Docker – Enterprise Container Platform

Course Outline

- Virtualization
- Container Fundamentals
- Why Docker?
- Docker Engine
- Docker Installations
- Building Image
- Sharing and Distribution
- Run Container
- Networks
- Storage
- Docker Compose
- Docker Swarm
- Docker Service

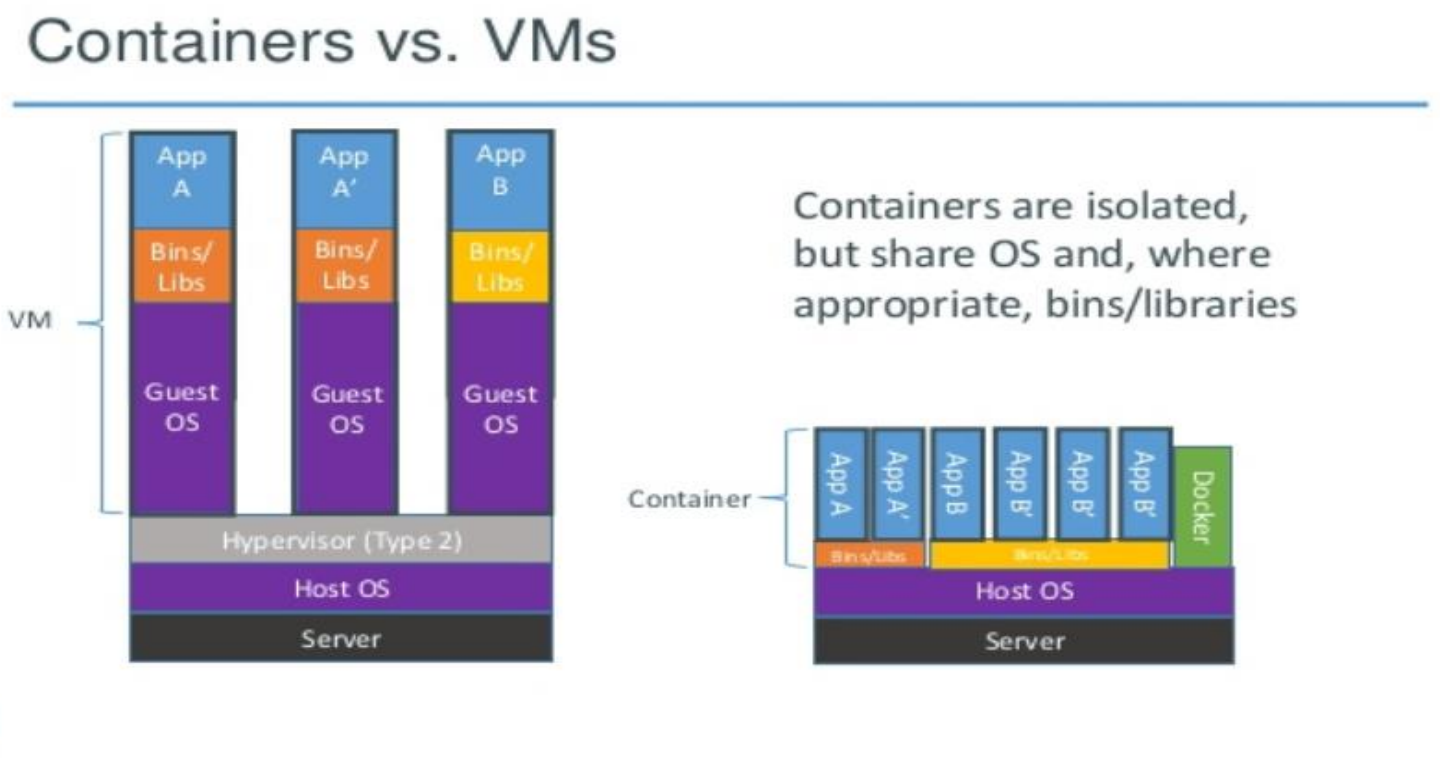
Problem Statement:

Developer would actually build their code and then they'd send it to the tester

But then the code doesn't work on tester's system , due to the differences in computer environments.

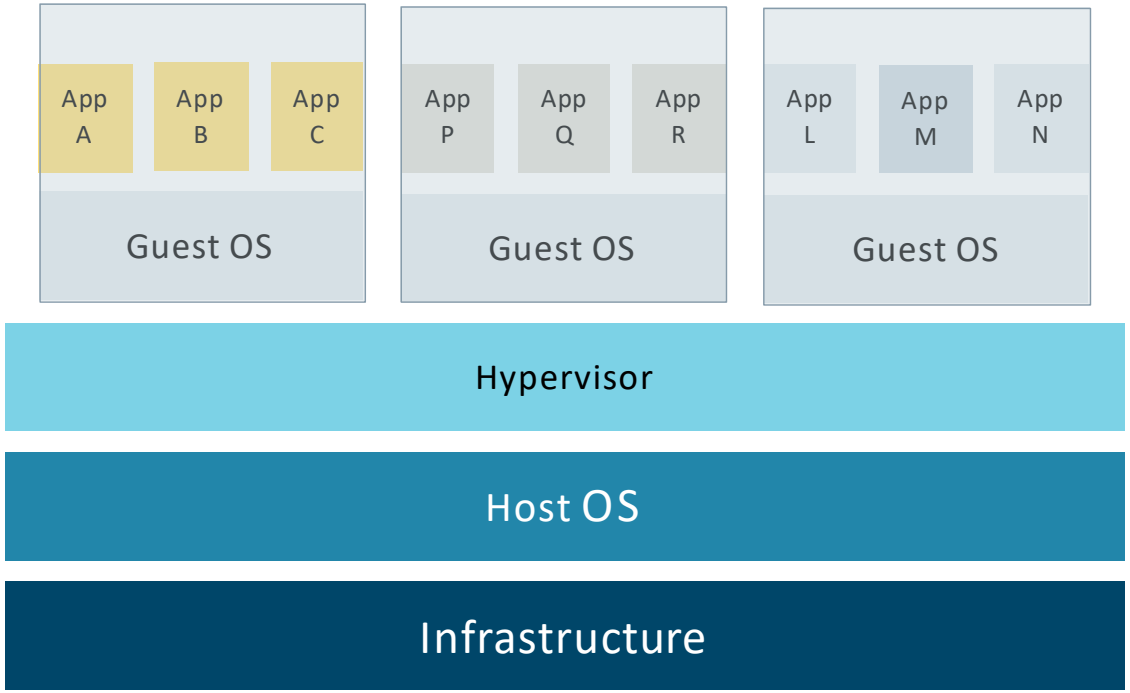
Solutions:

Virtual Machine and Docker

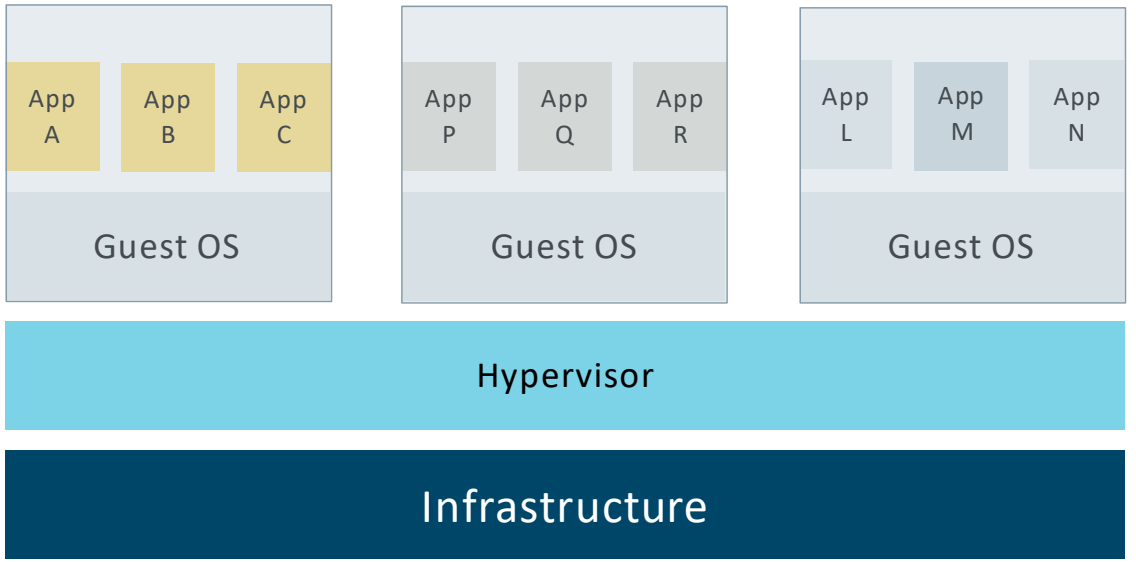


Virtualization

What is a Virtual Machine?



Hosted Hypervisor



Bare Metal /Embedded /Native Hypervisor

Why am I motivated to learn Docker?

- New initiatives
- Enhance knowledge
- Microservice
- Test automation
- Devops

Docker Lab

Docker Lab Details

10 mins

- Docker playground

  https://labs.play-with-docker.com/

  - Create an account with personal id

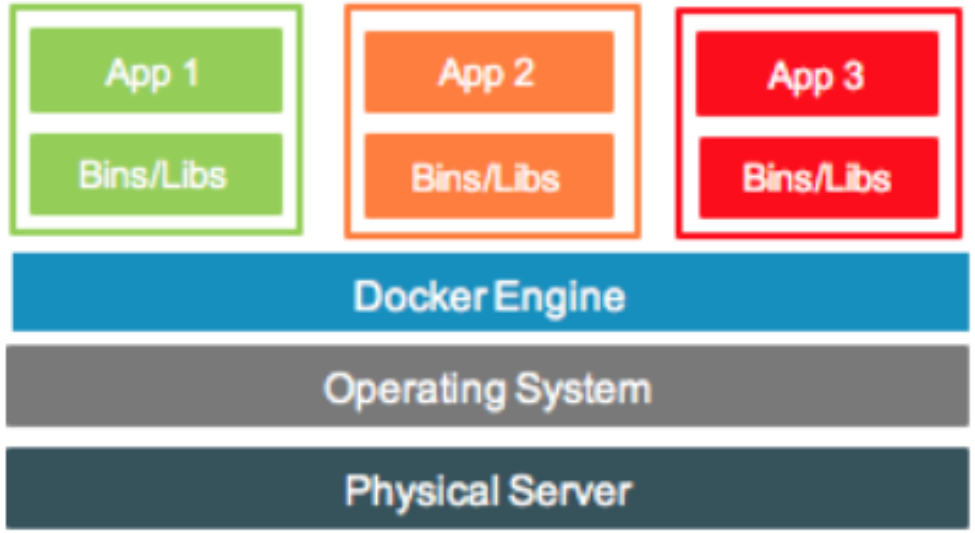
- Docker hub

  https://hub.docker.com/signup

  - Create an account with personal id

What is a Dockerization/Containerization?

Containerization is the technique of bringing virtualization to the operating system level. While Virtualization brings abstraction to the hardware, Containerization brings abstraction to the operating system. Do note that Containerization is also a type of Virtualization. Containerization is however more efficient because there is no guest OS here and utilizes a host's operating system, share relevant libraries & resources as and when needed unlike virtual machines.

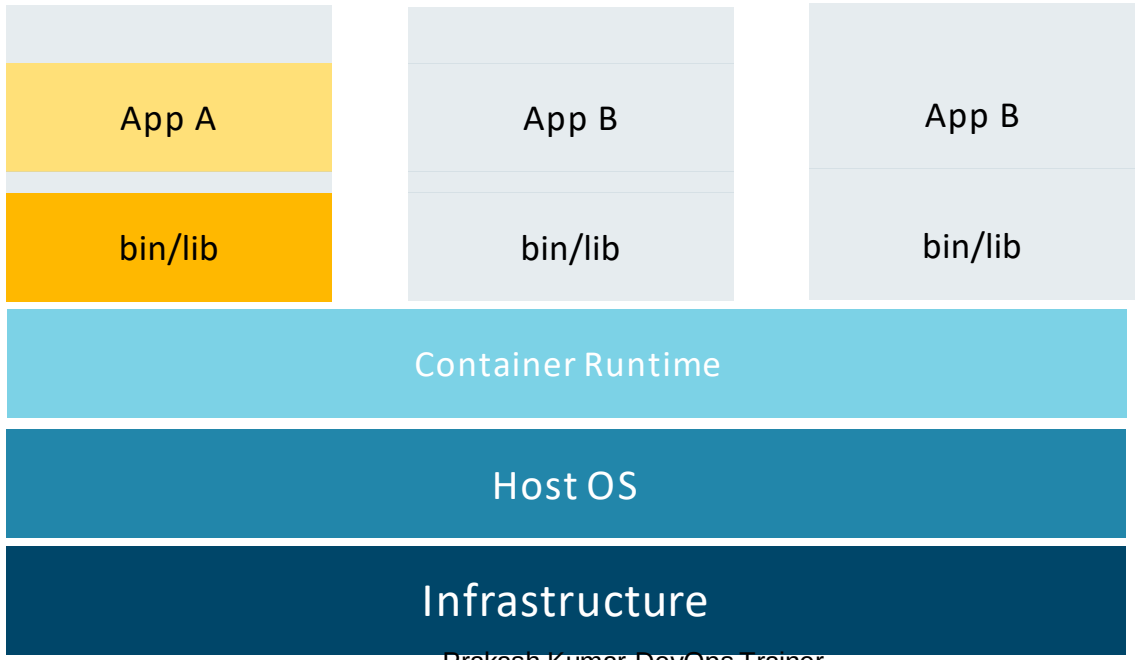


What is a Docker?

