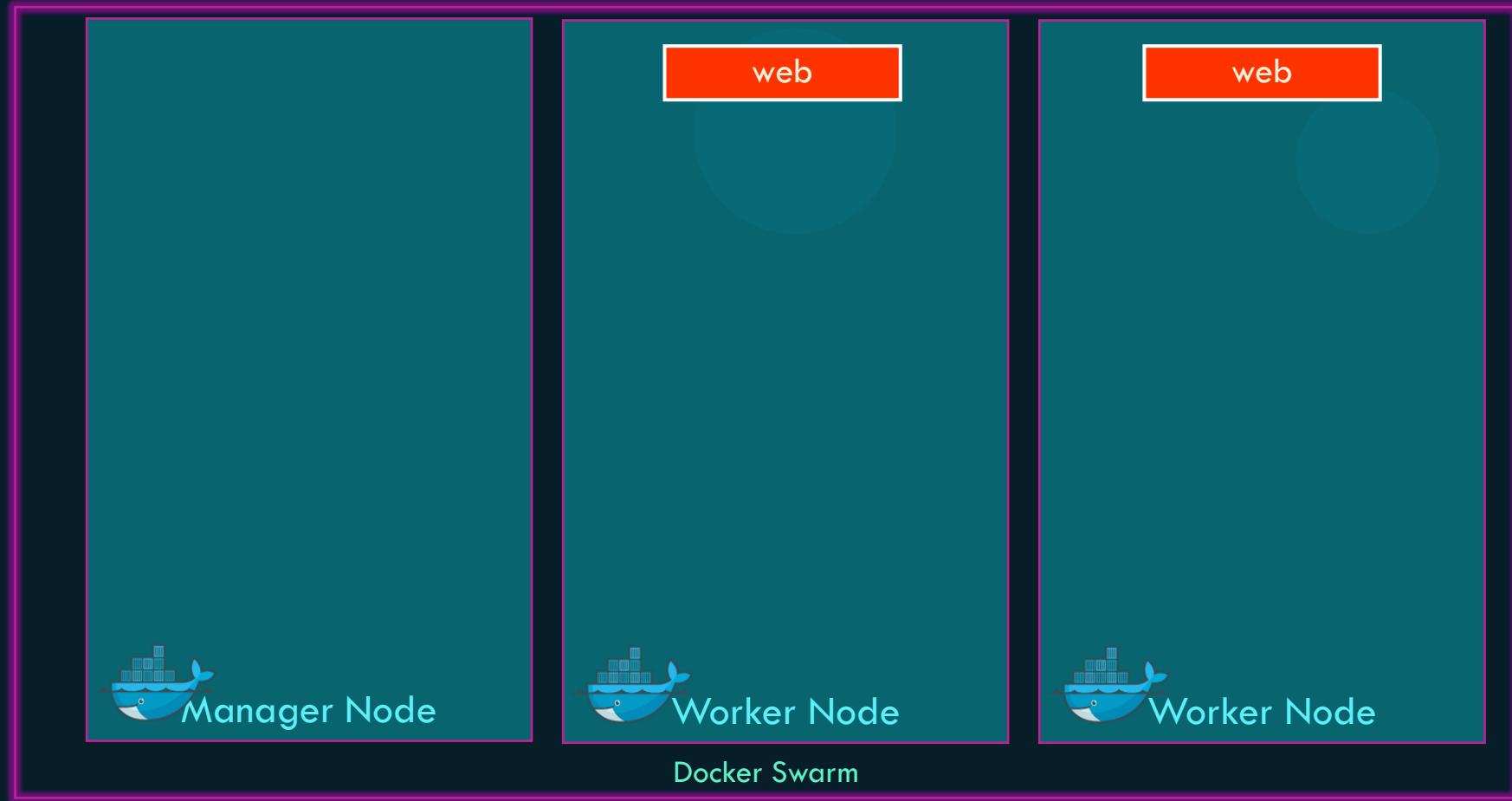


Docker Swarm - Recovery

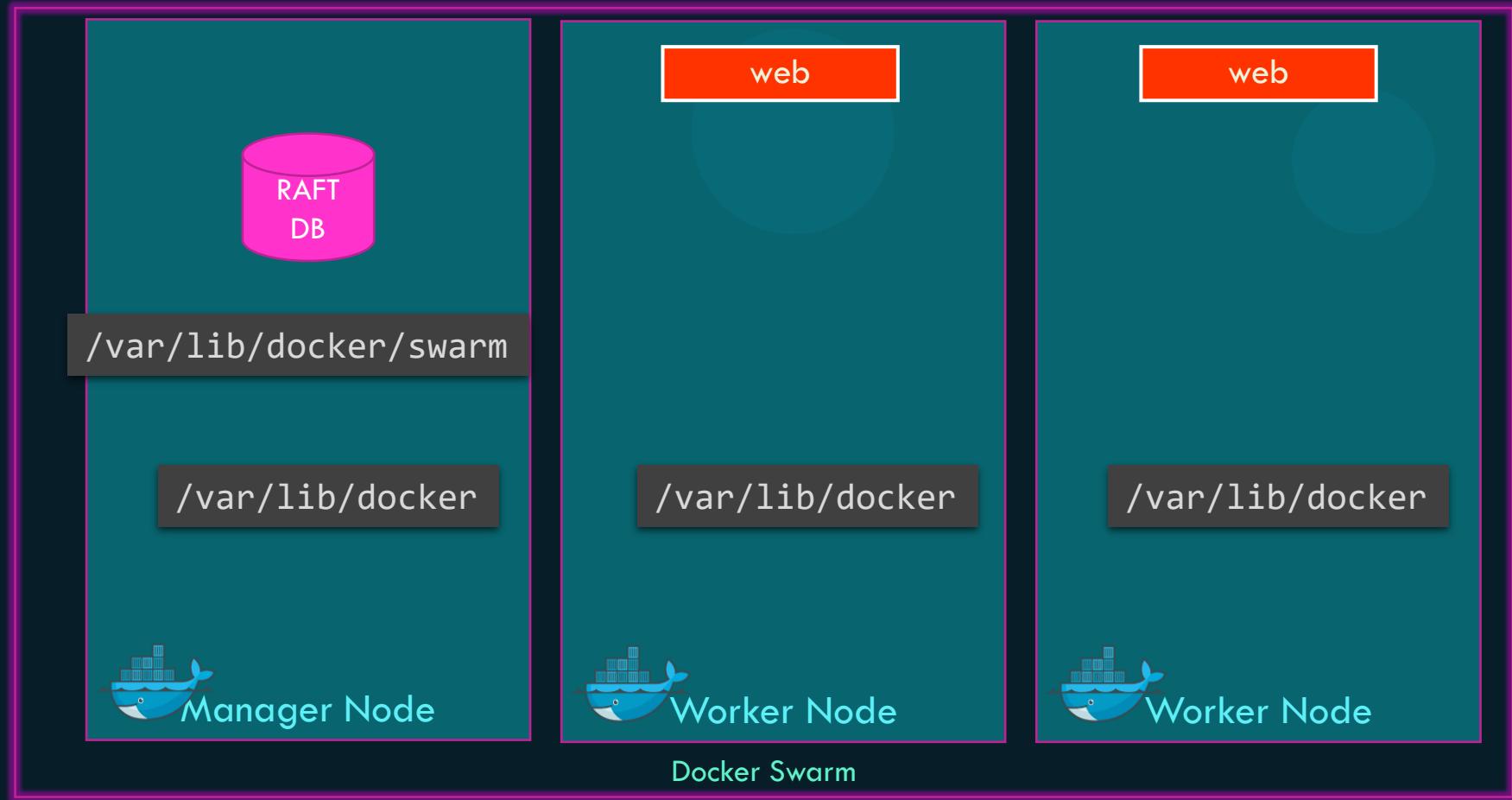
$$\text{Quorum of 3} = \frac{3 + 1}{2} = 2.5 = 2$$

