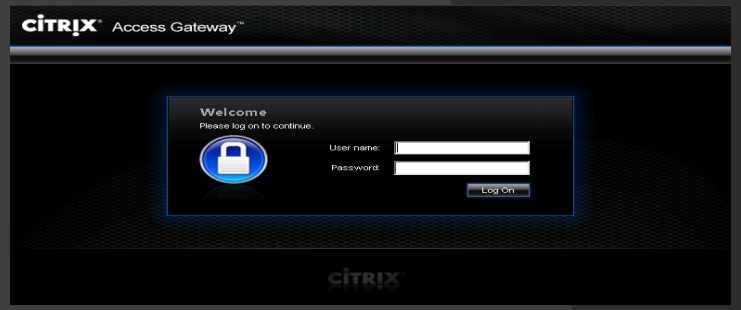
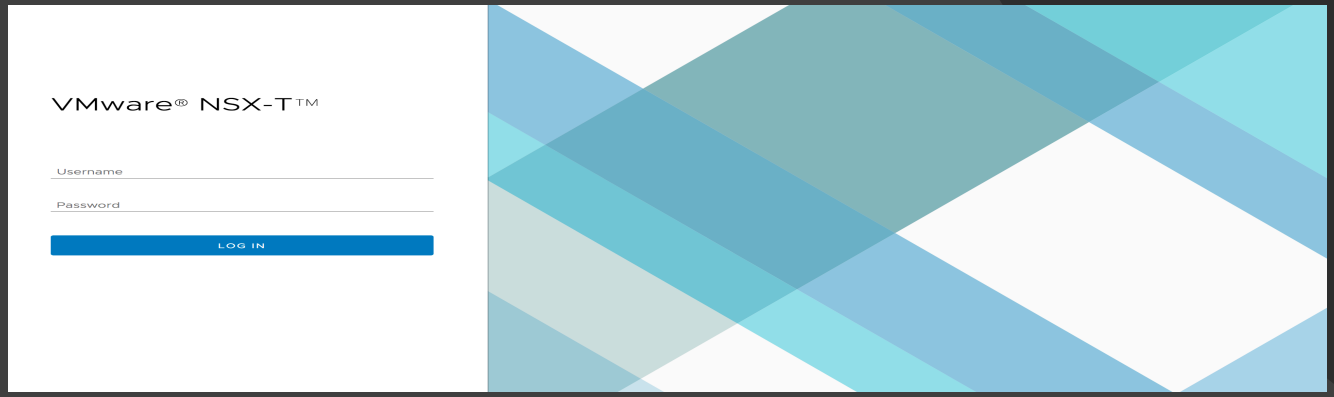
Docker Architecture & Docker Networking

Hussaina Begum N.

Staff Engineer, VMware

About me



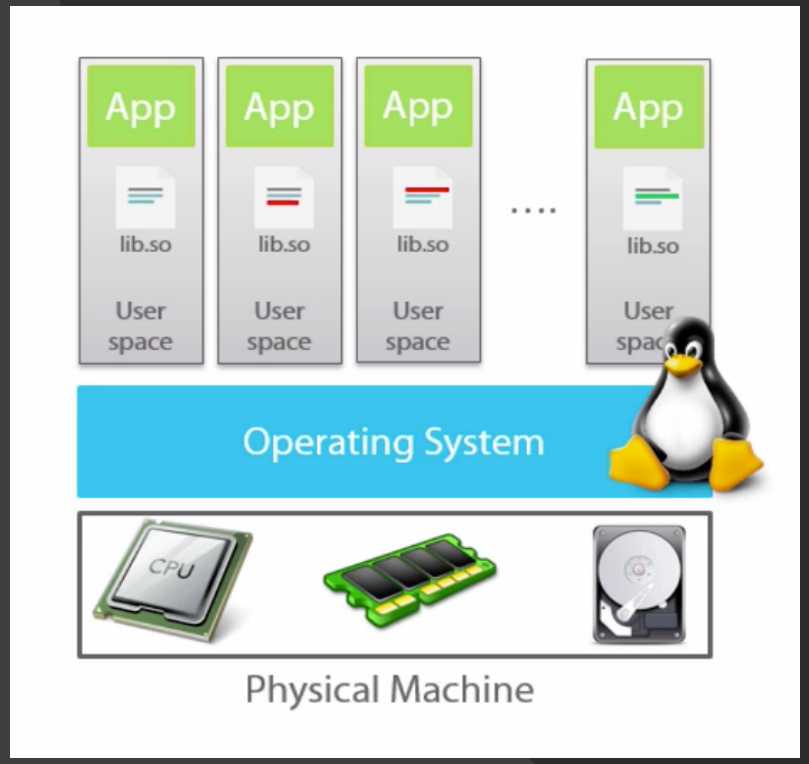


Agenda

- History of Compute Virtualizaion
- Linux Containers(LXC)
- Microservices architecture
- What are Dockers ?
- Docker Architecture
- Docker Hands-On
- Docker networking
- Single-Host, Multi-Host networking demos
- Q&A

