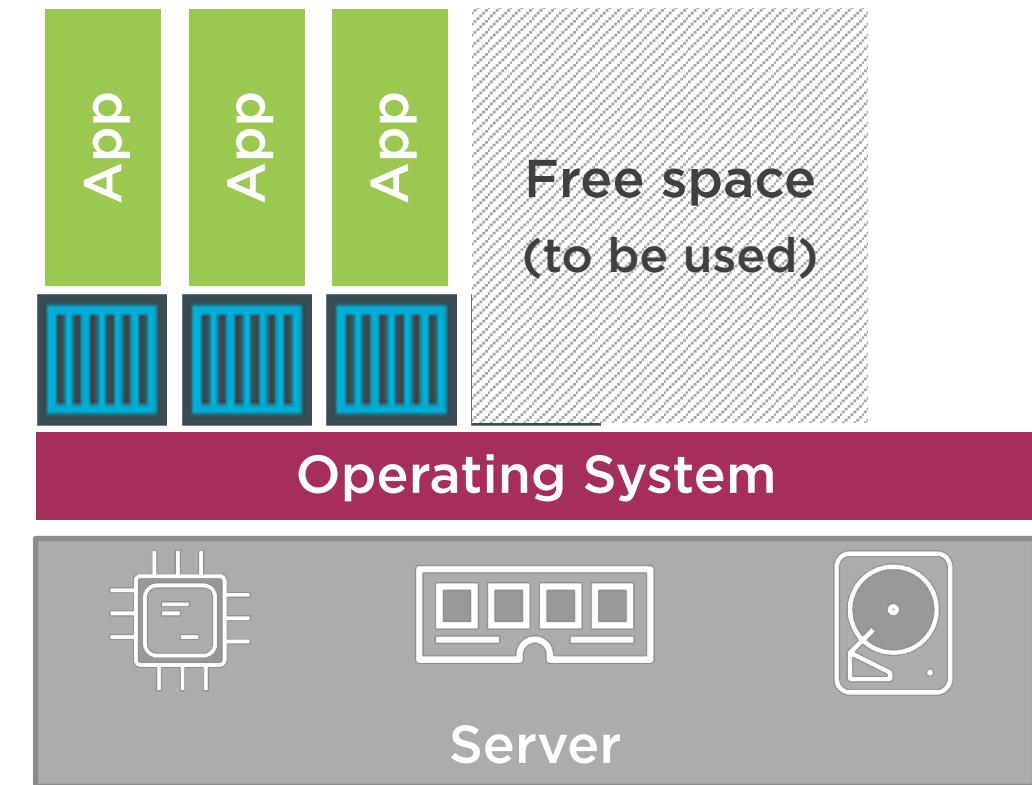
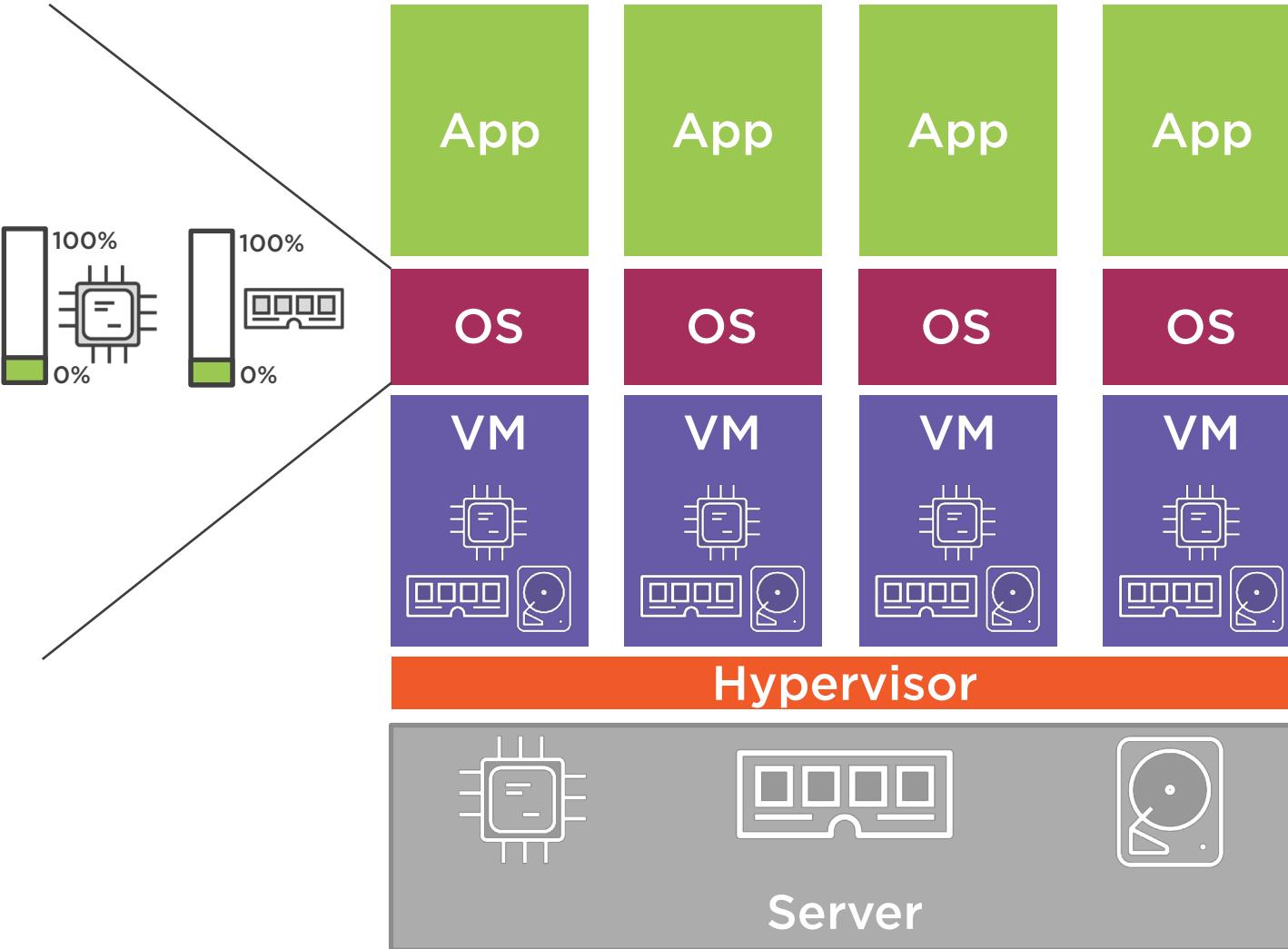
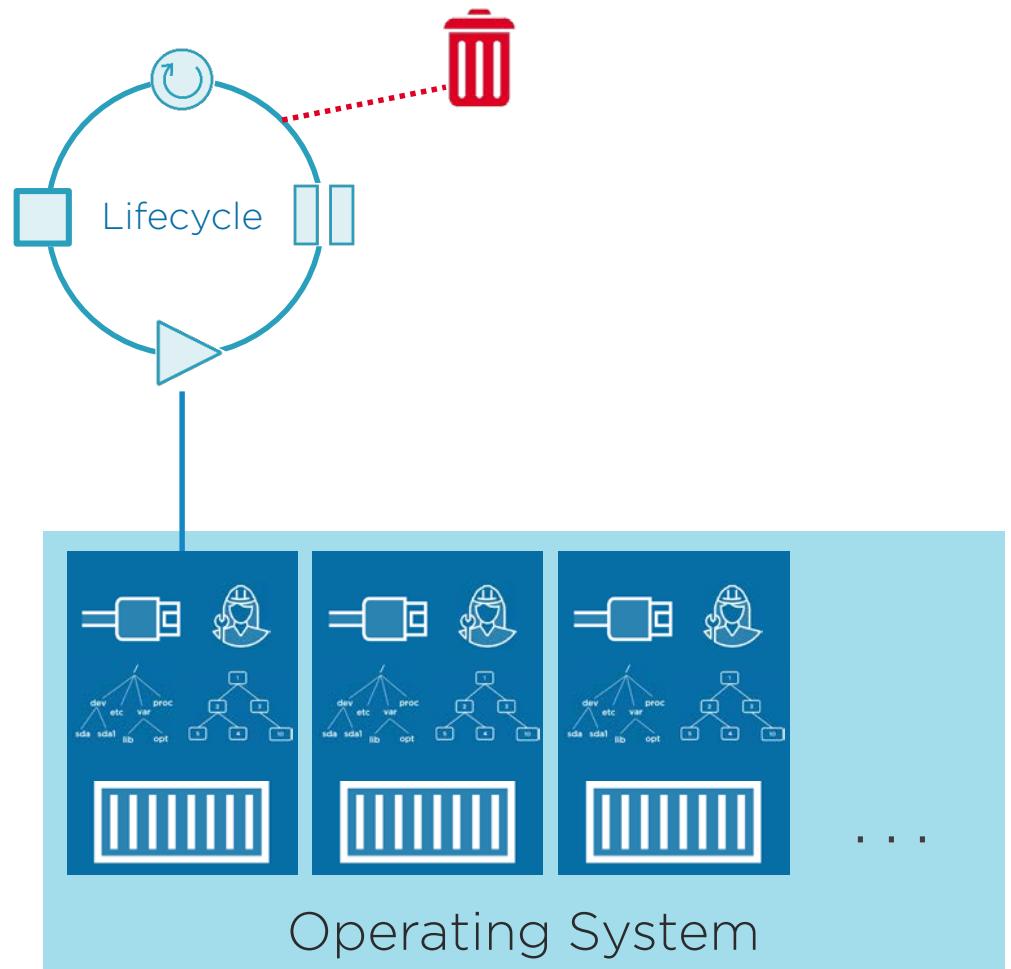
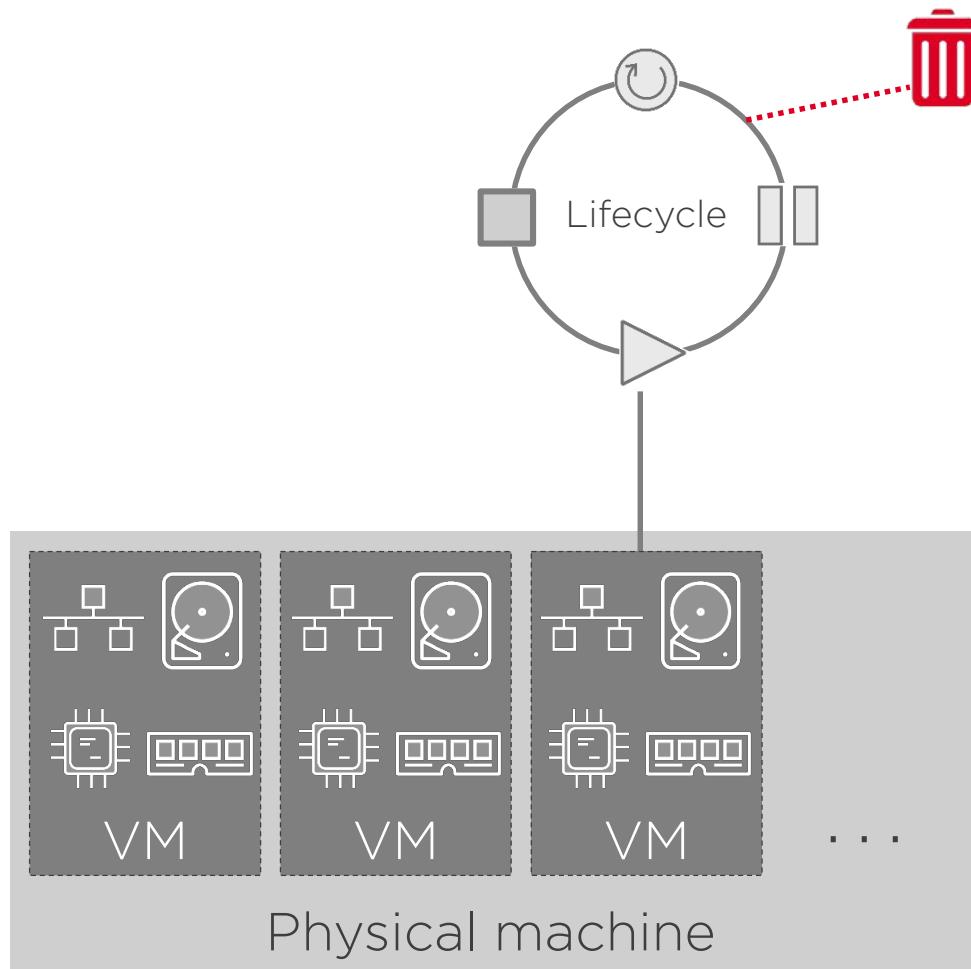


← → **Atomic units**







# The Benefits of Containers



**Resource isolation (sandboxing)**

**Less overhead (compared to VM)**

**Increased portability (run anywhere)**

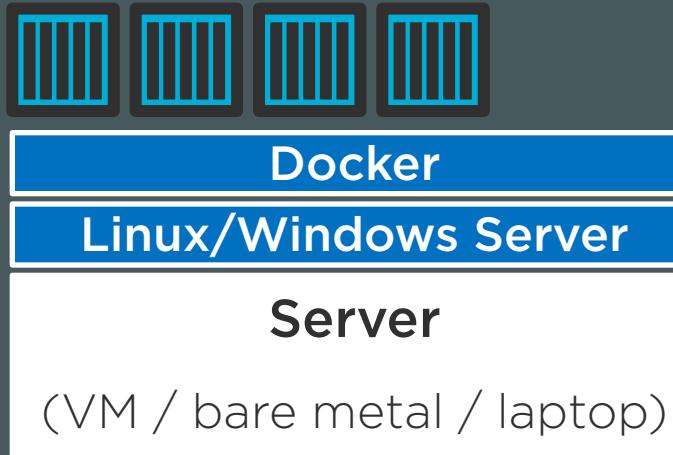
**Consistent environment**

**Greater efficiency**

**Elasticity**

**Constraints (memory, CPU, etc.)**

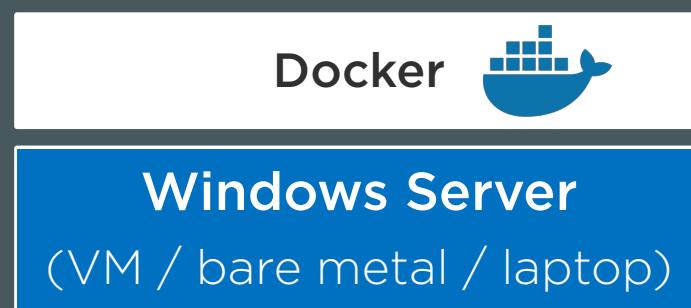




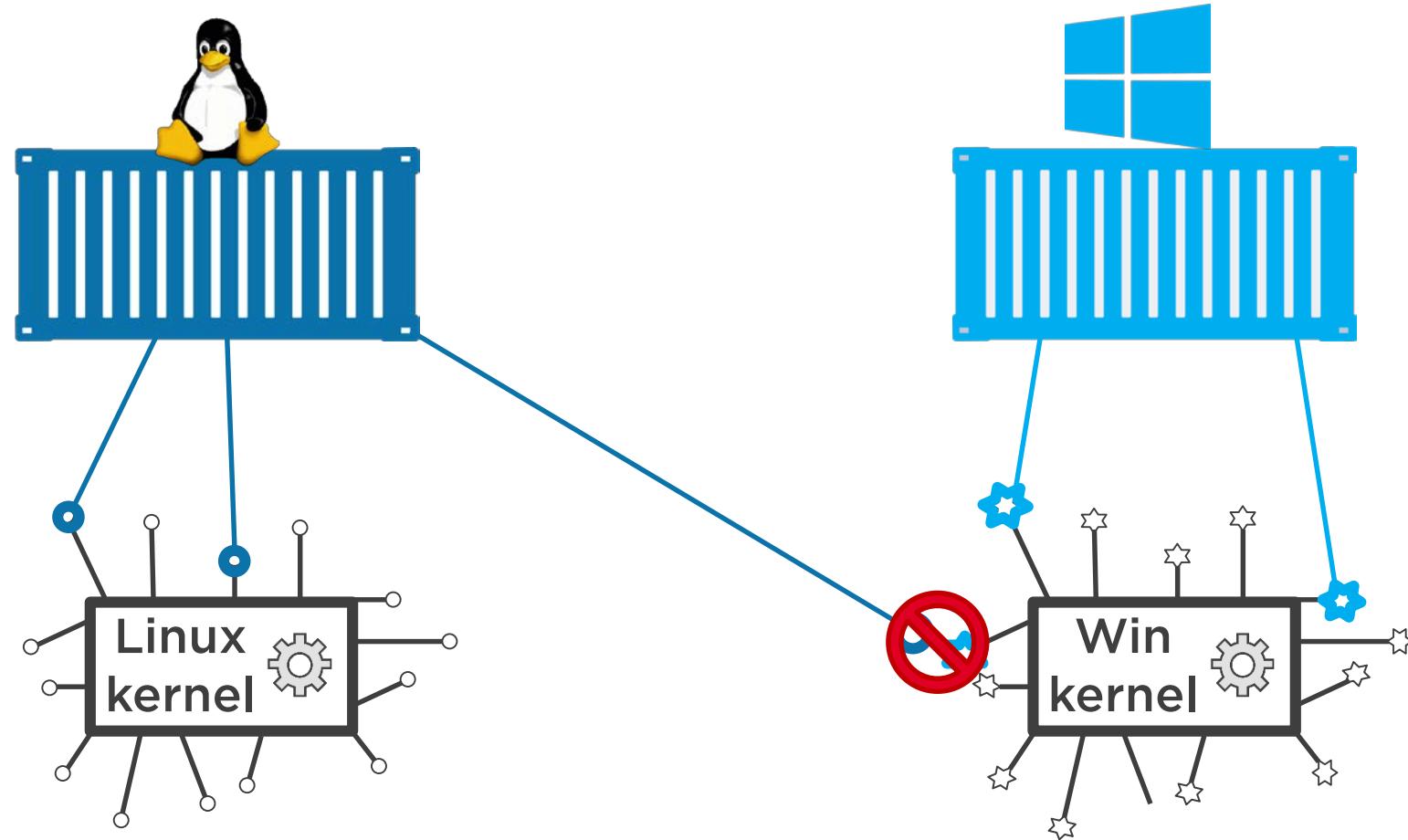
Windows apps

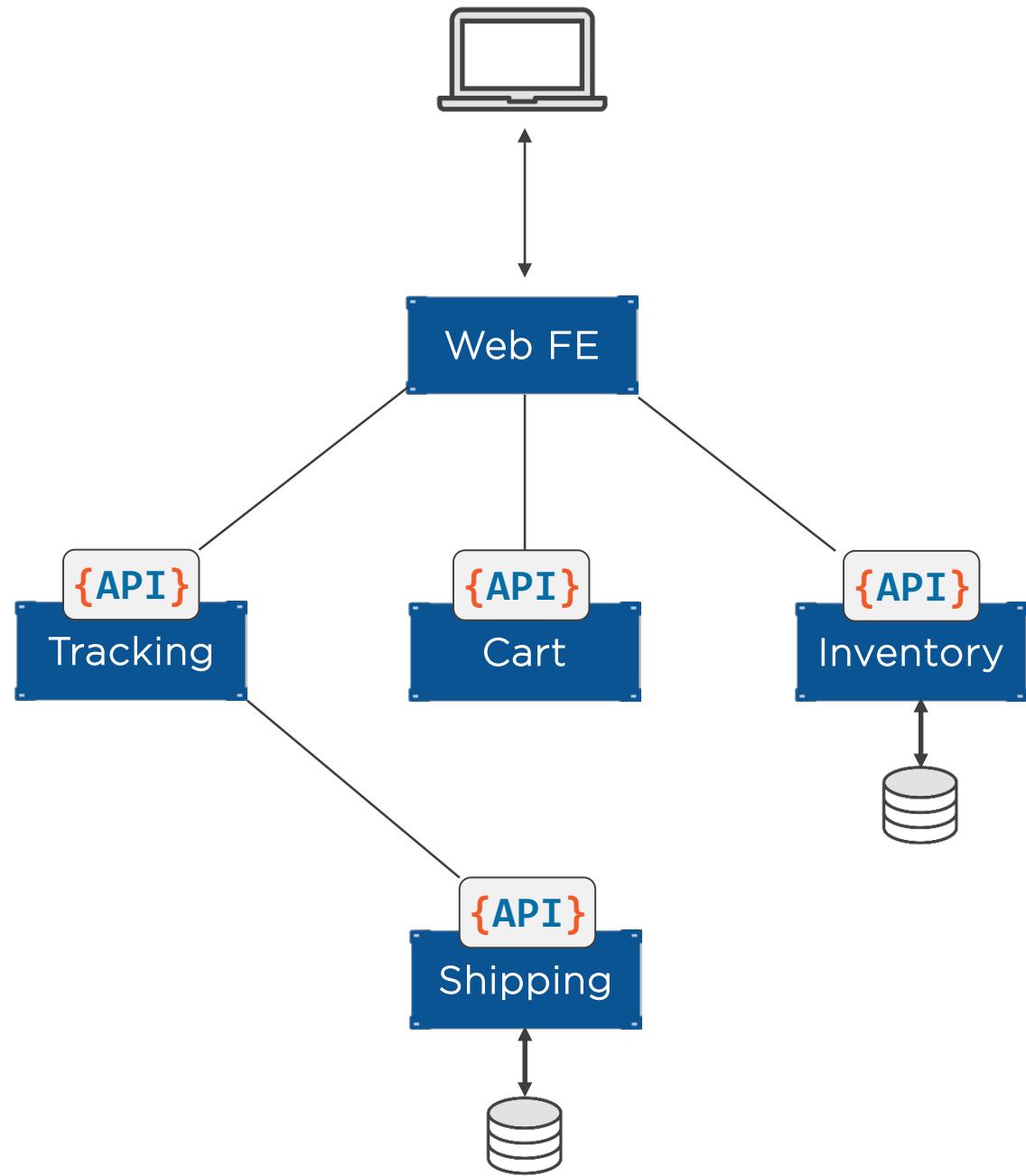
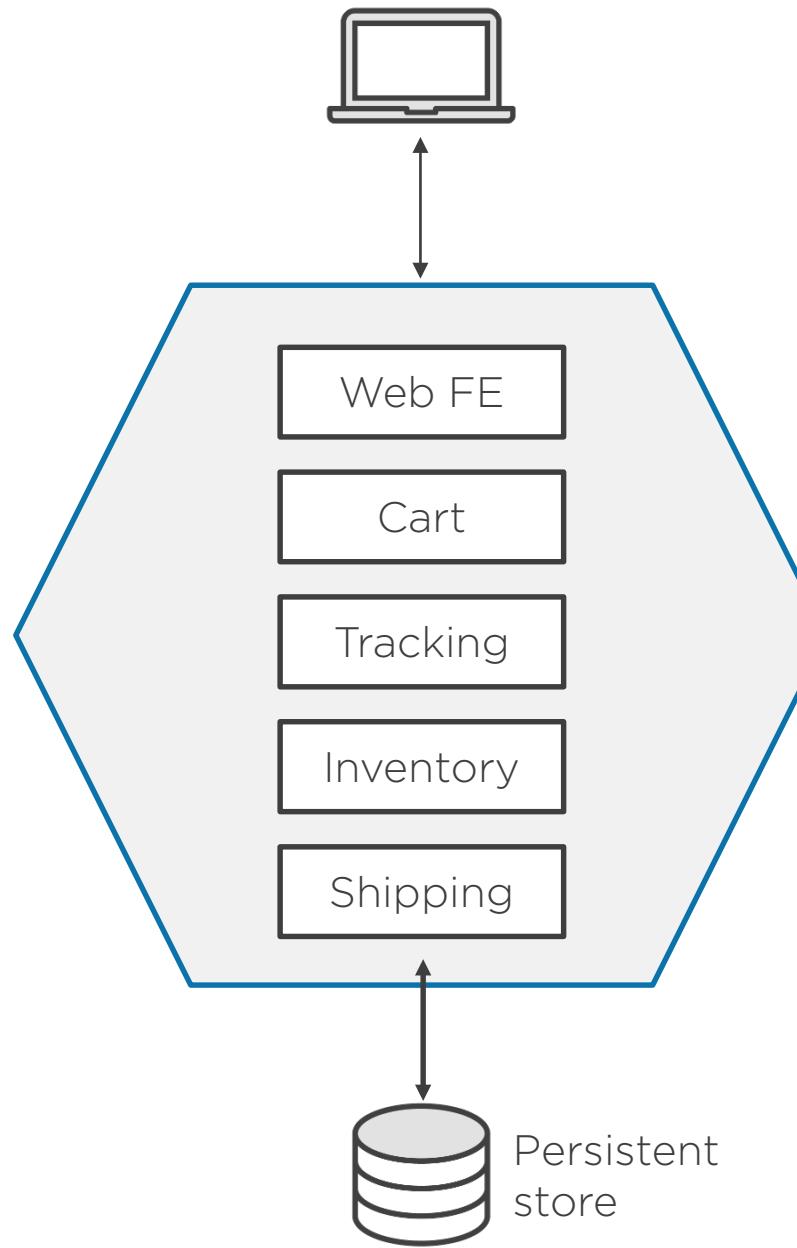


Linux apps



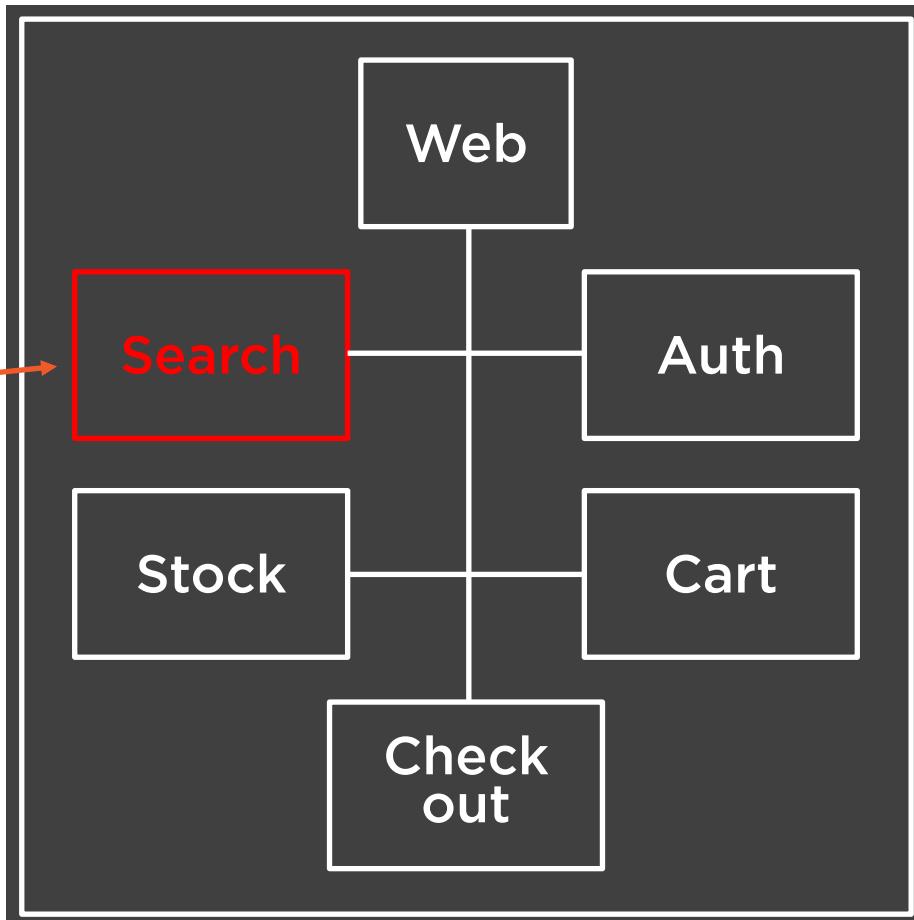
\*It may be possible to run your Linux apps on Docker on Windows