► docker node promote

► docker swarm init --force-new-cluster



Docker Swarm - Backup

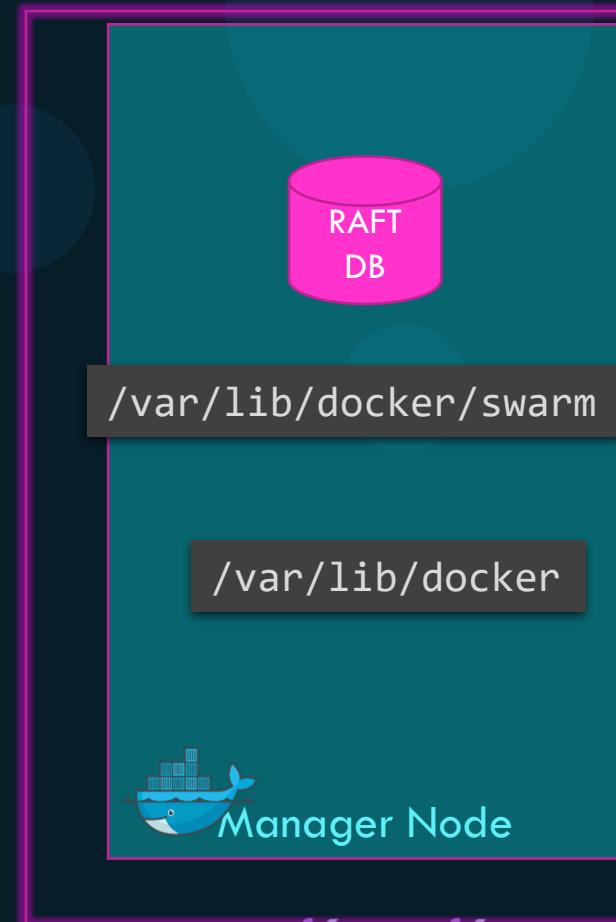


Docker Swarm - Backup

```
▶ systemctl stop docker
```

```
▶ tar cvzf /tmp/swarm-backup.tgz /var/lib/docker/swarm/
```

```
▶ systemctl start docker
```



Docker Swarm - Backup

Raft keys

Cluster Membership

Services

Networks

Configs

Secrets

Swarm unlock keys

```
▶ docker swarm init --autolock=true
```

```
▶ docker swarm update --autolock=true
```

Swarm updated.

To unlock a swarm manager after it restarts, run the `docker swarm unlock` command and provide the following key:

SWMKEY-1-7K9wg5n85QeC4Zh7rZ0vSV0b5MteDsUvpVhG/lQnb10

Please remember to store this key in a password manager, since without it you will not be able to restart the manager.



/var/lib/docker/swarm

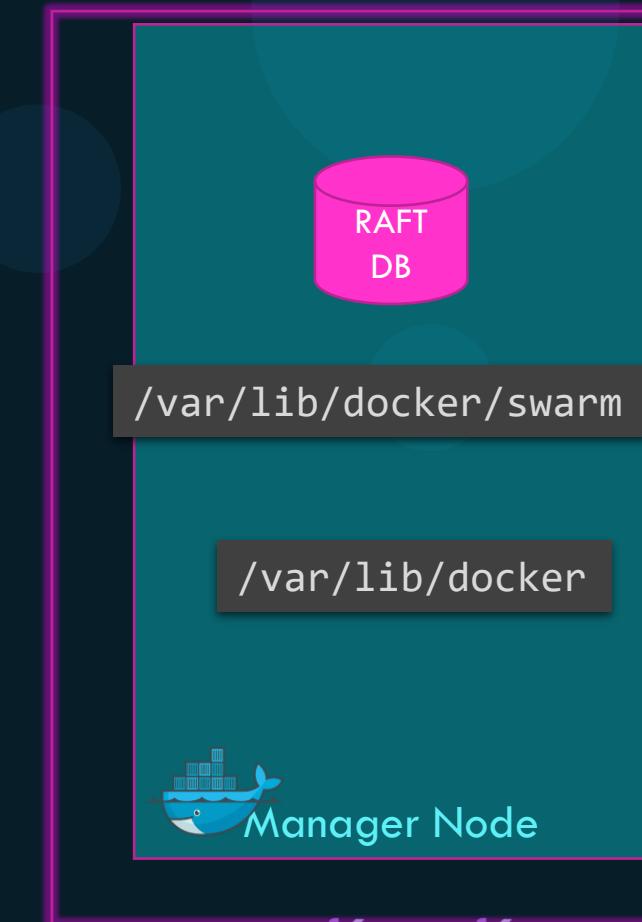
/var/lib/docker



Manager Node

Docker Swarm - Restore

```
▶ systemctl stop docker  
▶ tar xvzf /tmp/swarm-backup.tgz -C /  
▶ systemctl start docker  
▶ docker swarm init --force-new-cluster
```



References

<https://docs.mirantis.com/docker-enterprise/v3.0/dockeree-products/ucp/admin/disaster-recovery/backup-swarm.html>

<https://docs.mirantis.com/docker-enterprise/v3.0/dockeree-products/ucp/admin/disaster-recovery/restore-swarm.html>