Docker is a containerization platform that packages your application and all its dependencies together in the form of Containers to ensure that your application works seamlessly in any environment.

What is a Container?

A runtime instance of an image – what image becomes in memory when executed.
Docker Container is a standardized unit which can be created on the fly to deploy a particular application or environment. It could be an Ubuntu container, CentOs container, etc. to full-fill the requirement from an operating system point of view. Also, it could be an application-oriented container like CakePHP container or a Tomcat-Ubuntu container etc



What is an Image?

An executable package that includes everything needed to run an application – code, runtime, libraries, environment variables, and configuration files.

Docker image is combination of multiple layer, and each layer will consist of some binary and library files.

Virtual Machine v/s Container

❑ Virtual Machine

- Contains a complete operating system and applications
- Can take up several GBs depends on the guest the OS
- Use of hypervisors to share and manage hardware
- Have their own kernel and they don't use and share the kernel of the host OS
- Can run different operating systems
- Virtualizing underlying hardware

❑ Container

- Shares the kernel of the host OS to access the hardware
- Bound by the host operating system, containers on the same server use the same OS
- Virtualizing underlying operating system

Container Characteristics

- Own process space
- Own network interface
- Can run stuff as root
- Can install packages
- Can run services
- Can set own routing and iptables

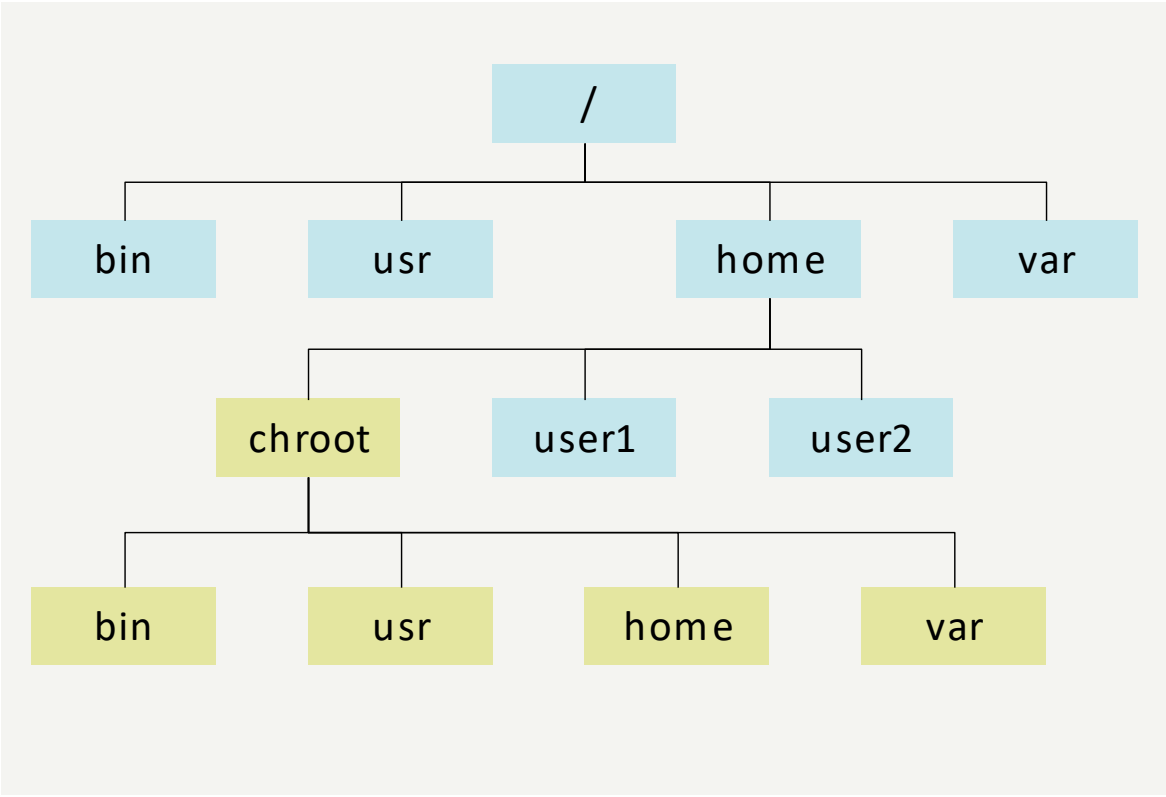
- Uses the host kernel
- Can't boot a different OS
- Can't have its own modules
- Doesn't need init as PID 1

https://www.youtube.com/watch?v=sK5i-N34im8

Container Fundamentals

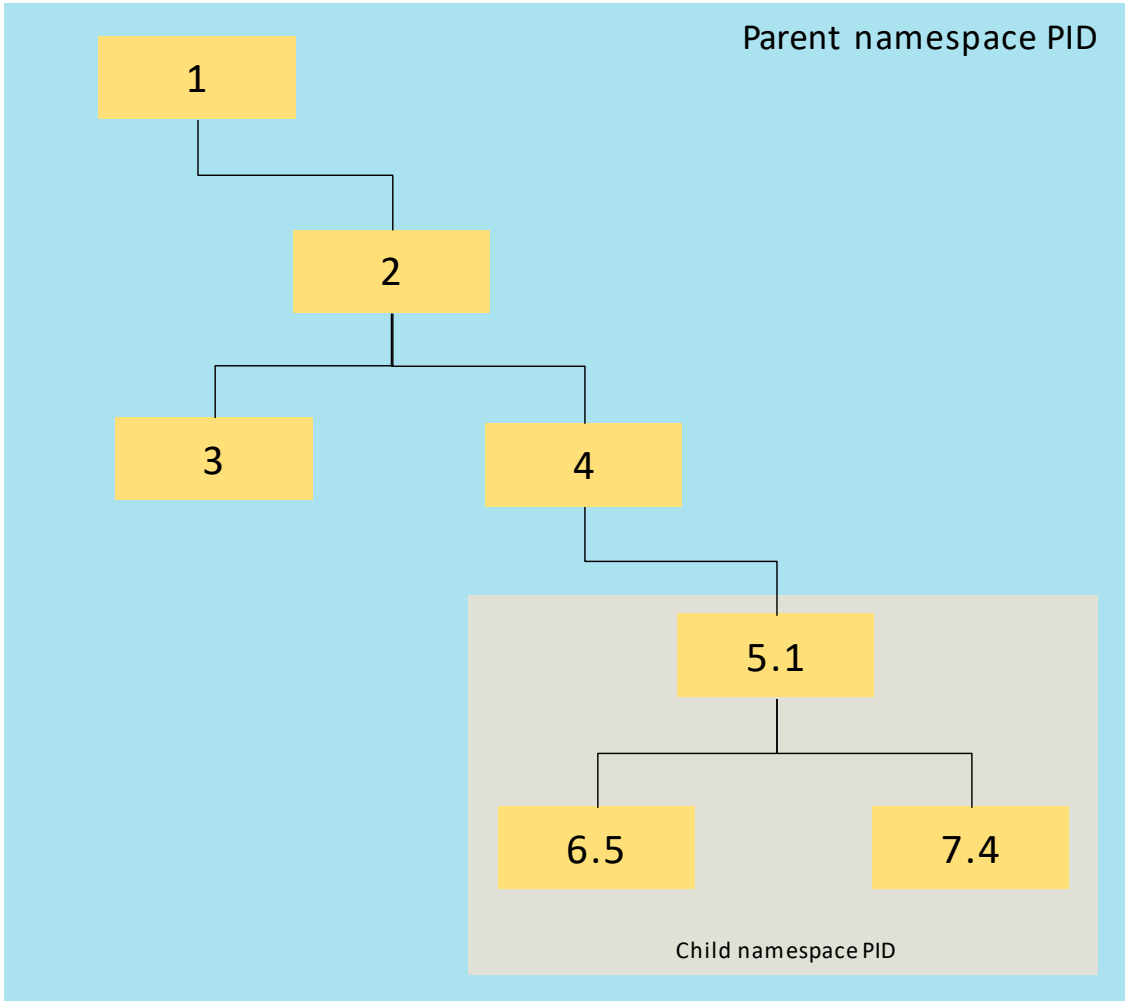
chroot

- Changes the apparent root directory for the current running process and their children
- Running program cannot access files and commands outside
- Global space knows internals of the directory and can modify contents
- Running program uses global system resources
- Not a security feature



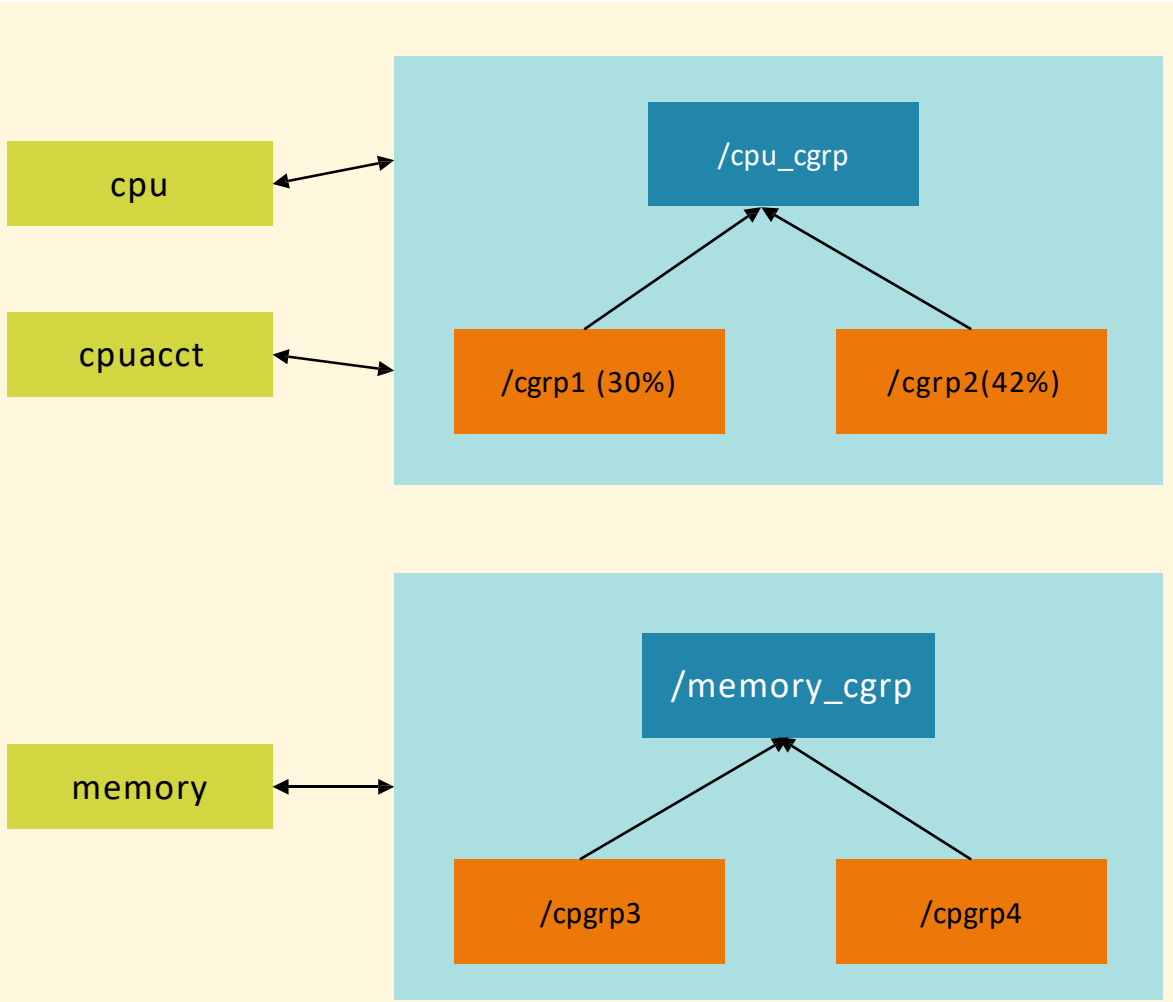
Namespaces

- Wraps a global system resource in an abstraction
- Provides processes with their own system view
- Processes in a namespace can only interact with processes that are part of the same namespace
- Each process is in one namespace of each type
- E.g. PID, mount, UTS, user, network, IPC, cgroup