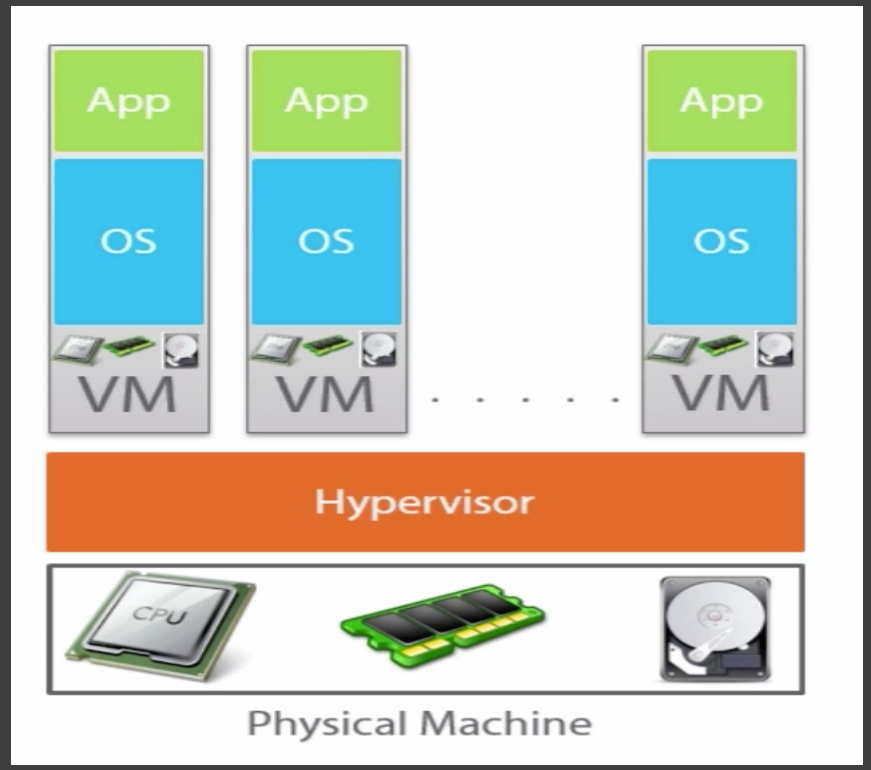
Long time ago in a galaxy far far away …

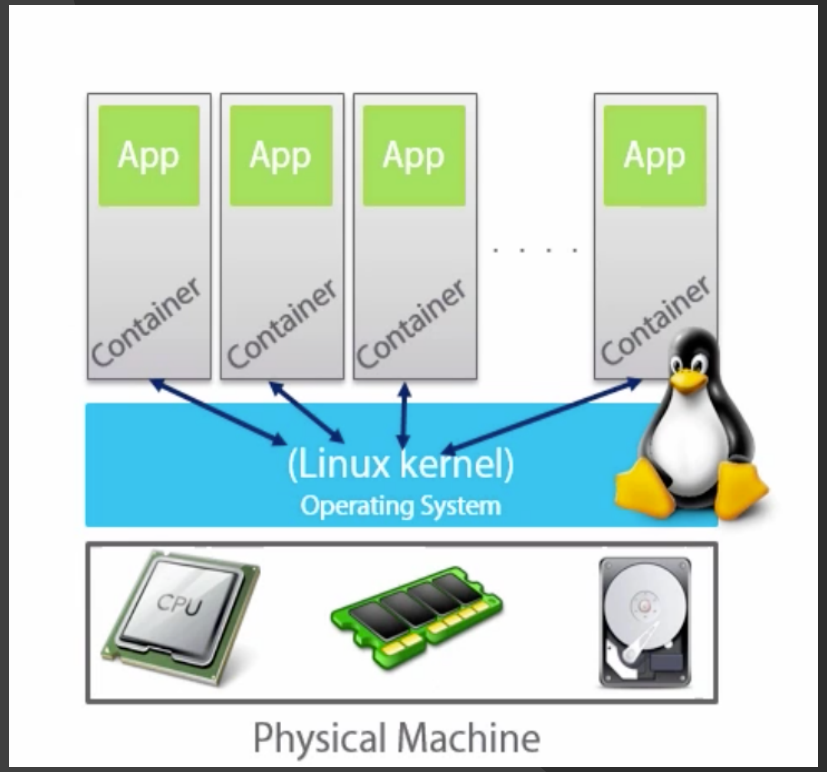
- Physical servers
- Monolithic applications
- Scaling is hard and expensive