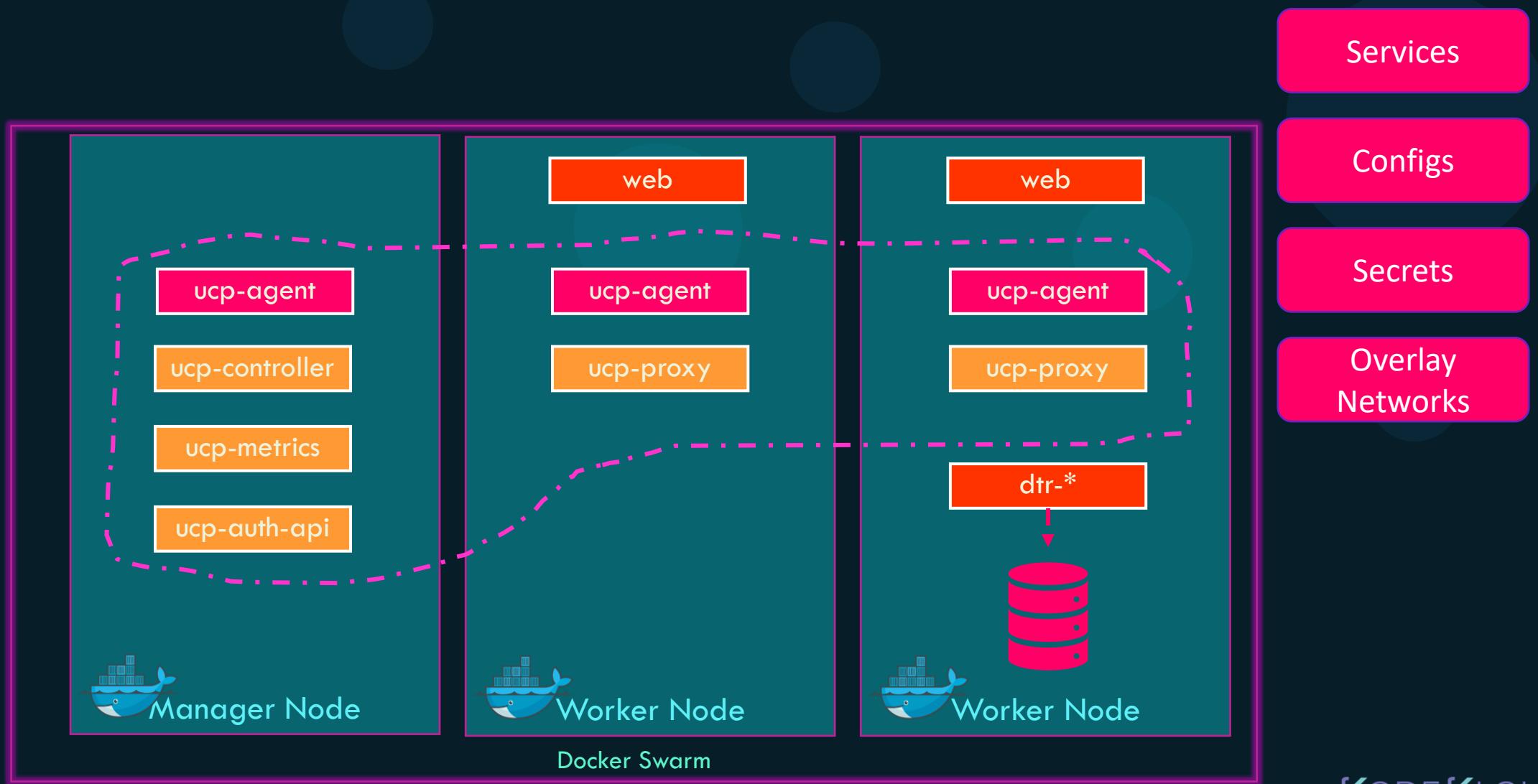




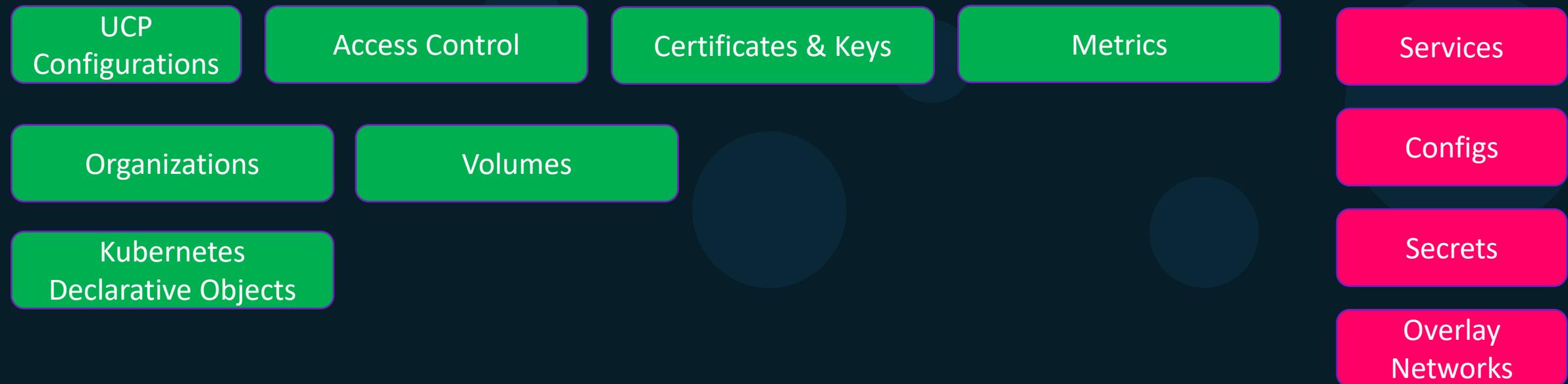
Disaster Recovery UCP



Disaster Recovery - UCP



Backup - UCP



The screenshot shows the Docker Enterprise Universal Control Plane (UCP) dashboard at 52.90.239.129/manage/dashboard. The left sidebar includes links for Dashboard, Access Control, Shared Resources, Kubernetes, and Swarm. The main area displays Manager and Worker nodes, with 1 Manager Node currently active. A summary bar indicates Max CPU 12.51%, Max Memory 13.26%, and Max Used Disk 20.57% over the last 6 hours. On the right, there are sections for SWARM Services (Active 0, Errors 0, Updating 0) and KUBERNETES default (Pods Running 0, Errors 0). A 'DEKLOUD' watermark is visible in the bottom right corner.

UCP - Backup

```
▶ docker container run \
  --rm \
  --log-driver none \
  --name ucp \
  --volume /var/run/docker.sock:/var/run/docker.sock \
  --volume /tmp:/backup \
  docker/ucp:3.2.5 backup \
  --file mybackup.tar \
  --passphrase "secret12chars" \
  --include-logs=false
```

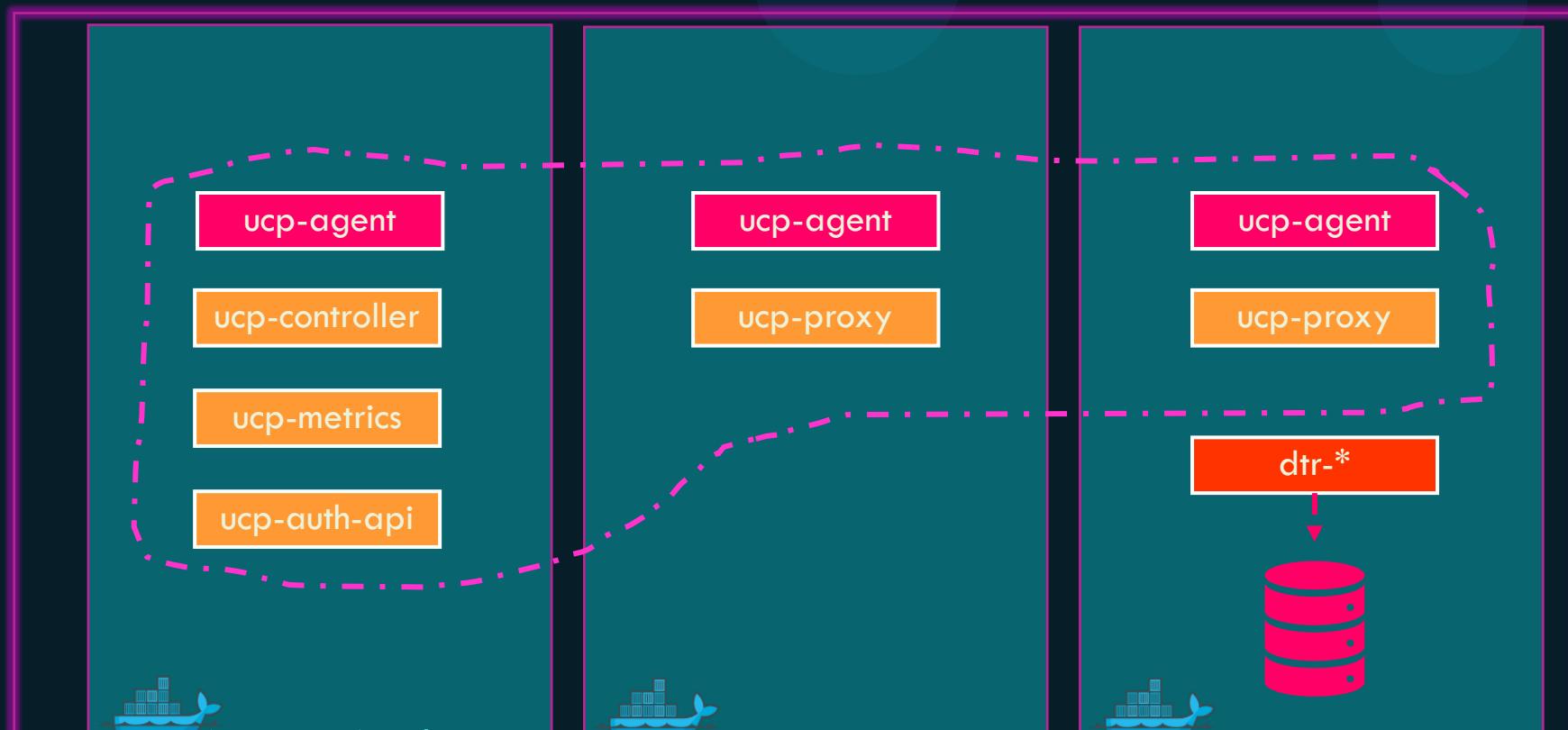
The screenshot shows the 'Content Trust Settings' page. At the top right, there is an 'Admin Setting' button. Below it, a list of options includes: Swarm, Certificates, Layer 7 Routing, Cluster Configuration, Authentication & Authorization, Logs, Audit Logs, License, and Backup. The 'Backup' option is highlighted with a blue oval. On the right side, there is a section titled 'Content Trust Settings' with a checkbox labeled 'Run only signed images' followed by a help icon.