Cgroups

- Allows processes to be organized in hierarchical groups
- Enable limiting and monitoring of system usage by process groups
- Subsystem, kernel component modifies the behavior of the process in a cgroup
- E.g. blkio, cpu, cpuacct, cpuset, memory, net_cls, freezer, devices, perf_event, hugetlb



Exercise: Process Namespace (Process Id)

10 mins

- docker container run -d --name my_cont nginx

- Host Shell
  - ps -aef |grep nginx
  - docker container exec -it my_cont /bin/bash

- Container Shell
  - which ps
  - apt-get update
  - apt-get install -y procps
  - ps –aef

Exercise: Process Namespace (Port)

10 mins

- docker container run -d --name nginx-1 nginx
- docker container run -d --name nginx-2 nginx
- Host Shell
  - ifconfig
  - netstat -tln
  - curl localhost
- ps –aef |grep nginx
- ls -ltr /proc/[pid]/ns

List the links for master and worker processes of nginx

- Container Shell
  - docker container exec -it my_cont /bin/bash
  - apt-get update
  - apt-get install -y net-tools
  - apt-get install –y curl
  - ifconfig
  - curl localhost

Exercise: Memory Cgroup

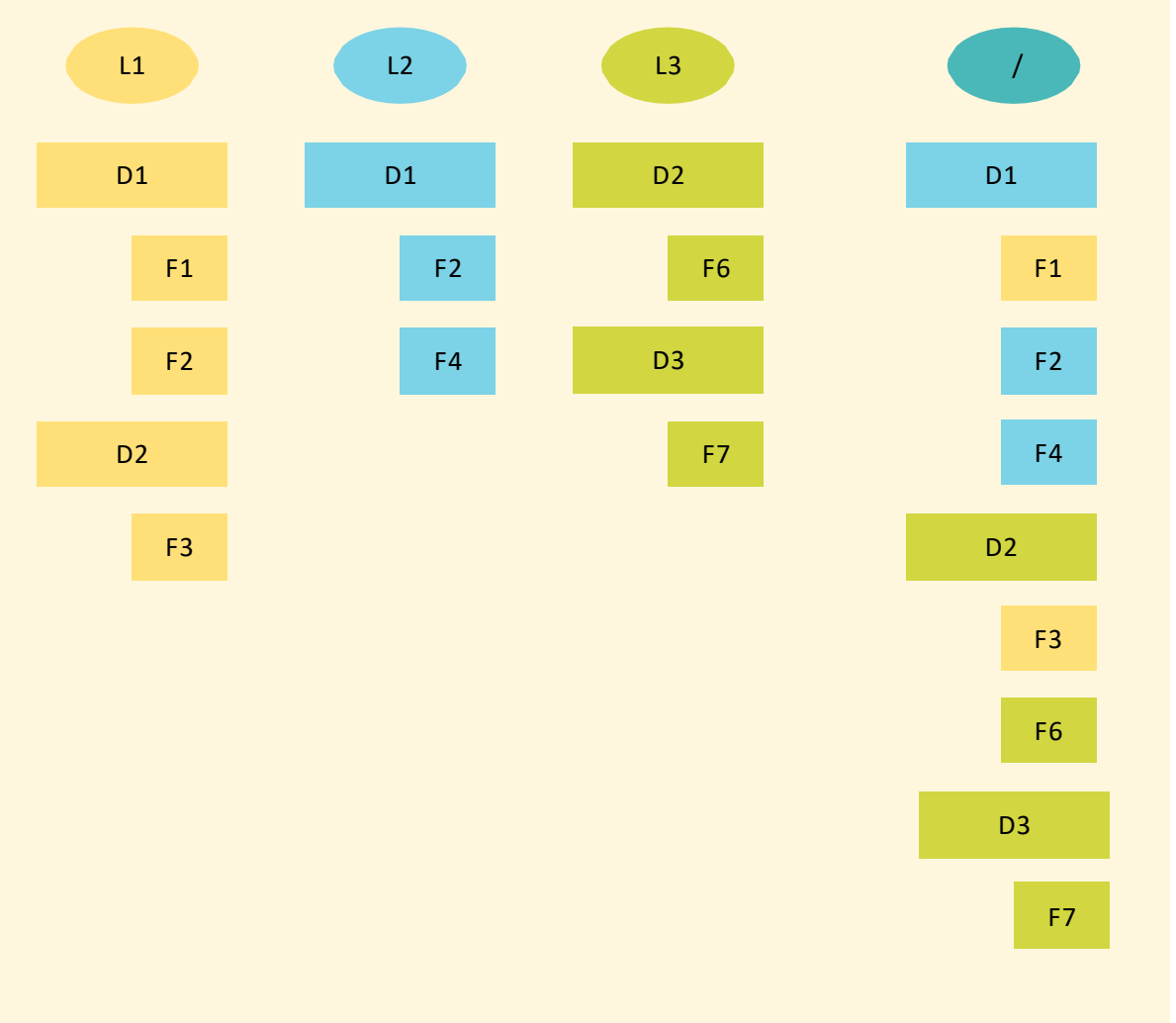
10 mins

- Host Shell
  - docker stats
  - cd /sys/fs/cgroup
  - cd /sys/fs/cgroup/memory/docker
  - ls
  - docker container run -d --name high_memory --memory 20m nginx
  - docker container run -d --name low_memory --memory 10m nginx
  - docker stats

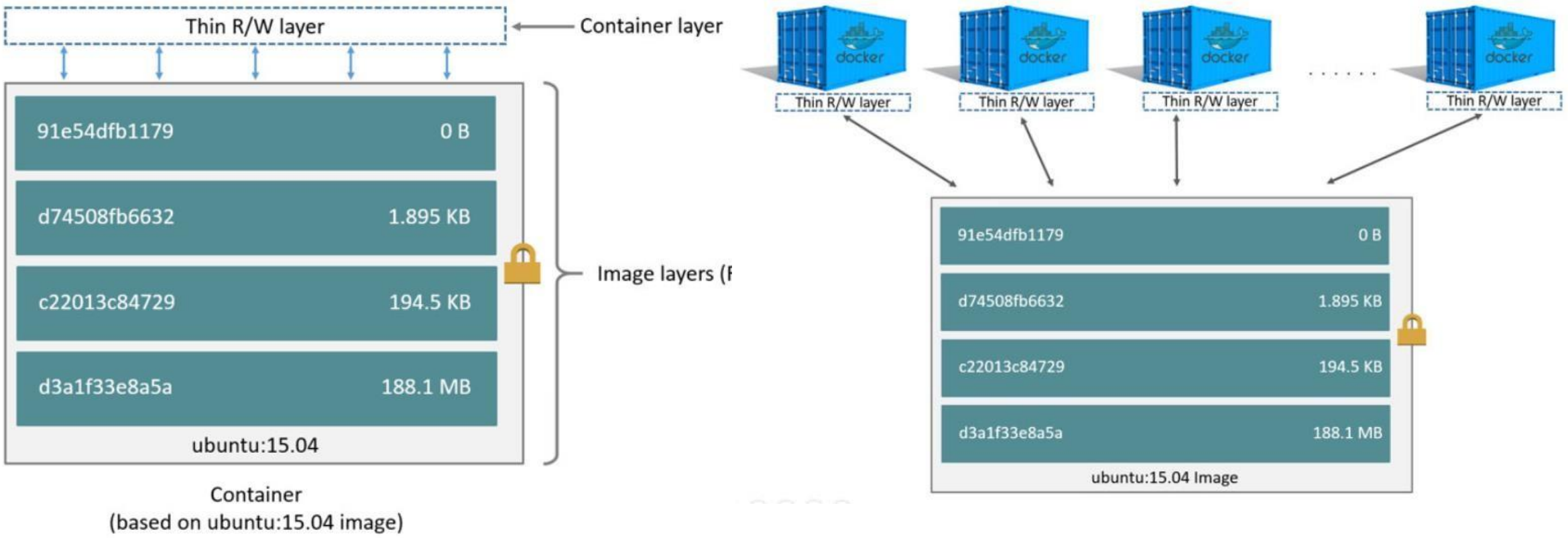
- cd XXXX
- ps –aef |grep nginx
- cat cgroup.procs
- cat cat memory.limit_in_bytes

UnionFS

- Implements a union mount for other file system
- Allows files and directories of separate file systems, to be transparently overlaid forming a single coherent file system
- Contents of directories having same path within merged branches seen together in merged branch directory
- Avoids duplicating a complete set of files each time image is run as a container
- Isolate changes to a container filesystem in its own layer



Images and Layers



https://docs.docker.com/storage/storagedriver/

Copy-on-write (CoW)

- Sharing and copying files for maximum efficiency
- Read access: Two images share the layers (files and directories) they have in common
- Write access: Shared file gets copied into the respective layer and then modified
- Sharing promotes smaller images
- Copying makes container efficient

Exercise: UnionFS, overlay2 Storage Driver

10 mins

- Host Shell
  - docker system prune -a
  - cd /var/lib/docker/overlay2
  - ls -ltr
  - docker image pull ubuntu
  - ls -ltr
  - ls -ltr l

- Special Directories and Files
  - committed
  - diff
  - link
  - lower
  - work

Why Docker?

Docker Use Cases

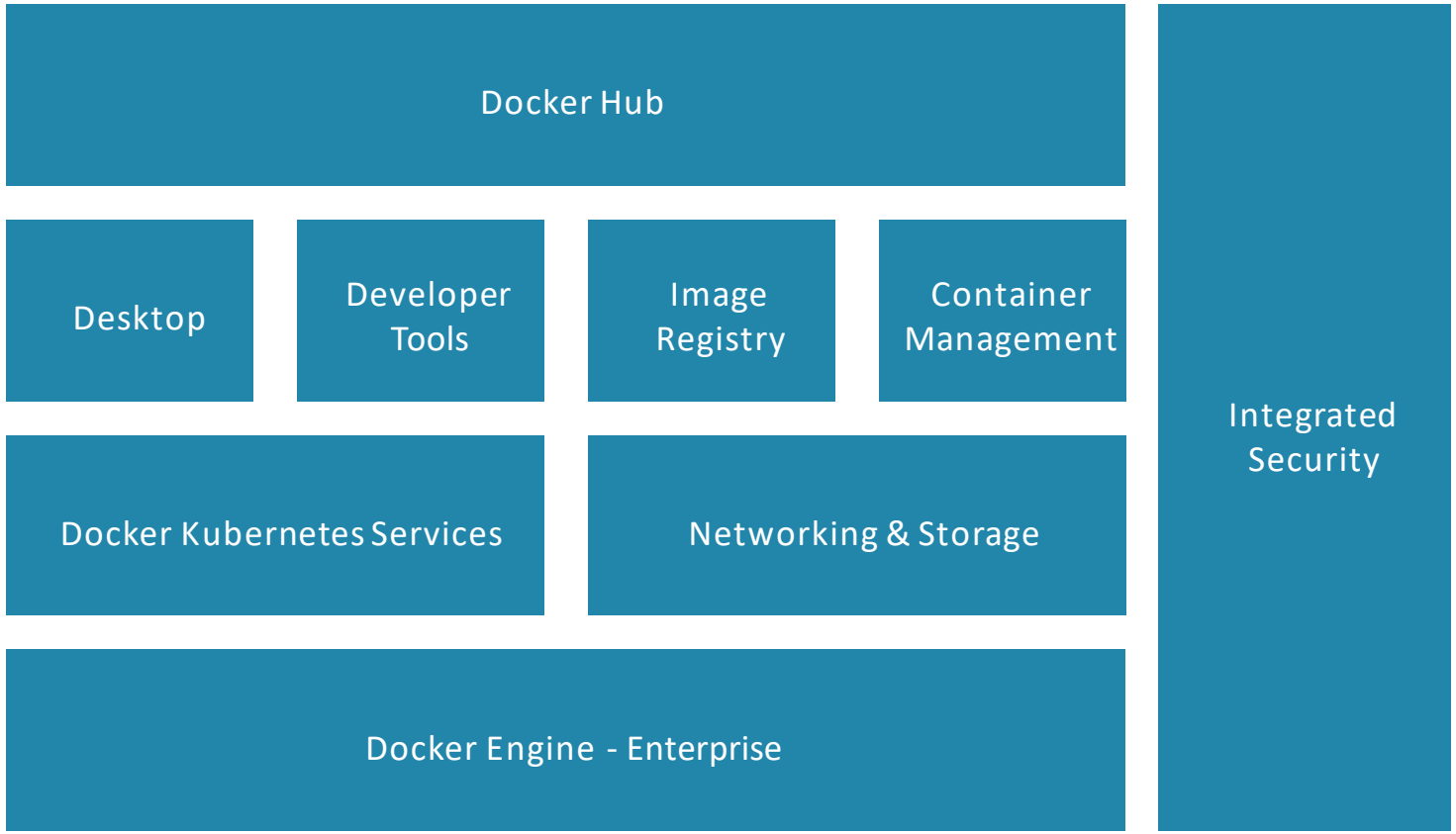
Traditional Applications	Microservices	CI/CD	Data Science
Edge Computing	Cloud Migration	Digital Transformation	Windows Server Migration

What is Docker?