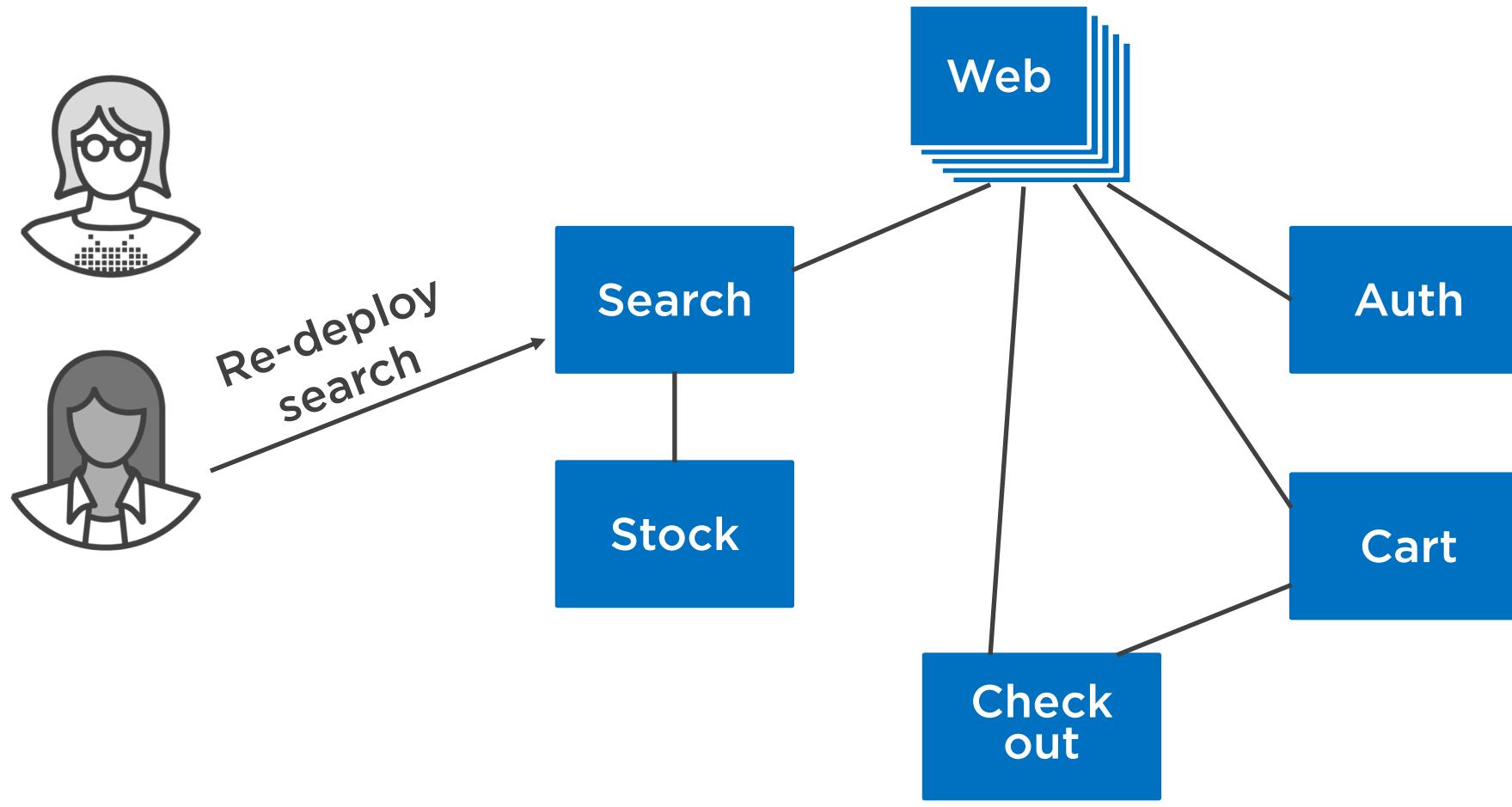


Fix search



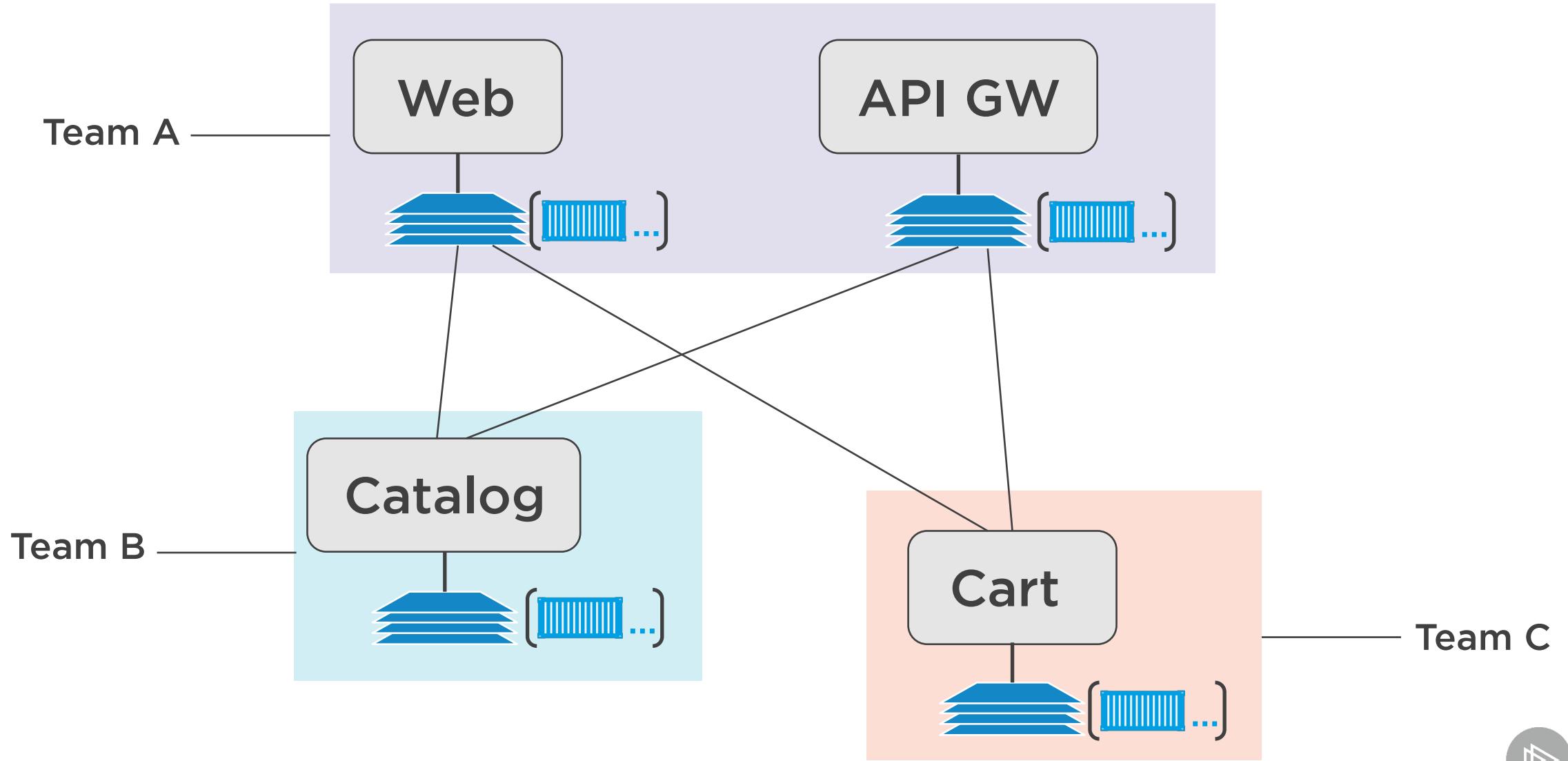
**Monolith/Legacy App**  
(Single binary)





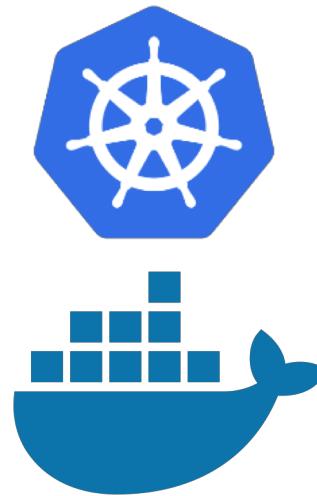
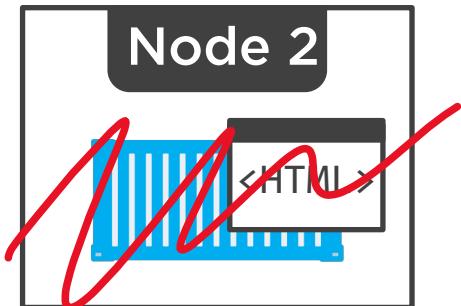
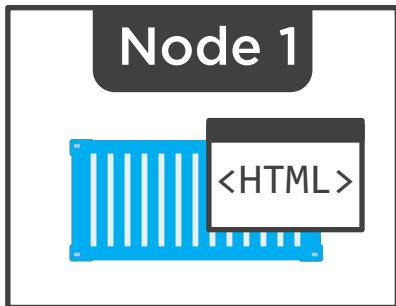
On-prem / cloud





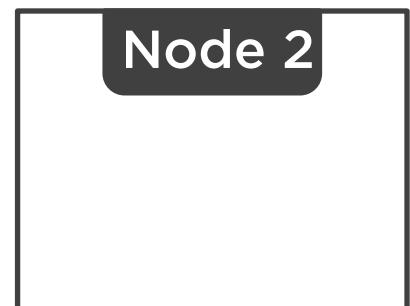
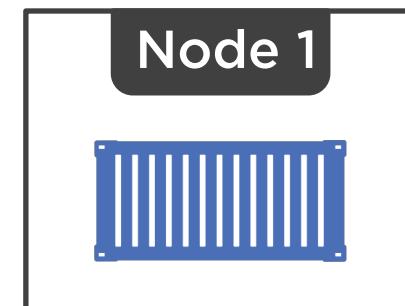
# ✓ Stateless

- Doesn't remember stuff

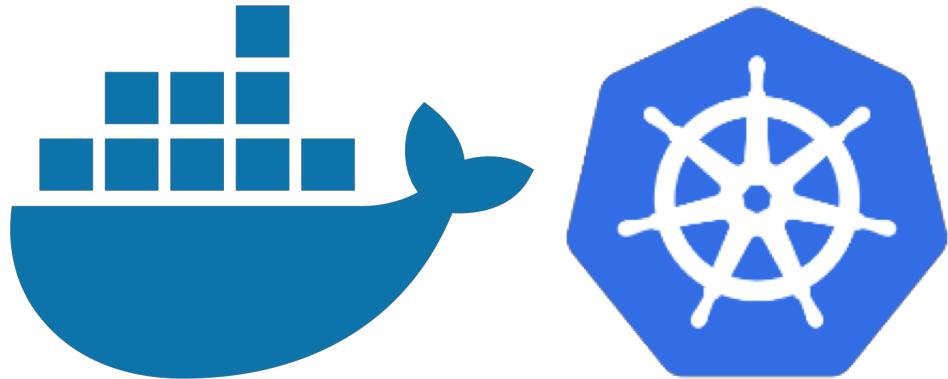


# Stateful ✓

- Has to remember stuff

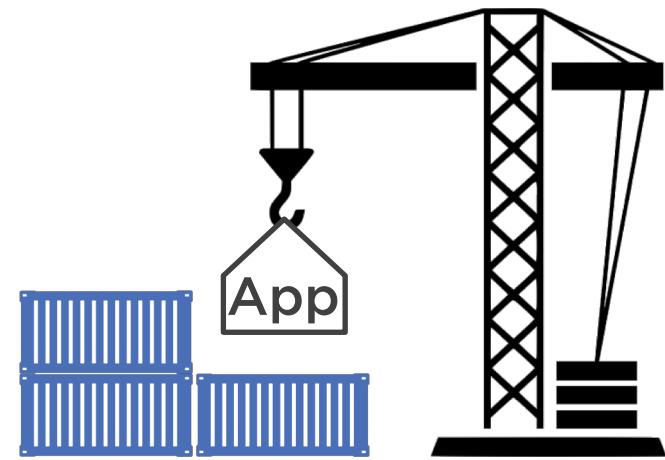


# Stateful



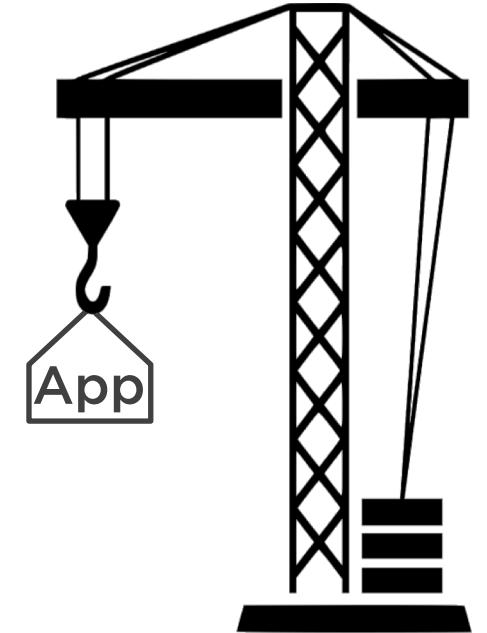
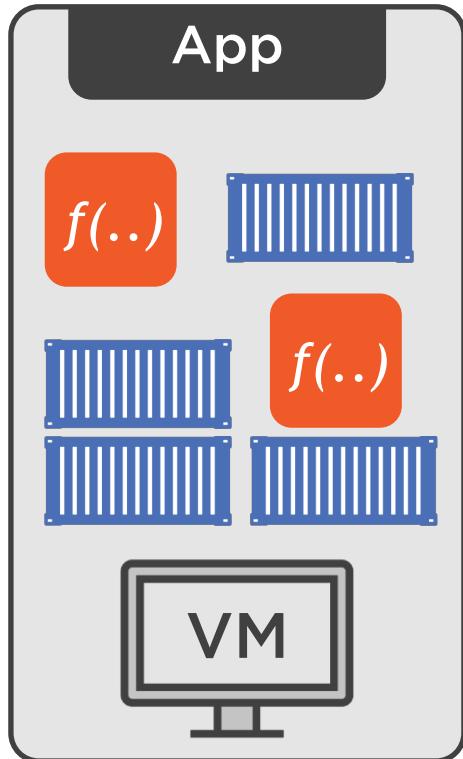
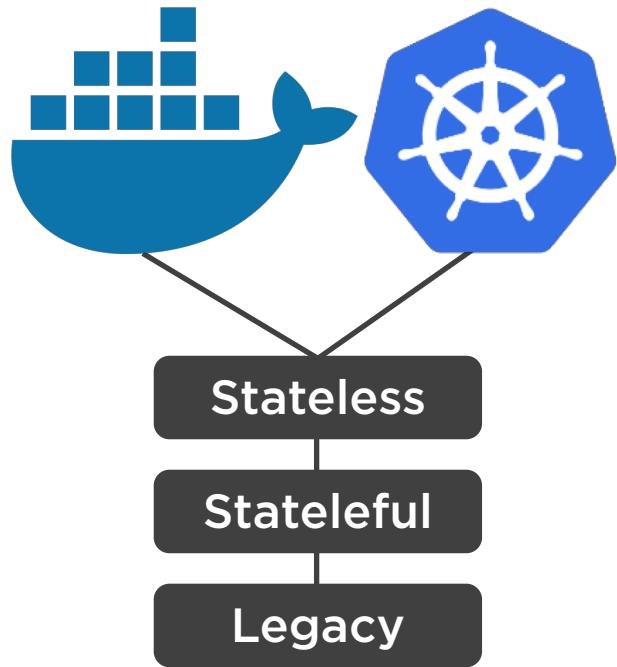
Added features for  
stateful apps

# Legacy

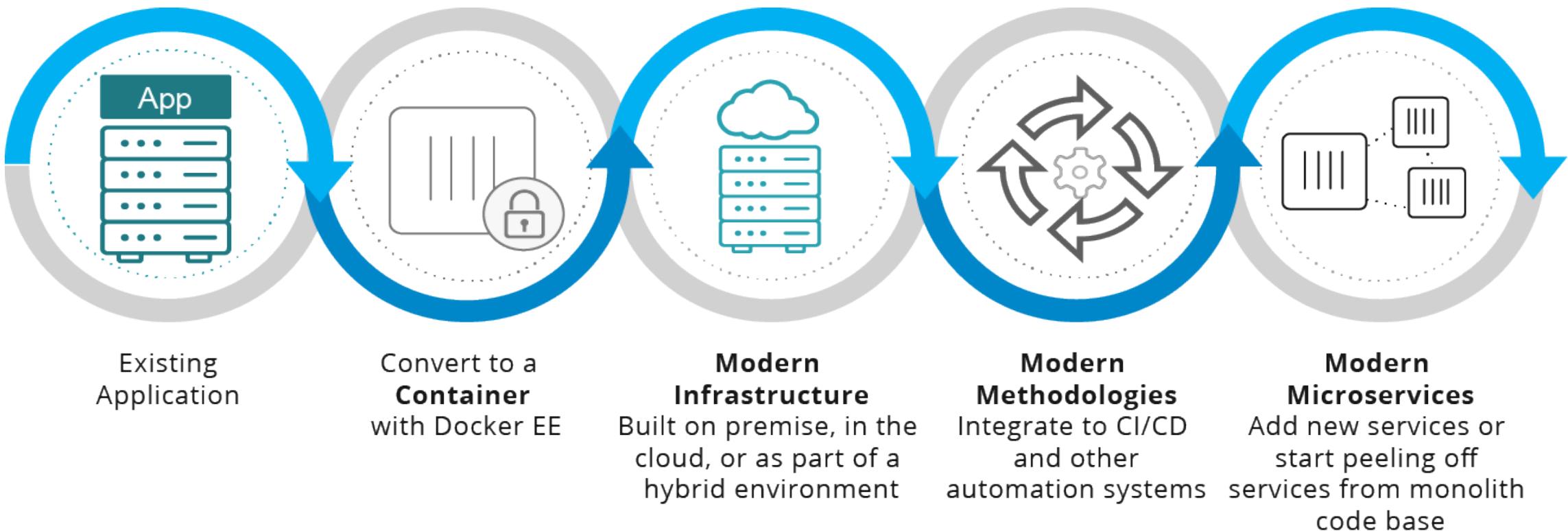


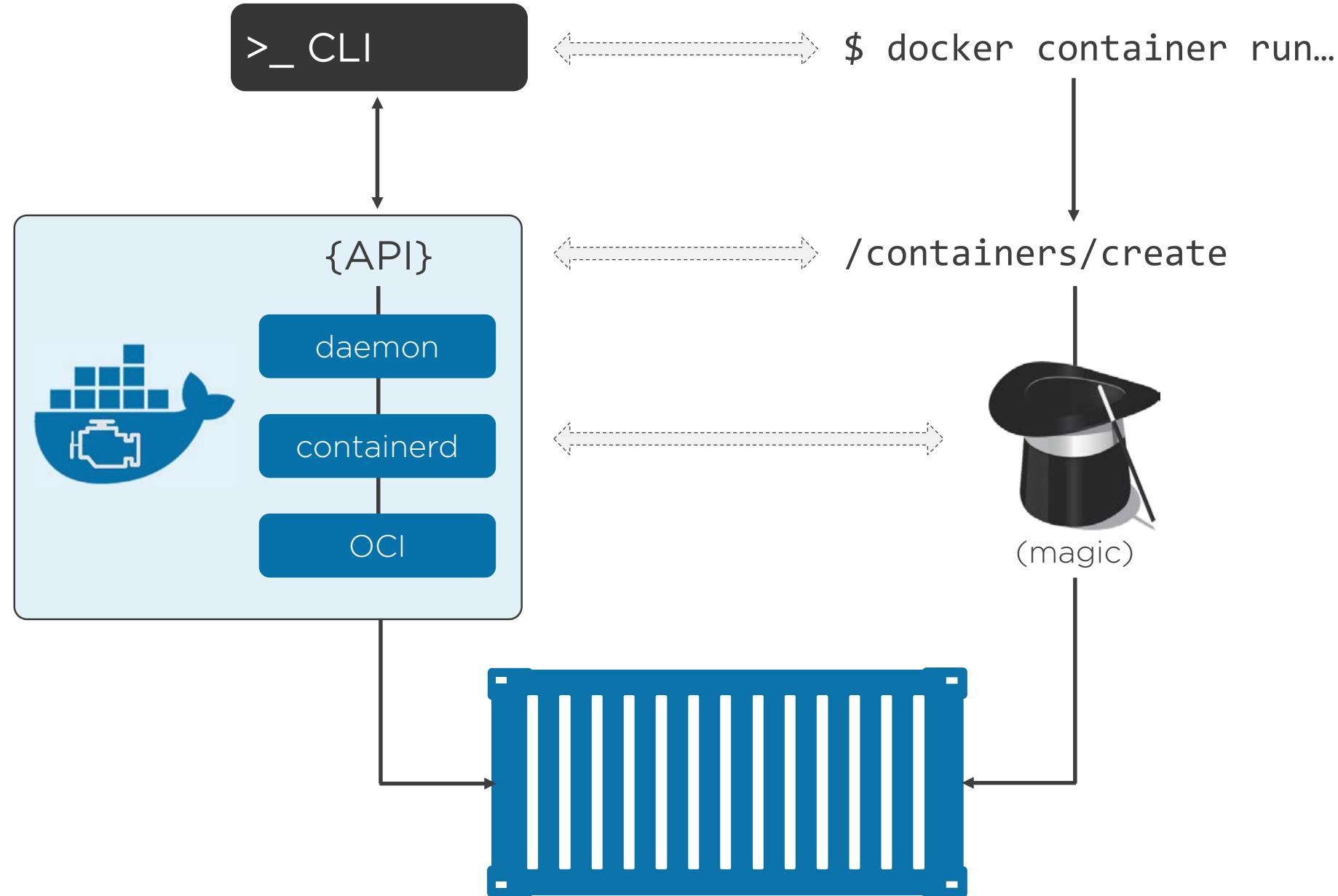
Some apps can be  
migrated directly to  
containers

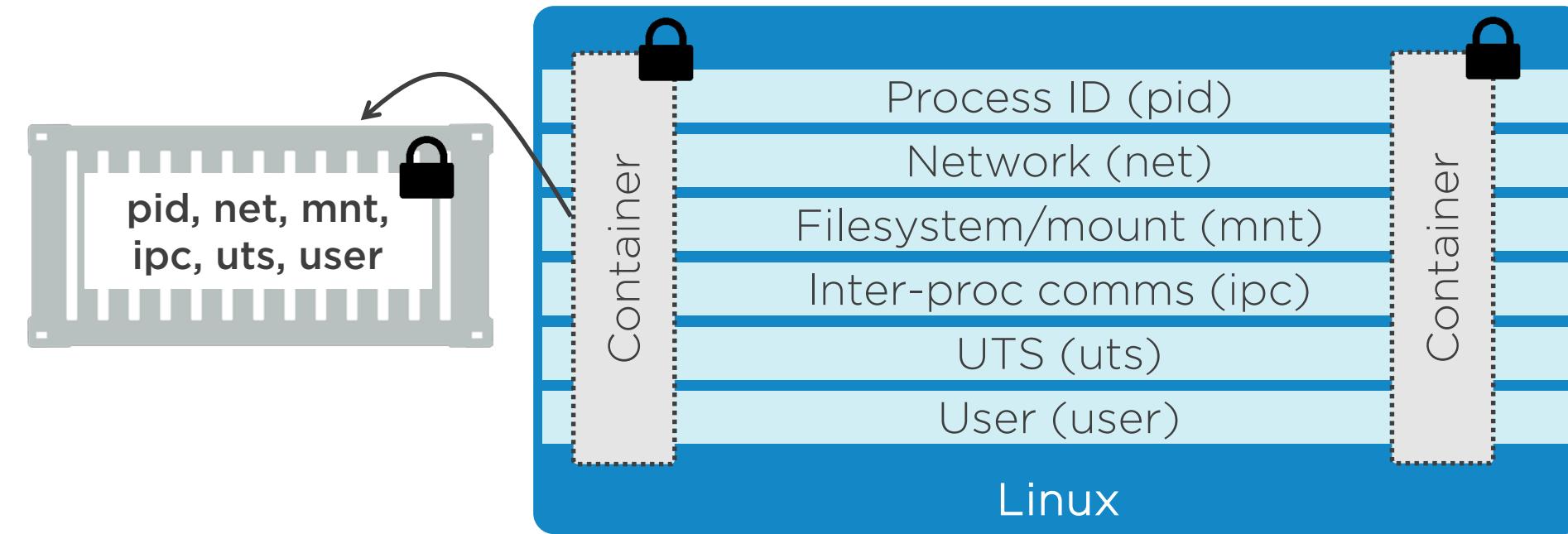




# Modernize Traditional Apps





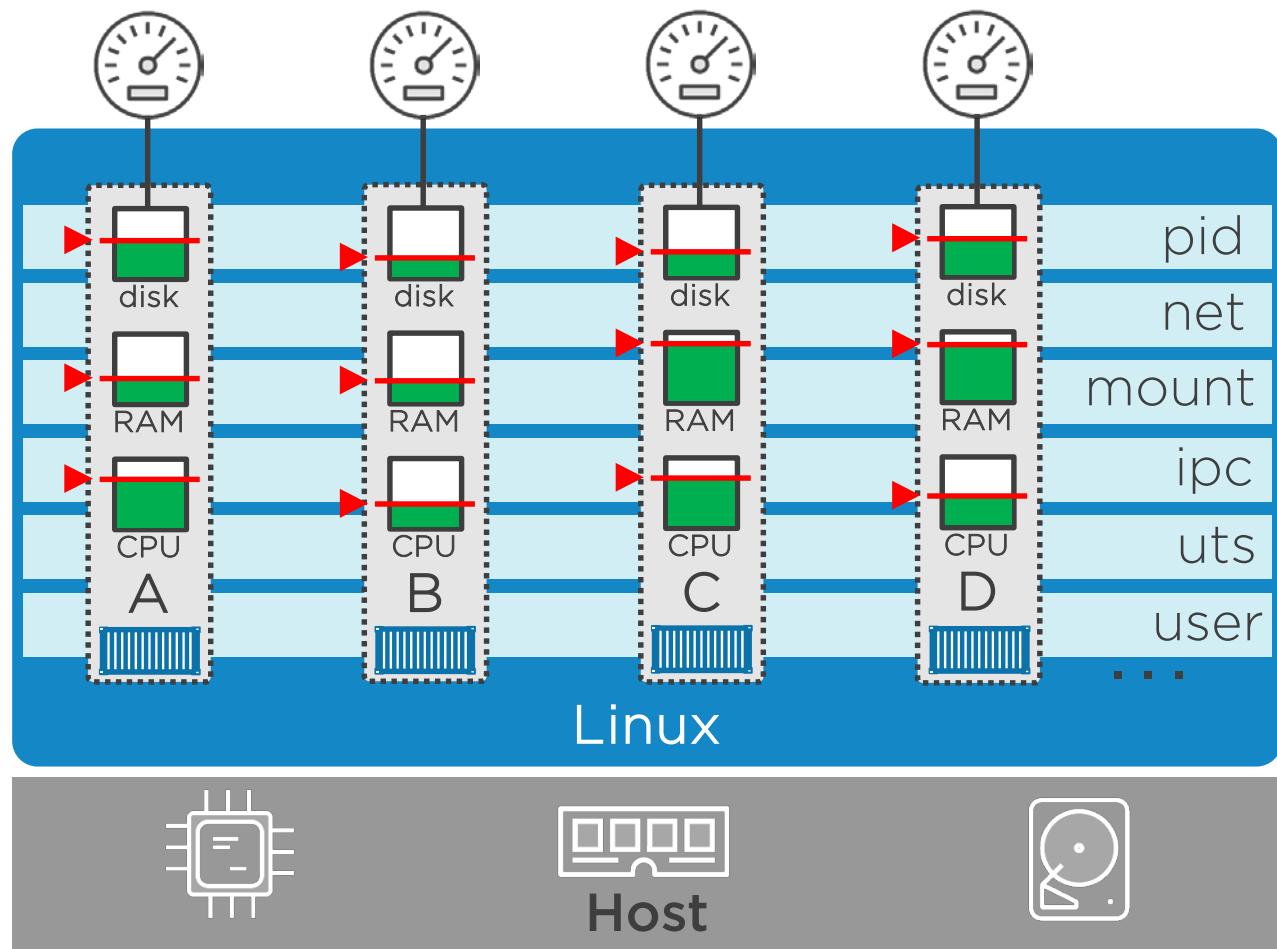


Namespaces

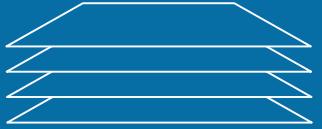


Control Groups





## Layers



Union fs/mount  
& CoW



## Linux

- pid
- net
- mount
- ...