Content Trust Settings
<input type="checkbox"/> Run only signed images <small>?</small>



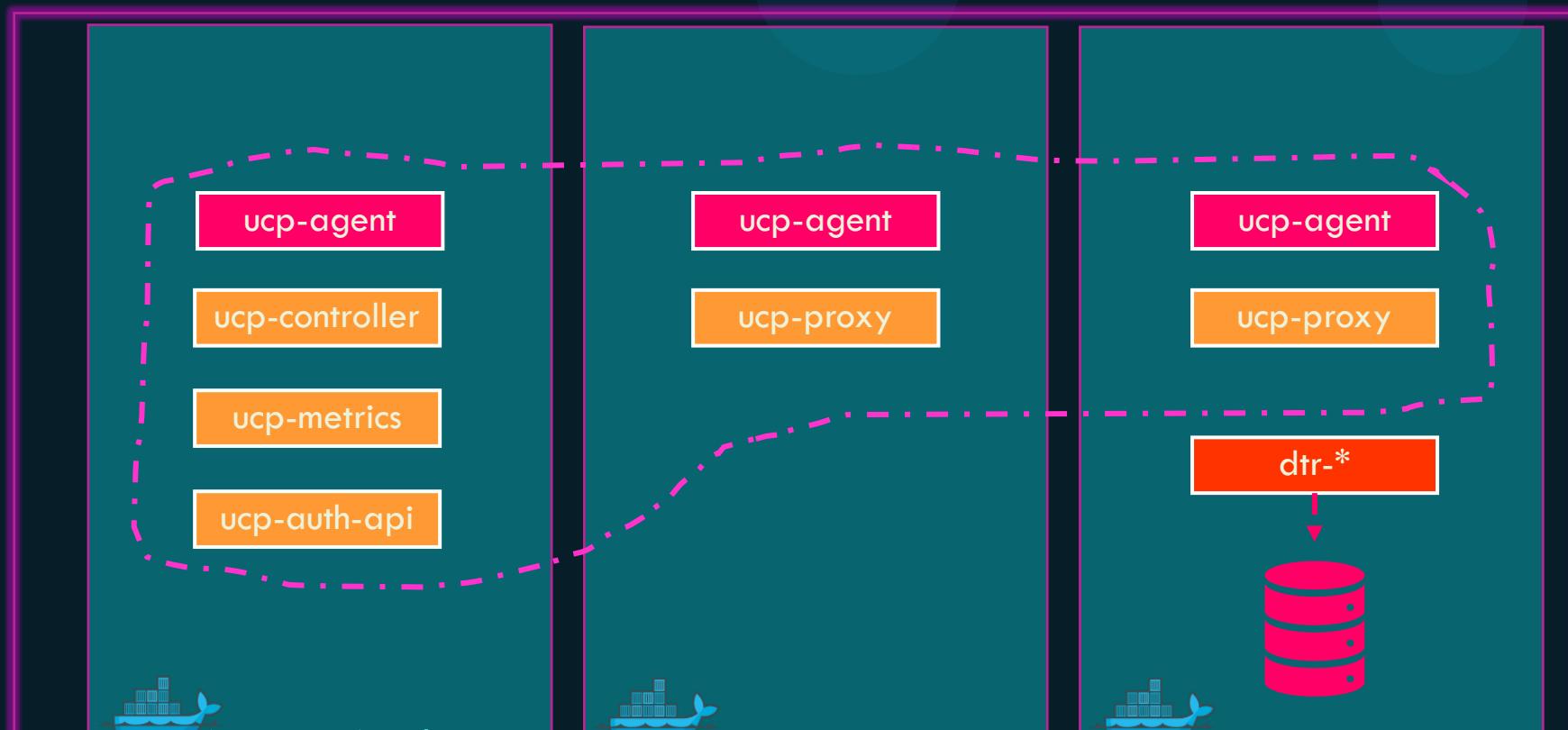
UCP - Restore

```
▶ docker container run \  
  --rm -it \  
  --name ucp \  
  -v /var/run/docker.sock:/var/run/docker.sock \  
  docker/ucp \  
  uninstall-ucp
```



UCP - Restore

```
▶ docker container run \
--rm \
--interactive \
--name ucp \
--volume /var/run/docker.sock:/var/run/docker.sock \
docker/ucp:3.2.5 restore < /tmp/mybackup.tar
```



Notes

- One backup at a time
- UCP does not backup swarm workloads. Swarm workloads are backed up with Swarm backup
- Cannot take a backup of a cluster that's already crashed.
- Restore to the same version of Docker Enterprise as that of the one that was used during backup
- Restore either to the same swarm cluster or to a Docker host and swarm will be initialized automatically



References

<https://docs.mirantis.com/docker-enterprise/v3.0/dockeree-products/ucp/admin/disaster-recovery/disaster-recovery-ucp.html>

<https://docs.mirantis.com/docker-enterprise/v3.0/dockeree-products/ucp/admin/disaster-recovery/backup-ucp.html>

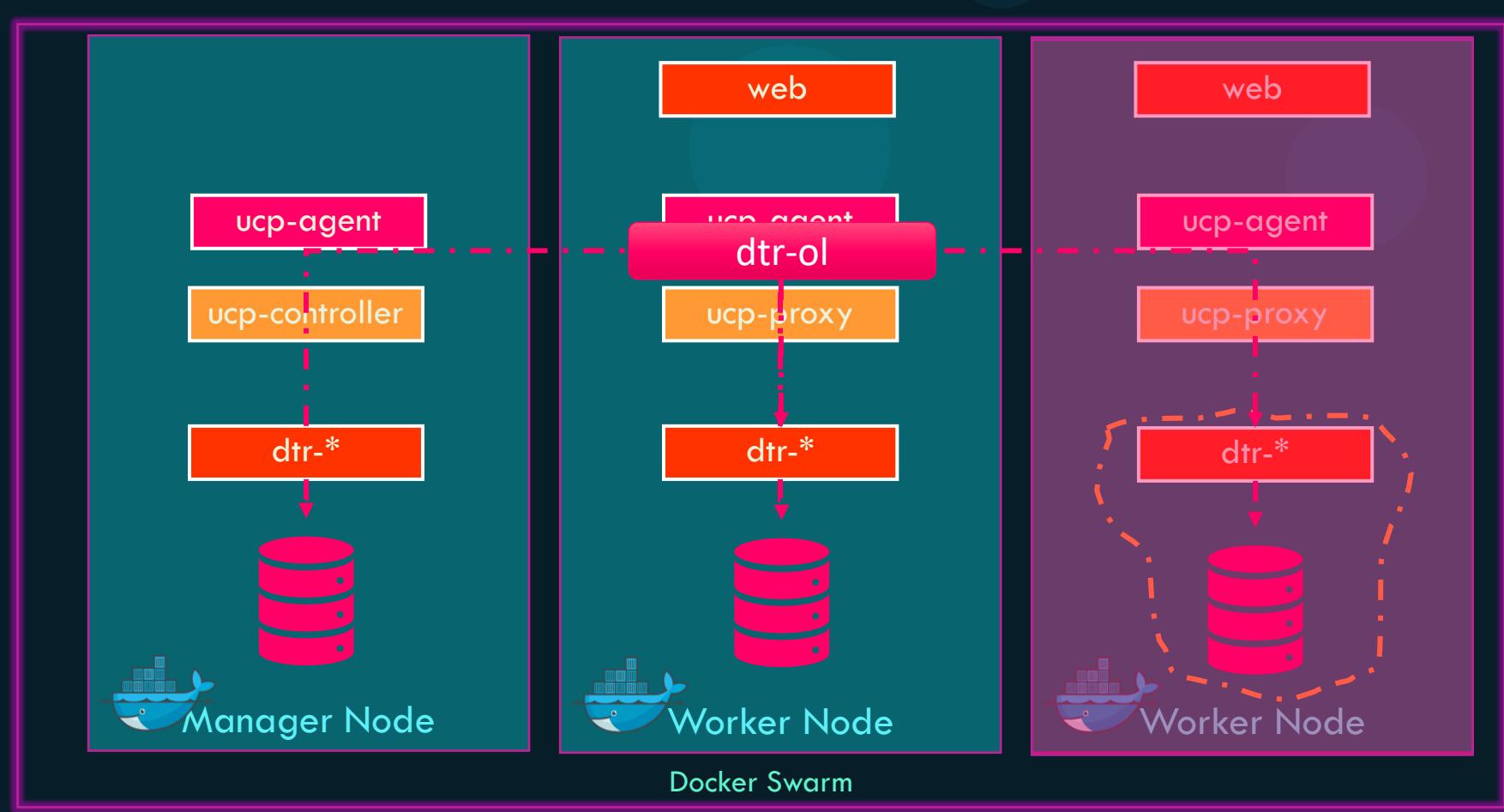
<https://docs.mirantis.com/docker-enterprise/v3.0/dockeree-products/ucp/admin/disaster-recovery/restore-ucp.html>