History

- VMs vs. containers





@nigelpoulton

Linux Containers (LXC)

- Something in between chroot and VM
- Uses the below constructs to achieve containers
  - Kernel namespaces (ipc, uts, mount, pid, network and user)
  - Apparmor and SELinux profiles
  - Seccomp policies
  - Chroots (using pivot_root)
  - Kernel capabilities
  - Cgroups (control groups)

Linux Containers (LXC) → Docker

- LXC + aufs → dotCloud (dc)
- Changes to LXC broke dc
- dc came up with libcontainer abstraction
- dc -> docker daemon – 2013
- docker daemon is bloated(monolith)
- Open Container Initiative (runtime spec, image spec) - 2015
- docker refactoring

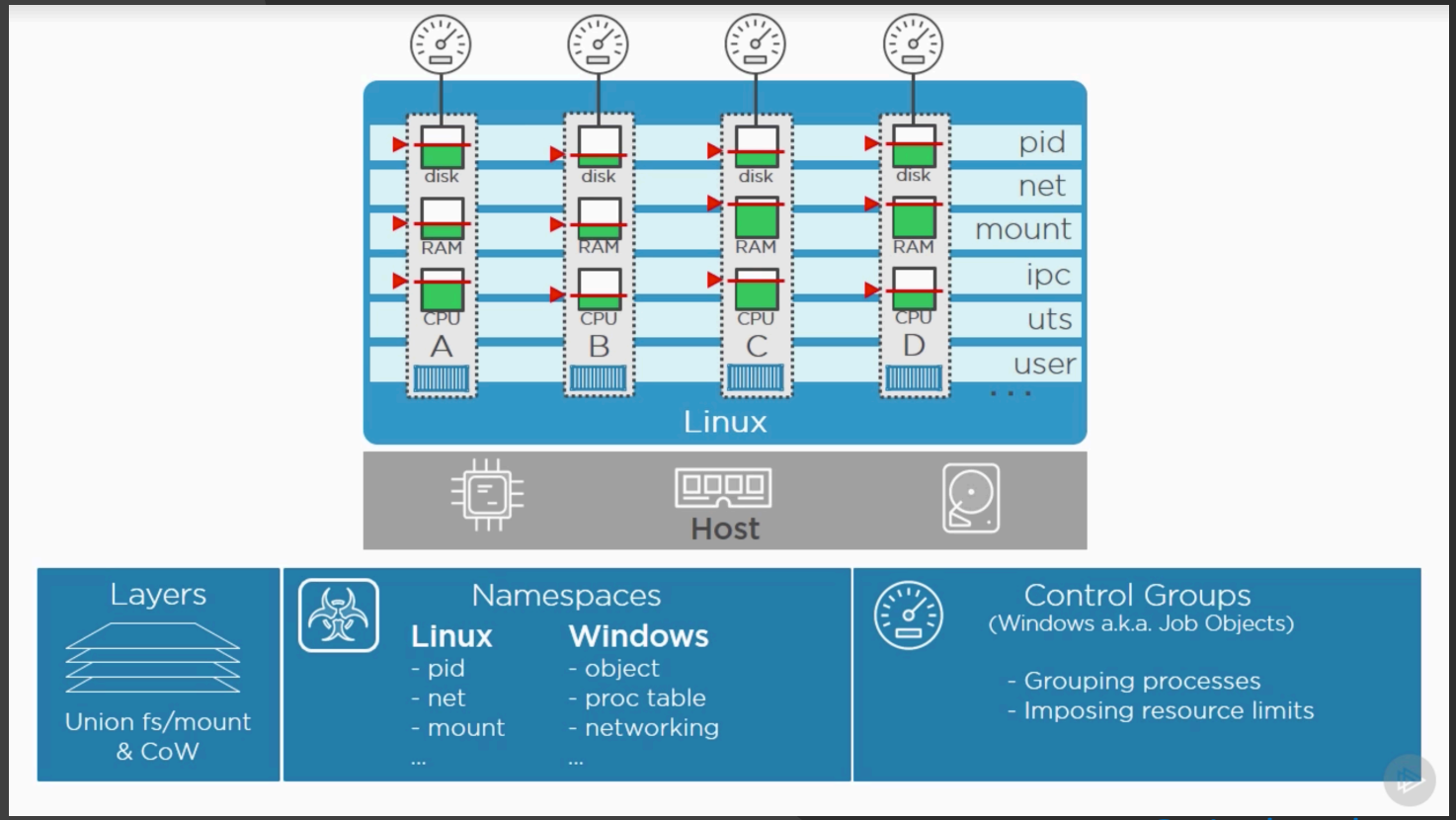
Microservices Architecture

- Monolithic applications are
  - Inflexible
  - Unscalable
  - Unreliable
  - Slow development(same framework for all components)
- Microservices
  - Small autonomous services
  - Independent and loosely coupled components
  - Can scale each component
  - Stateless components
  - Redeployment
  - Small focused, loosely coupled, language independent, bounded context

https://blog.newrelic.com/technology/microservices-what-they-are-why-to-use-them/

Dockers

- OS level virtualization
- Shares host kernel
- Multi-layered FS
  - Union FS(overlay2), mount, CoW
- Namespaces
  - pid, net, mnt, ipc, uts, user
- Cgroups
  - Grouping procs, rlimits