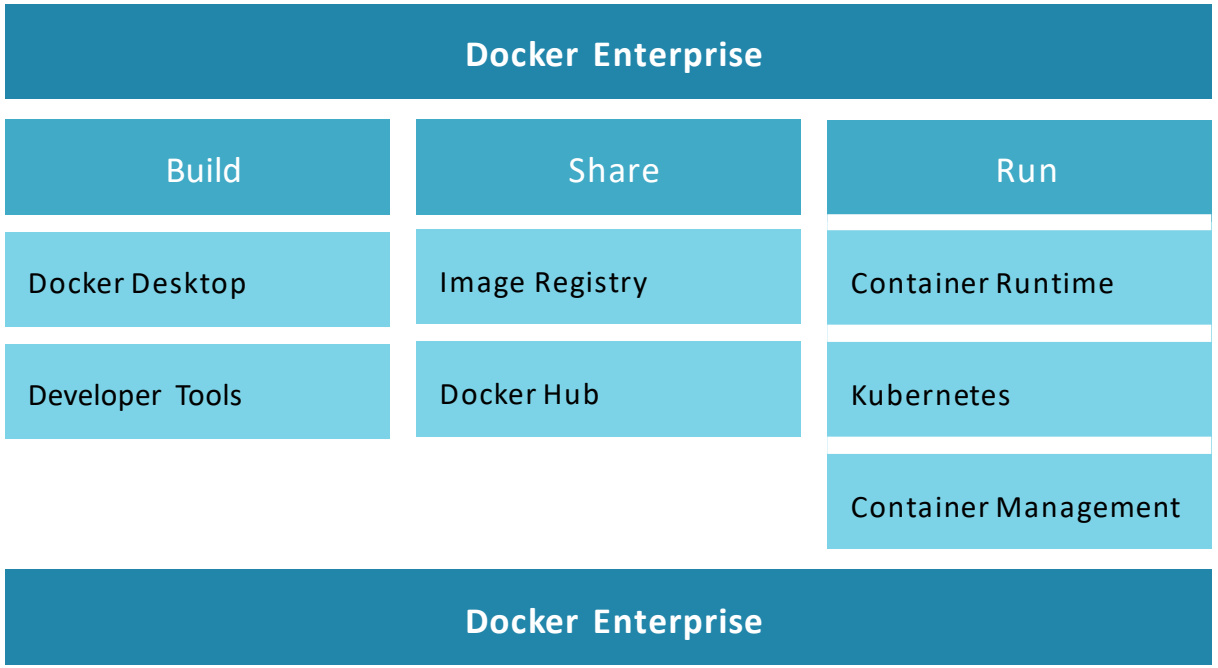
Enterprise container platform that securely

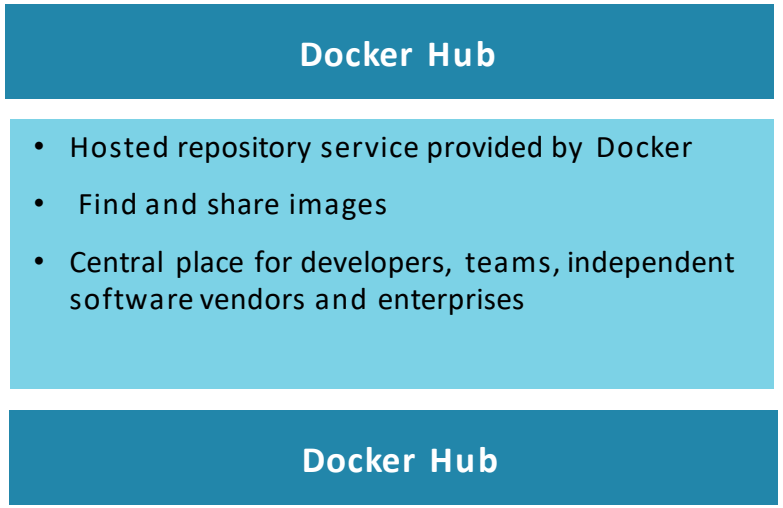
- Builds
- Shares
- Runs

any application anywhere



Docker Offerings





Docker Editions

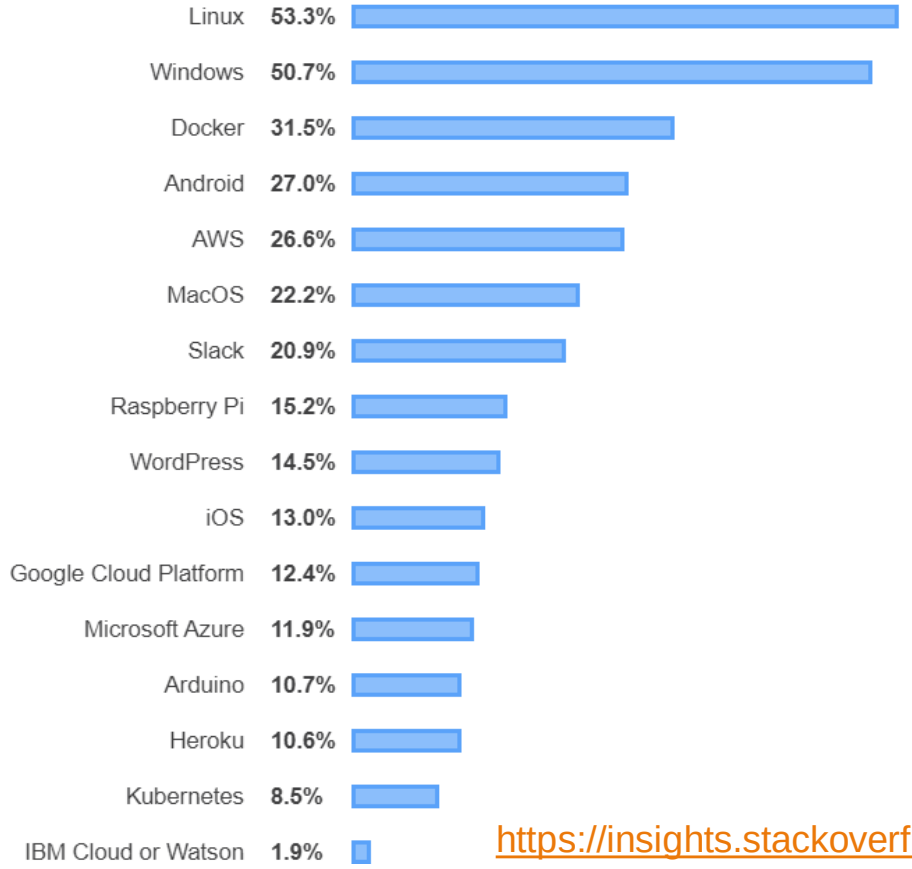
Capabilities	Docker Engine - Community	Docker Engine - Enterprise	Docker Enterprise
Container engine and built in orchestration, networking, security	✓	✓	✓
Certified infrastructure, plugins and ISV containers		✓	✓
Image management			✓
Container app management			✓
Image security scanning			✓

Reference: https://docs.docker.com/install/overview/

Stack Overflow: Developer Survey 2019

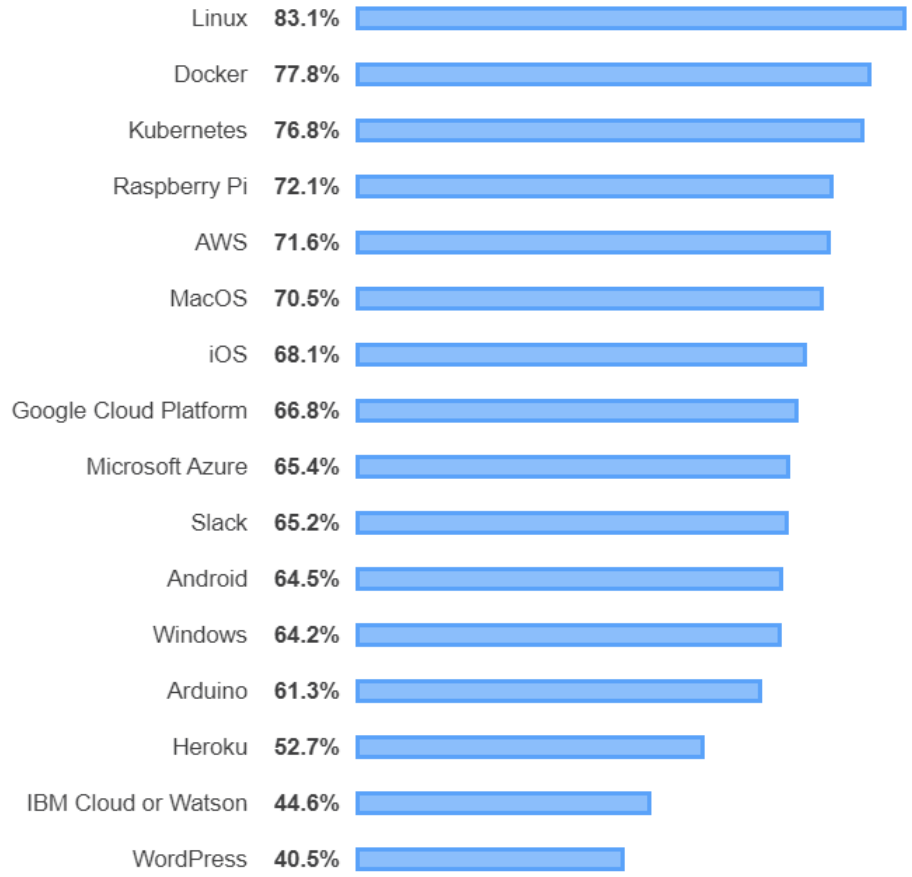
Platforms

All Respondents | Professional Developers



Most Loved, Dreaded, and Wanted Platforms

Loved | Dreaded | Wanted



https://insights.stackoverflow.com/survey/2019

Docker Enterprise Platform – Acquired !!!

MIRANTIS

Products Use Cases Delivery Approach Training Resources

Mirantis Acquires Docker Enterprise Platform Business

November 13, 2019

By Dave Van Everen

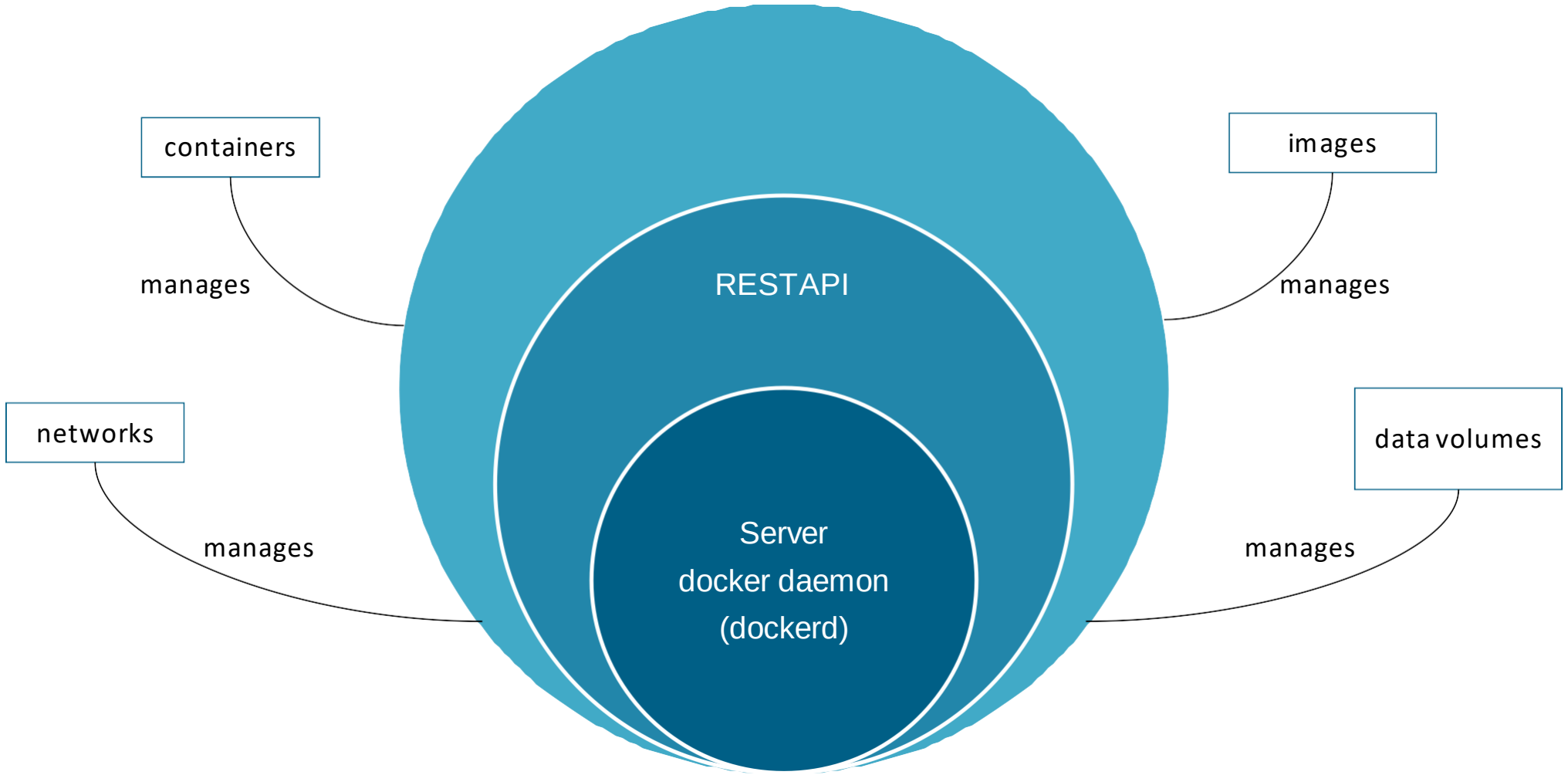
Industry-leading Docker Enterprise container platform complements existing Kubernetes technology from Mirantis