Disaster Recovery Docker Trusted Registry

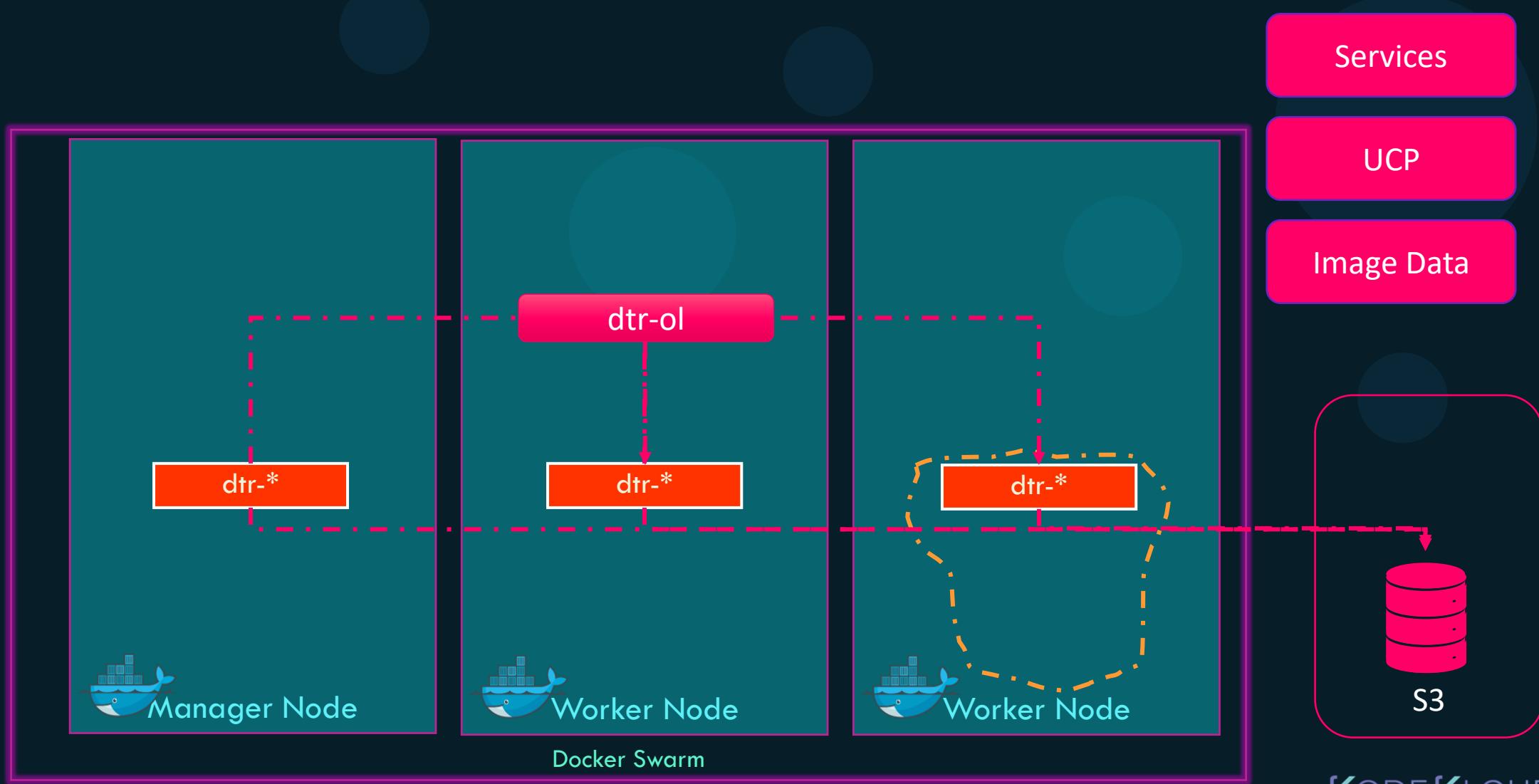
DTR - Backup and Restoration



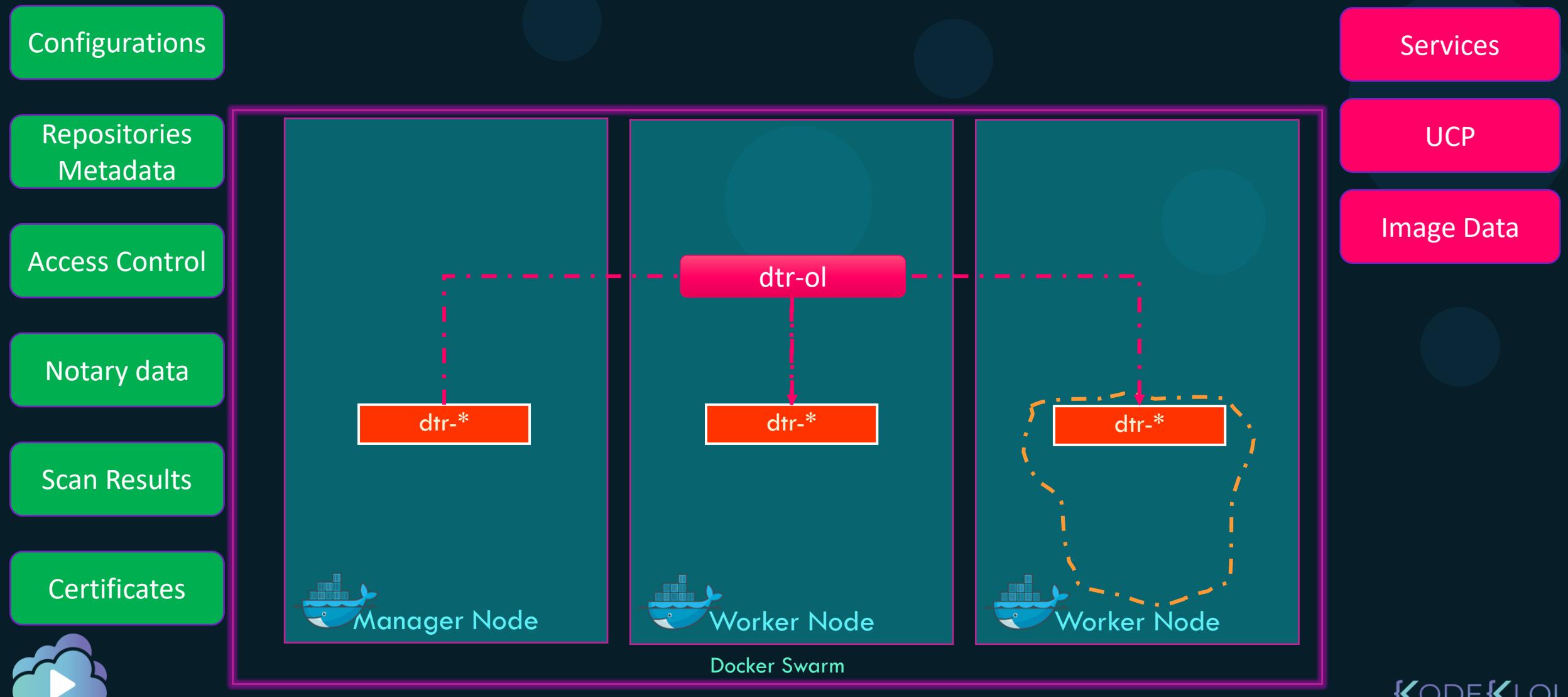
Services

UCP

DTR - Backup

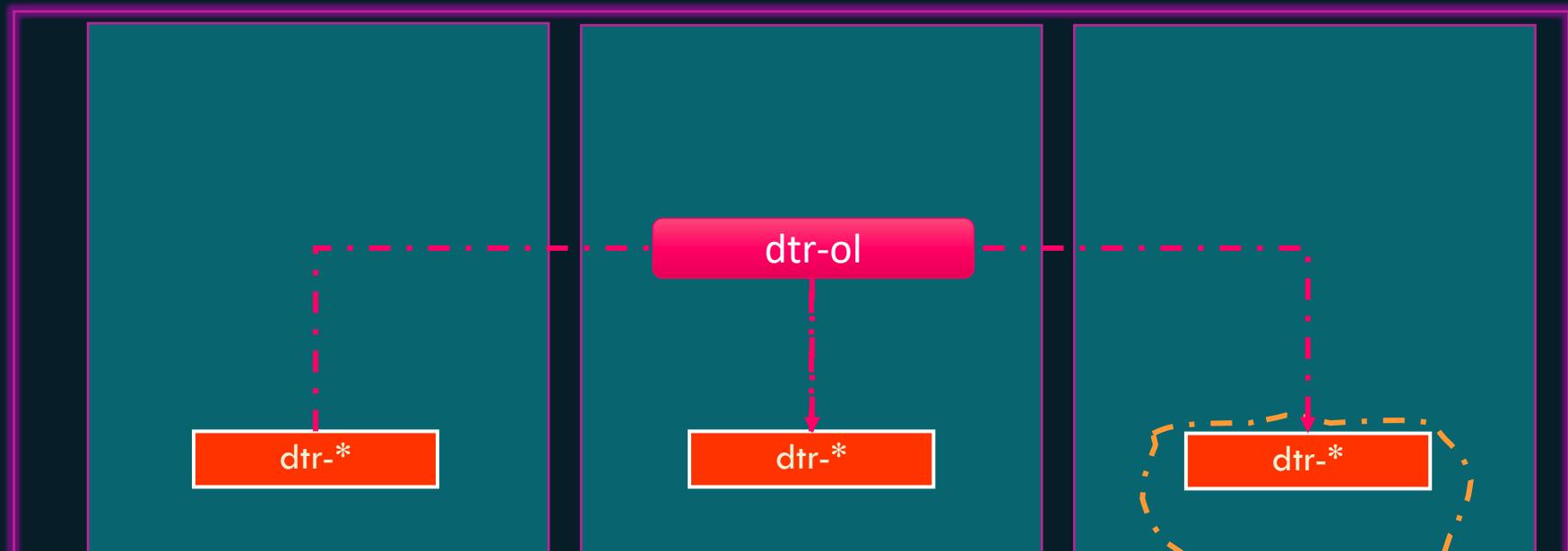


DTR - Backup



DTR - Backup

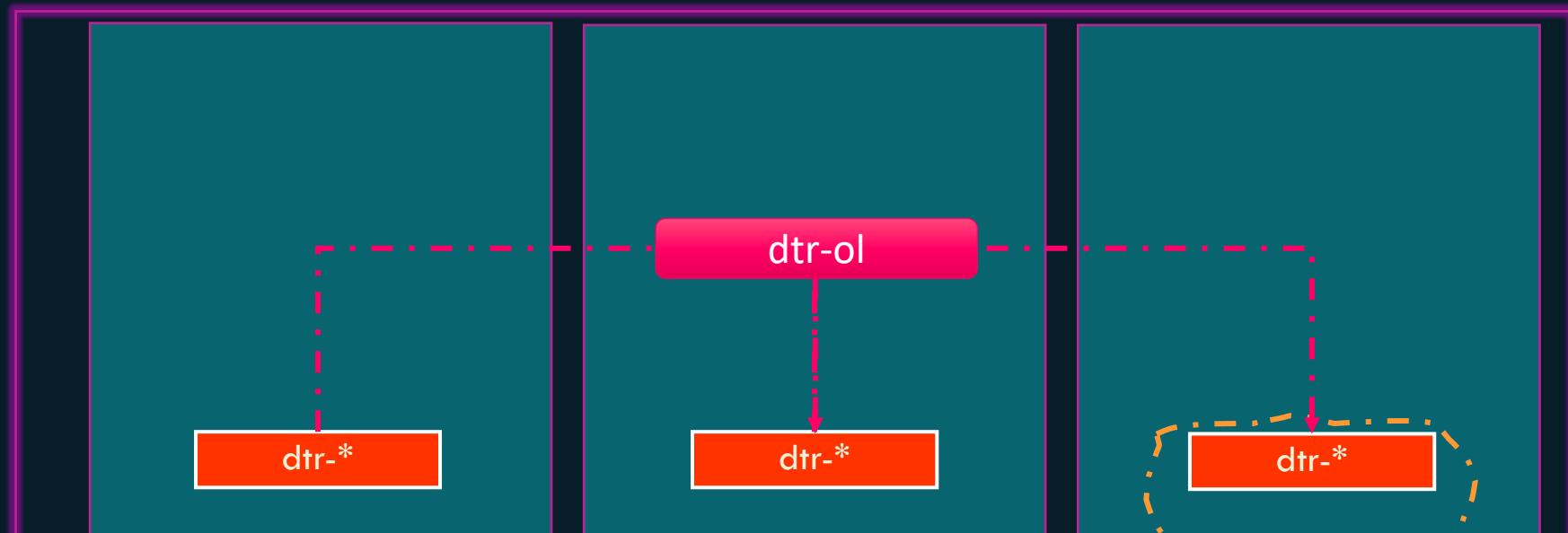
```
▶ docker run \
  docker/dtr backup \
  --existing-replica-id $REPLICA_ID > dtr-metadata-backup.tar
```



DTR - Backup

```
▶ docker run --rm \
--env UCP_PASSWORD=$UCP_PASSWORD \
docker/dtr backup \
--ucp-username $UCP_ADMIN \
--ucp-url $UCP_URL \
--ucp-ca "$(curl https://$UCP_URL/ca)" \
--existing-replica-id $REPLICA_ID > dtr-metadata-backup.tar
```

