Containerization

Introduction to Containers, Docker and Kubernetes

EECS 768

Apoorv Ingle

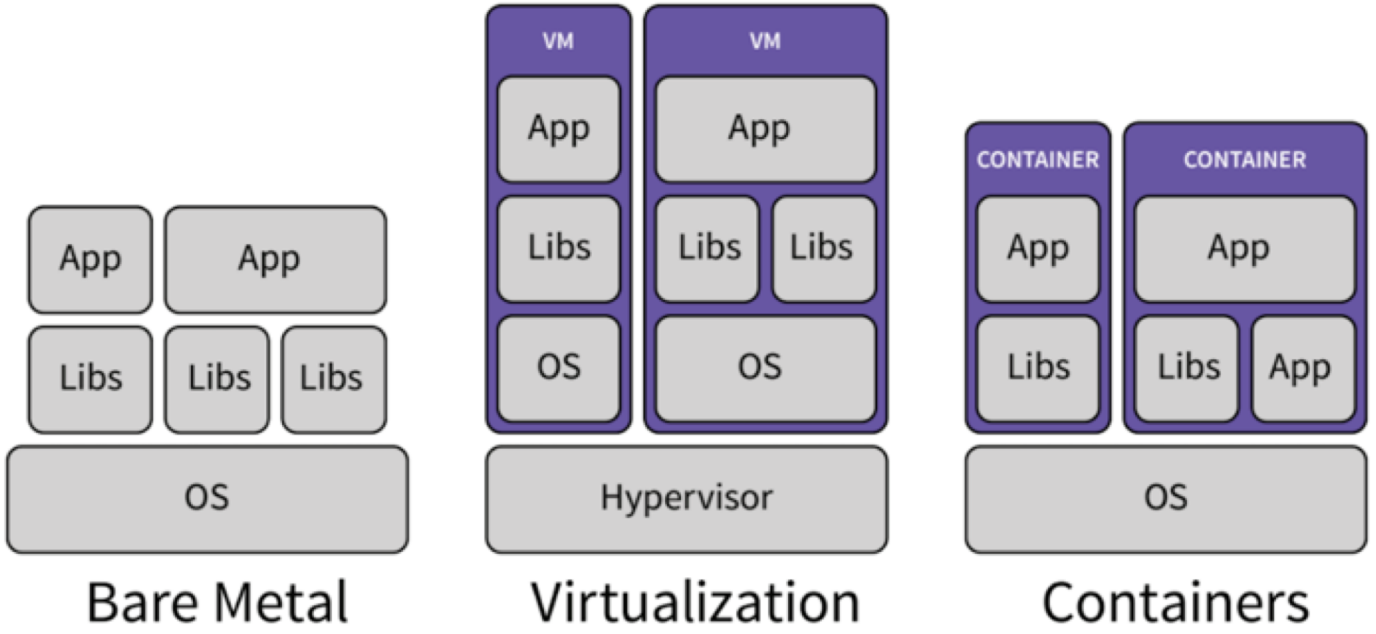
ani@ku.edu

Containers

- Containers – lightweight VM or chroot on steroids
  - Feels like a virtual machine
    - Get a shell
    - Install packages
    - Run applications
    - Run services
  - But not really
    - Uses host kernel
    - Cannot boot OS
    - Does not need PID 1
  - Process visible to host machine

Containers

- VM vs Containers



Containers

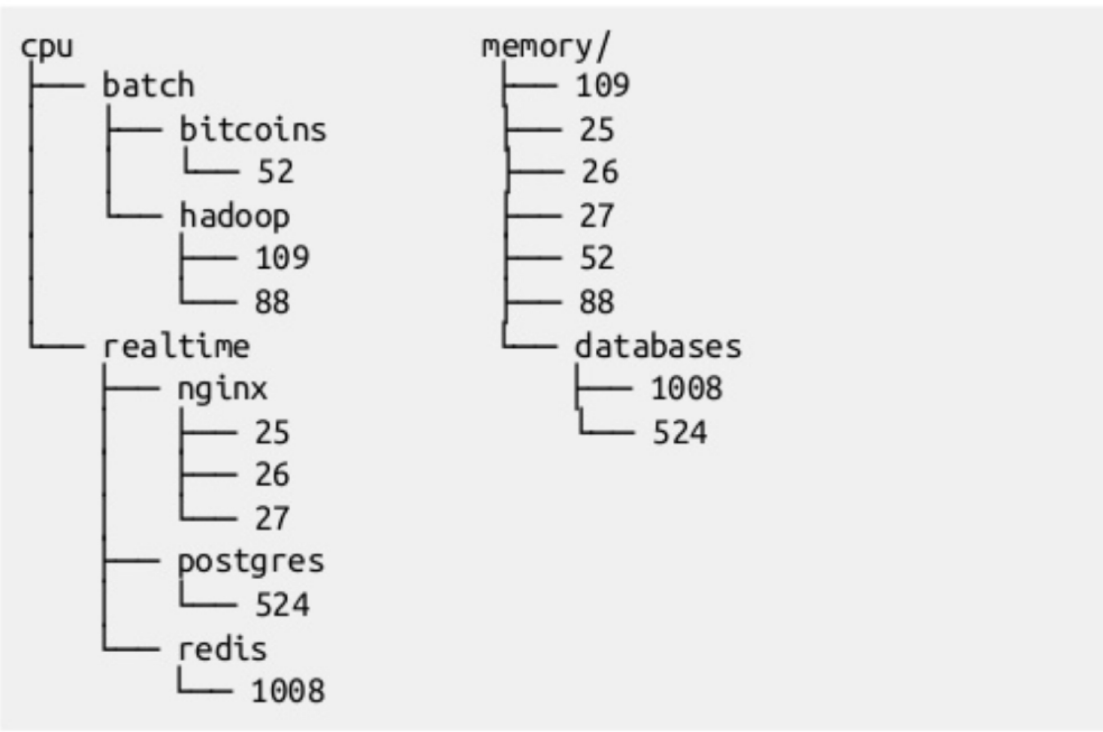
- Container Anatomy
  - cgroup: limit the use of resources
  - namespace: limit what processes can see (hence use)