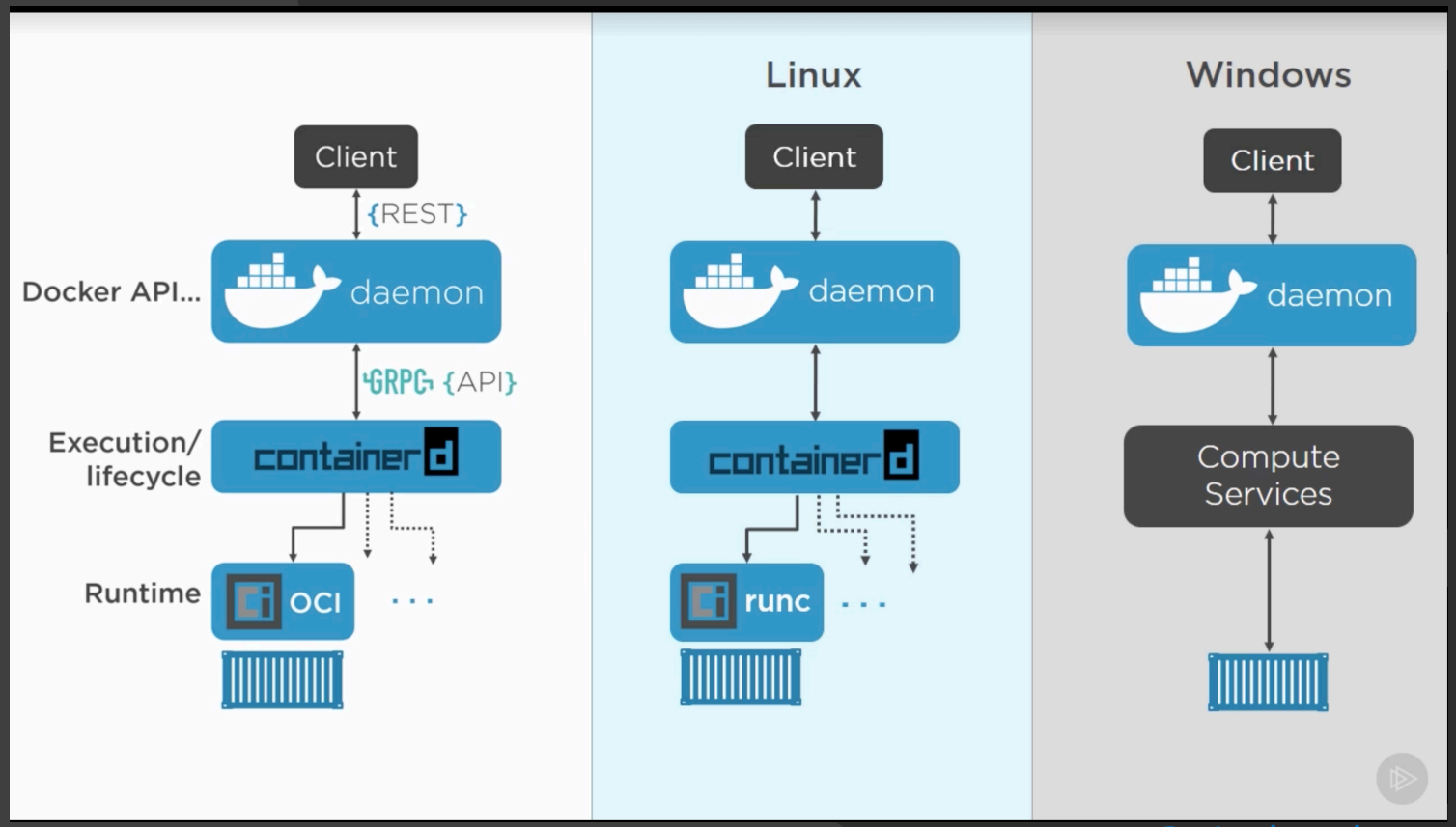
@nigelpoulton

Docker on Windows/Mac

- Docker Toolbox (not a native solution)
  - Virtual Env, boot2docker, Docker Engine, Docker Machine, Docker compose
  - Network, storage is not shared from host
- Mac Native Docker (2016)
  - HW virtualization, Embedded Linux, unikernel, docker engine
- Windows Native Docker(2016)
  - Windows Server based
    - Leverages namespace, cgroup of windows kernel
    - Host kernel is shared
  - Hyper-V based
    - Containers run in highly optimized VM
    - Host kernel is not shared

Docker Engine

- API server
- Docker daemon
- containerd
- Runtime
  - runc
  - Rkt – retired
  - Cri-o (for k8s)
  - kata-runtime