## Namespaces

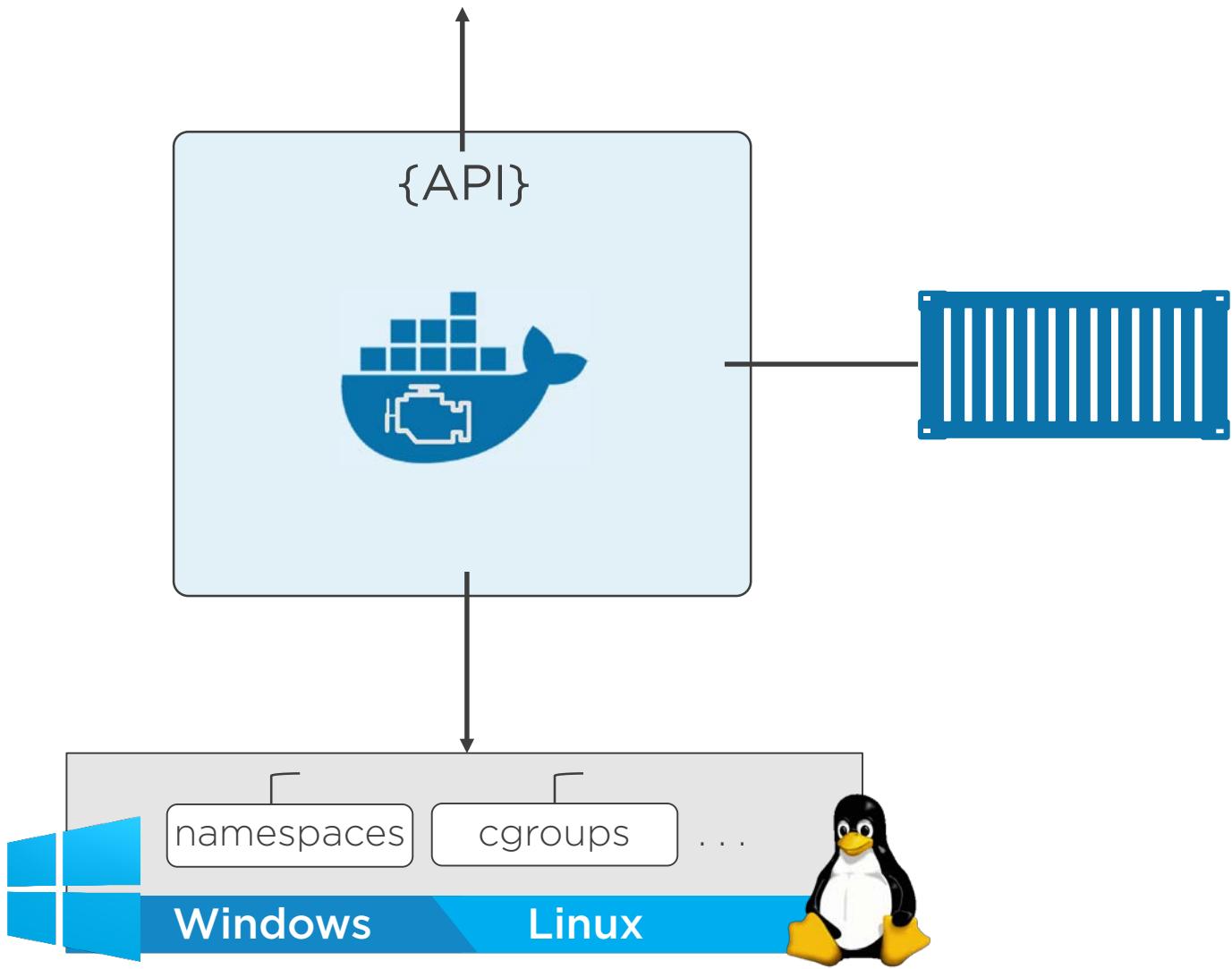
### Windows

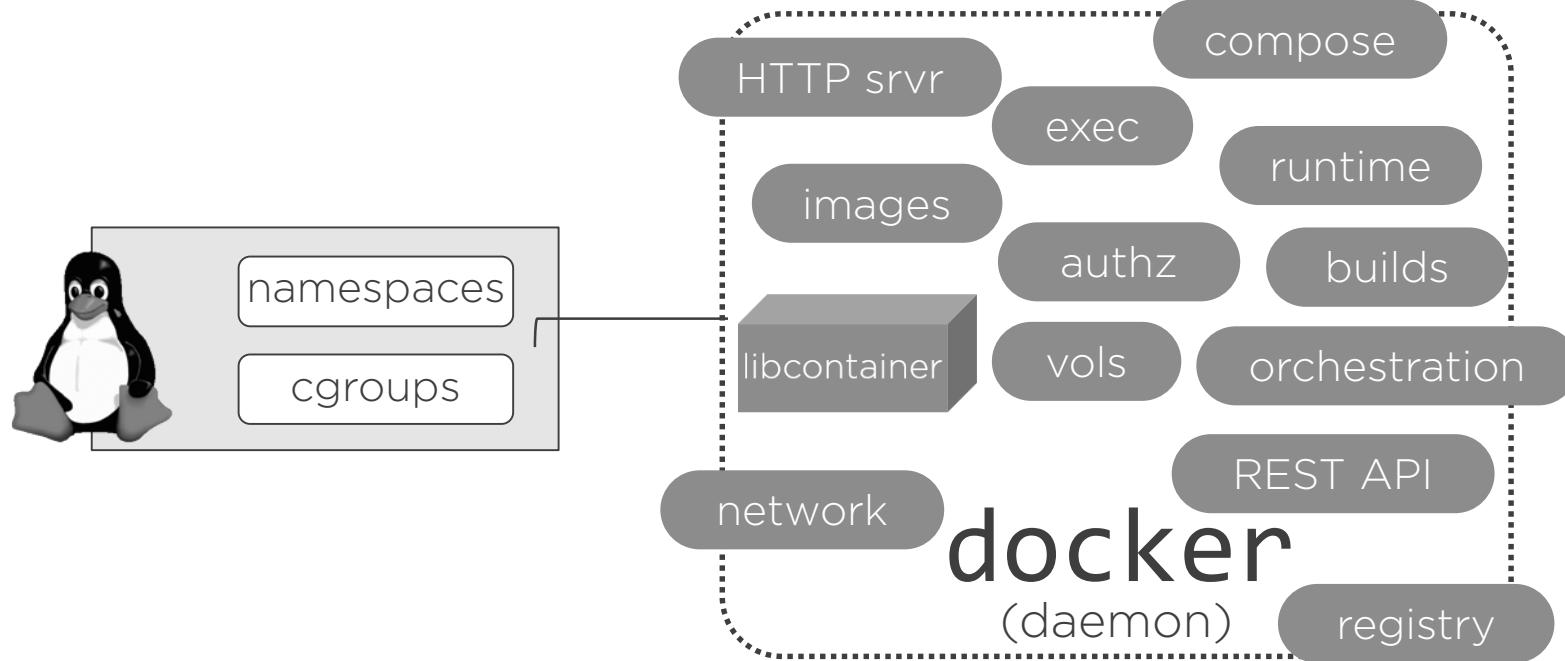
- object
- proc table
- networking
- ...

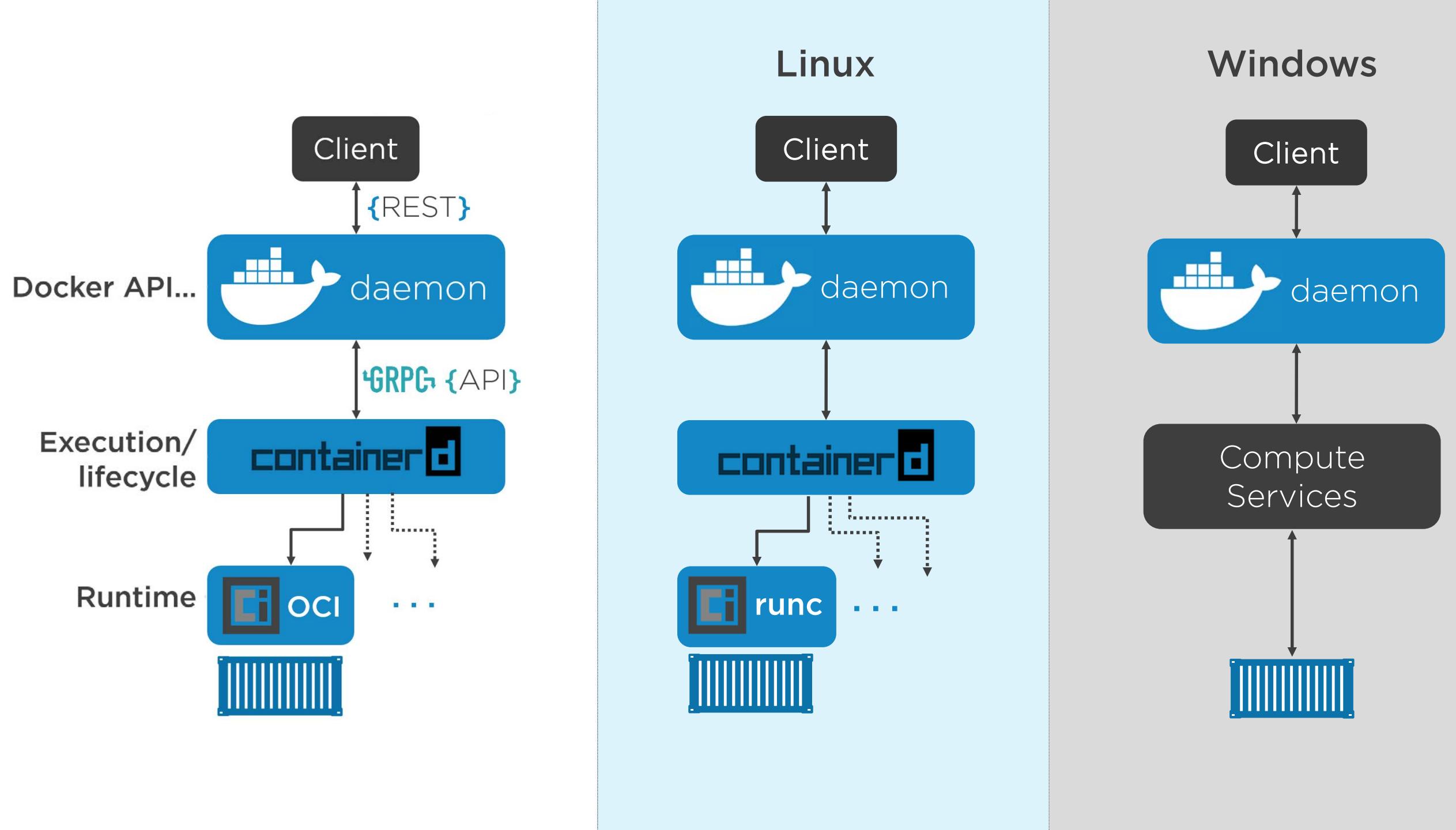


## Control Groups (Windows a.k.a. Job Objects)

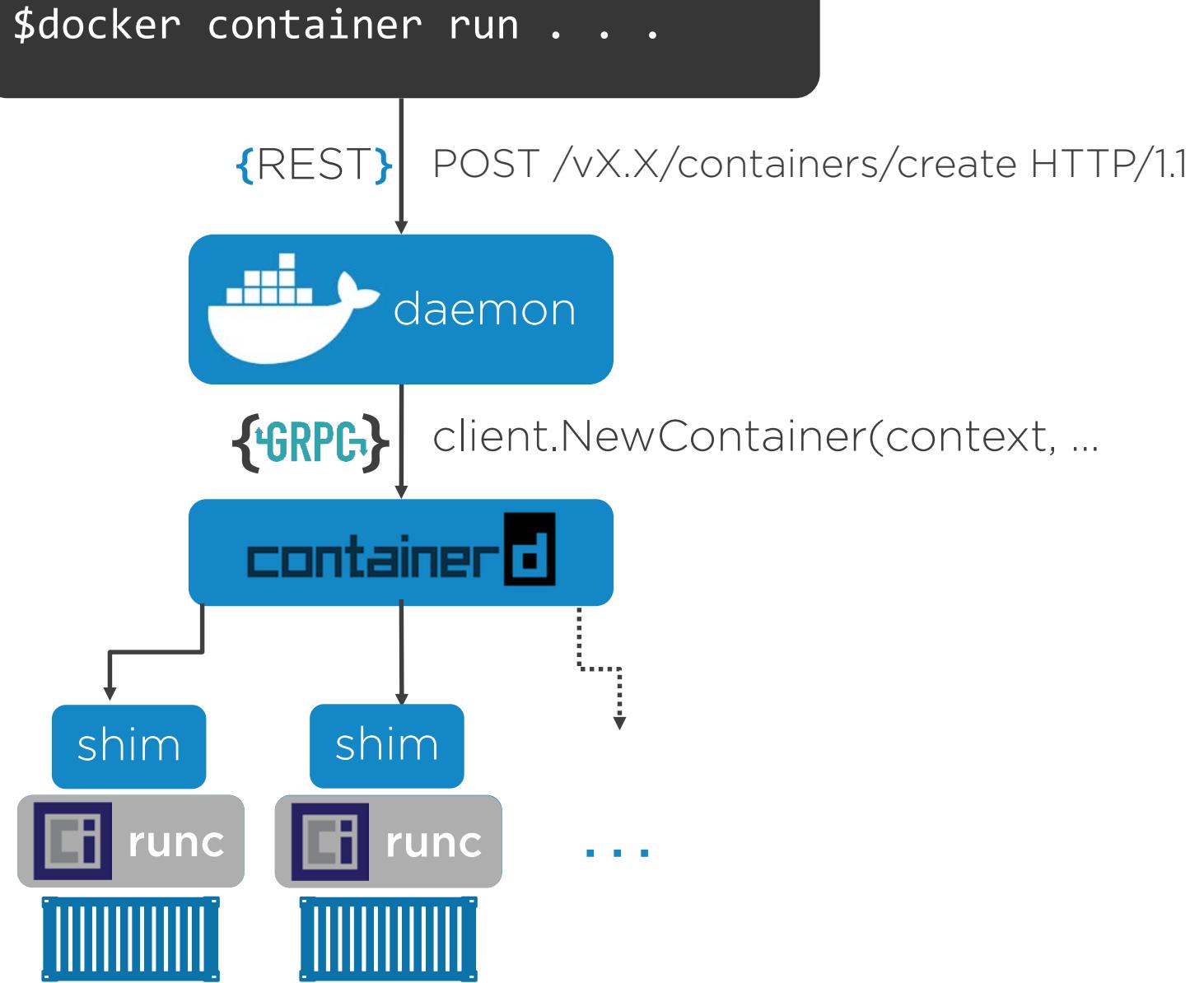
- Grouping processes
- Imposing resource limits



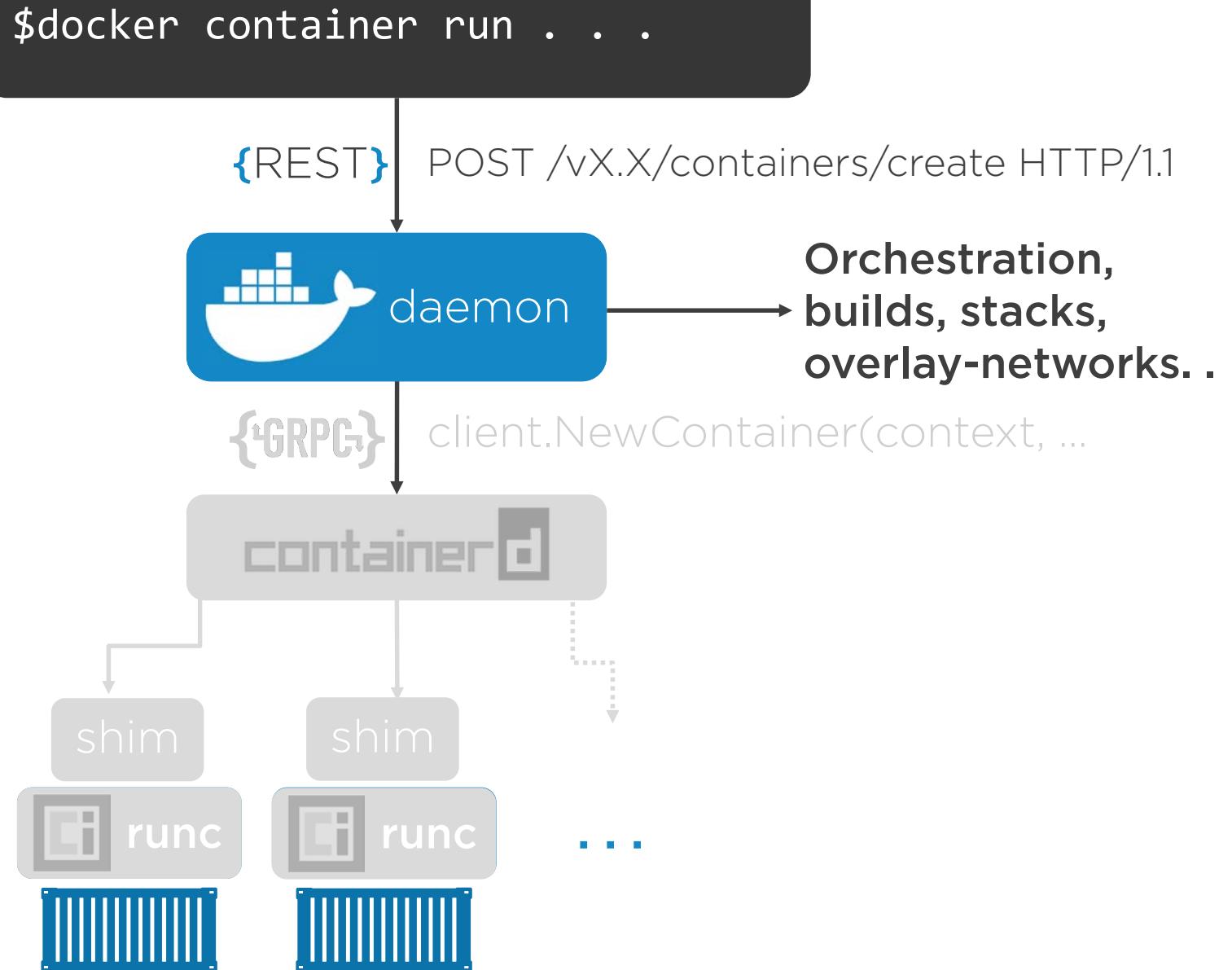




**Example:**  
Creating a new  
container on  
Linux



**Example:**  
Creating a new container on  
Linux

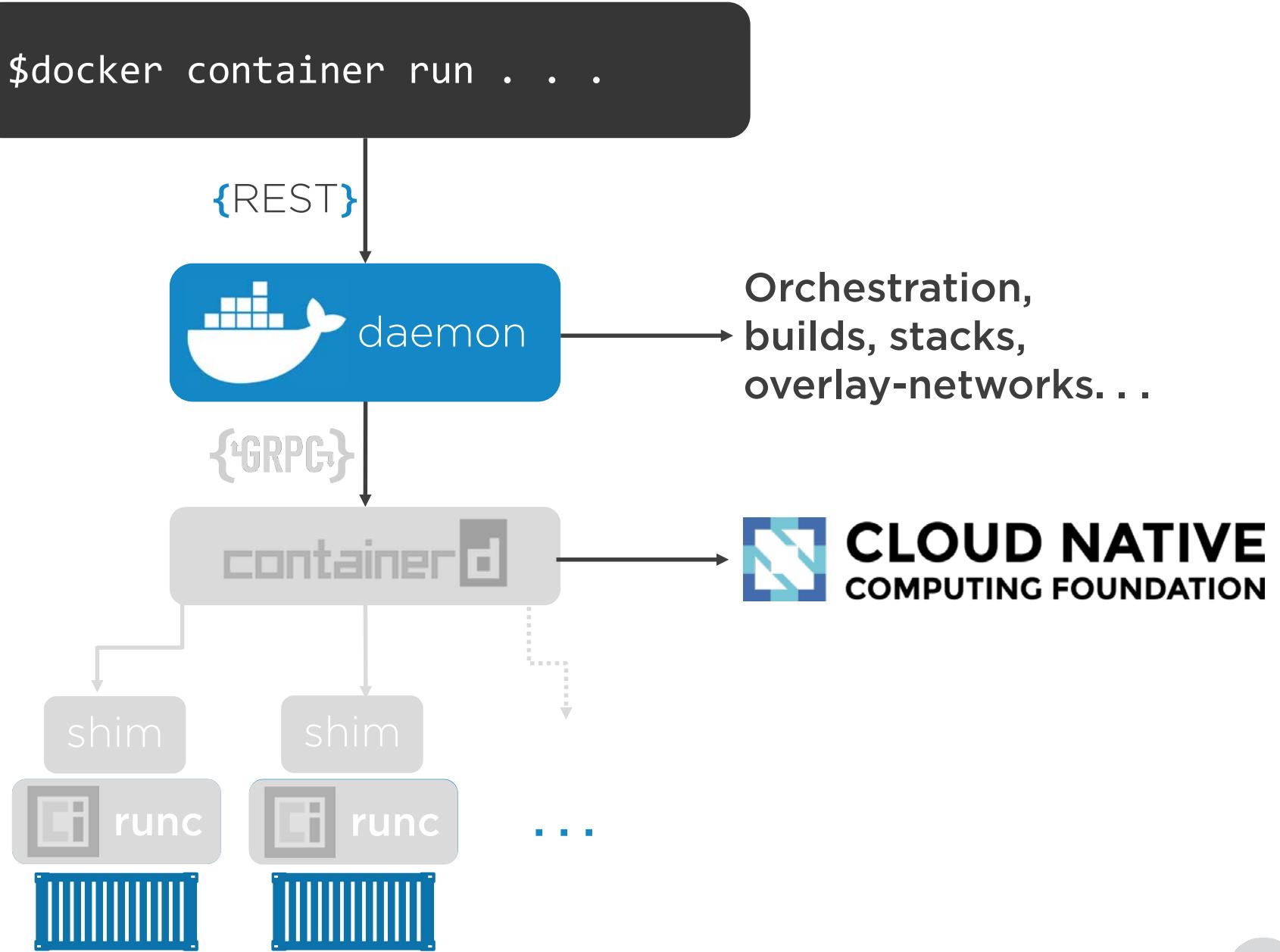


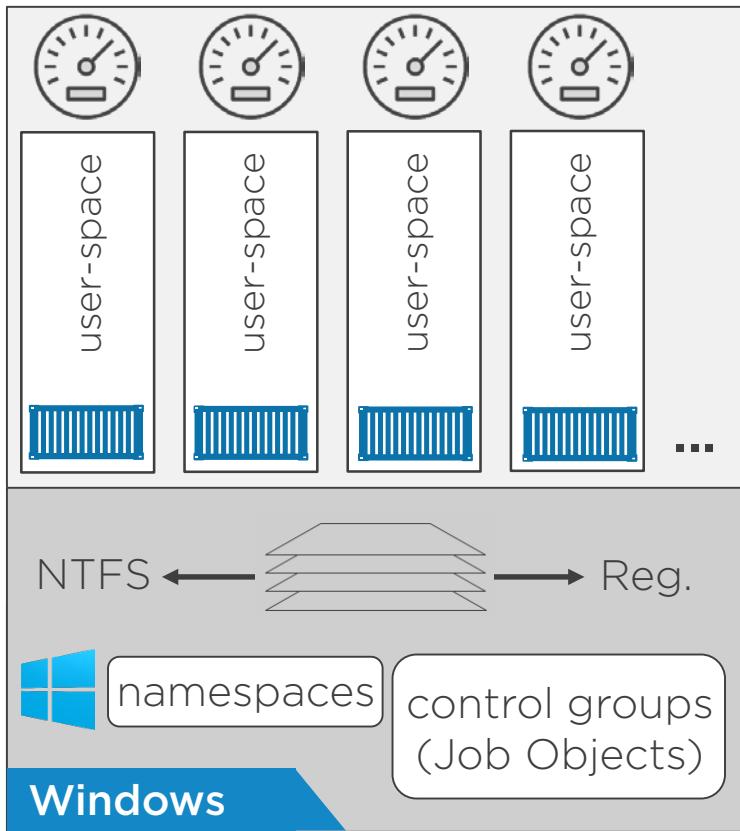
```
$docker container run . . .
```

There's a branch of `containerd` that's implementing things like:

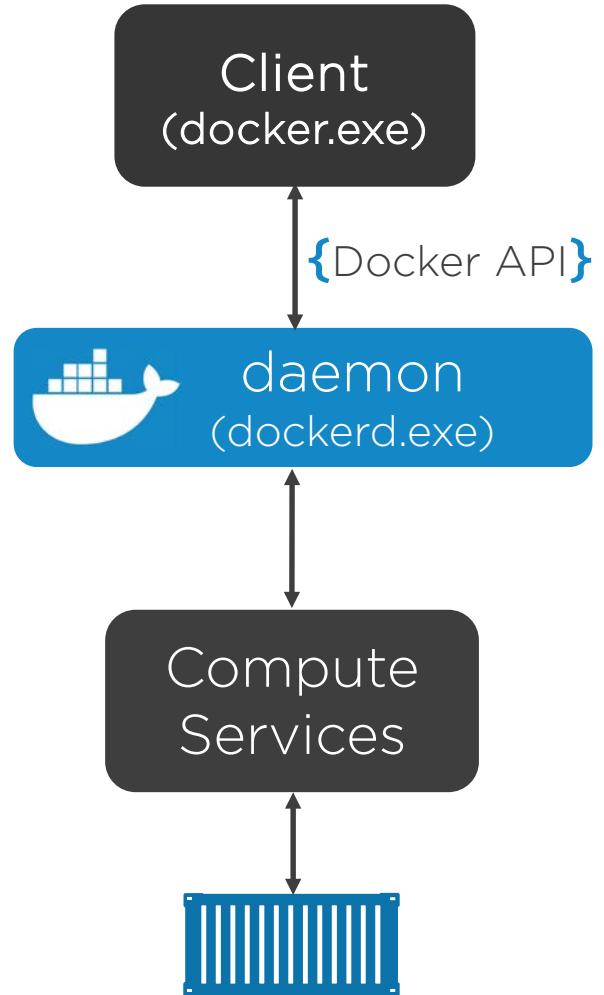
- Basic networking
- Basic storage
- Basic image distribution
- ...