Campbell, Calif – November 13, 2019 – Mirantis, the open cloud company, announced today its acquisition of Docker's Enterprise Platform business. Its industry leading container platform, employees and hundreds of enterprise customers will accelerate Mirantis' goal to deliver Kubernetes-as-a-Service with a consistent experience for developers on any cloud and on-prem infrastructure. Terms of the deal are confidential.

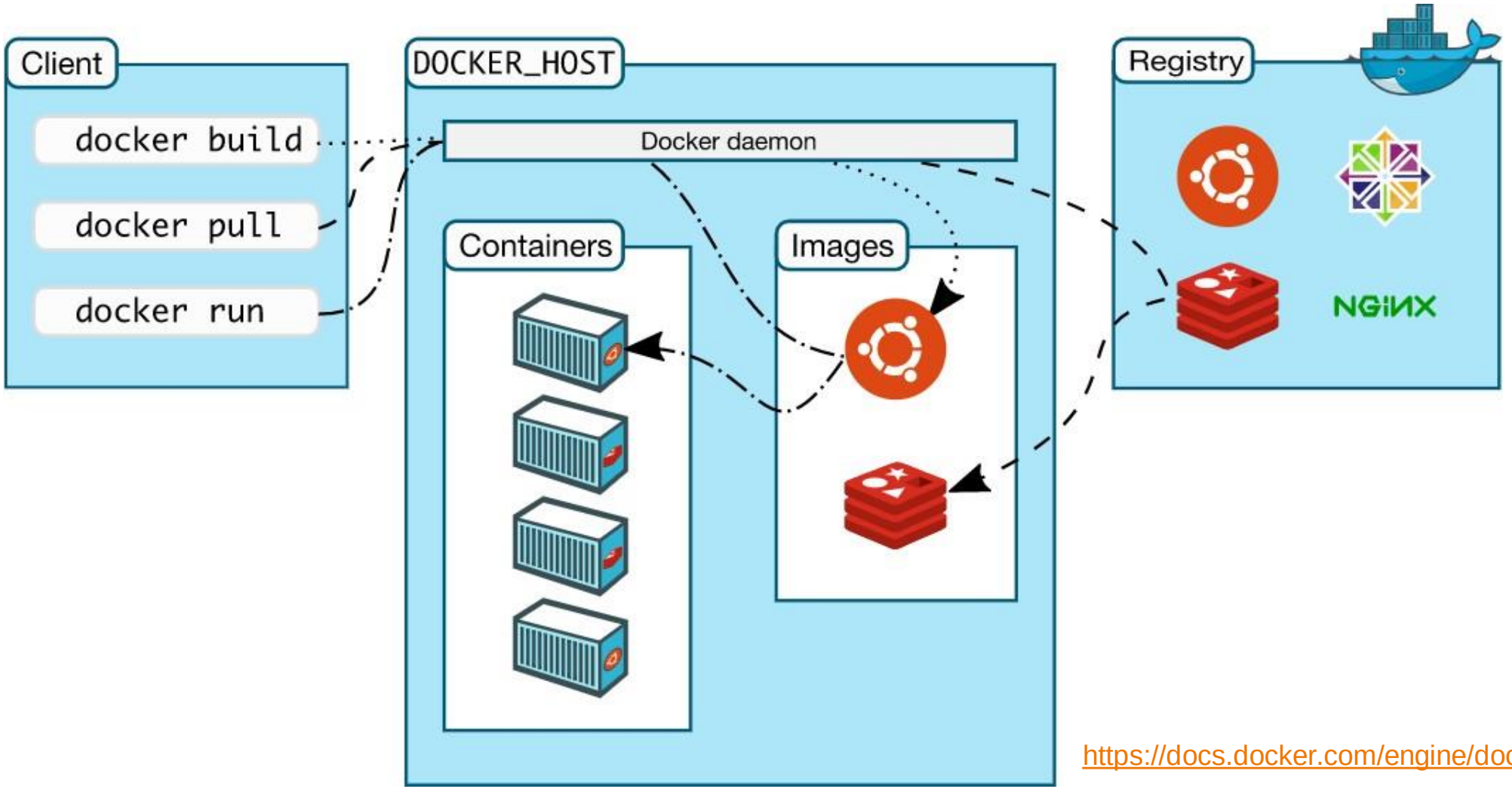
https://www.mirantis.com/company/press-center/company-news/mirantis-acquires-docker-enterprise/

Docker Engine

Docker Engine



Docker Architecture



https://docs.docker.com/engine/docker-overview/

Docker Installations

Install Docker Desktop for Windows
(Docker Engine - Community)

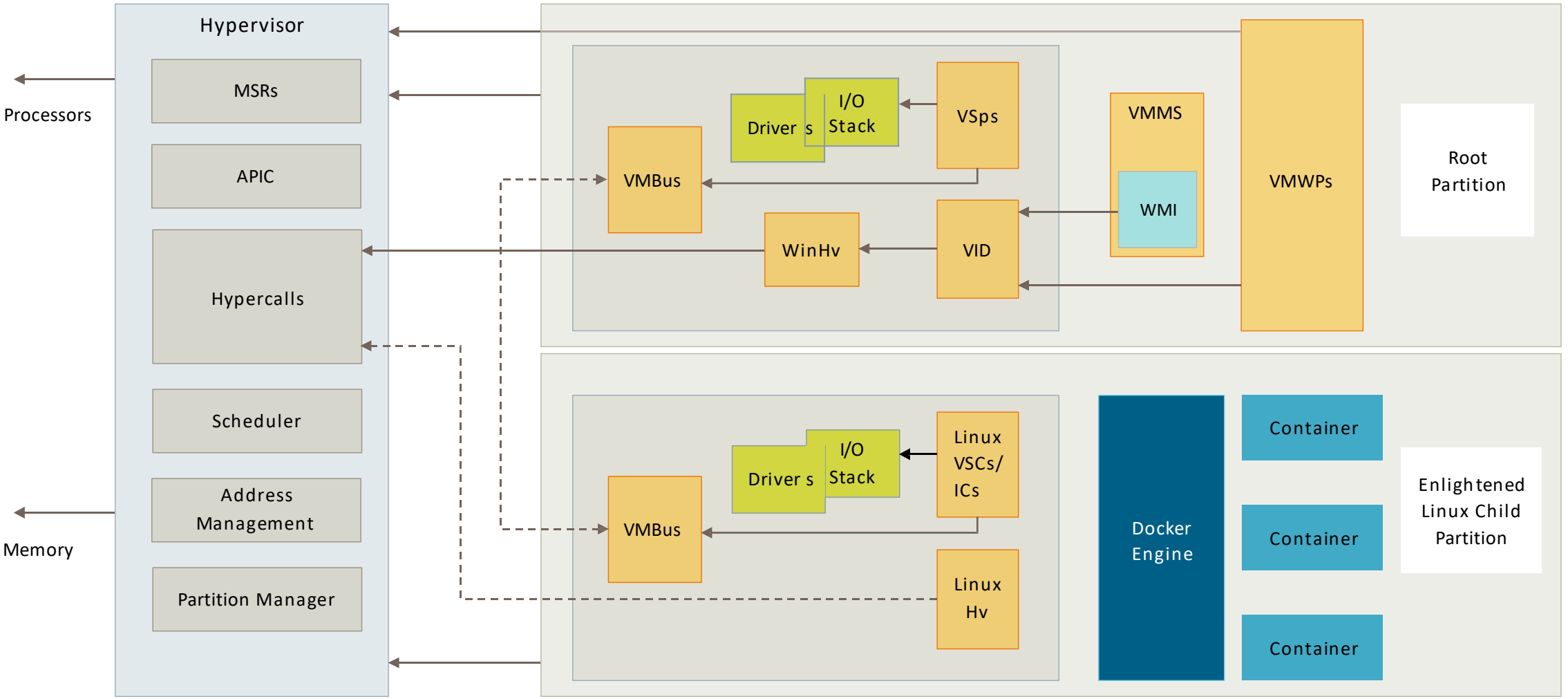
- System Requirements
  - Windows 10 64-bit
    - Pro
    - Enterprise
    - Education
  - Hyper-V
  - Containers Windows features enabled

- Components
  - Docker Engine
  - Docker Compose
  - Kubernetes
  - Notary
  - Credential Helpers
  - Machine

- Containers
  - Switch between Windows and Linux containers

https://docs.docker.com/docker-for-windows/install/

Architecture - Docker Desktop for Windows
(Docker Engine - Community)

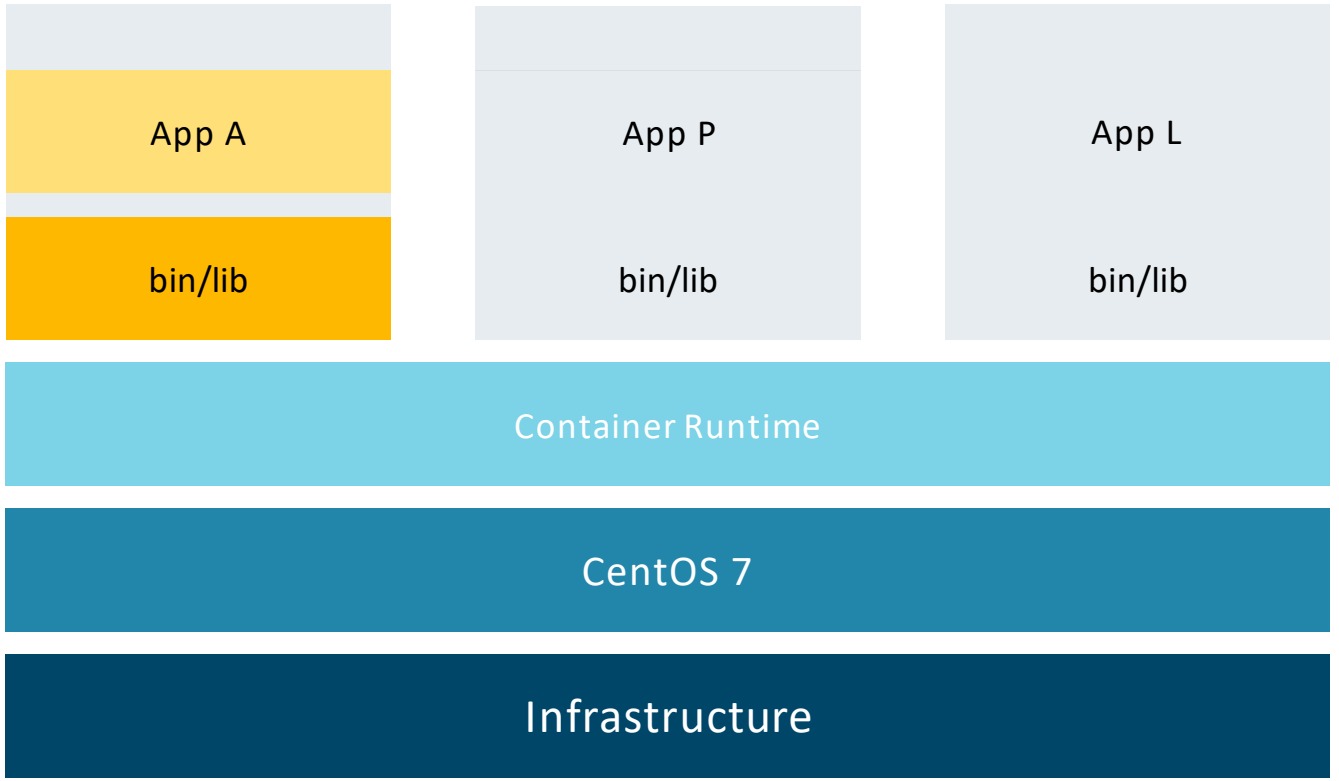


https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/reference/hyper-v-architecture