Docker workflow

- Image
- Container
- Registry

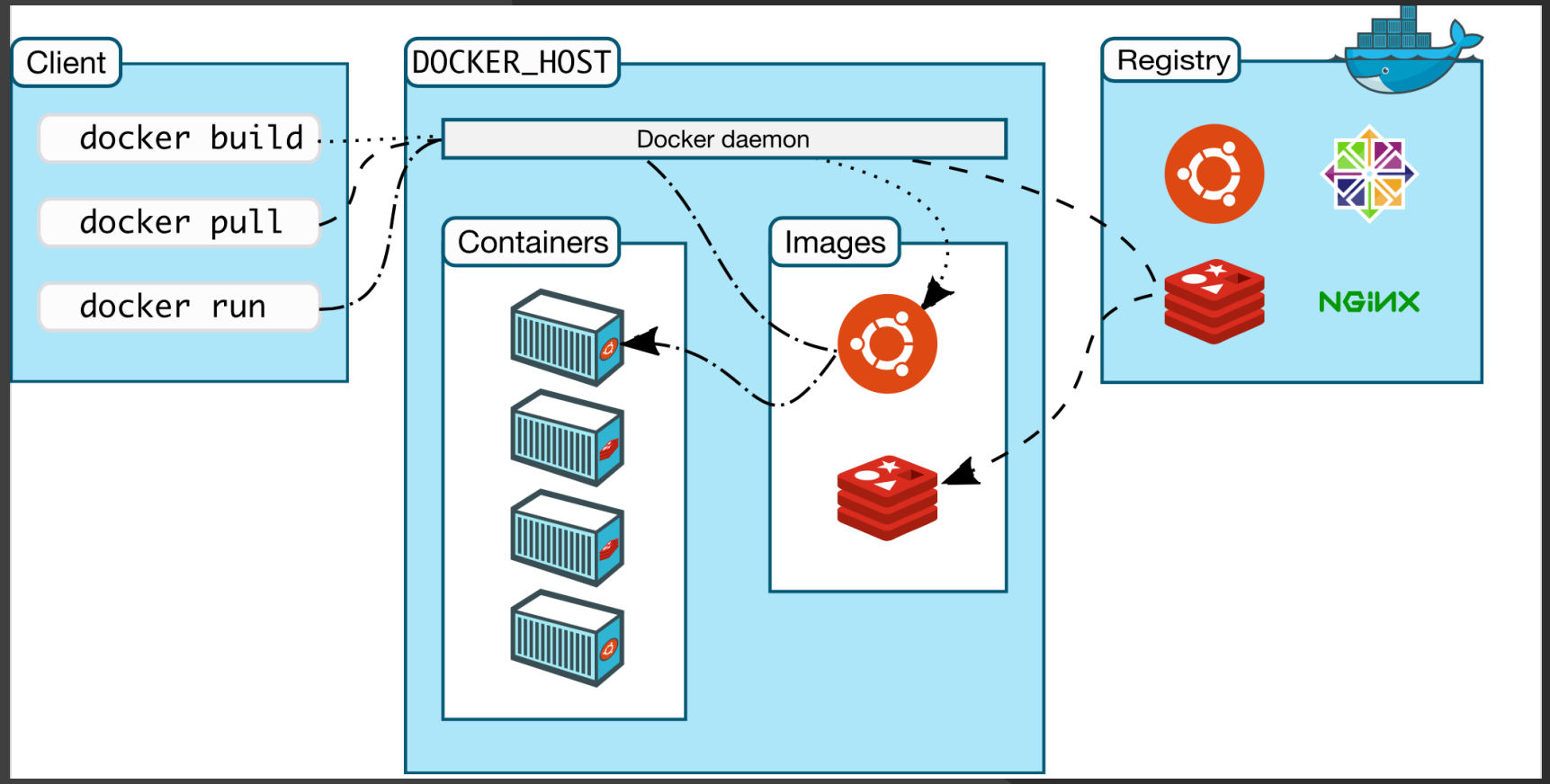


Image @docker.com

Dockers – Install/HandsOn

- Docker install on Ubuntu/Mac/Windows
- docker info – lists the docker engine details on the host(client and server versions)
- docker pull <image> – pulls image to docker host
- docker run – starts a new container