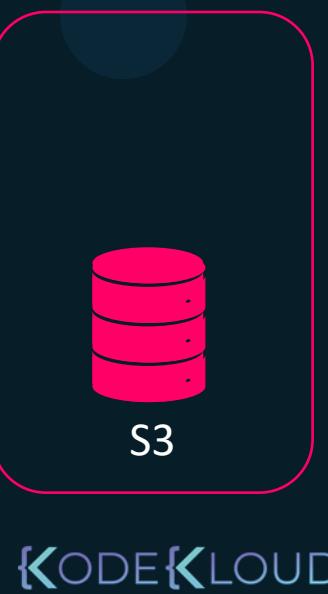
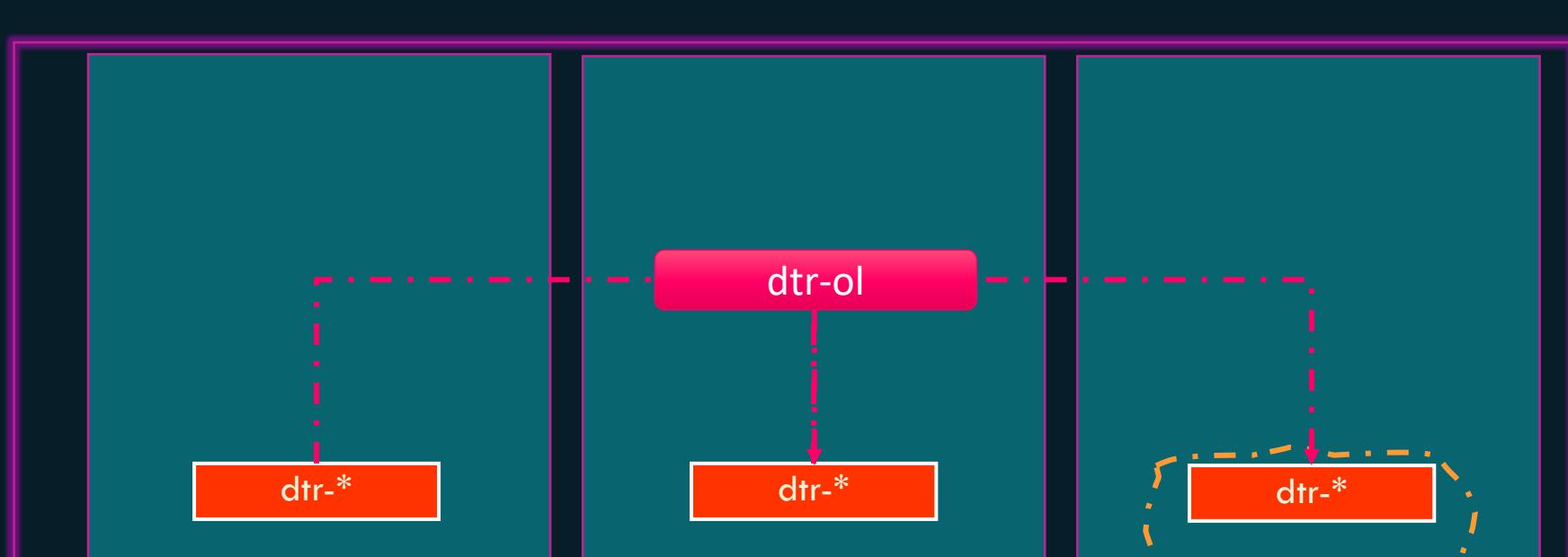
<https://docs.mirantis.com/docker-enterprise/v3.0/dockeree-products/dtr/dtr-admin/disaster-recovery/create-a-backup.html>



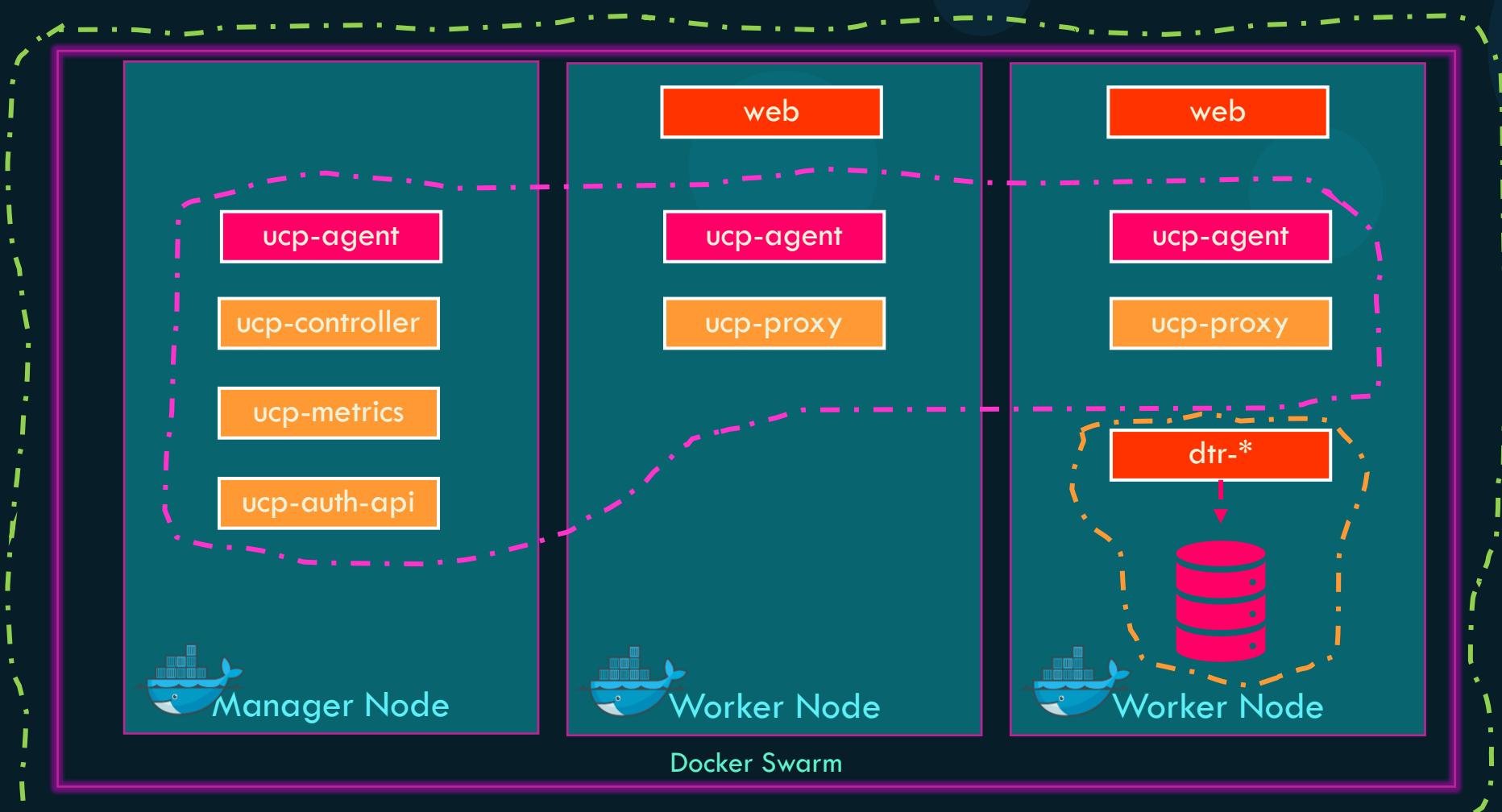
DTR - Restore

- ▶ `docker run -it --rm \
 docker/dtr destroy \
 --ucp-insecure-tls`
- ▶ `docker run -i --rm \
 docker/dtr restore < dtr-metadata-backup.tar`

<https://docs.mirantis.com/docker-enterprise/v3.0/dockeree-products/dtr/dtr-admin/disaster-recovery/restore-from-backup.html>



Backup and Restoration







title

sample

```
▶ docker run ubuntu
```

```
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
fc7181108d40: Already exists
d2e987ca2267: Pull complete
0b760b431b11: Pull complete
Digest:
sha256:96fb261b66270b900ea5a2c17a26abbfabe95506e73c3a3c65869a6dbe83223a
Status: Downloaded newer image for nginx:latest
```



Sample - Commands

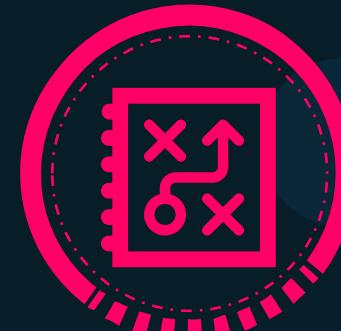
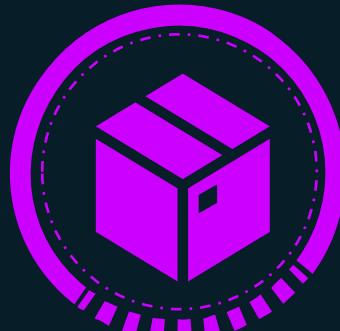
```
▶ docker run nginx
```

```
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
fc7181108d40: Already exists
d2e987ca2267: Pull complete
0b760b431b11: Pull complete
Digest:
sha256:96fb261b66270b900ea5a2c17a26abbfabe95506e73c3a3c65869a6dbe83223a
Status: Downloaded newer image for nginx:latest
```



Sample - Containers

▶ docker run ubuntu



▶ docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
--------------	-------	---------	---------	--------	-------

▶ docker ps -a

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
45aacca36850	ubuntu	"/bin/bash"	43 seconds ago	Exited (0) 41 seconds ago	