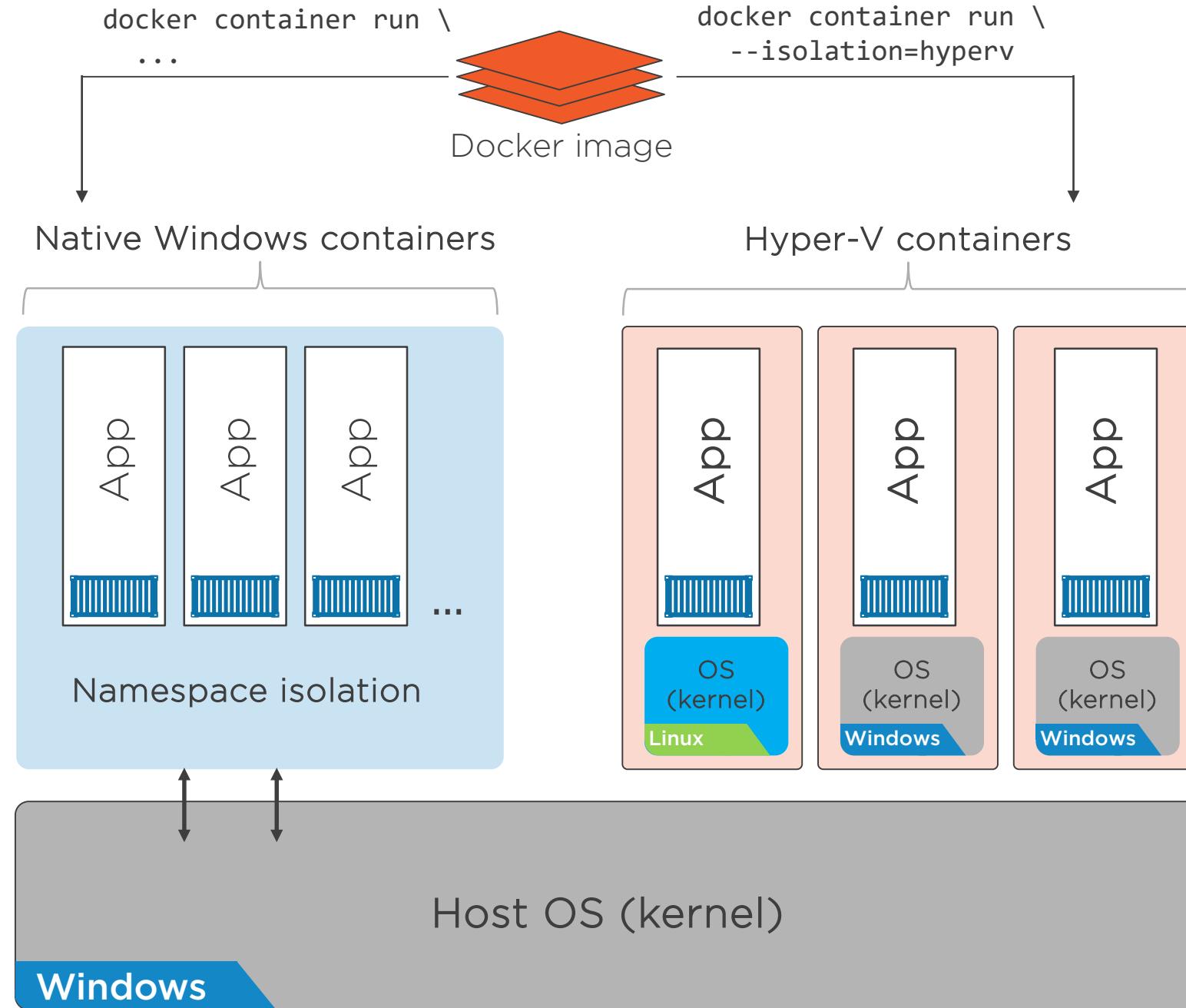
Each one is well-defined and not tightly coupled.



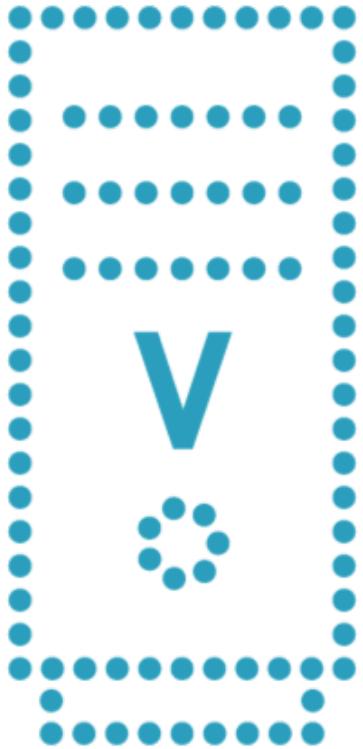


```
PS C:\> ps
Handles NPM(K) PM(K) WS(K) CPU(s) Id SI ProcessName
---- - - - - - - - - - -
0 5 964 1292 0.00 4716 4 CExecSvc
0 5 592 956 0.00 4524 4 csrss
0 0 0 4 0 0 Idle
0 18 3984 8624 0.13 700 4 lsass
0 52 26624 19400 1.64 2100 4 powershell
0 38 28324 49616 1.69 4464 4 powershell
0 8 1488 3032 0.06 2488 4 services
0 2 288 504 0.00 4508 0 smss
0 8 1600 3004 0.03 908 4 svchost
0 12 1492 3504 0.06 4572 4 svchost
0 15 20284 23428 5.64 4628 4 svchost
0 15 3704 7536 0.09 4688 4 svchost
0 28 5708 6588 0.45 4712 4 svchost
0 10 2028 4736 0.03 4840 4 svchost
0 11 5364 4824 0.08 4928 4 svchost
0 0 128 136 37.02 4 0 System
0 7 920 1832 0.02 3752 4 wininit
0 8 5472 11124 0.77 5568 4 WmiPrvSE
```





# Hyper-V Containers



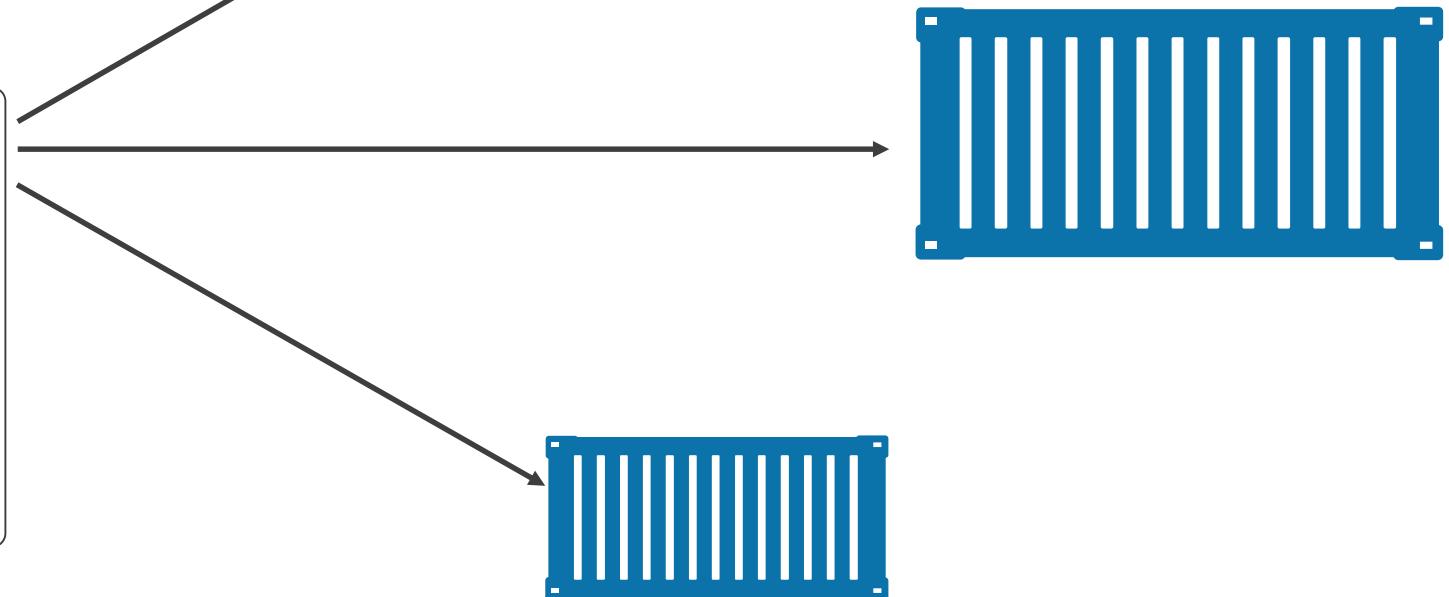
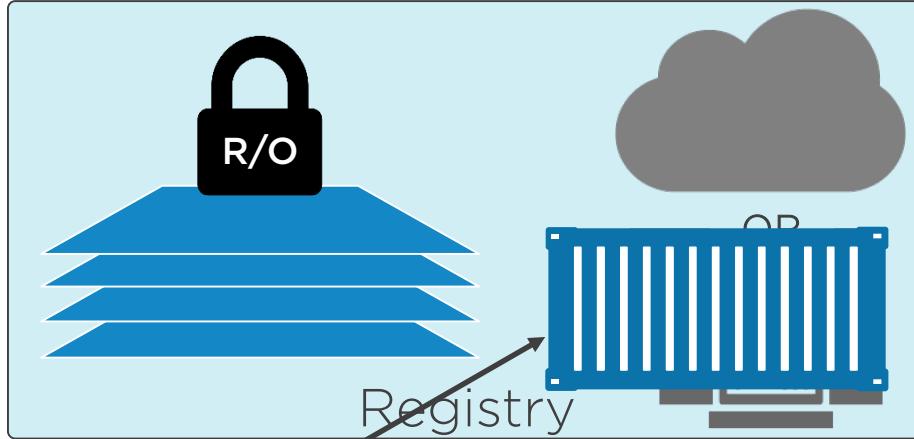
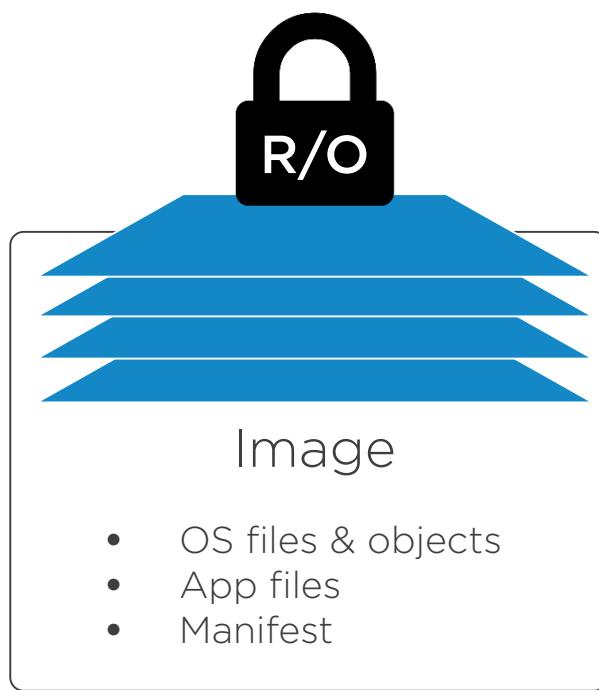
**Based on Hyper-V Virtual Machines**

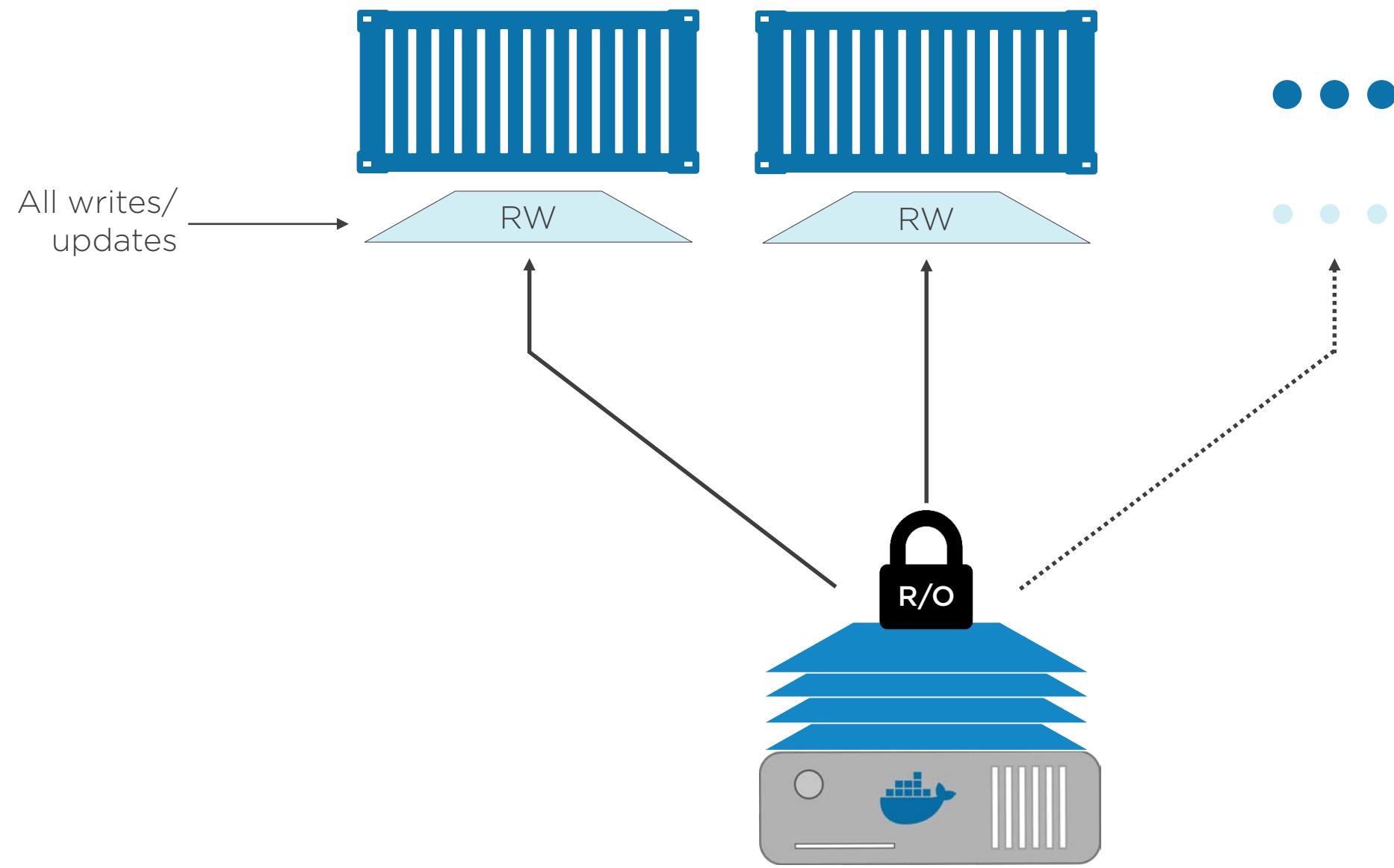
**Highly Optimized**

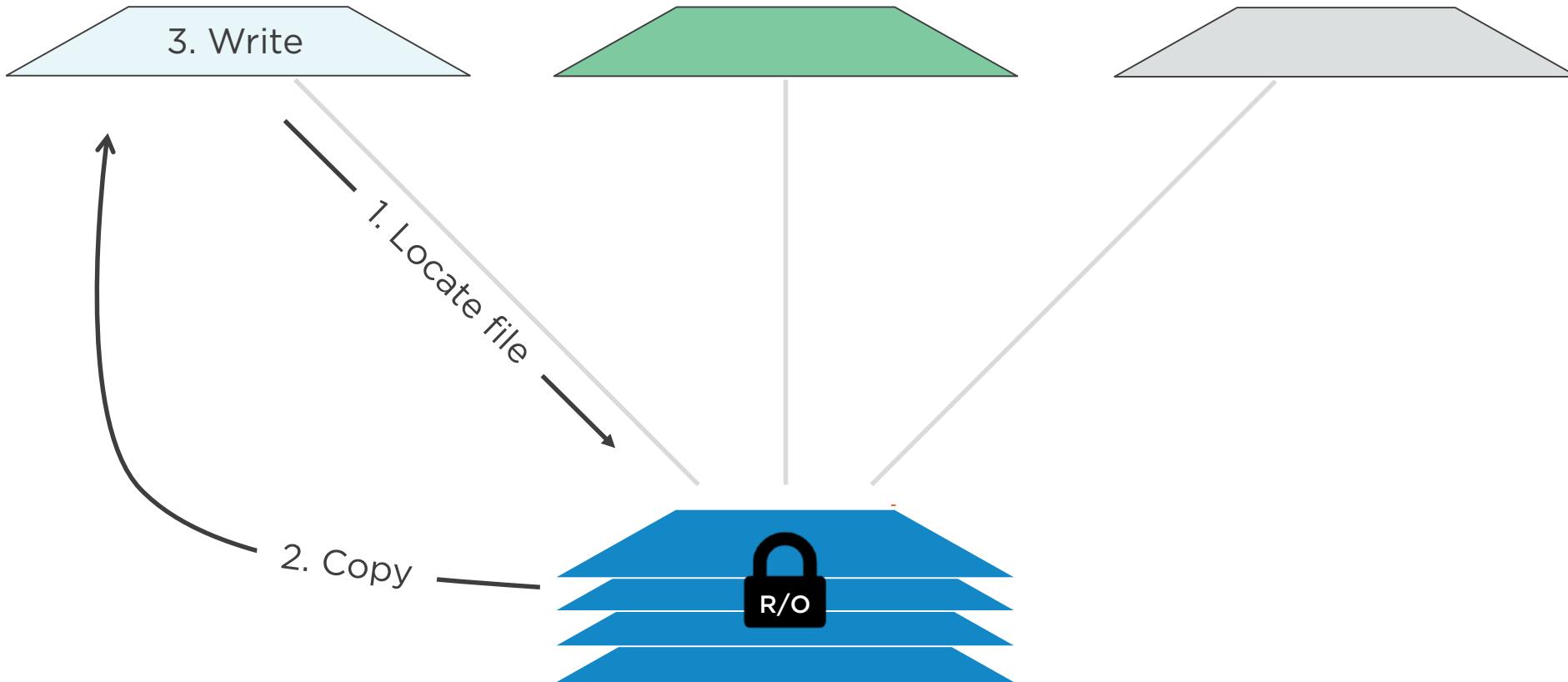
**Allow older images to work on newer versions of Windows**

**Resource utilization worse than Windows Server Containers**









## Image Manifest

```
{
  "schemaVersion": 2,
  "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
  "config": {
    "mediaType": "application/vnd.docker.container.image.v1+json",
    "size": 7023,
    "digest": "sha256:b5b2b2c507a0944348e0303114d8d93aaaa081732b86451d9bc1f432a537bc?"
  },
  "layers": [
    {
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
      "size": 32654,
      "digest": "sha256:e692418e4cbaf90ca69d05a66403747baa33ee08806650b51fab815ad7fc3"
    },
    {
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
      "size": 16724,
      "digest": "sha256:3c3a4604a545cdc127456d94e421cd355bca5b528f4a9c1905b15da2eb4a4c"
    },
    {
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
      "size": 73189,
      "digest": "sha256:ec4b8955958665577945c89419d1af06b5f7636b4ac3da7f12184802ad867"
    },
    {
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
      "size": 73109,
      "digest": "sha256:ec4b8955958665577945c89419d1af06b5f7636b4ac3da7f12184802ad867"
    }
  ]
}
```

Files & objects



Files & objects



Files & objects



Files & objects



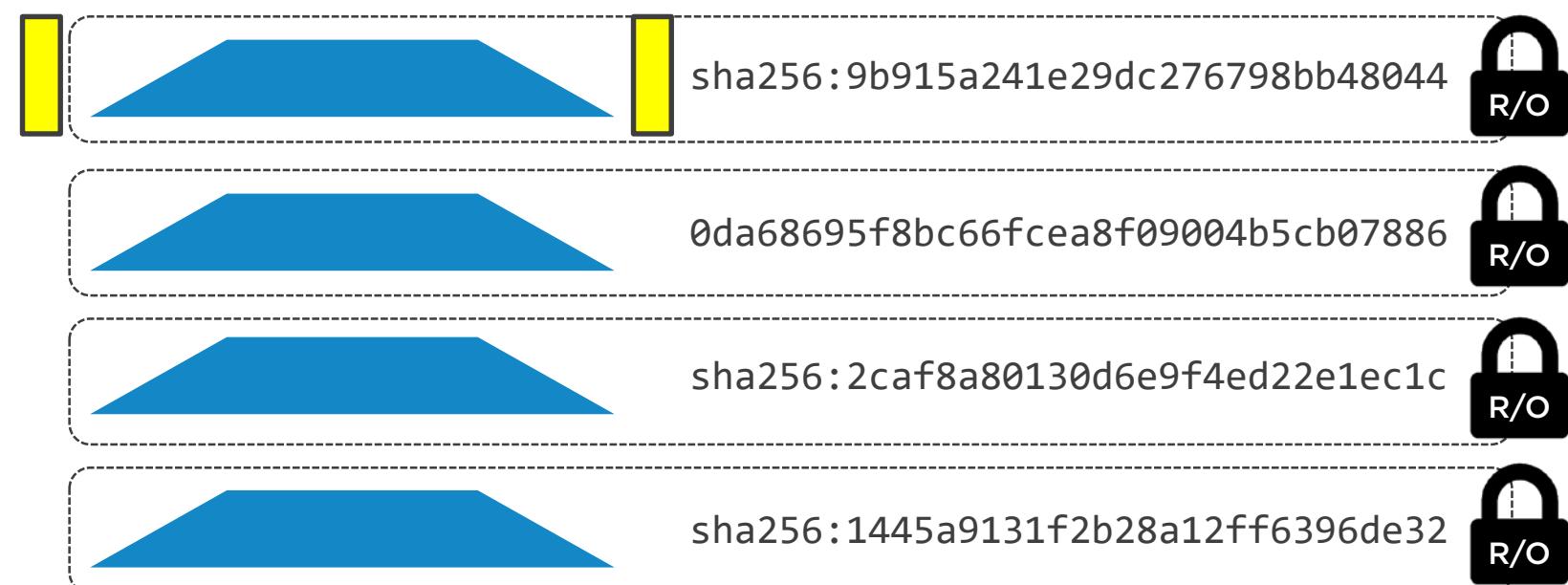
Layers

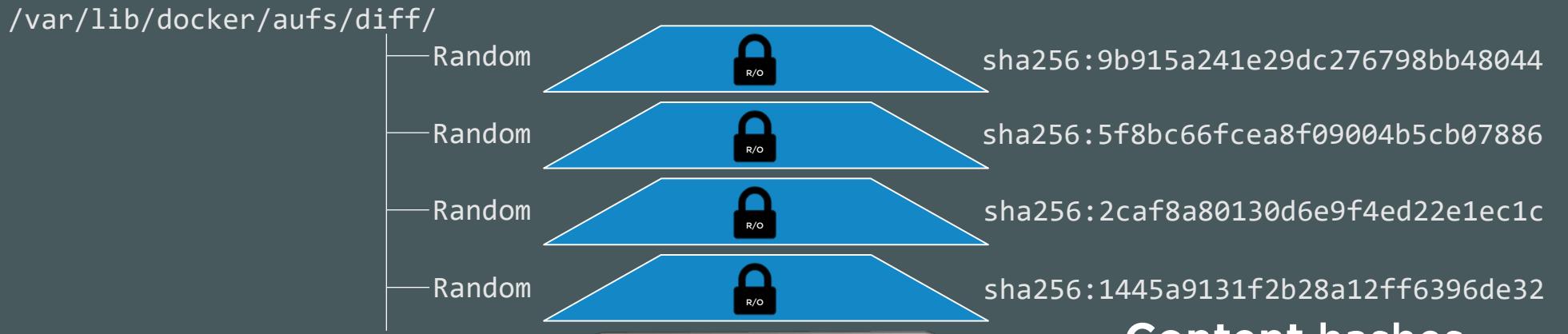
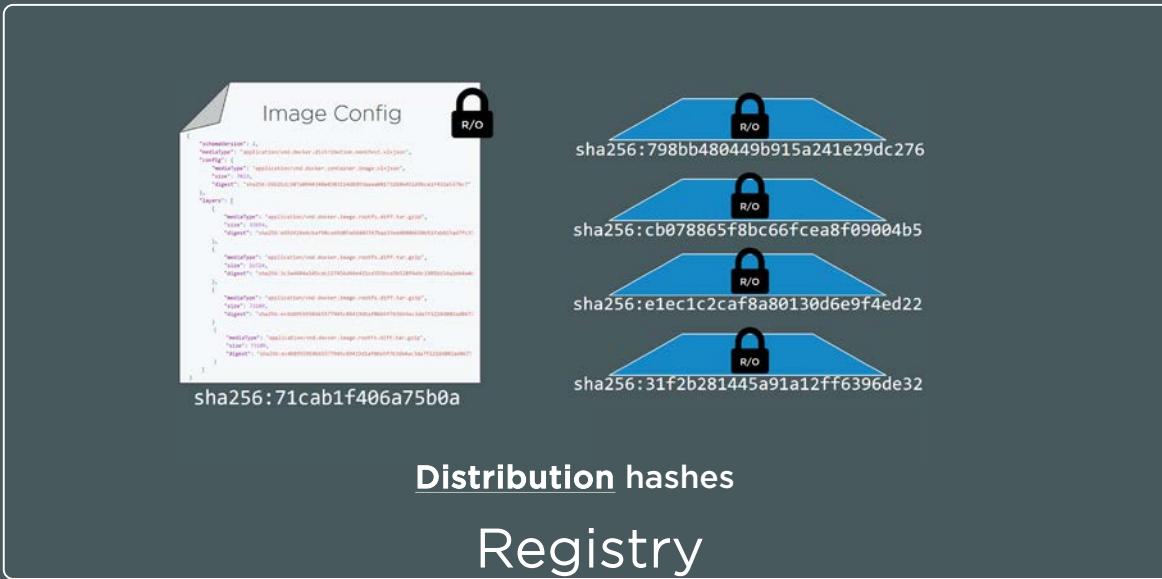