- docker stop – stops the container
- docker ps – lists current running containers
- docker images – lists the images on docker host
- docker rmi – removes image from docker host

Dockerfile

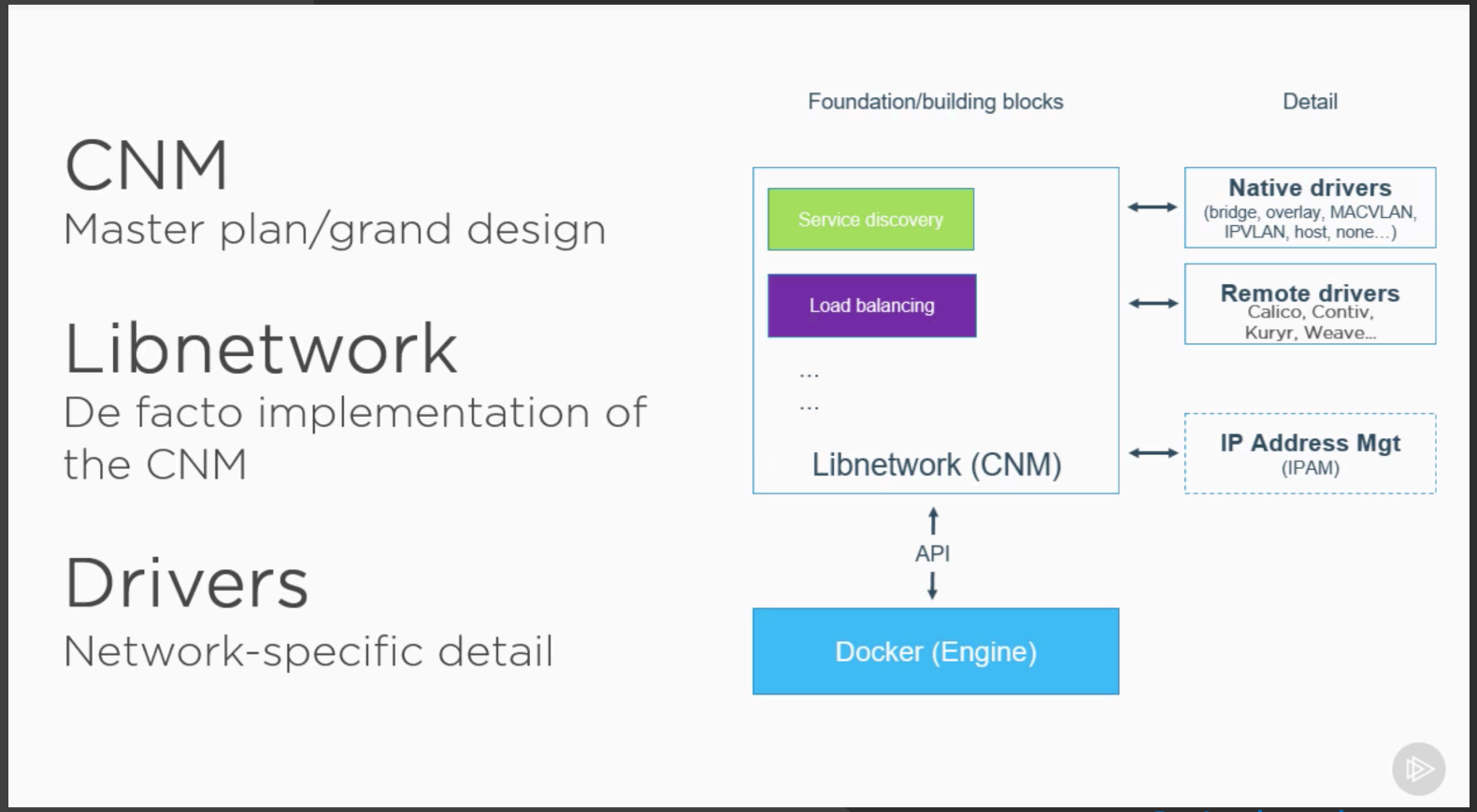
- CAPITALIZED instructions
- <INSTRUCTION> - <value>
- FROM
- LABEL
- RUN
- COPY
- WORKDIR
- ENTRYPOINT
- CMD