Install Docker on CentOS
(Docker Engine - Community)

- System Requirements
  - CentOS 7
  - Enabled centos-extras repository
  - overlay2 storage driver
- Components
  - Docker Engine
  - Docker CLI client
  - Docker Compose
- Containers
  - Linux containers

Architecture - Docker for CentOS
(Docker Engine - Community)



Installation: Docker Engine on CentOS

- sudo yum update –y
- sudo yum remove docker \
  docker-client \
  docker-client-latest \
  docker-common \
  docker-latest \
  docker-latest-logrotate \
  docker-logrotate \
  docker-engine
- sudo yum install -y yum-utils \
  device-mapper-persistent-data \
  lvm2
- sudo yum-config-manager \
  --add-repo \
  https://download.docker.com/linux/centos/docker-ce.repo
- sudo yum install docker-ce docker-ce-clicontainerd.io
- sudo systemctl start docker
- sudo docker run hello-world

Building Images

docker build …

- Builds an image from a Dockerfile and a context
- Context, a set of files at a specified location PATH or URL
- Docker daemon builds an image
- Sends entire context recursively to daemon
- Best practice, context to have only needed files
- <USER>/<REPOSITORY>:<TAG>
- TAG : Pointer to specific image

- Dockerfile Instructions
  - FROM
  - WORKDIR
  - ADD
  - COPY
  - VOLUME
  - ENV
  - EXPOSE
  - LABEL
  - STOPSIGNAL
  - USER

- Commands
  - docker build –f
  - docker build –t
  - …
  - docker pull ubuntu
  - docker pull registrydomain:port/ubuntu
  - docker container commit

Sample Dockerfiles

- Docker Hub

  https://hub.docker.com/

- Nginx
- https://github.com/prakashk0301/docker-static-webpage/blob/master/Dockerfile
  https://github.com/nginxinc/docker-nginx/blob/fe97d699daae7e04f916771ac520f7cf25ab2b27/mainline/buster/Dockerfile

- MySQL

  https://github.com/docker-library/mysql/blob/6659750146b7a6b91a96c786729b4d482cf49fe6/8.0/Dockerfile

- OpenJDK with MongoDB
- https://github.com/prakashk0301/spring-boot-mongo-docker/blob/master/Dockerfile

Demo: Build Image

- Build PMM image
  - Prepare context
  - Author dockerfile
  - Build image with different tags
  - Run built image

Exercise: Build Custom Nginx Image

- Dockerfile

```
FROM nginx:latest

COPY index.html /usr/share/nginx/html

EXPOSE 80

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

- index.html

```
<html>
    <body>Custom container</body>
</html>
```

- Steps
  - mkdir custom
  - cd custom
  - touch Dockerfile
  - vi Dockerfile
  - *Copy contents to Dockerfile*
  - touch index.html
  - vi index.html
  - *Copy contents to html*

- Command
  - docker container run -d -p 8080:80 nginx
  - curl localhost:8080
  - **docker image build --tag custnginx:1.0 .**
  - **docker container run -d –p 9090:80 custnginx**
  - **curl localhost:9090**
  - **docker push custnginx:1.0**

Exercise: Image, few more commands

- docker history nginx:latest
- doker search nginx
- docker image inspect
- docker image history sanesaur/mcqs:0.1
- docker image inspect sanesaur/mcqs:0.1
- docker image tag nginx sanesaur/nginx
- docker image push sanesaur/nginx

- docker login
- /root/.docker/config.json
- docker system df

Sharing & Distribution

Docker Hub

- Repositories
- Teams & Organizations
- Official Images
- Publisher Images
- Builds
- Webhooks

https://hub.docker.com/u/pkw0301

Docker Registry

- What is it
  - Containerized application, available on Docker Hub
  - Stateless, highly scalable server side application
  - Stores and lets you distribute Docker images
  - Open source
- Why use it
  - Tightly control where your images are being stored
  - Fully own your images distribution pipeline
  - Integrate image storage and distribution tightly into your in-house development workflow
- Alternatives
  - Docker Hub
  - Docker Trusted Registry

Exercise: Docker Registry

- docker search registry
- docker run -d -p 5000:5000 --name registry registry:2
- docker pull ubuntu
- docker image tag ubuntu localhost:5000/my-ubuntu
- docker push localhost:5000/my-ubuntu
- docker pull localhost:5000/my-ubuntu
- docker container stop registry && docker container rm -v registry

Docker Trusted Registry (DTR)

- Enterprise-grade image storage solution from Docker
- In-built role based access control (RBAC)
- Web user interface enables browsing of dockerfiles, images and repository events
- Caches images closer to users
- Built in security scanner
- Image signing

Run Container

docker container run…

```
docker container run –d –p 8080:80 nginx
```

- Looks for image locally in image cache
- Then looks in remote repository
- Downloads latest/given version
- Creates new container
- Gives virtual IP on private network
- Opens port on host and forwards to port in container
- Starts container using CMD or ENTRYPOINT

- docker container ls/ps
- ps –aef |grep XXX
- docker container exec -it
- docker stats

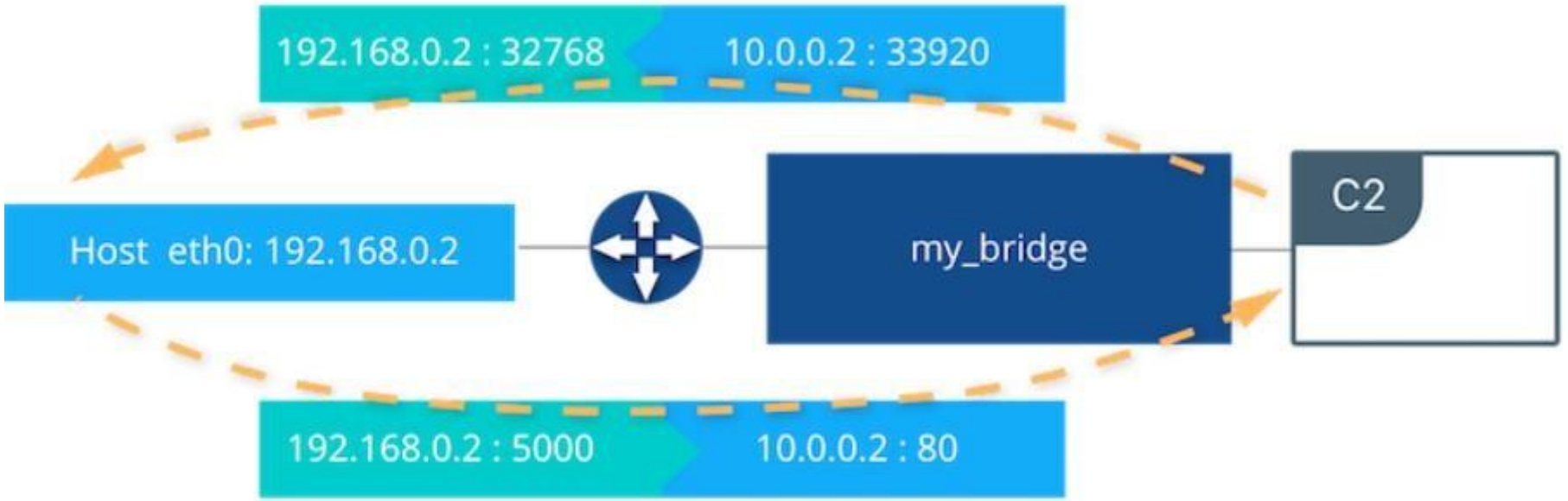
Exercises: Run Container

- nginx – map port 80 of host
- Httpd – map port 8080 of host
- mongodb – map default port 27017 of host

Network

Port Mapping