## Image Config

```
{  
  "schemaVersion": 2,  
  "mediaType": "application/vnd.docker.distribution.manifest.v2+json",  
  "config": {  
    "mediaType": "application/vnd.docker.container.image.v1+json",  
    "size": 7023,  
    "digest": "sha256:b5b2b2c507a0944348e0303114d8d93aaaa081732b86451d9bce1f432a537bc7"  
  },  
  "layers": [  
    {  
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",  
      "size": 32654,  
      "digest": "sha256:e692418e4cbaf90ca69d05a66403747baa33ee08806650b51fab815ad7fc3"  
    },  
    {  
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",  
      "size": 16724,  
      "digest": "sha256:3c3a4604a545cdc127456d94e421cd355bca5b528f4a9c1905b15da2eb4a4c"  
    },  
    {  
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",  
      "size": 73109,  
      "digest": "sha256:ec4b8955958665577945c89419d1af06b5f7636b4ac3da7f12184802ad8677"  
    },  
    {  
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",  
      "size": 73109,  
      "digest": "sha256:ec4b8955958665577945c89419d1af06b5f7636b4ac3da7f12184802ad867"  
    }  
  ]  
}
```

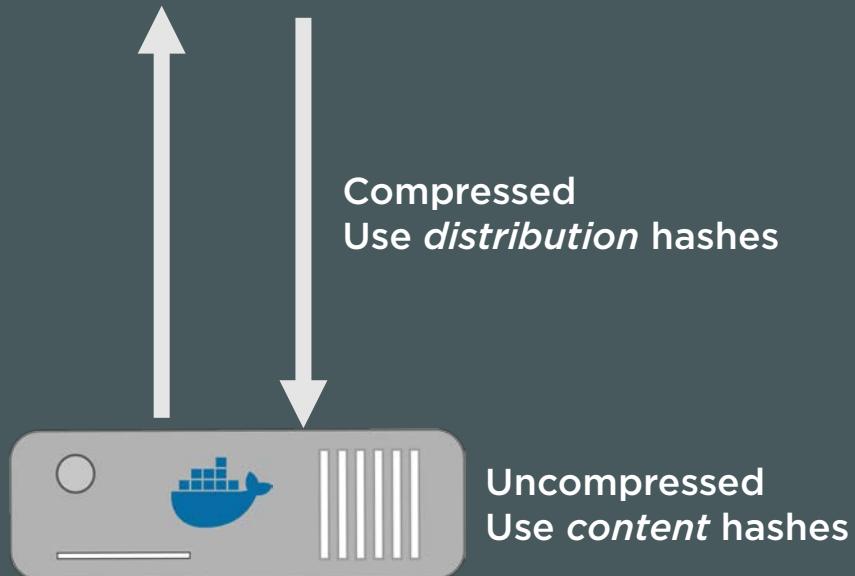
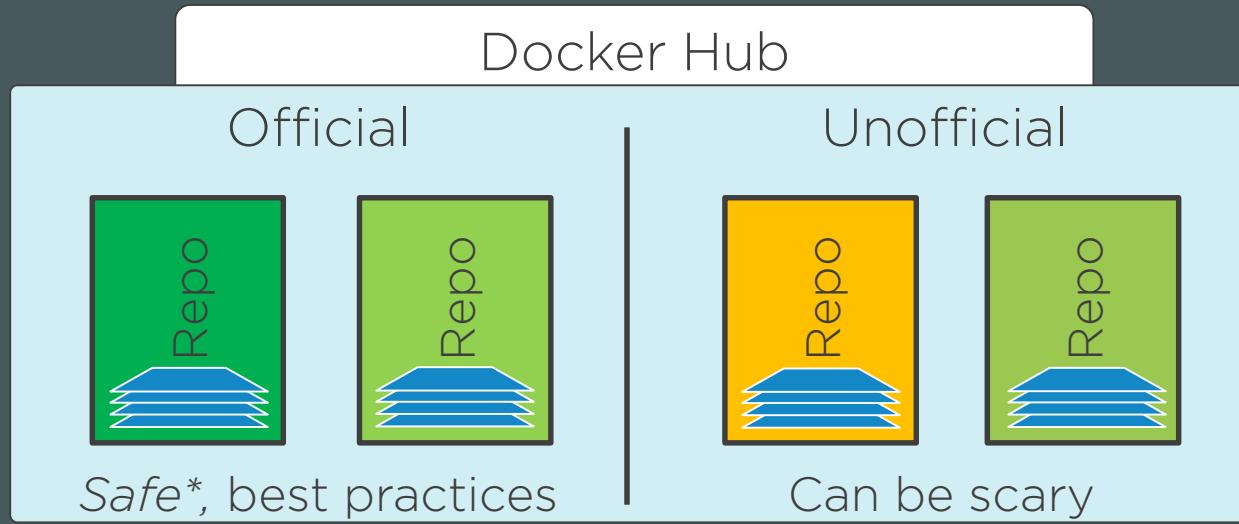
sha256:d124781fc06a73b05

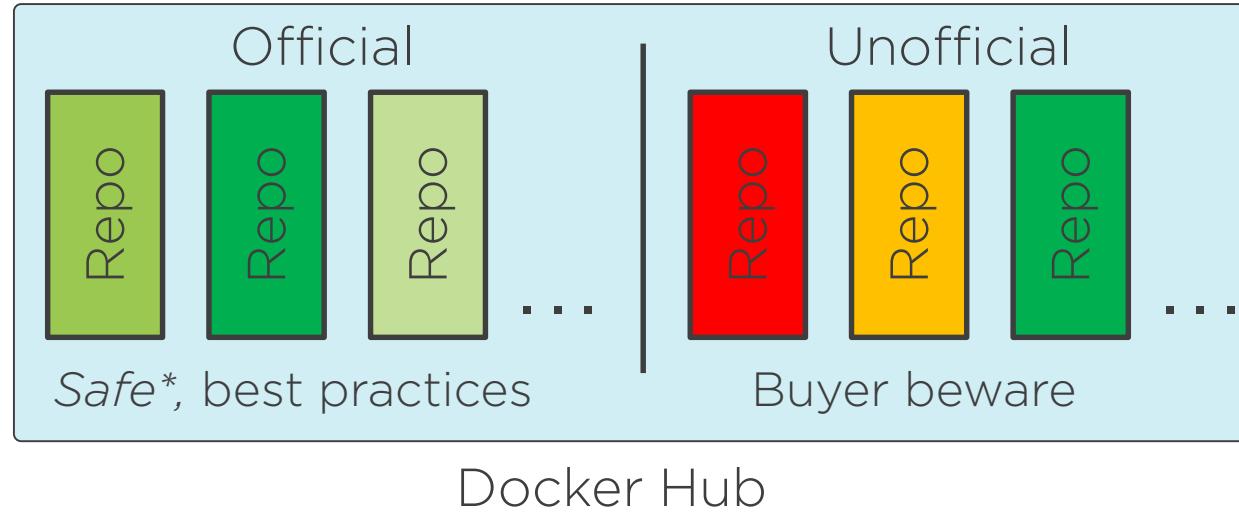




**Content hashes**







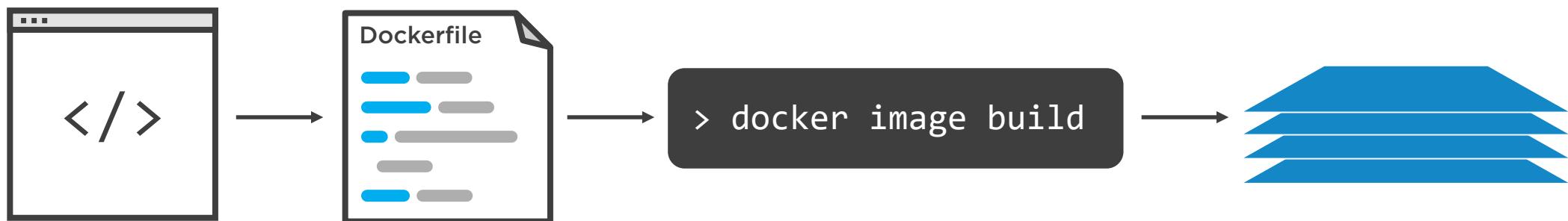
```
$ docker image push
```

```
$ docker image pull
```

```
$ docker image inspect
```

```
$ docker image rm
```





# Key Dockerfile Instructions

**FROM**

**LABEL**

(ex: maintainer/author)

**RUN**

**COPY**

**ENTRYPOINT**

**WORKDIR**

**EXPOSE**

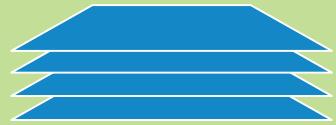
**ENV**

**VOLUME**



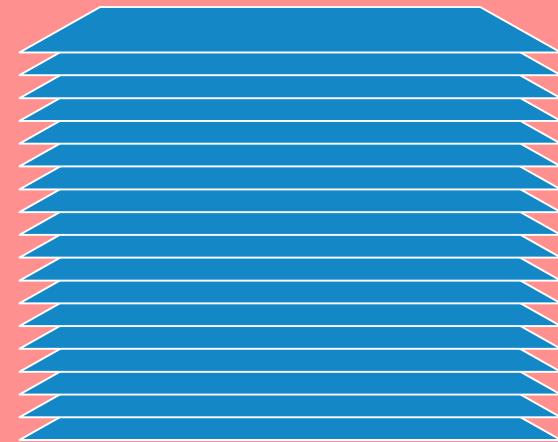


## Small Images



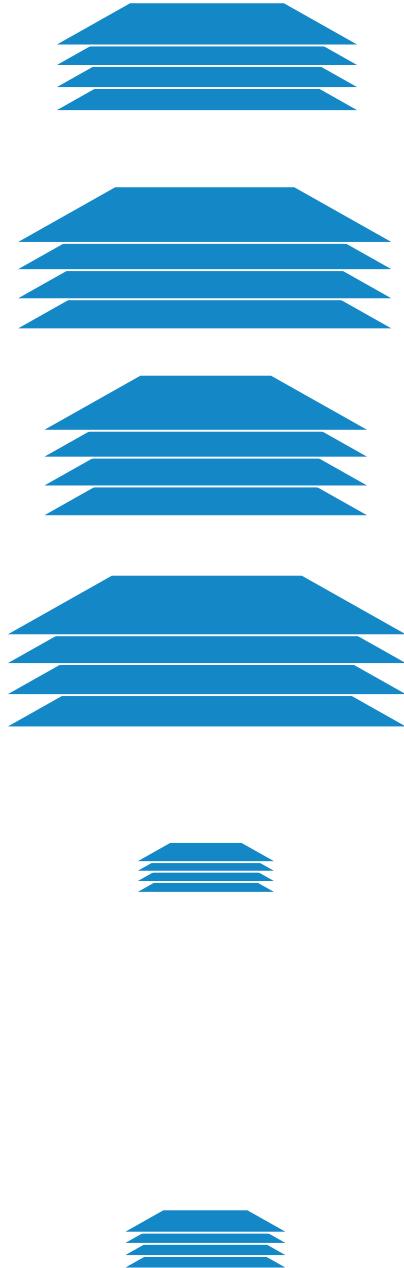
Minimal OS  
and packages

## Bloated Images



Too much OS  
Too many packages





```
1 FROM node:latest AS storefront
2 WORKDIR /usr/src/atsea/app/react-app
3 COPY react-app .
4 RUN npm install
5 RUN npm run build
6
7 FROM maven:latest AS appserver
8 WORKDIR /usr/src/atsea
9 COPY pom.xml .
10 RUN mvn -B -f pom.xml -s /usr/share/maven/ref/settings-docker.xml dependency:resolve
11 COPY . .
12 RUN mvn -B -s /usr/share/maven/ref/settings-docker.xml package -DskipTests
13
14 FROM java:8-jdk-alpine AS production
15 RUN adduser -Dh /home/gordon gordon
16 WORKDIR /static
17 COPY --from=storefront /usr/src/atsea/app/react-app/build/ .
18 WORKDIR /app
19 COPY --from=appserver /usr/src/atsea/target/AtSea-0.0.1-SNAPSHOT.jar .
20 ENTRYPOINT ["java", "-jar", "/app/AtSea-0.0.1-SNAPSHOT.jar"]
21 CMD ["--spring.profiles.active=postgres"]
```

# Dockerfile

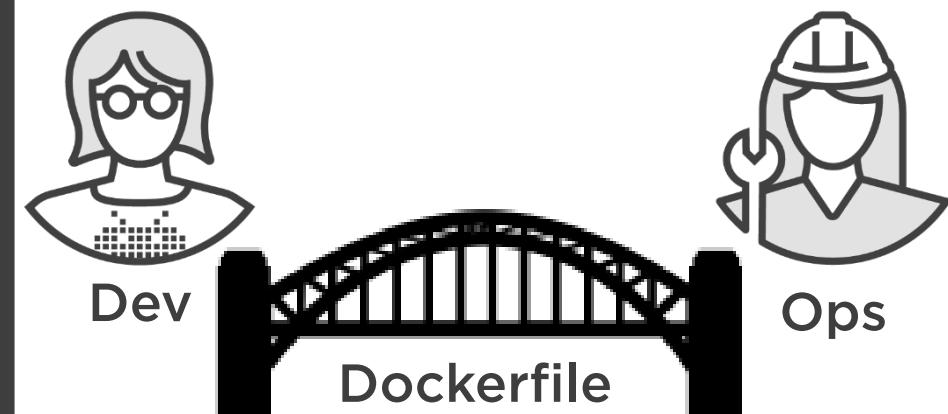
```
FROM ... AS Build-Stage-0
```



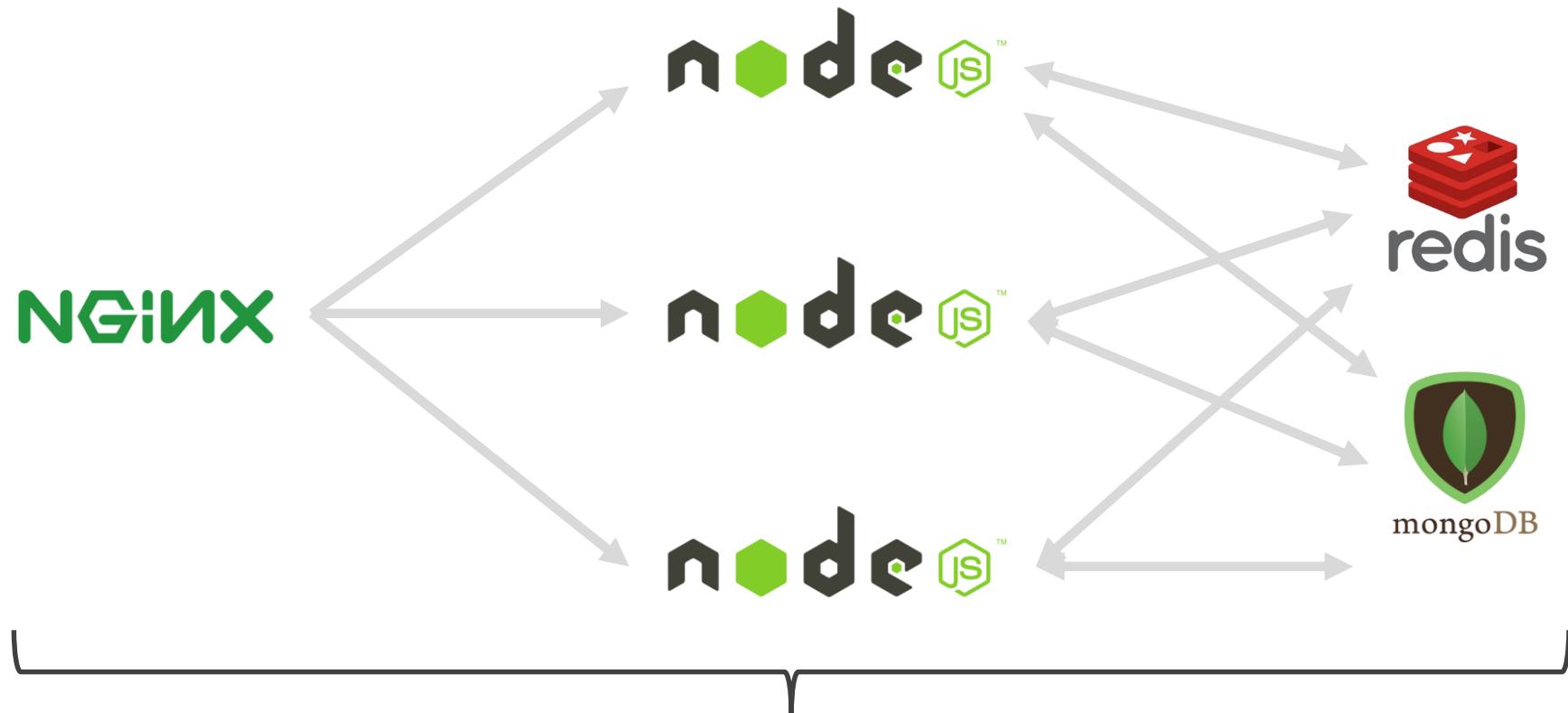
```
FROM ... AS Build-Stage-1
```



```
FROM ... AS Production-Stage
```



# The Need For Docker Compose



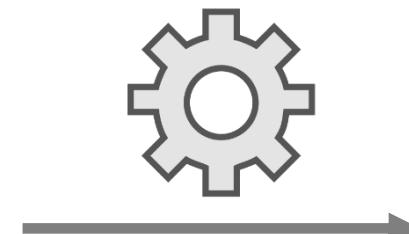
Docker Compose  
(`docker-compose.yml`)



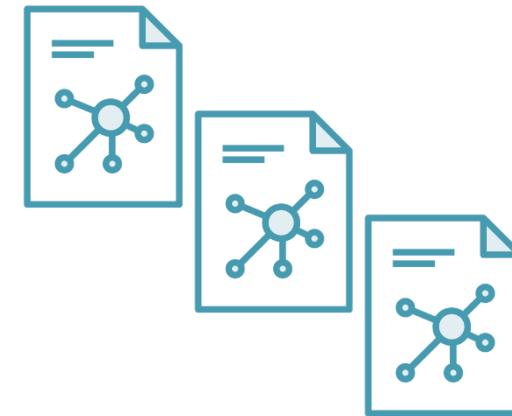
# The Role of the Docker Compose File



**docker-compose.yml**  
**(service configuration)**



**Docker Compose  
Build**



**Docker Images  
(services)**



# Key Service Configuration Options

**build**

**environment**

**image**

**networks**

**ports**

**volumes**



# docker-compose.yml Example

```
version: "3.x"
services:
  node:
    build:
      context: .
      dockerfile: node.dockerfile
    networks:
      -nodeapp-network

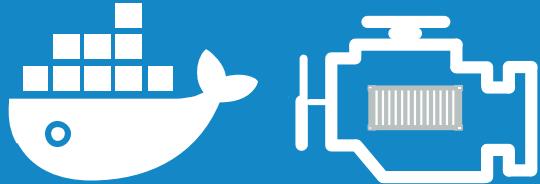
  mongodb:
    image: mongo
    networks:
      - nodeapp-network

networks:
  nodeapp-network
  driver: bridge
```



# Logging

## Engine/daemon



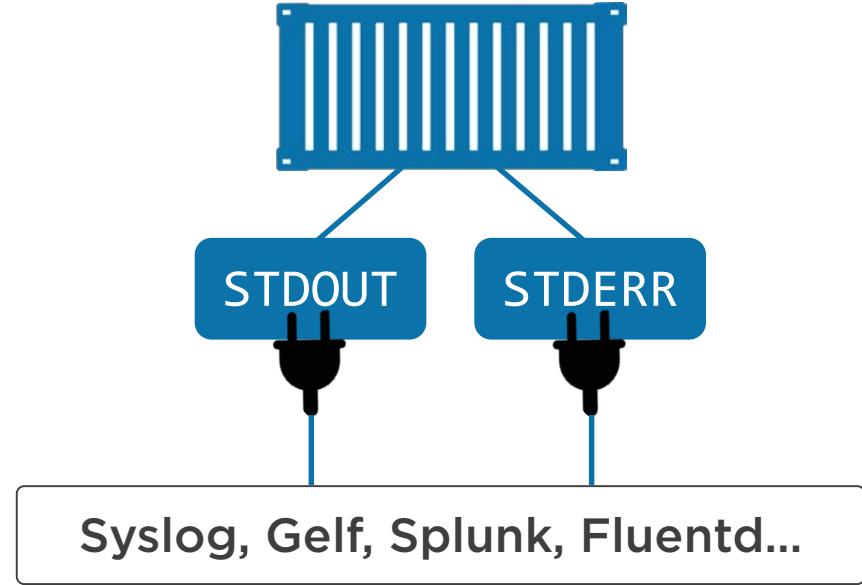
### Linux

- **systemd:**  
`journalctl -u docker.service`
- **Non-systemd:**  
Try `/var/log/messages`

### Windows:

`~/AppData/Local/Docker`

## Container/App

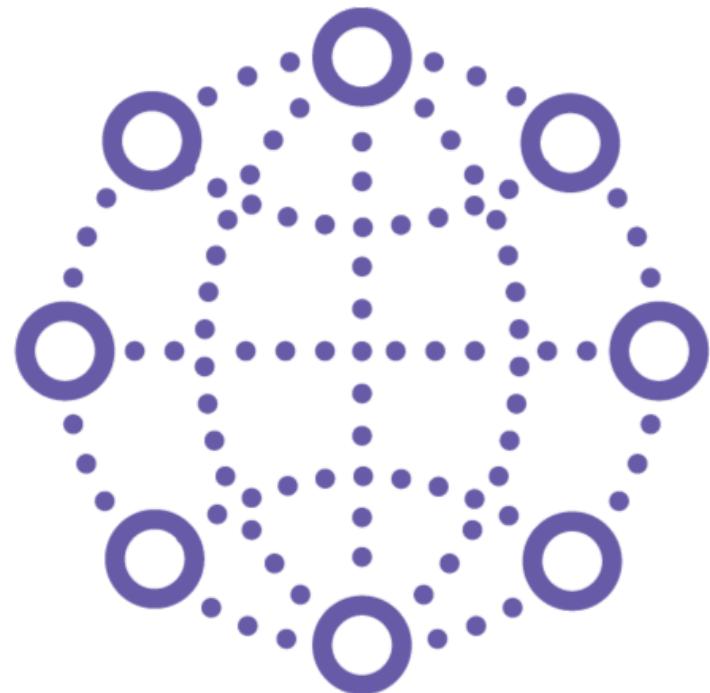


**Set default logging driver in `daemon.json`**

**Override per-container with  
`--log-driver --log-opt`**

**Inspect logs with `docker logs <container>`**  
*- Doesn't work with all drivers*

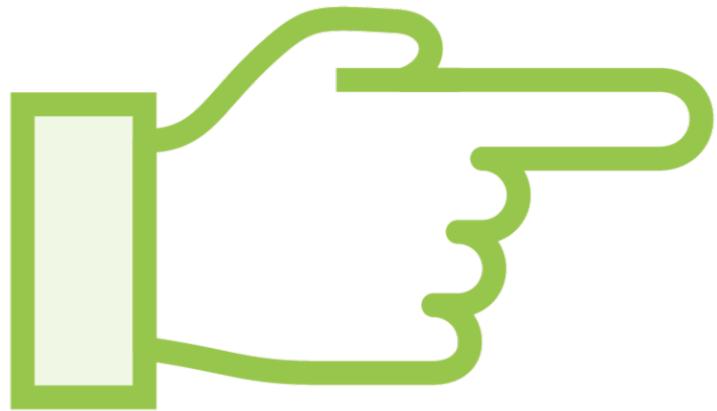
# Orchestration



- Deploying at scale**
- Scheduling workload**
- Monitoring health**
- Automated failover**
- Service discovery**
- Coordinated upgrades**
- Cluster node affinity**



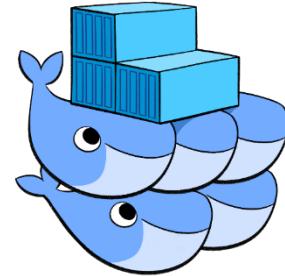
# The Benefits of Clusters



- Auto-scaling**
- Load-balancing**
- Abstracting individual machines**
- Scalability**
- Zero-downtime deployments**
- Deployment rollback**
- Simplified networking**
- High availability**



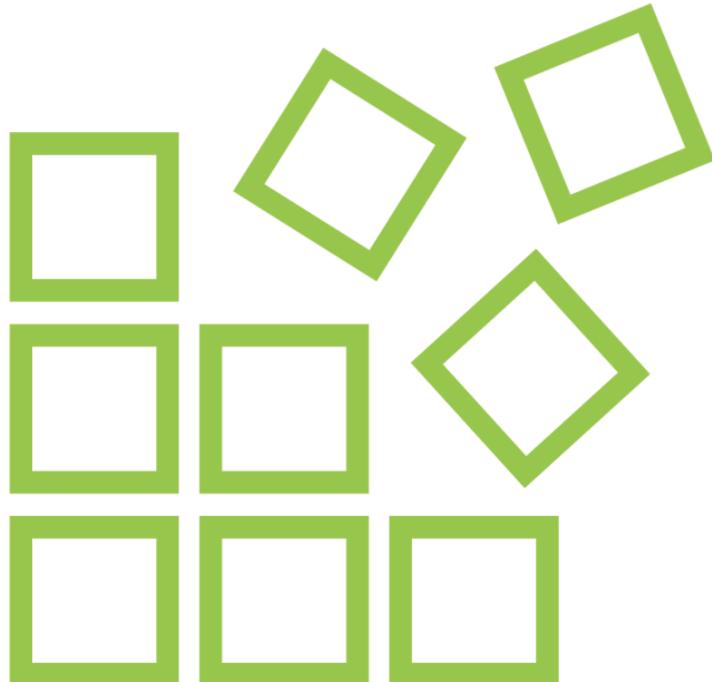
# Operating at scale is hard!



Here to help!