Container Networking Models

- CNM – by Docker
  - Sandbox
  - Endpoint
  - Network
- CNI – by coreOS (approved by CNCF)
  - CNI Specification - https://github.com/containernetworking/cni/blob/master/SPEC.md
  - Plugins
  - Library

Dockers – Networking

- CNM
  - Sandbox – namespaces
  - Endpoint - interfaces
  - Network – connected endpoints
- Libnetwork
- Drivers
  - host
  - bridge
  - none
  - Remote(overlay, macvlan)
  - plugins



Dockers – Networking CLIs

- docker network ls
- docker info
- docker network create
- docker network connect
- docker network disconnect

Dockers – Single Host Networking

Drivers that are available on host
- Bridge(default)
- Host
- None

1. Create two containers
2. Create a bridge n/w with subnet
3. Attach bridge n/w to two containers
4. Containers can talk to each other over bridge n/w. i.e, by ip or by container-name

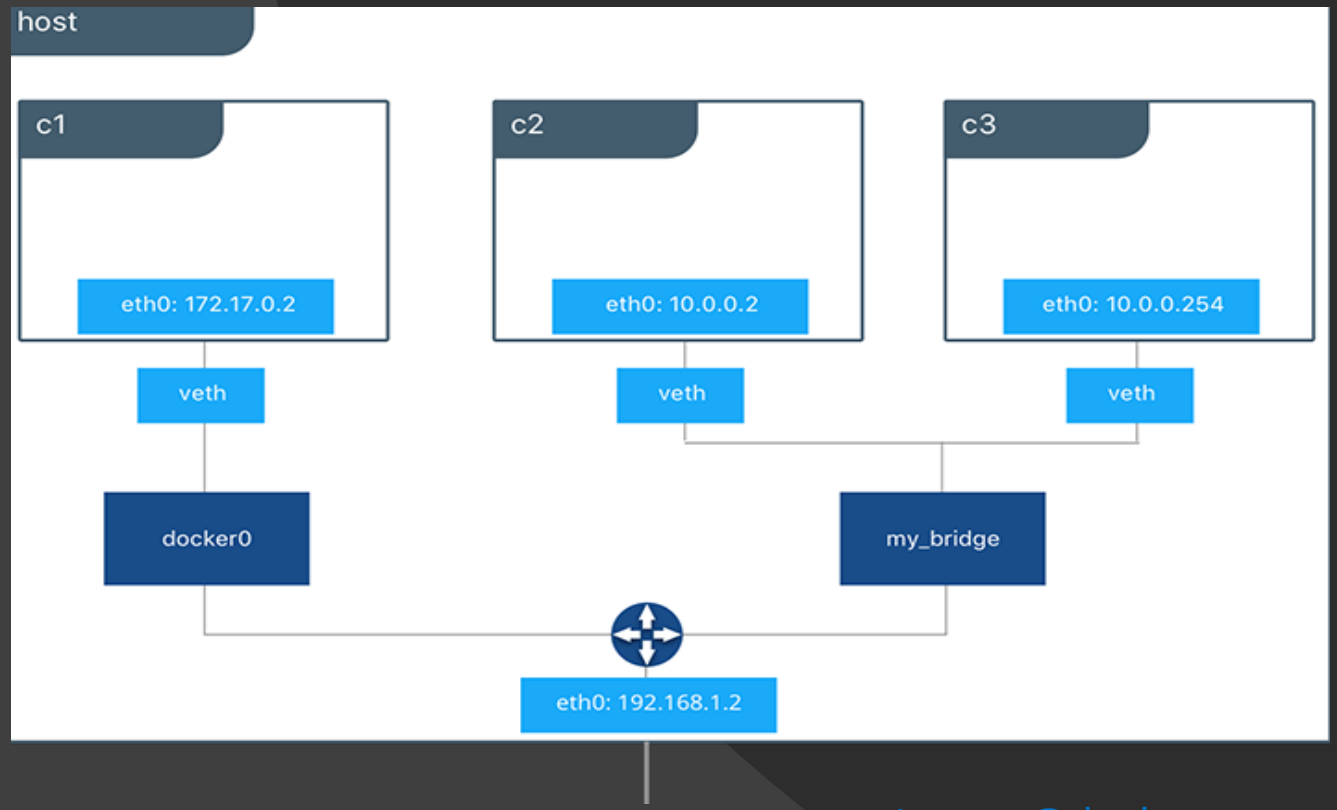
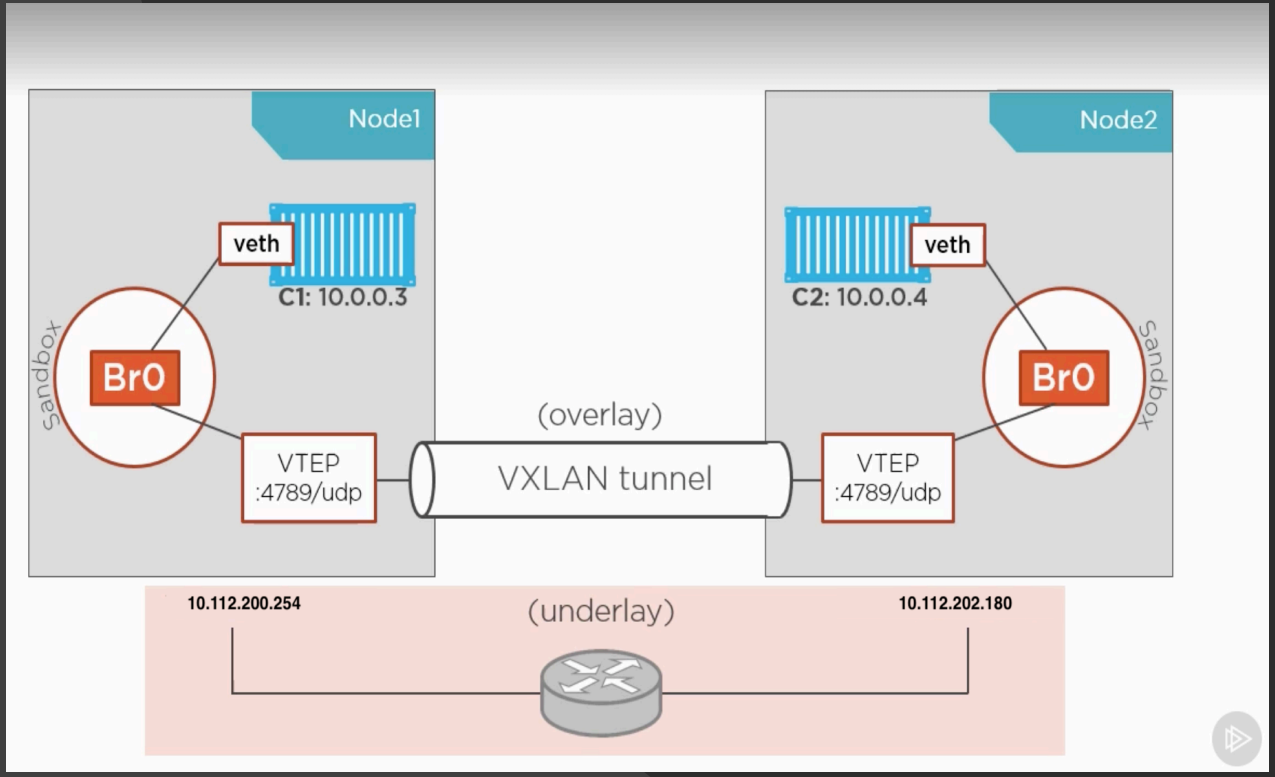


Image @docker.com

Dockers – Multi Host Networking

Overlay driver needs to be used for connectivity.

1. Create two containers on two different nodes
2. Node1 - docker swarm init
3. Node2 - docker swarm join
4. Create network using ‘overlay’ driver
5. Create a docker service with 2 replicas
6. Containers can talk to each other, using overlay network



Dockers – Orchestration

- docker compose(single host)
  - -link command
- docker swarm (multi-host)
  - swarm init
  - swarm join
- kubernetes

References

- https://docs.docker.com
- https://success.docker.com/article/networking
- Pluralsight courses
  - https://app.pluralsight.com/profile/author/nigel-poulton
    - Docker Deep Dive
    - Docker Networking
- Linux Foundation courses
  - Container Fundamentals (LFS253) - https://vmware.sabacloud.com/Saba/Web_spf/NA1PRD0121/common/ledetail/cours000000000035007
  - Kubernetes Fundamentals - https://vmware.sabacloud.com/Saba/Web_spf/NA1PRD0121/common/ledetail/cours000000000035010

Thank you!!
Questions ?