```
docker run -d --name C2 --net my_bridge -p 5000:80 nginx
```



https://success.docker.com/article/networking

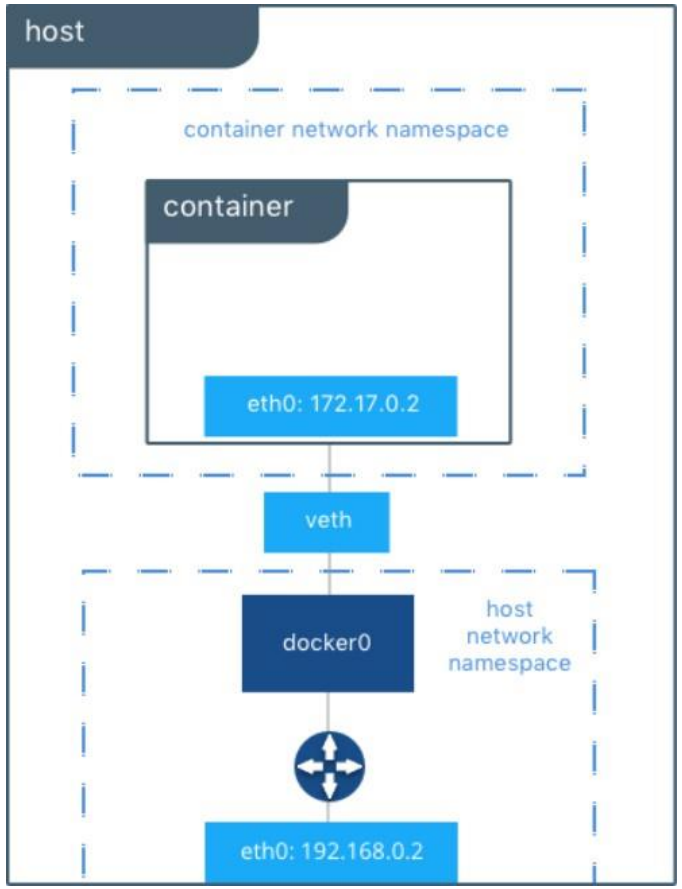
Bridge Network

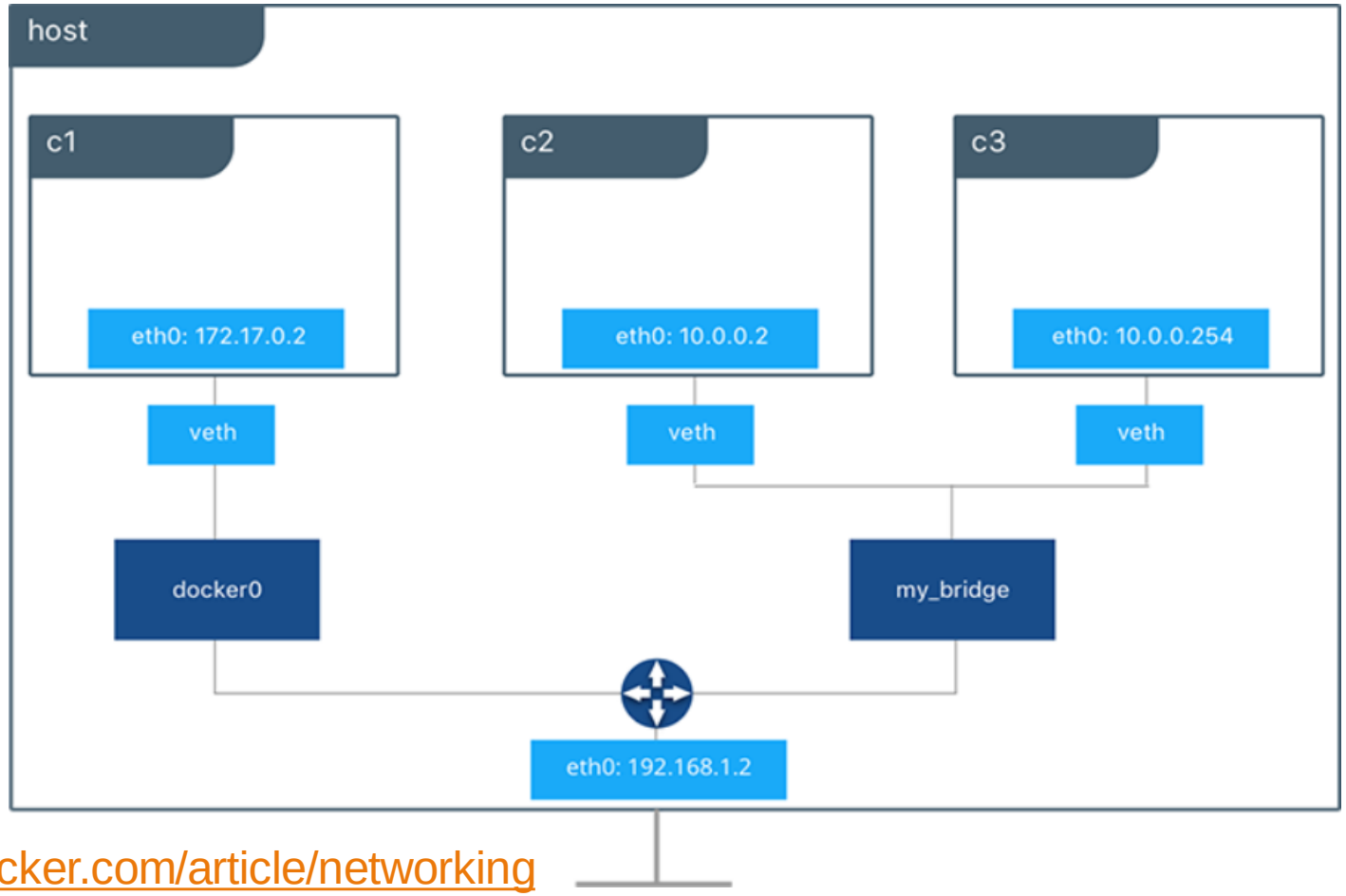
- Containers on different network cannot communicate with each other
- Software bridge allows containers connected to the same bridge network to communicate
- Provides isolation from containers not connected to the same bridge
- Applicable to containers running on the same Docker daemon host

- Default bridge
  - Does not have in built DNS
  - Explicit link needs to be created to connect containers

- User Defined Bridge
  - Expose all ports to each other not to external world
  - Can access each other by IP addresses, can resolve each other by name
  - Containers can be attached and detached during lifetime

Bridge Network (continued…)





https://success.docker.com/article/networking

Exercise: Bridge Network

- docker container run -d --name db-host mongo
- docker container run -d --name mcqs-app -p 8080:8080 tomcat
- docker container ls
- docker logs XXXX
- docker container run -d --name my-app -p 8080:8080 --**link** db-host:**db-host** tomcat

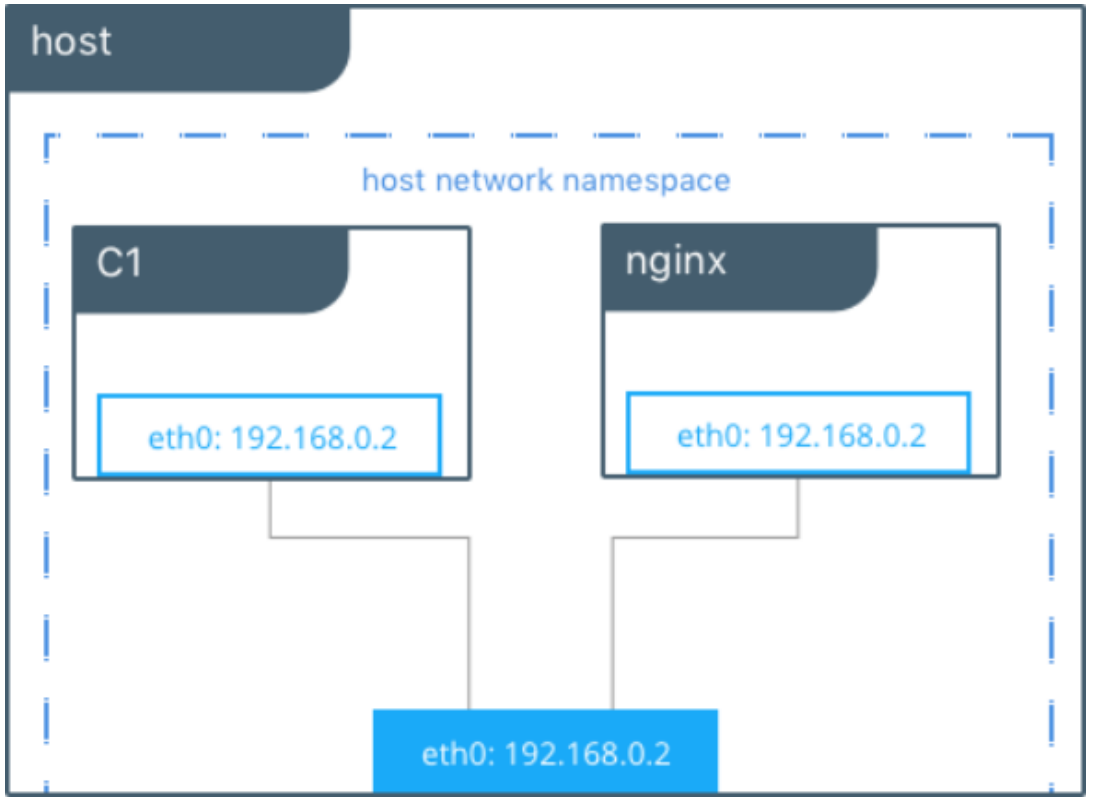
- docker network create --driver bridge mcqs-app-bridge
- docker container run -d --name db-host --network mcqs-app-bridge mongo
- docker container run -d --name my-app -p 8080:8080 --network mcqs-app-bridge tomcat
- docker network inspect XXXX
- docker logs XXXX

Exercise: Bridge Network

- curl -v -F id="0003“ -F subject="java" -F question="What is the output of this program, please tell?“ -F choices="2 2, 3 3, Runtime Error, Compilation Error“ http://XXXXXXX/mcqs
- curl http://XXXXXXX/mcqs/java
- curl -X DELETE http://XXXXXXX/mcqs/java

Host Networks

- Container's network stack not isolated from Docker host
- Container does not get own IP address
- Only work on Linux hosts
- Used where container needs to handle a large range of ports



https://success.docker.com/article/networking

Exercise: Host Network

- docker run -d --network host nginx

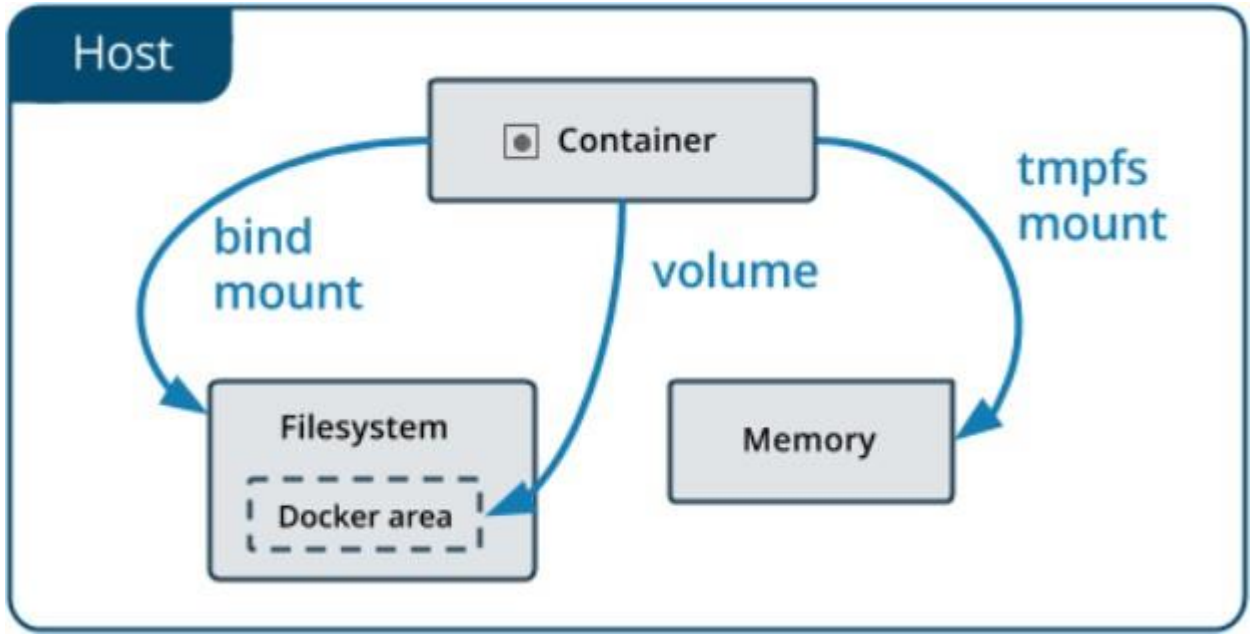
Overlay Networks

- Distributed network across multiple Docker daemon hosts
- Sits on the top of host specific network, allows containers connected to it communicate
- On initialization of swarm or joining Docker host to swarm overlay (ingress) network gets created

Storage

Data Persistence

- Data doesn't persist when that container no longer exists
- Tight coupling between container's R/W layer and underlying host
- Driver required for container's R/W layer reduces performance



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

Volume

- A new directory is created within Docker's storage directory on the host machine
- Docker manages directory's contents
- Can be mounted into multiple containers simultaneously
- Not removed automatically
- Store your container's data on a remote host or a cloud provider
- Back up, restore, or migrate data from one Docker host to another

Exercise: Volume

- docker container run -d --name=volume_test--mount source=vol1,target=/app nginx:latest
- docker container ls
- docker container inspect XXX
- docker container exec -it XXX /bin/bash
- ls -ltr
- cd /app
- echo "test volume" >> test_volume.txt
- exit
- docker container rm -f XXX
- cat /var/lib/docker/volumes/vol1/_data/test_volume.txt

Bind Mounts

- File or directory on the host machine mounted into a container
- File or directory is referenced by its full or relative path on the host machine
- File or directory does not need to exist on the Docker host already. It is created on demand if it does not yet exist.
- Performant, rely on the host machine's file system
- Possible to change the host filesystem via processes running in a container
- Sharing configuration files, source code from the host machine to containers e.g. maven target dir

Exercise: Bind Mount (httpd)

- docker container run -d -p 8080:80 httpd
- curl localhost:8080
- touch index.html
- vi index.html
- Update index.html with
- <html><body><h1>Its overwritten</h1></body></html>
- cd /
- chmod –R 777 ./root

- docker container run -d -p 9080:80 --mount type=bind,source="$(pwd)",target=/usr/local/apache2/htdocs httpd
- curl localhost:9080
- Update index.html again
- curl localhost:9080

**click on the publish port link available on Play-with-docker*

Demo: Bind Mount (TcPMM)

- Execute image as is, log levels set to 0.
- sudo docker container run -d -p 8080:90 tcpmm114:1.1
- sudo docker container ls
- sudo docker logs XXXX
- sudo docker container exec XXXX -it /bin/bash

- cd /home/prakash/workdir/tcpmm/test/bin
- sudo docker container run -d -p 8085:90 --mount type=bind,source="$(pwd)",target=/apps/tcpmm114/tcbom_server/bin tcpmm114:1.1
- sudo docker logs XXXX
- sudo docker container exec -it XXXX /bin/bash
- Change logging settings in ebom.xml
- sudo docker container stop XXXX
- sudo docker container start XXXX

tmpfs

- In-memory i.e. non persistent
- Available during life-time of container
- Used to store non consistent or sensitive information

Exercise: tmpfs

- docker container run -d --name=tmpfs_test --mount type=tmpfs,destination=/app nginx:latest
- docker container ls
- docker container inspect XXX
- docker container exec -it XXX /bin/bash
- ls -ltr
- cd /app
- echo "test tmpfs" >> test_tmpfs.txt
- cat test_tmpfs.txt
- exit
- docker container rm -f XXX

Docker Compose

Docker Compose

- Easiest way to deploy an application on a single host
- Tool for defining and running multi-container Docker applications
- Use a YAML file to configure your application's services
- Versions: 1, 2, 2.1, 3, 3.1
- Development and Testing
  - Isolated environment with multiple services in a single step
  - Deploy and test changes on complete stack
  - Automates test suites

Exercise: Docker Compose

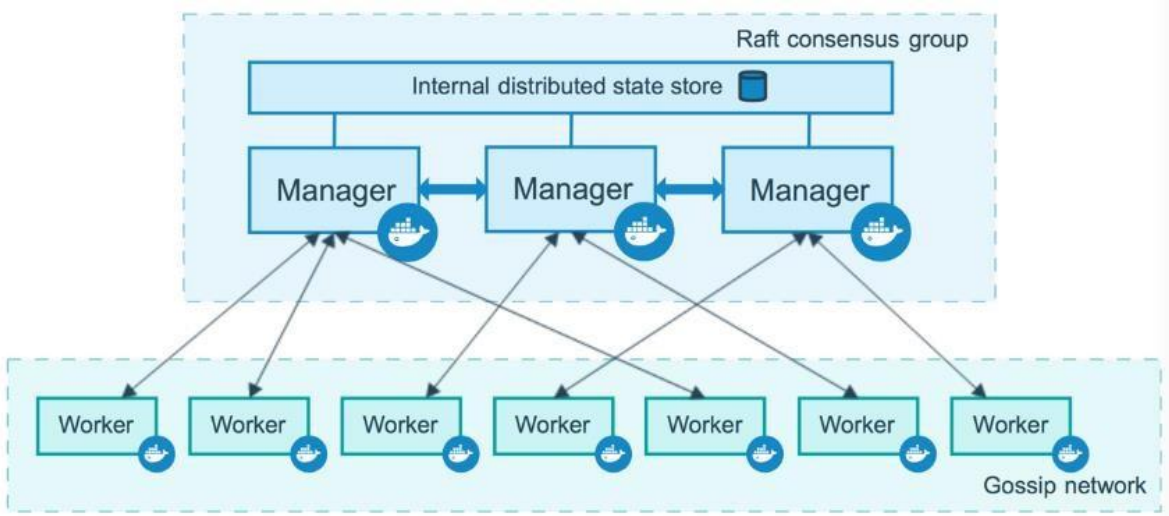
```
version: "3.0"
services:
  dockerapp:
    build: .
    ports:
      - "5000:5000"
    depends_on:
      - redis
  redis:
    image: redis:3.2.0
```

- docker-compose up -d
- http://<host>:<port>/mcqs
- docker-compose images
- docker-compose top
- docker-compose logs
- docker-compose config
- docker-compose down
- https://github.com/prakashk0301/dockerapp/blob/master/docker-compose.yml

Docker Swarm

Swarm

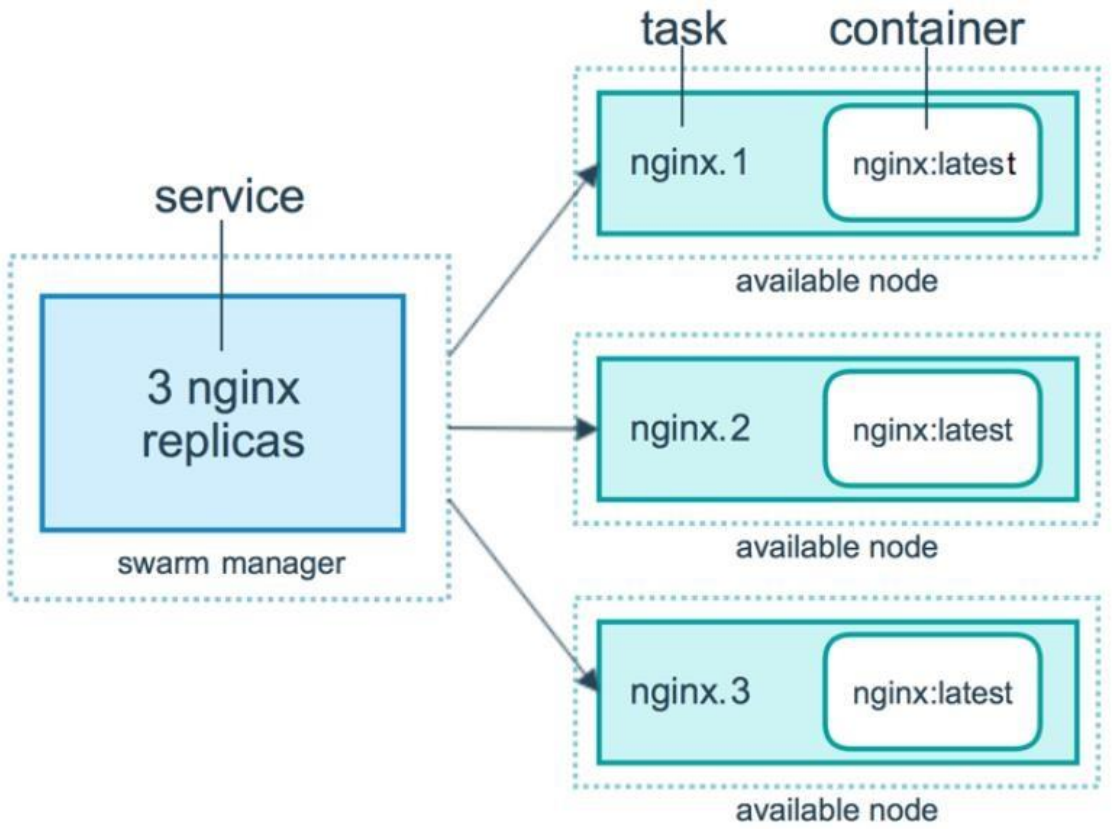
- Cluster management and orchestration feature embedded in Docker Engine
- Multiple Docker hosts which run in swarm mode
- Node acts as manager or worker or both
- Updated configuration change without need to manually restart service – network, volume etc
- Service definition submitted to a manager node, manager node dispatches tasks to worker nodes
- Worker nodes receive and execute tasks dispatched from manager node
- Raft maintains internal state of swarm and services



https://docs.docker.com/engine/swarm/how-swarm-mode-works/nodes/

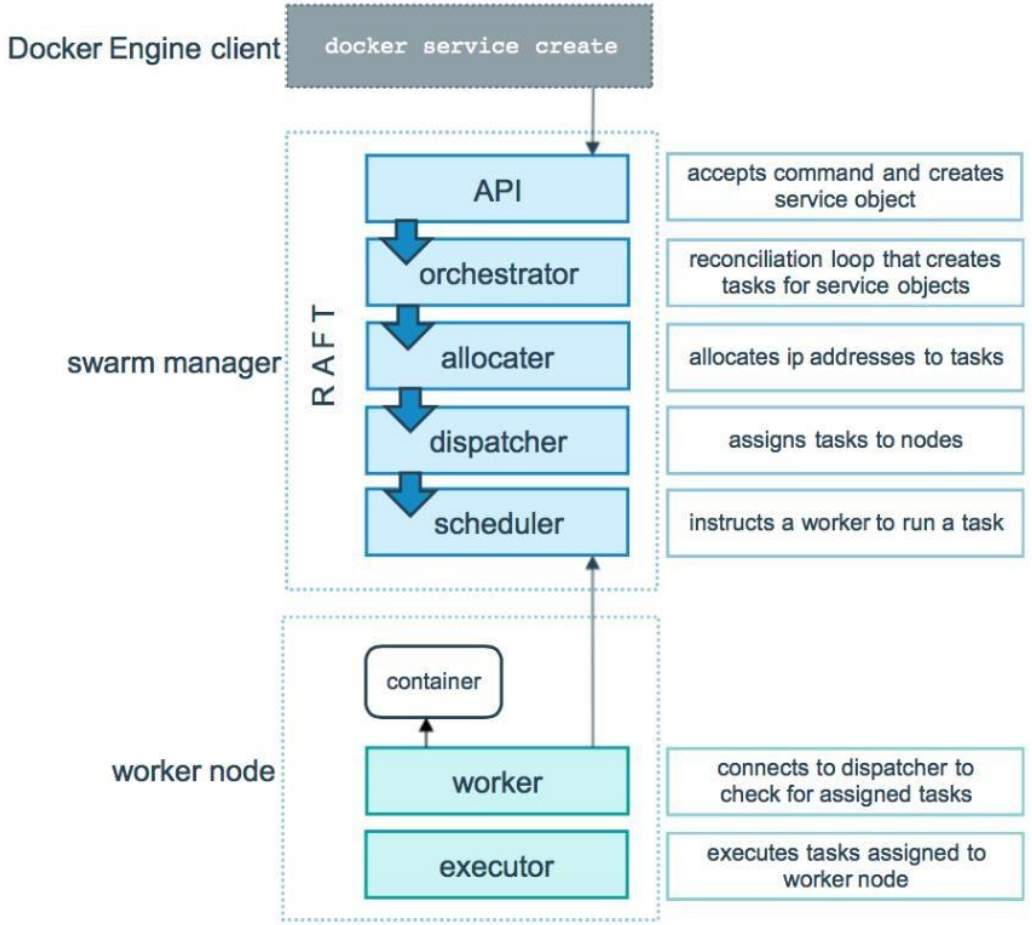
Services and Tasks

- Service is the definition of the tasks to execute on the manager or worker nodes.
- Replicated services model, the swarm manager distributes a specific number of replica tasks among the nodes based upon the scale
- Global services, the swarm runs one task for the service on every available node in the cluster
- Task carries a Docker container and the commands to run inside the container



https://docs.docker.com/engine/swarm/how-swarm-mode-works/services/

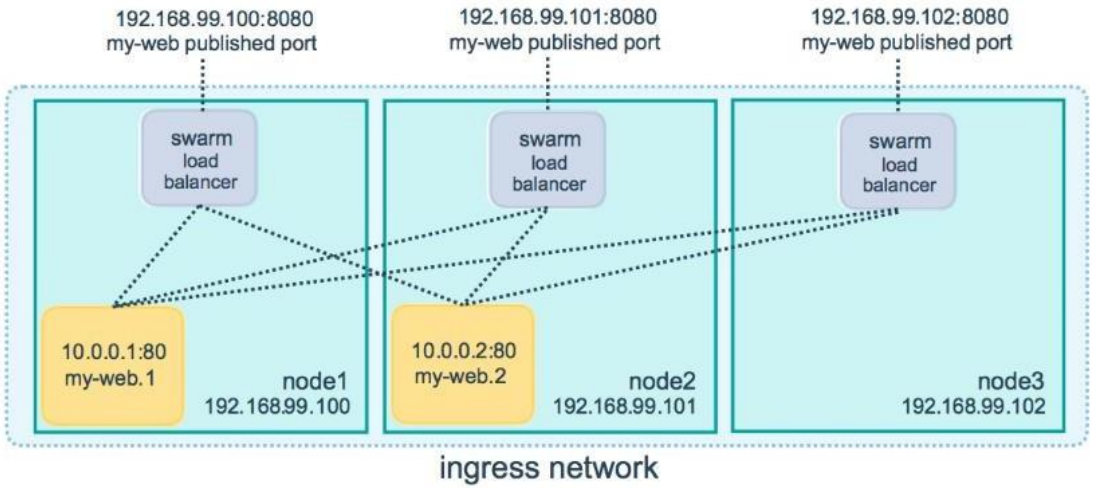
Tasks and Scheduling



https://docs.docker.com/engine/swarm/how-swarm-mode-works/services/

Load Balancing

- Uses ingress load balancing to expose the services to make available externally to the swarm
- Service gets automatically assigned a PublishedPort or it is configured explicitly for service
- Routing mesh enables each node in the swarm to accept connections on published ports for service
- Internal DNS component automatically assigns each service in the swarm a DNS entry
- TCP port 2377 : cluster management communications
- Port 7946 TCP/UDP : container network discovery.
- Port 4789 UDP : the container ingress network.



https://docs.docker.com/engine/swarm/ingress/

Exercise: Swarm

- Node I
  - docker swarm init --advertise-addr <LEADER HOST IP>
- Node II
  - docker swarm join --token <SWMTKN><LEADER HOST IP>:2377
- Node III
  - docker swarm join --token <SWMTKN><LEADER HOST IP>:2377

- Node I
  - docker service create -p 8080:80 --replicas 3 --name nginx-load-balanced nginx
  - docker service ls
  - docker service ps nginx-load-balanced
- Browse to http://<host>:<PORT>
- docker node ls

Docker Service

Docker Service

- Services are really just "***containers in production***."
- A service only runs ***one image***, but it codifies the way that image runs e.g. ports, replicas
- A single service stack running on a ***single host***
- Change number of running containers to scale the service
- Runs in swarm mode

- docker stack ls
- docker stack deploy -c <composefile> <appname>
- docker service ls
- docker service ps

Exercise: Docker Service