# Docker Swarm



**Created by Docker Inc.**

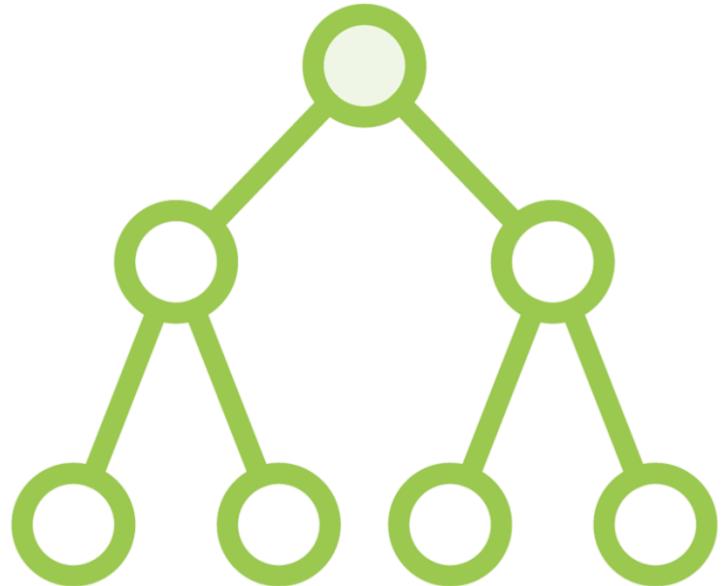
**Part of a default installation**

**Uses the same Docker CLI and API**

**Works with Docker Compose services**



# Kubernetes



**Created by Google**

**Based on Borg (internal tool used to run Google Search)**

**Introduces new concepts such as pods and deployments**

**Features own API and CLI**

**Several distributions and managed services exist**

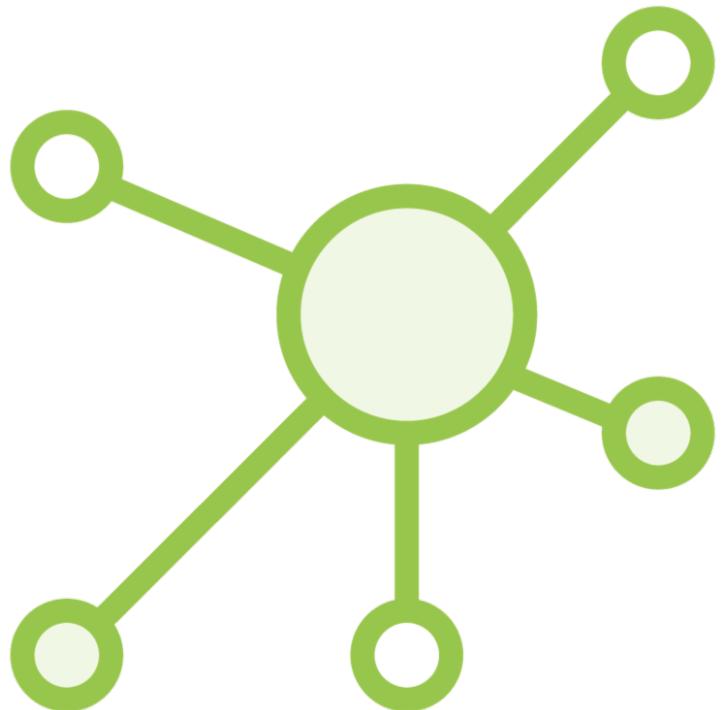


# Swarm vs. Kubernetes

	Swarm	Kubernetes
Easier to migrate from Docker Compose		Harder to migrate from Docker Compose
Easier to set up if you use Docker		Harder to set up from scratch
Less features		More features
Smaller community		Large community
Very few integrations and extensions		A lot of integrations and extensions
Less popular as a managed service		Popular as a managed service
Simple		Complex



# Mixed Linux/Windows Clusters



**Docker Swarm is compatible with Windows containers**

**Kubernetes control plane runs on Linux**

**Worker nodes can run Windows**

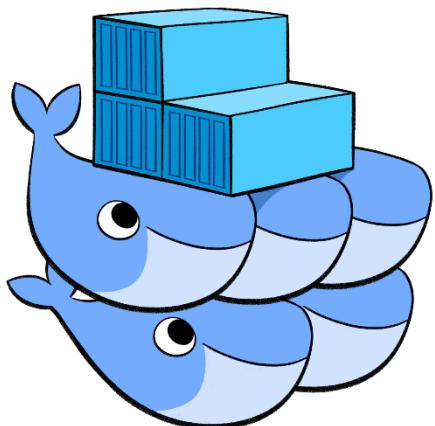
**Windows Server 2016 is unsupported**

**Different Windows SAC releases are supported by different versions of Kubernetes**



# Swarm

The future of Docker.

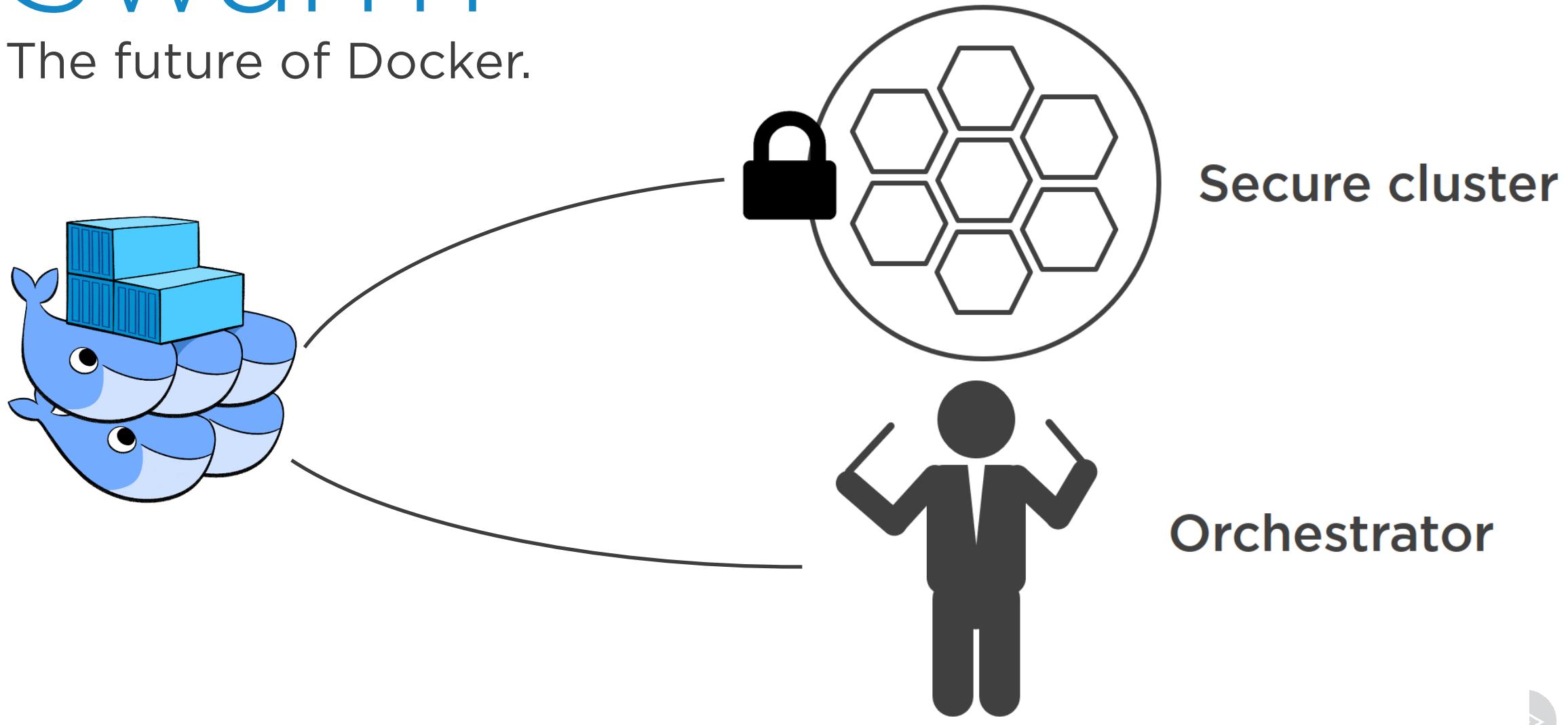


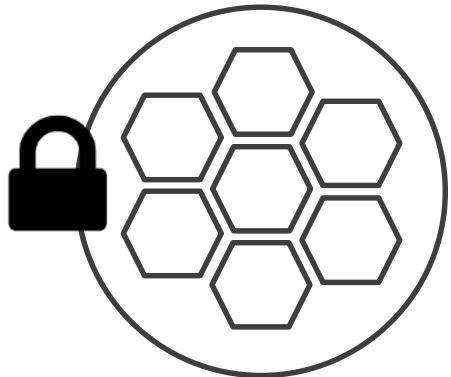
VS



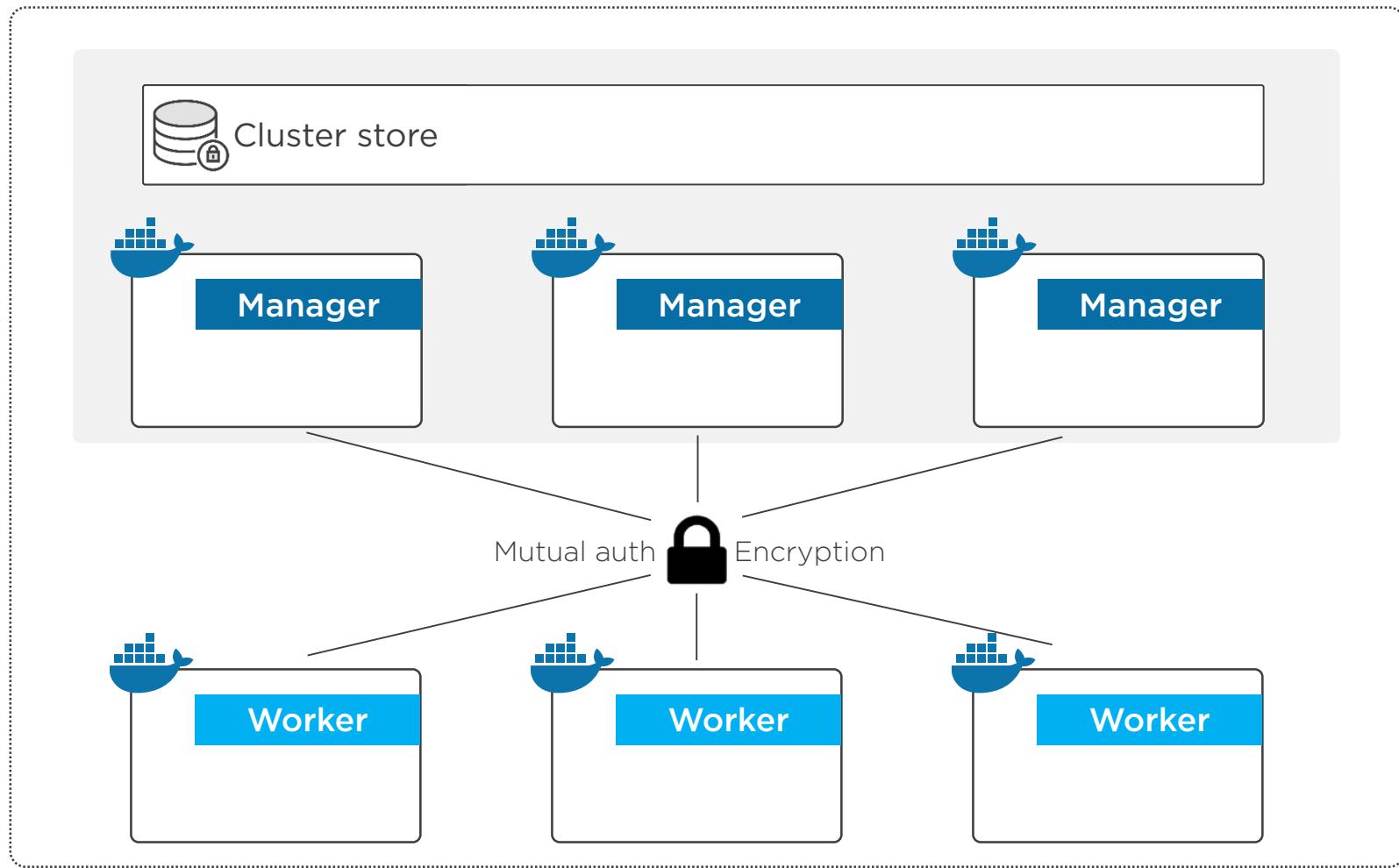
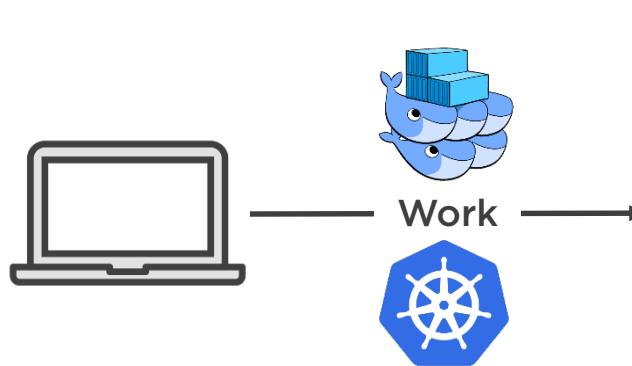
# Swarm

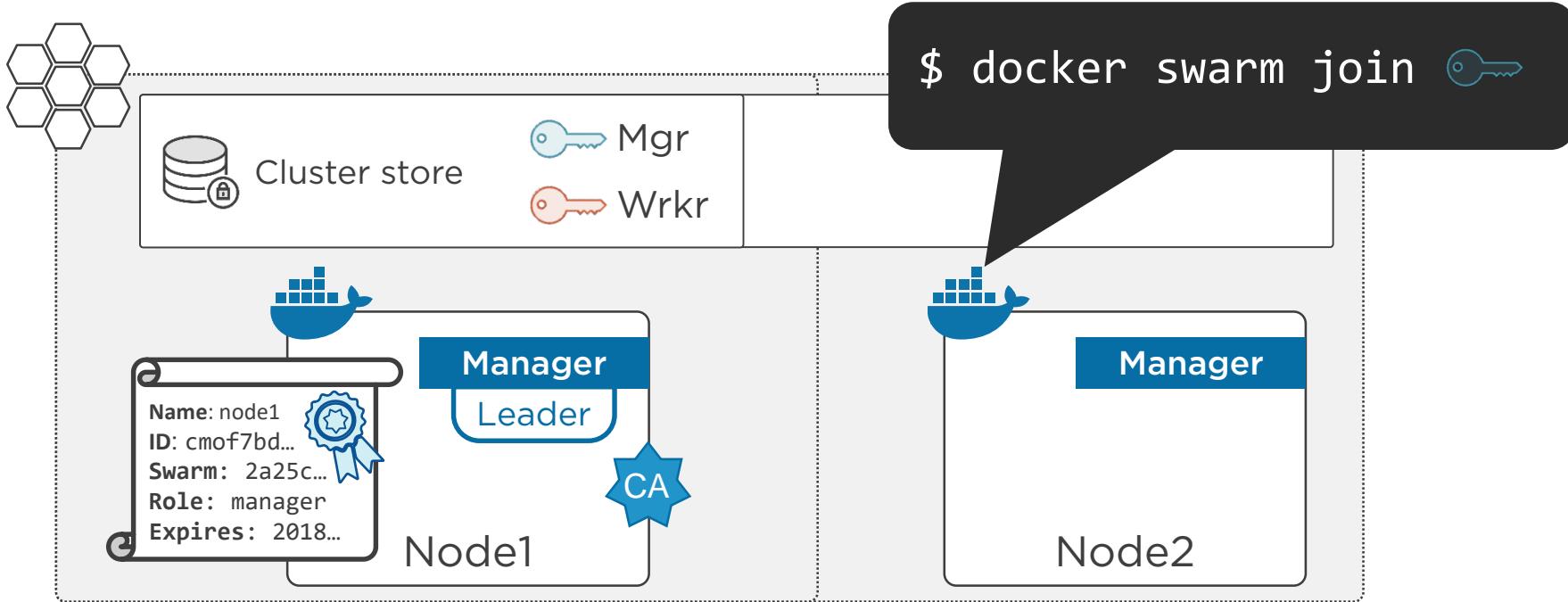
The future of Docker.

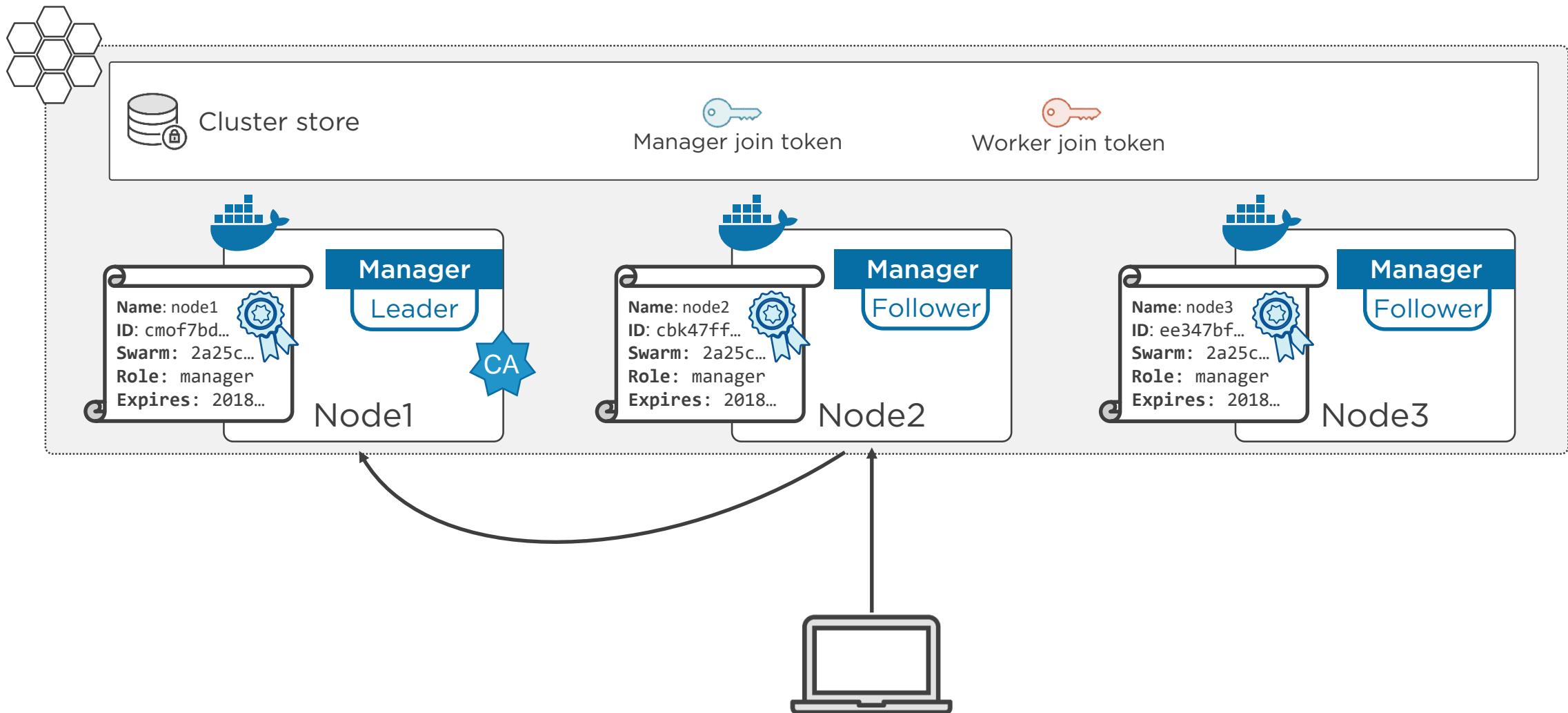


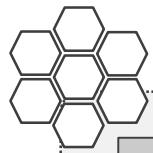


## Secure Swarm cluster

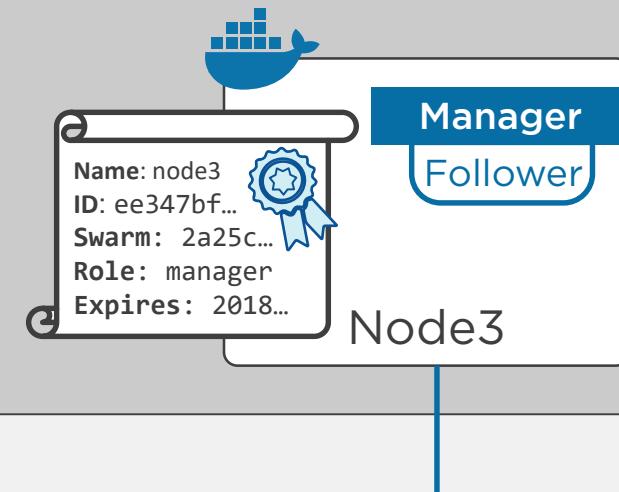
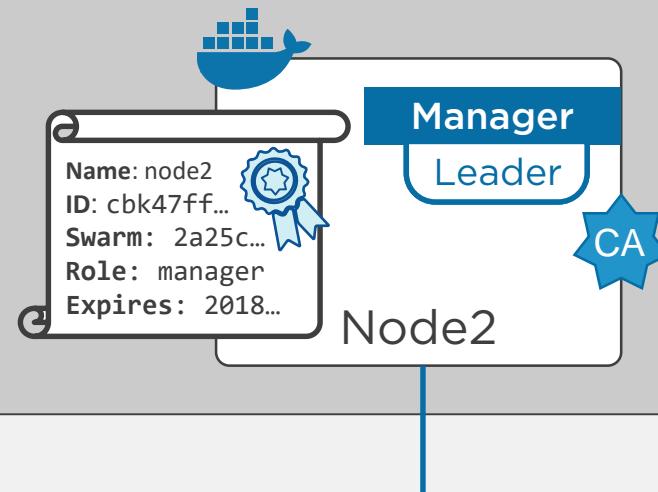
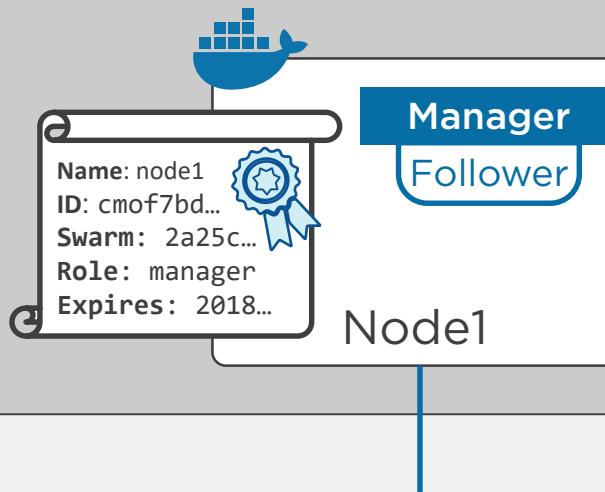
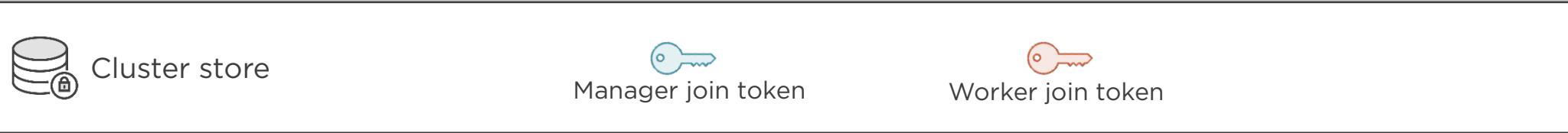






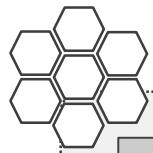


## Raft Consensus Group

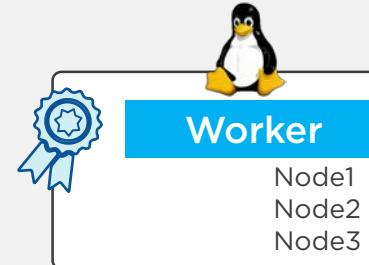
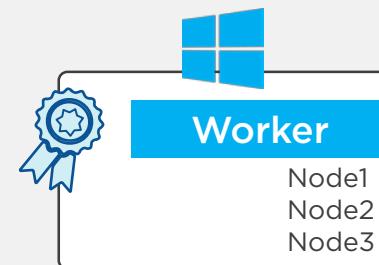
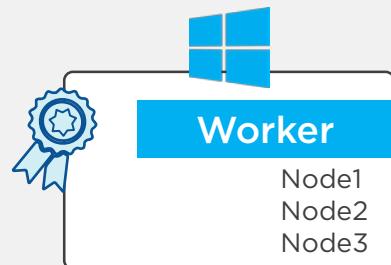
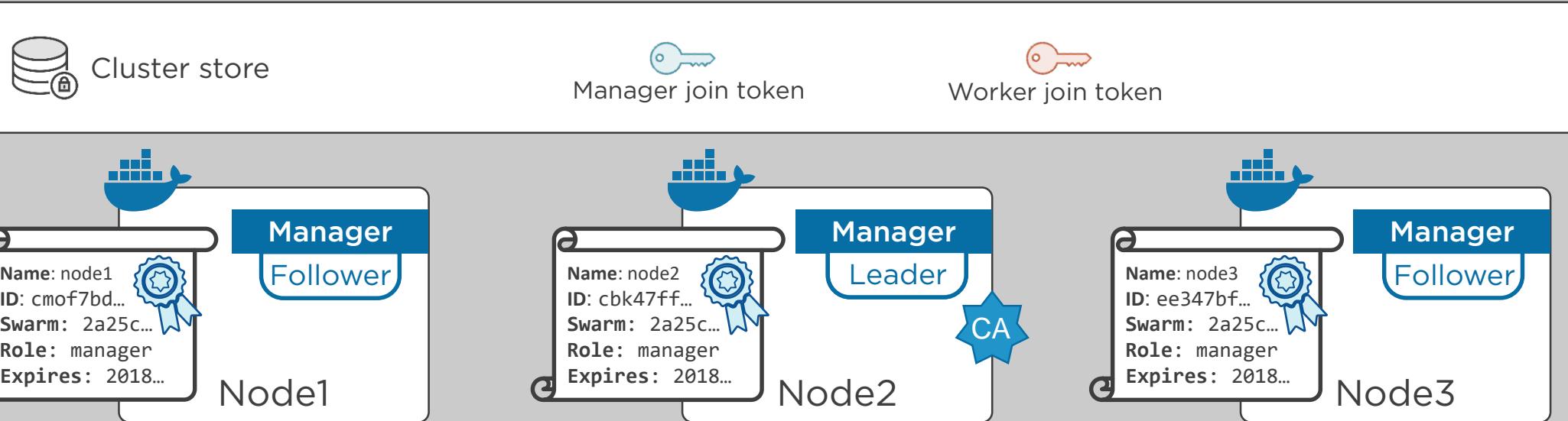