Sample – Highlighting command/output

```
docker run redis

Using default tag: latest
latest: Pulling from library/redis
f5d23c7fed46: Pull complete
Status: Downloaded newer image for redis:latest

1:C 31 Jul 2019 09:02:32.624 # o000o000o000o Redis is starting o000o000o000o
1:C 31 Jul 2019 09:02:32.624 # Redis version=5.0.5, bits=64, commit=00000000, modified=0, pid=1, just started
1:M 31 Jul 2019 09:02:32.626 # Server initialized
```

```
docker run redis:4.0 TAG
Unable to find image 'redis:4.0' locally
4.0: Pulling from library/redis
e44f086c03a2: Pull complete
Status: Downloaded newer image for redis:4.0

1:C 31 Jul 09:02:56.527 # o000o000o000o Redis is starting o000o000o000o
1:C 31 Jul 09:02:56.527 # Redis version=4.0.14, bits=64, commit=00000000, modified=0, pid=1, just started
1:M 31 Jul 09:02:56.530 # Server initialized
```



Sample – Port Mappings

```
docker run kodekloud/webapp
```

```
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

http://172.17.0.2:5000

Internal IP

```
docker run -p 80:5000 kodekloud/simple-webapp
```

```
docker run -p 8000:5000 kodekloud/simple-webapp
```

```
docker run -p 8001:5000 kodekloud/simple-webapp
```

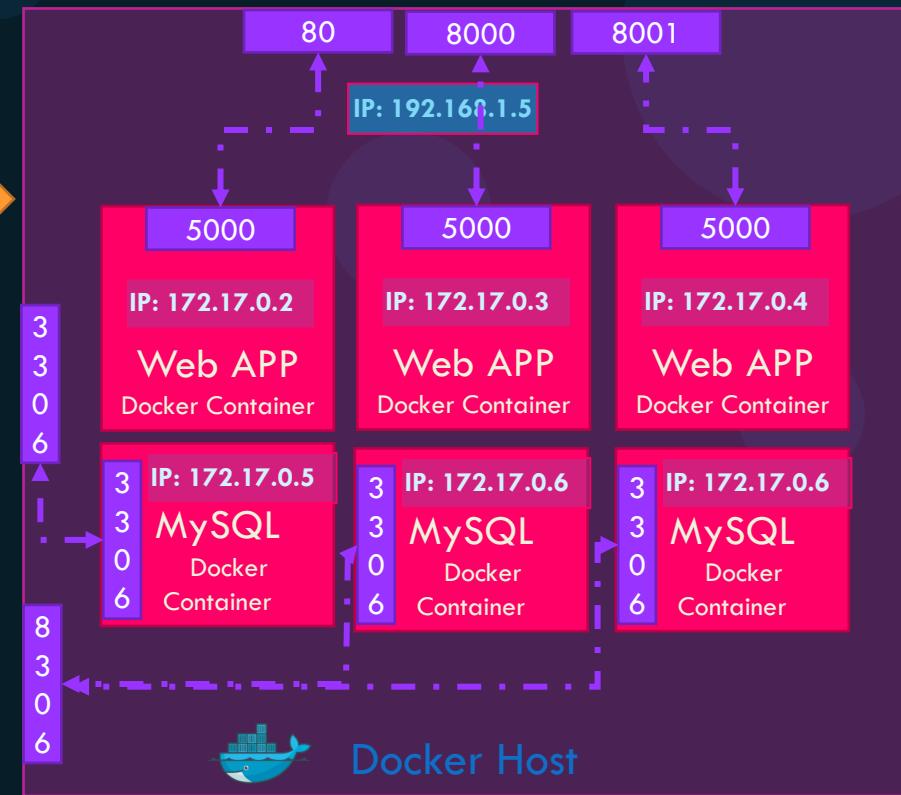
```
docker run -p 3306:3306 mysql
```

```
docker run -p 8306:3306 mysql
```

```
docker run -p 8306:3306 mysql
```



http://192.168.1.5:80



```
root@osboxes:/root # docker run -p 8306:3306 -e MYSQL_ROOT_PASSWORD=pass mysql
docker: Error response from daemon: driver failed programming external connectivity on endpoint boring_bhabha (5079d342b7e8ee11c71d46): Bind for 0.0.0.0:8306 failed: port is already allocated.
```



Inspect Container

```
▶ docker inspect blissful_hopper
```

```
[  
 {  
   "Id": "35505f7810d17291261a43391d4b6c0846594d415ce4f4d0a6ffbf9cc5109048",  
   "Name": "/blissful_hopper",  
   "Path": "python",  
   "Args": [  
     "app.py"  
   ],  
   "State": {  
     "Status": "running",  
     "Running": true,  
   },  
   "Mounts": [],  
   "Config": {  
     "Entrypoint": [  
       "python",  
       "app.py"  
     ],  
   },  
   "NetworkSettings": {...}  
 }
```



Sample – Application Code

app.py

```
import os
from flask import Flask

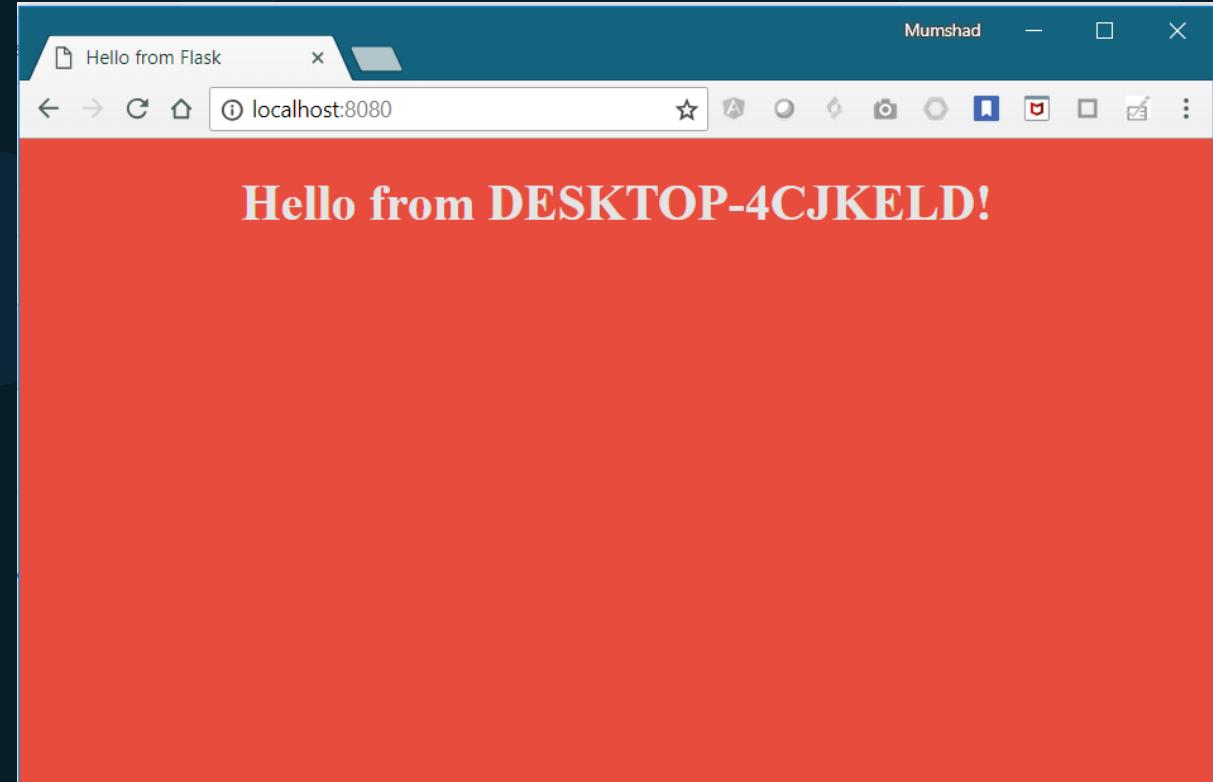
app = Flask(__name__)

...
...

color = "red"

@app.route("/")
def main():
    print(color)
    return render_template('hello.html', color=color)

if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```



▶ python app.py


Applying Finishing Touches

We will be here soon !