```
version: "3"
services:
    web:
        image: nginx
        deploy:
            replicas: 5
            resources:
                limits:
                    cpus: "0.1"
                    memory: 50M
            restart_policy:
                condition: on-failure
        ports:
            - "80:80"
        networks:
            - webnet
networks:
    webnet:
```

- docker swarm init --advertise-addr <IP ADDRESS>
- docker stack deploy -c docker-compose.yml service-stack
- docker service ls
- docker service ps
- docker stack services service-stack

QA

References

- Docker

https://docs.docker.com/get-started/

- Cgroups, namespaces, chroot, UnionFS

https://youtu.be/sK5i-N34im8

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/resource_management_guide/ch01

http://man7.org/linux/man-pages/man7/cgroups.7.html

http://man7.org/linux/man-pages/man7/namespaces.7.html

http://man7.org/linux/man-pages/man2/chroot.2.html

https://stackoverflow.com/questions/32775594/why-does-docker-need-a-union-file-system

- Hyper-v

https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/reference/hyper-v-architecture

References (continued…)

- Open Container Initiative

https://containerd.io/

- Docker Swarm Reference Architecture: Exploring Scalable, Portable Docker Container Networks

https://success.docker.com/article/networking

- Demystifying Docker overlay networking

http://blog.nigelpoulton.com/demystifying-docker-overlay-networking/

- Docker Mastery: with Kubernetes +Swarm from a Docker Captain

https://www.udemy.com/share/101WekAkAccVlUQXQ=/