Fast and reliable network





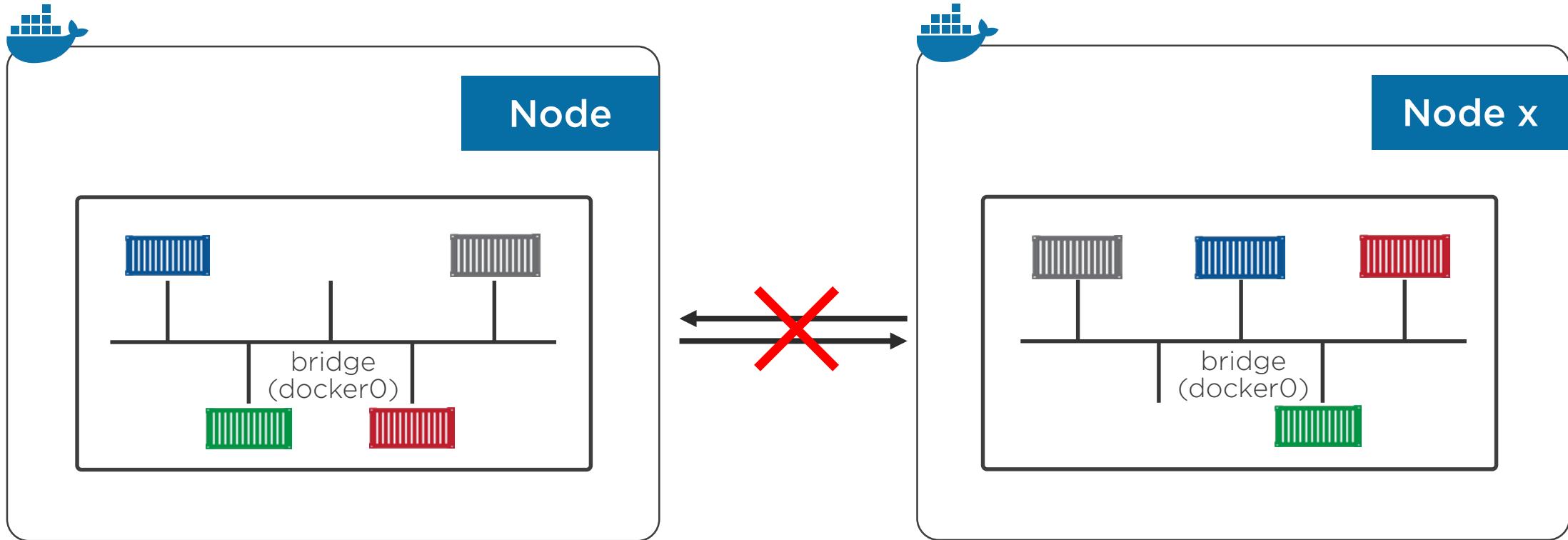
## Raft Consensus Group



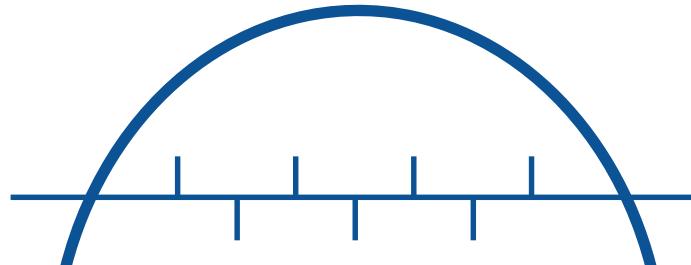
# Create a Swarm (anywhere)

- Create a Swarm manager
  - Assign it a crypto ID
  - Elect it as the Swarm leader
- Create an Swarm config DB
  - Encrypt it
  - Configure it to automatically replicate with all Swarm managers
- Create a Swarm join token for new workers
- Create a Swarm join token for new manager
- Configure a new Root CA on the leader
  - Configure a 90 day certificate rotation period

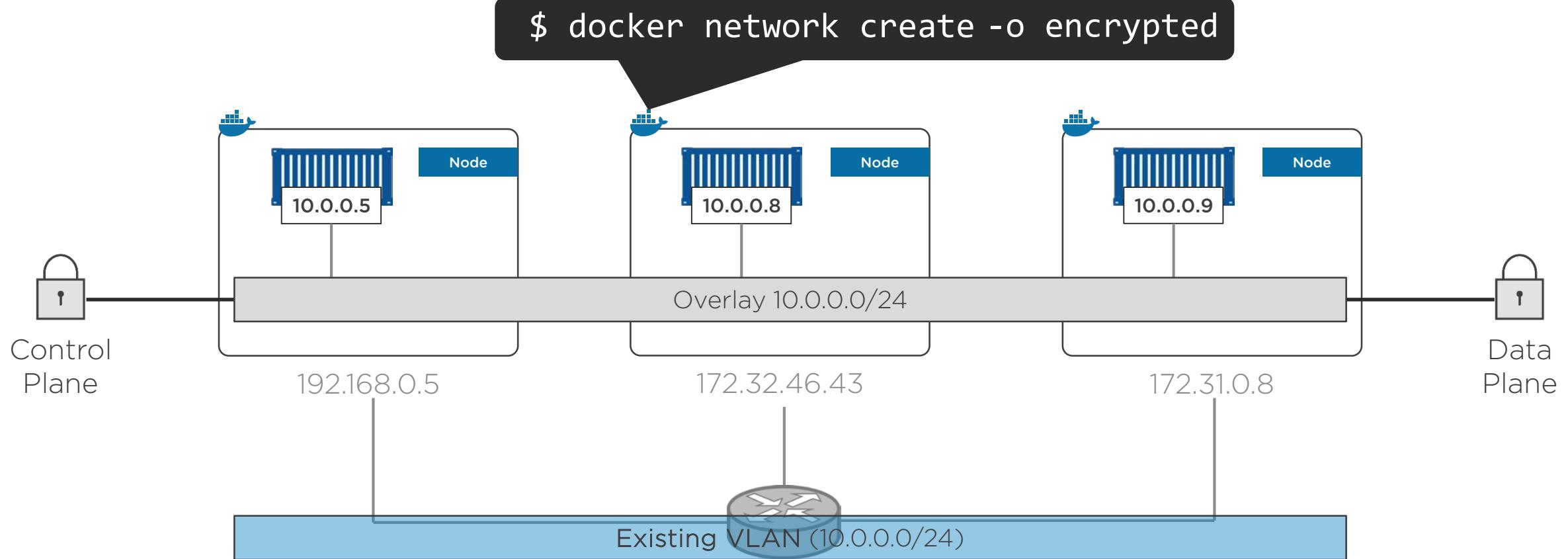
# Bridge Networking



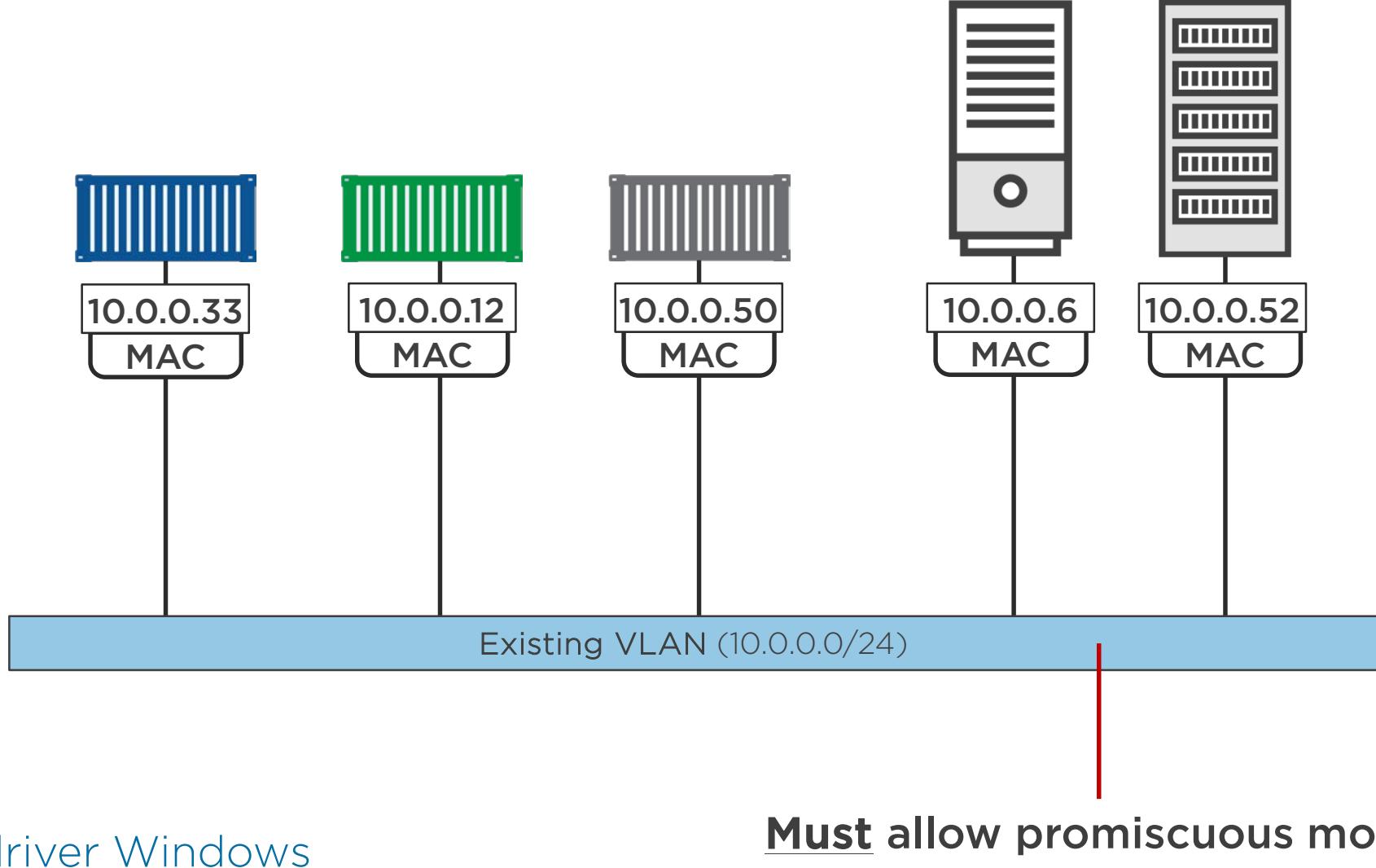
bridge driver Linux  
nat driver Windows



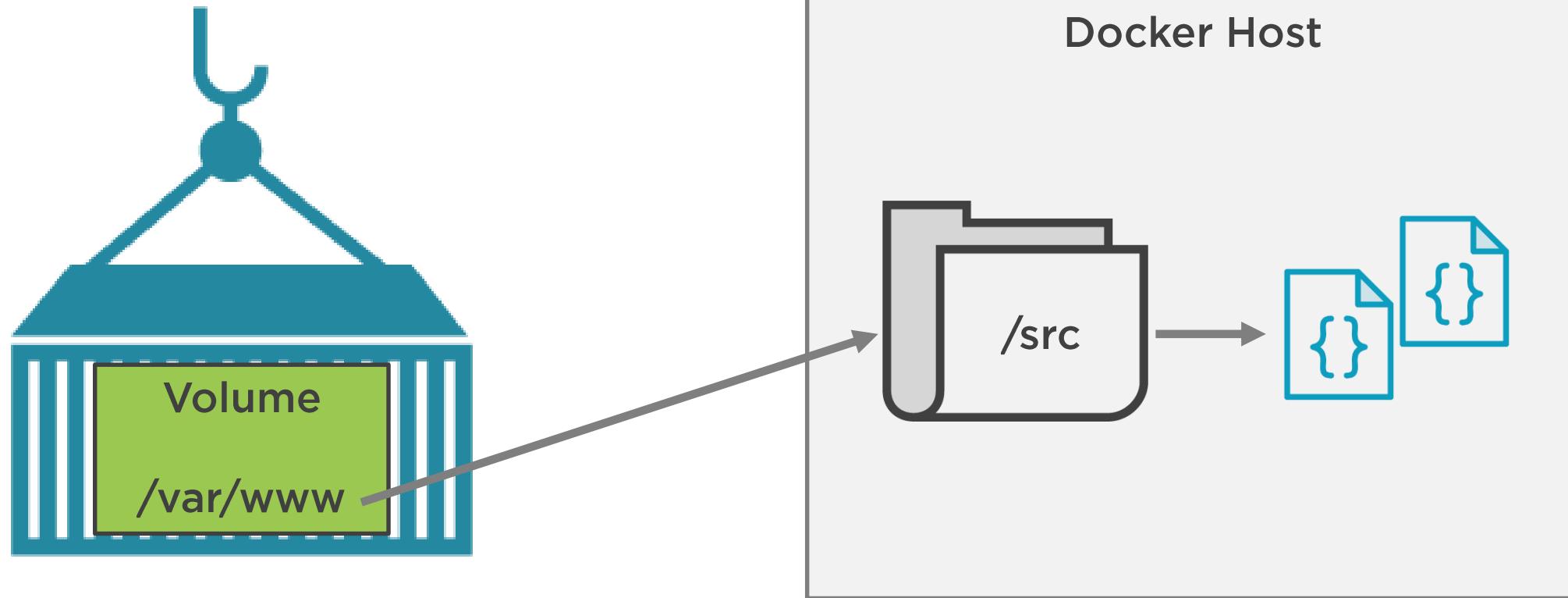
# Overlay Networking

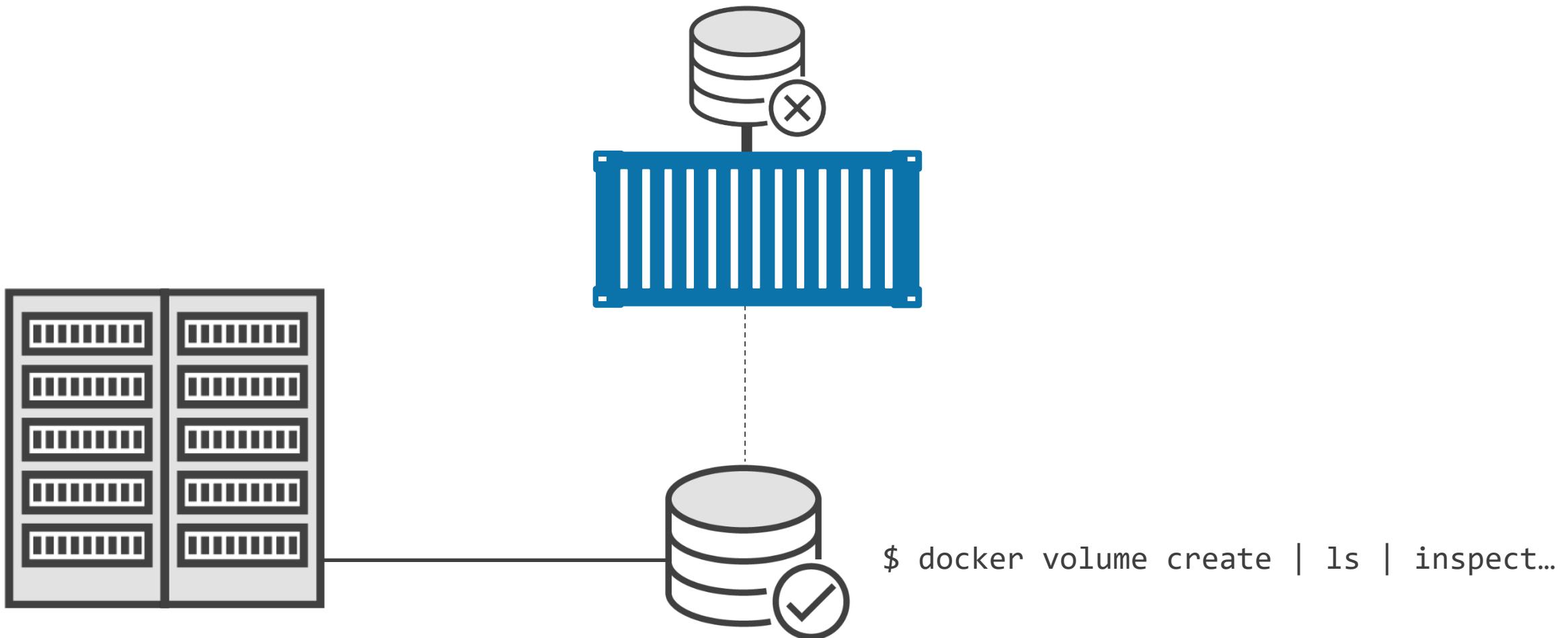


# MACVLAN



# Docker Volumes





# Docker Secrets

String. <=500K

Safe

Secure

Infrastructure independent

Requires *Swarm-mode*



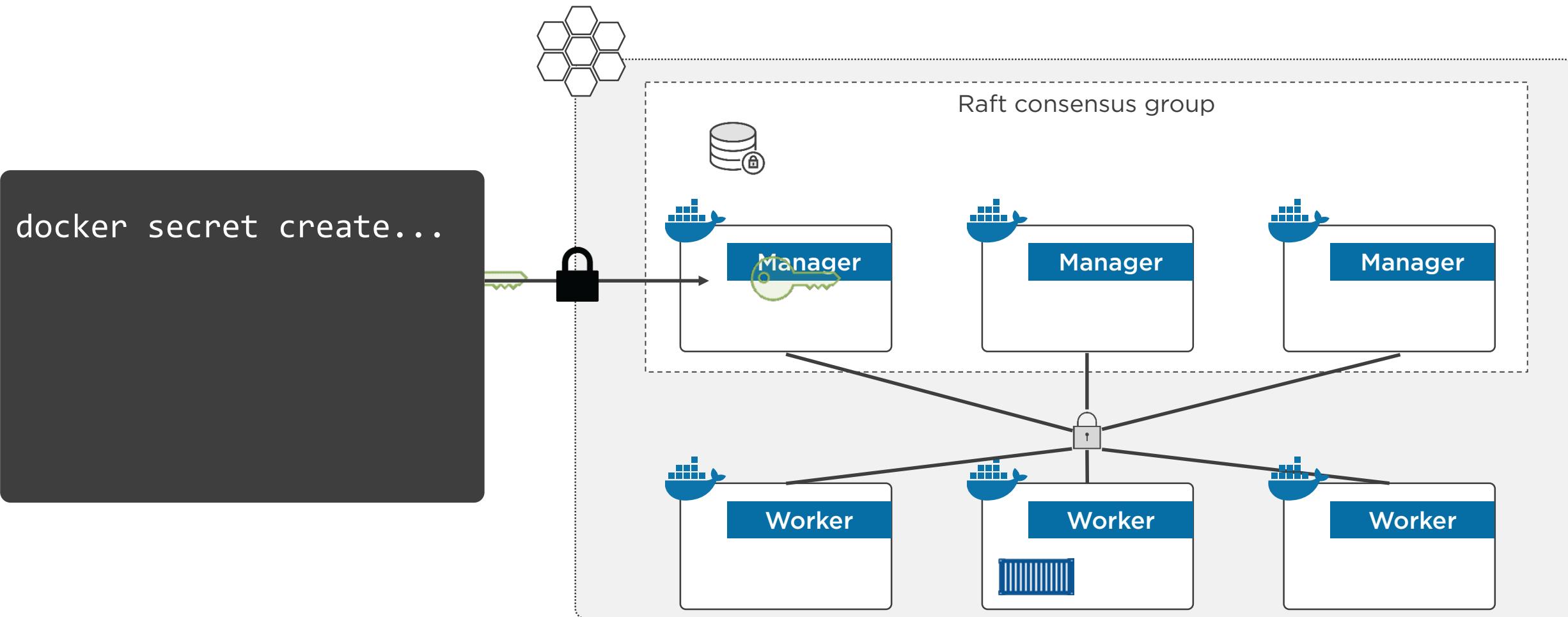
# Docker Secrets

Swarm-mode Services only

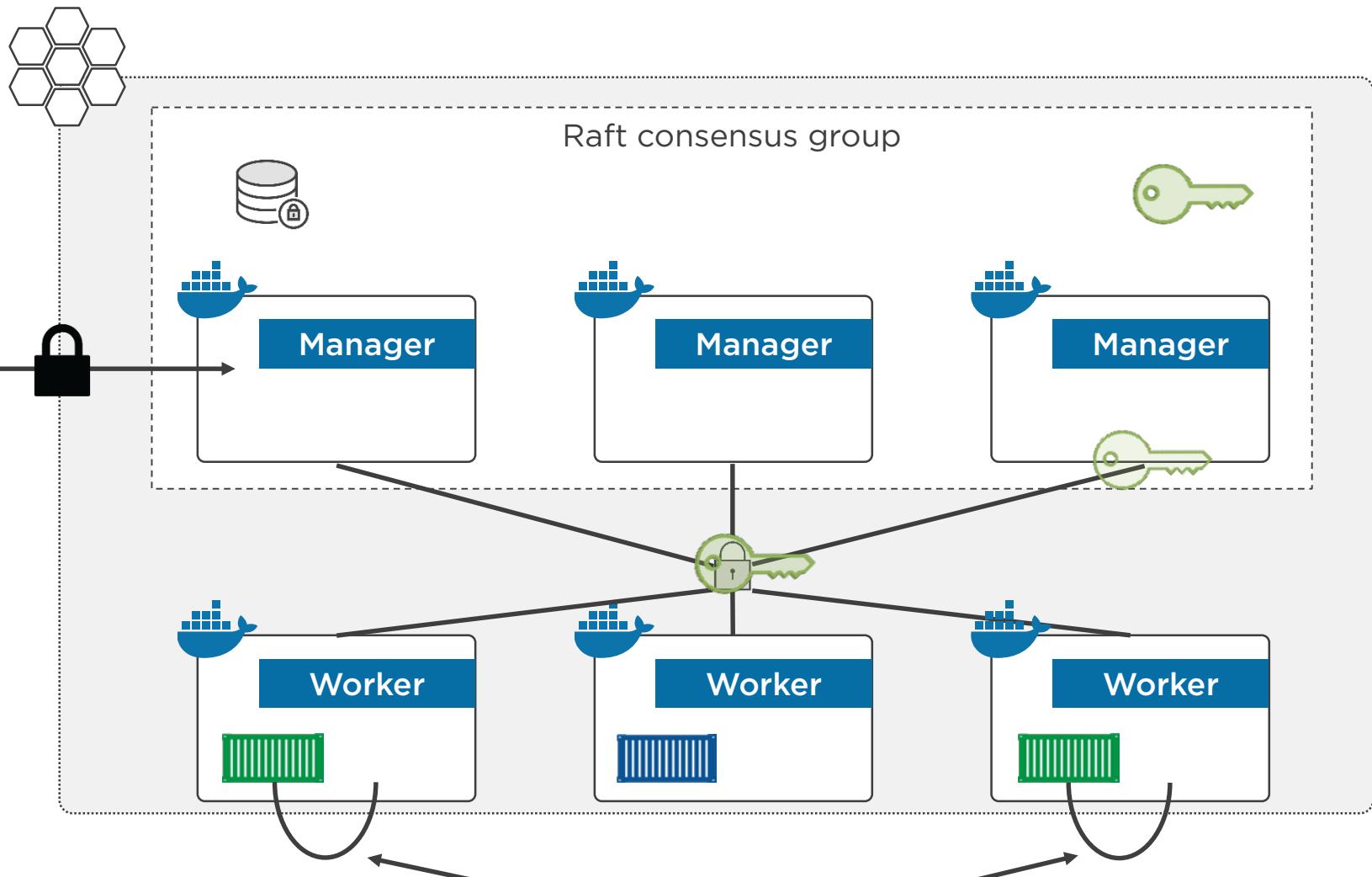
Linux: 1.13+

Windows: 17.06+



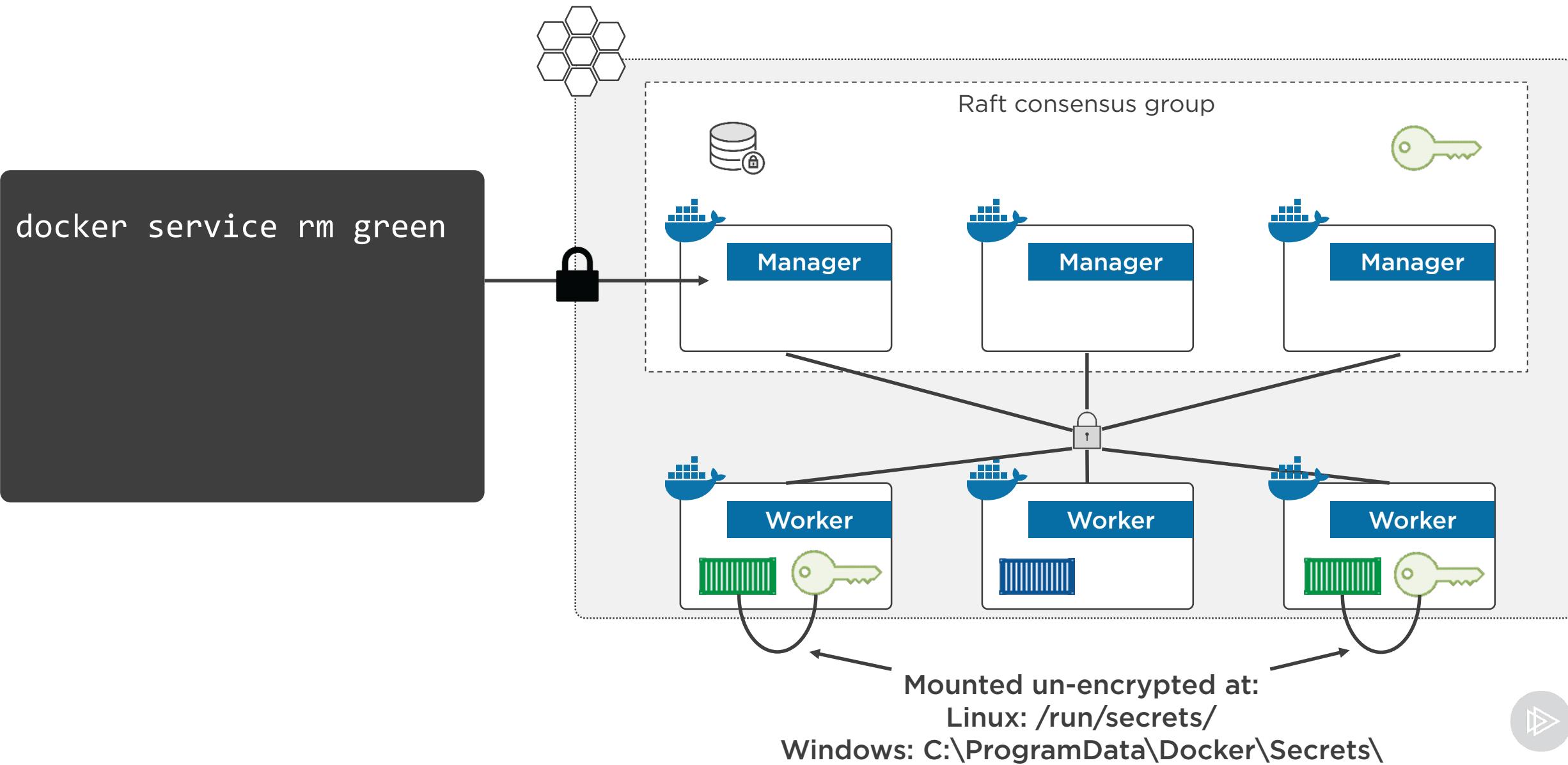


```
docker service create \  
  --name green \  
  --secret sec1 \  
  --replicas 2 \  
  ...
```

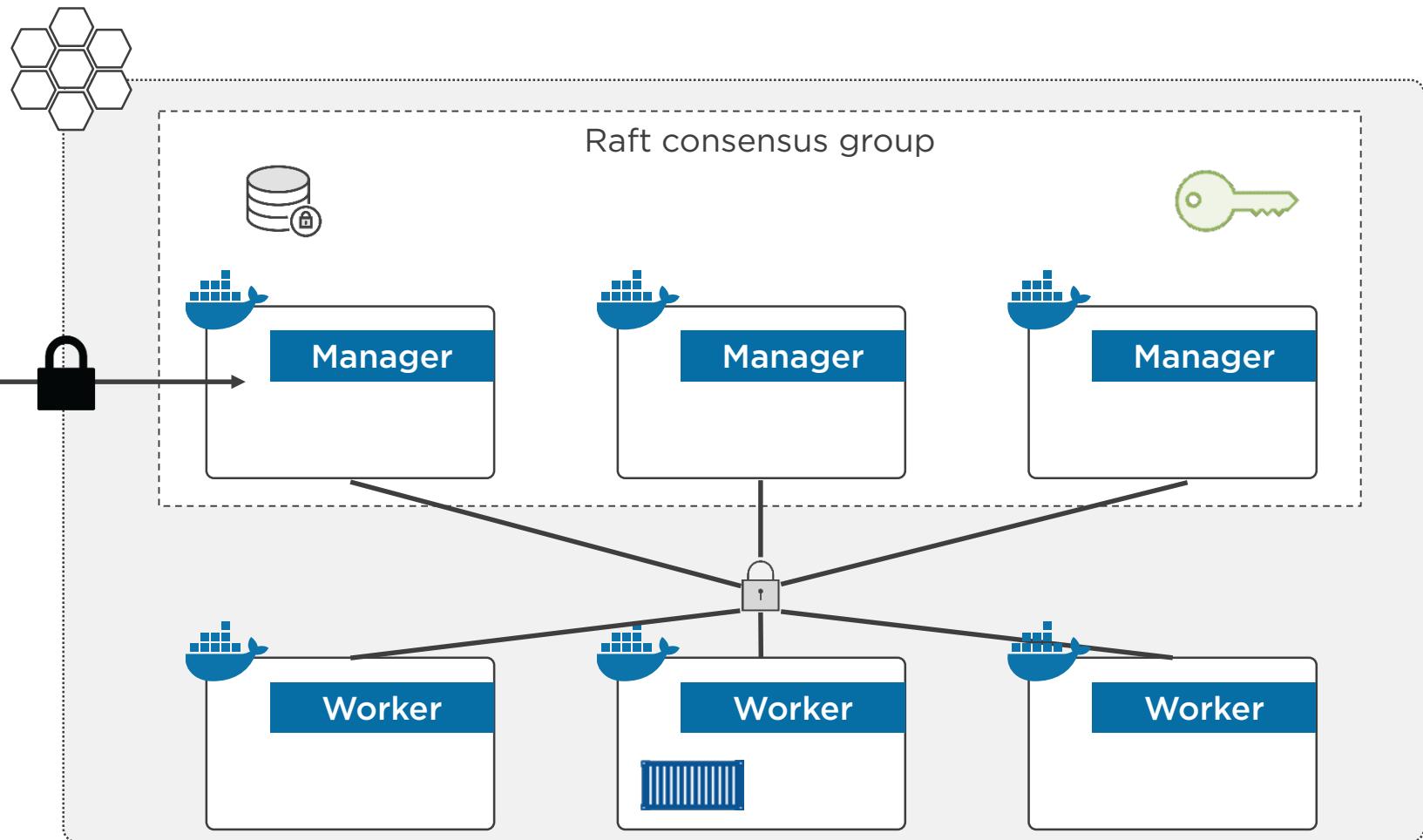


Mounted un-encrypted at:  
Linux: /run/secrets/  
Windows: C:\ProgramData\ Docker\Secrets\

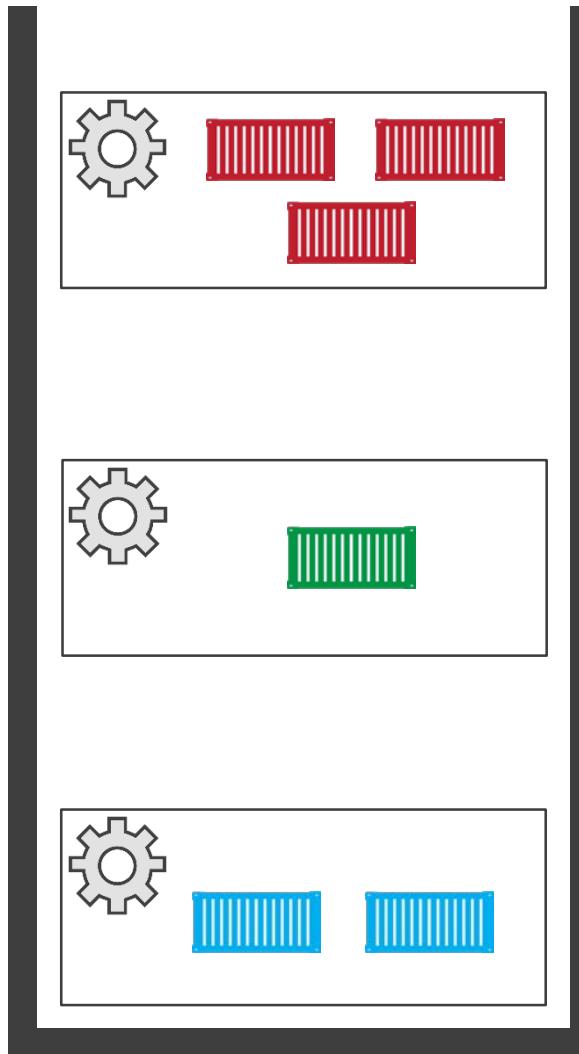




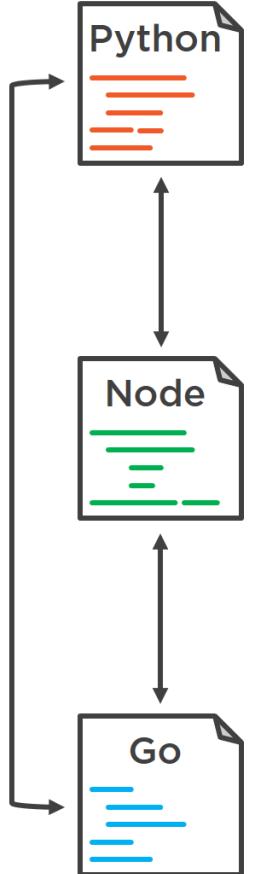
```
docker service rm green
```



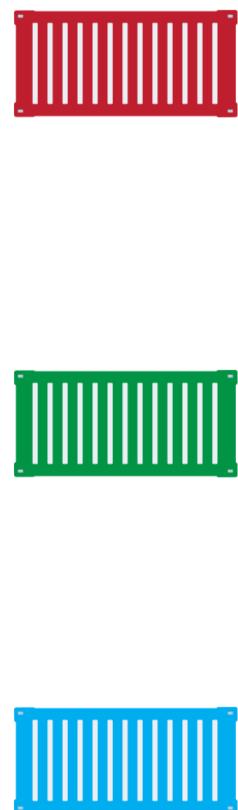
# Stack



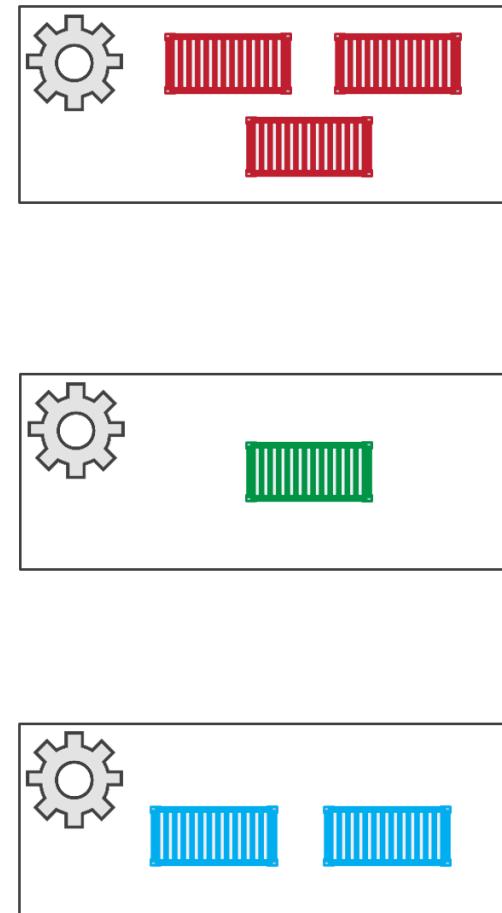
## Code



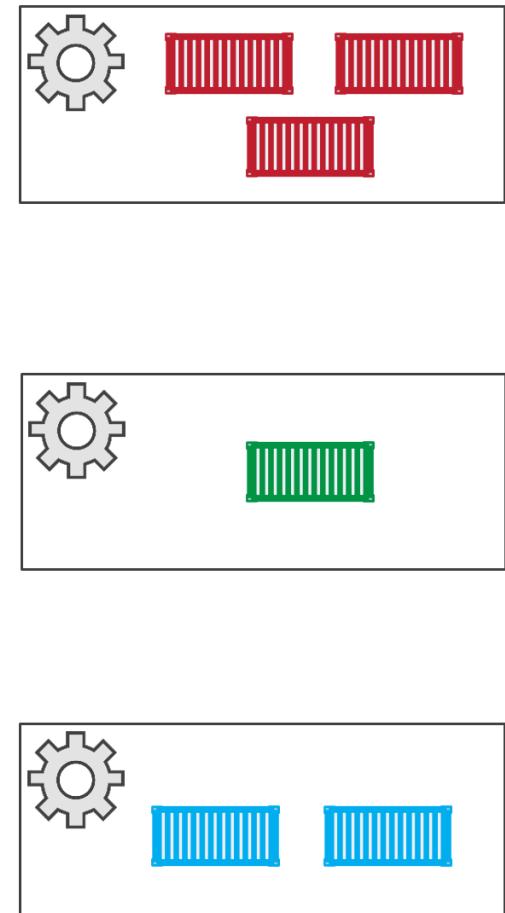
## Containers



## Services

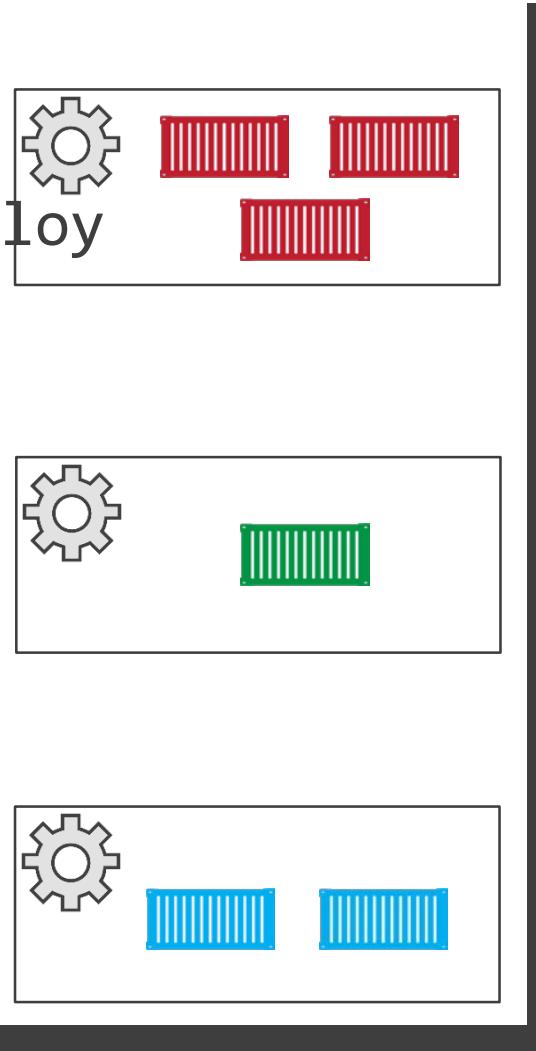


## Stack





```
$ docker stack de
```

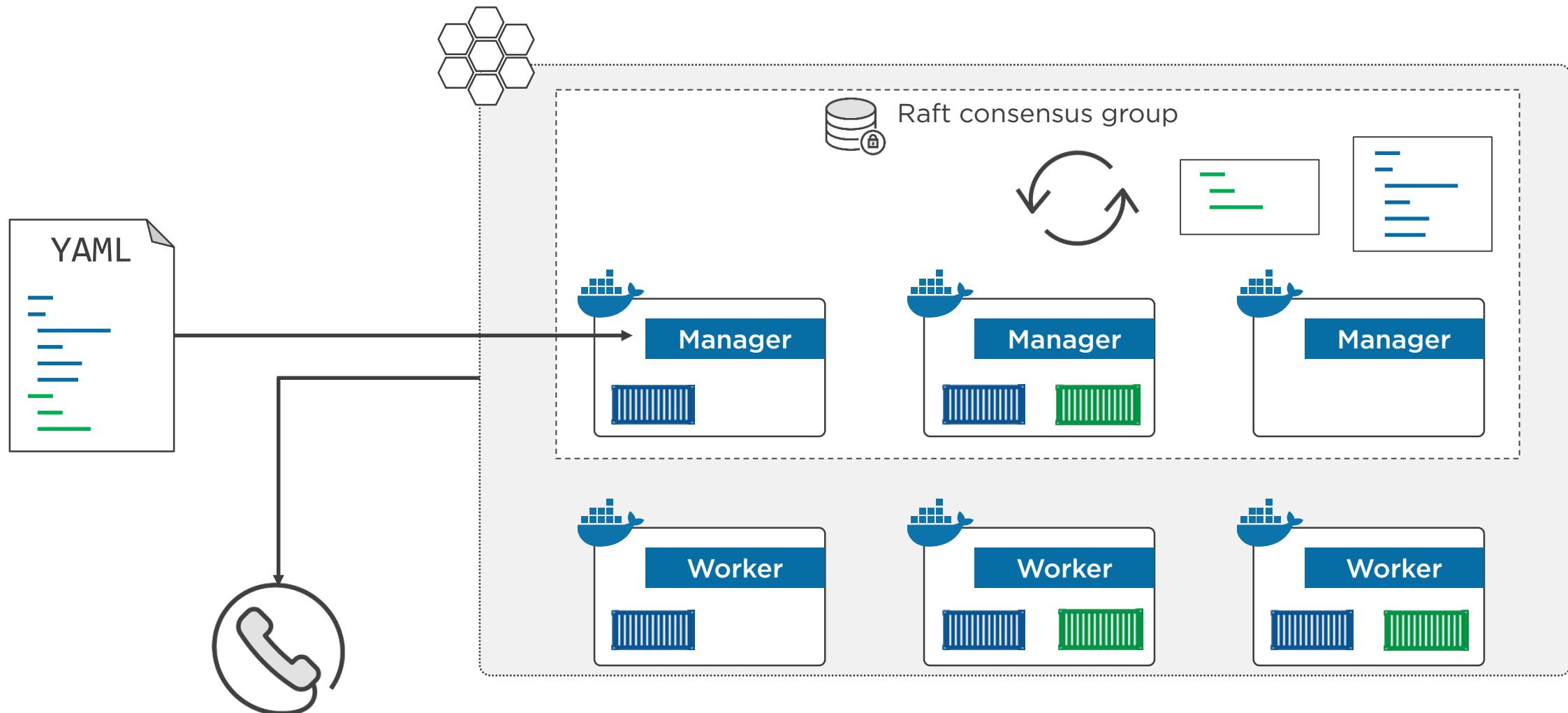


Docker UCP



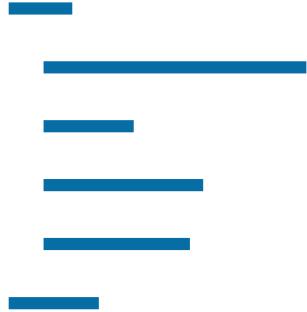
Docker Cloud





# YAML

Service

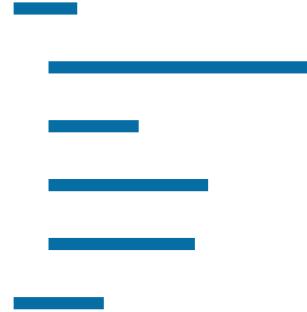


Networks  
Volumes

Dev

# YAML

Service

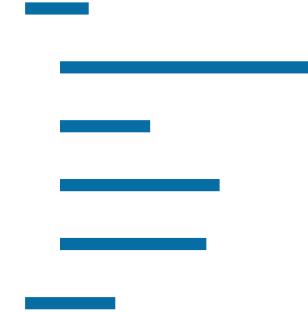


Networks  
Volumes

Test

# YAML

Service

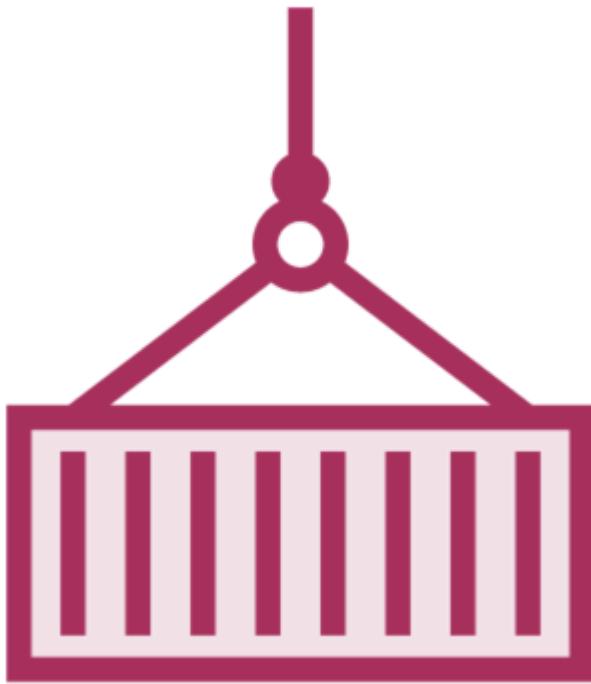


Networks  
Volumes

Prod



# Docker Distributions for Windows



**Docker Toolbox**  
**Docker in WSL2**  
**Docker Desktop**  
**Docker Engine Enterprise**



# Orchestration Solutions for Windows Containers

Azure Kubernetes  
Service (AKS)

Azure Service  
Fabric

Azure Stack with  
AKS Engine

Kubernetes

Docker Swarm



# Switching Between Linux and Windows Containers

To Linux:

```
[Environment]::SetEnvironmentVariable  
("LCOW_SUPPORTED", "1", "Machine")  
Restart-Service docker
```

To Windows:

```
[Environment]::SetEnvironmentVariable  
("LCOW_SUPPORTED", $null, "Machine")  
Restart-Service docker
```

# Maintenance

## Installing Specific Version

```
Install-Package -Name docker \  
-ProviderName DockerMsftProvider \  
-Force -RequiredVersion 19.03
```

## Updating Mirantis Container Engine

```
Install-Package -Name docker \  
-ProviderName DockerMsftProvider \  
-RequiredVersion 19.03 -Update -Force
```

# Checking the Windows Version

```
Get-ItemProperty -Path "HKLM:\Software\Microsoft\Windows NT\CurrentVersion" |  
Select ProductName, ReleaseId, InstallationType,  
CurrentMajorVersionNumber, CurrentMinorVersionNumber, CurrentBuild
```

# Windows Server Base Images

**windows**  
**servercore**  
**nanoserver**  
**iotcore**



# Windows Base Images

Using windows tag

**Desktop version**

**Contains full set of Windows APIs and system services**

**Required for some apps such as SharePoint Server 2019**



# Windows Server Core Base Images

Using servercore tag

**Headless**

**Smaller image**

**Suited for both VMs and containers**

**Contains a subset of the Windows Server APIs**

**Full .NET framework**

**Includes most server roles**



# Nano Server Base Images

Using nanoserver  
tag

**Headless**

**Smallest image**

**Designed for containers (only available as a  
container base image in the Semi-Annual  
Channel)**

**Optimized for .NET Core applications**

**PowerShell Core, .NET Core, and WMI are  
not included by default**



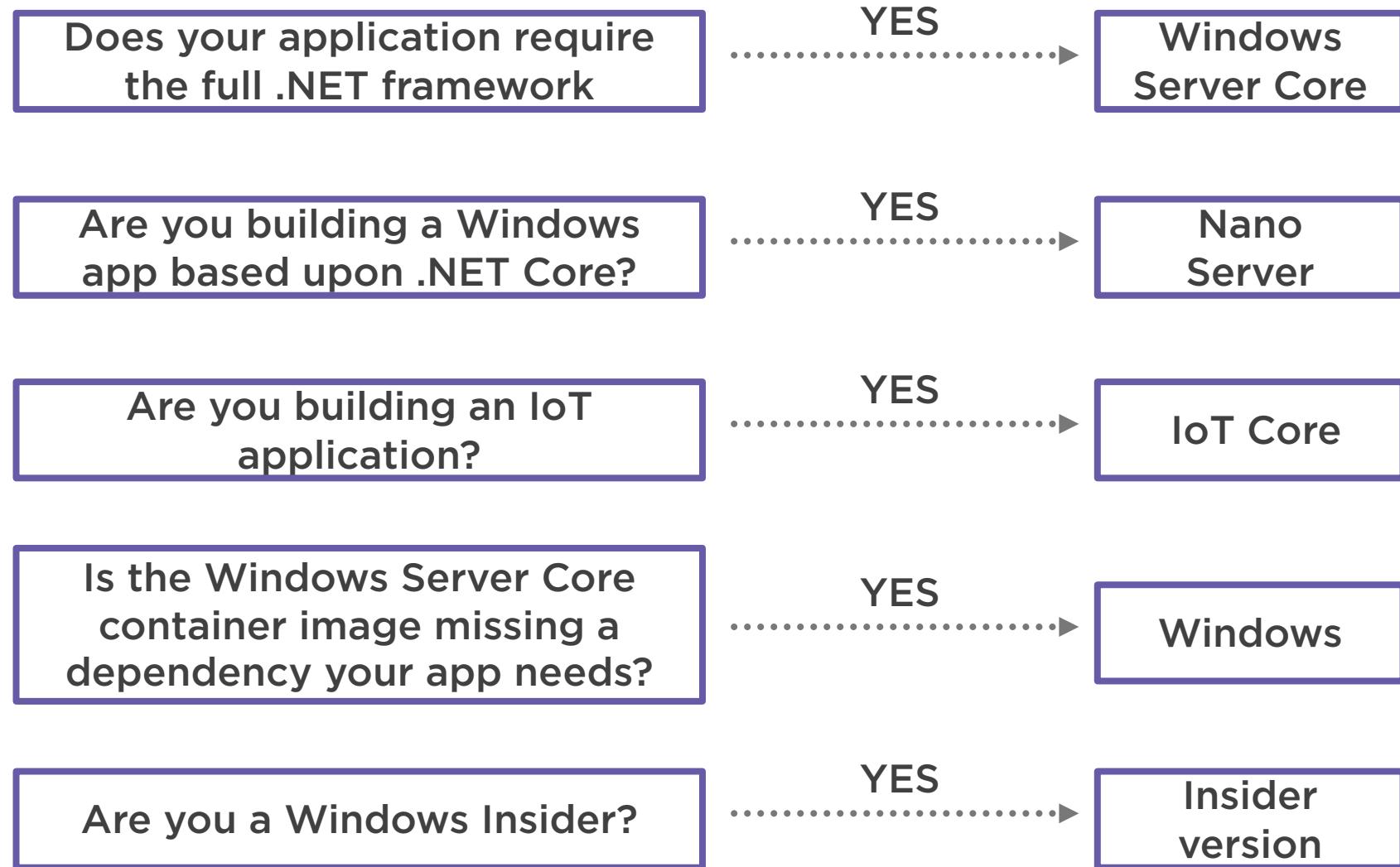
# Windows 10 IoT Core Base Images

Using iotcore tag

**Used for Internet of Things devices**  
**Supports ARM and x64 processors**



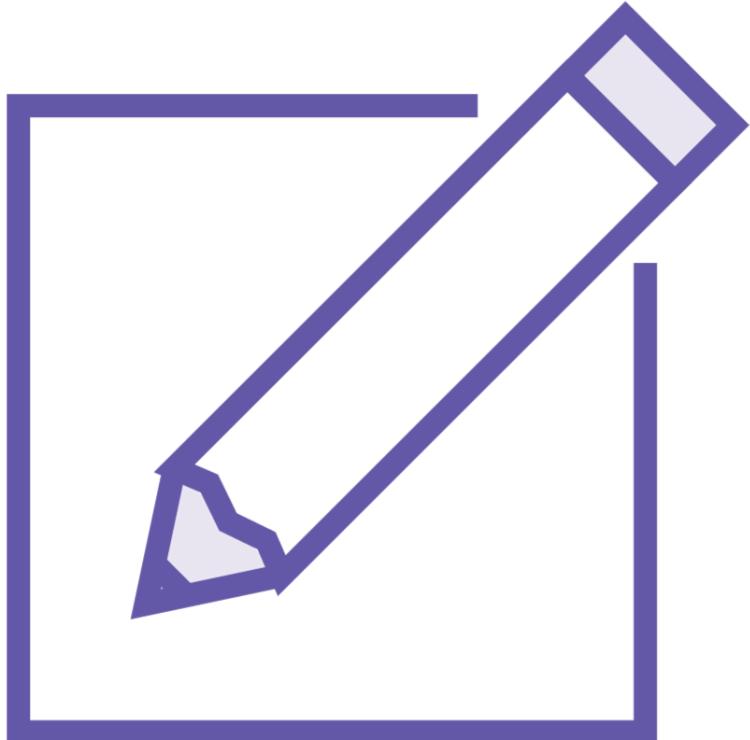
# Choosing a Base Image



# Other Base Images



- Dotnet
- IIS
- **azure-functions**



## Licensing:

- Container images may be used with properly licensed Windows host software
- The Semi-Annual Channel is available for cloud providers and volume-licensed customers



```
FROM mcr.microsoft.com/powershell:nanoserver-$SERVERCORE
SHELL [ "pwsh", "-Command", "$ErrorActionPreference = 'Stop'; $ProgressPreference = 'SilentlyContinue';" ]
COPY --from=builder /nodejs /nodejs
COPY ["app.js", "c:/web/"]
EXPOSE 8080
WORKDIR /web
CMD Start-Process -NoNewWindow -Wait -FilePath c:/nodejs/node.exe -ArgumentList c:/web/app.js
```

## Powershell in Dockerfile

**Source:**

<https://github.com/rancher/rancher/blob/v2.5.3/tests/validation/tests/Dockerfiles/windows/metrics/Dockerfile>



```
# escape=`

FROM mcr.microsoft.com/nanoserver

ARG VS_REMOTE_DEBUGGER_PATH

ARG VS_OUT_DIR

ADD $VS_REMOTE_DEBUGGER_PATH "c:\windows"

VOLUME $VS_OUT_DIR

WORKDIR $VS_OUT_DIR

ENTRYPOINT msvsmon.exe /noauth /anyuser /silent /nostatus /noclrwarn /nosecuritywarn /nofirewallwarn /nowowwarn
/timeout:36000
```

## Shell Subset in Dockerfile

**Source:**

**<https://github.com/microsoft/NanoServerTools/blob/v0.1/DevTools/ContainerDebug/dockerfile>**