Containers

- cgroup
  - Resource metering and limiting
    - CPU
    - IO
    - Network
    - etc..
  - $ ls /sys/fs/cgroup

Containers

- Separate Hierarchies for each resource subsystem (CPU, IO, etc.)
  - Each process belongs to exactly 1 node
  - Node is a group of processes
    - Share resource



Containers

- CPU cgroup
  - Keeps track
    - user/system CPU
    - Usage per CPU
  - Can set weights
- CPUset cgroup
  - Reserve to CPU to specific applications
  - Avoids context switch overheads
  - Useful for non uniform memory access (NUMA)

Containers

- Memory cgroup
  - Tracks pages used by each group
  - Pages can be shared across groups
  - Pages “charged” to a group
  - Shared pages “split the cost”
  - Set limits on usage

```
a553i967@cycle3 ~ $ cat /proc/1/cgroup
11:memory:/init.scope
10:pids:/init.scope
9:devices:/init.scope
8:blkio:/init.scope
7:hugetlb:/
6:cpuset:/
5:cpu,cpuacct:/init.scope
4:net_cls,net_prio:/
3:perf_event:/
2:freezer:/
1:name=systemd:/init.scope
```

Containers

- Namespaces
  - Provides a view of the system to process
  - Controls what a process can see

  - Multiple namespaces
    - pid
    - net
    - mnt
    - uts
    - ipc
    - usr

Containers

- PID namespace
  - Processes within a PID namespace see only process in the same namespace
  - Each PID namespace has its own numbering staring from 1
  - Namespace is killed when PID 1 goes away
  - Nesting of namespaces possible
    - Each process gets a multiple PID depending on the namespace
- Mnt namespace
  - choot – each process gets its own root

Containers

- Namespaces
  - <ns>:[<inode>]
  - Same inode => same ns
- Namespaces manipulation
  - $ nsenter

```
a553i967@cycle3 ~ $ ps
  PID TTY          TIME CMD
24177 pts/66   00:00:00 bash
24183 pts/66   00:00:02 zsh
27919 pts/66   00:00:00 emacs
28901 pts/66   00:00:00 ps
a553i967@cycle3 ~ $ ll /proc/24183/ns
total 0
lrwxrwxrwx 1 a553i967 a553i967_g 0 Apr 16 23:25 cgroup -> cgroup:[4026531835]
lrwxrwxrwx 1 a553i967 a553i967_g 0 Apr 16 23:25 ipc -> ipc:[4026531839]
lrwxrwxrwx 1 a553i967 a553i967_g 0 Apr 16 23:25 mnt -> mnt:[4026531840]
lrwxrwxrwx 1 a553i967 a553i967_g 0 Apr 16 23:25 net -> net:[4026531957]
lrwxrwxrwx 1 a553i967 a553i967_g 0 Apr 16 23:25 pid -> pid:[4026531836]
lrwxrwxrwx 1 a553i967 a553i967_g 0 Apr 16 23:25 user -> user:[4026531837]
lrwxrwxrwx 1 a553i967 a553i967_g 0 Apr 16 23:25 uts -> uts:[4026531838]
a553i967@cycle3 ~ $ ll /proc/27919/ns
total 0
lrwxrwxrwx 1 a553i967 a553i967_g 0 Apr 17 00:36 cgroup -> cgroup:[4026531835]
lrwxrwxrwx 1 a553i967 a553i967_g 0 Apr 17 00:36 ipc -> ipc:[4026531839]
lrwxrwxrwx 1 a553i967 a553i967_g 0 Apr 17 00:36 mnt -> mnt:[4026531840]
lrwxrwxrwx 1 a553i967 a553i967_g 0 Apr 17 00:36 net -> net:[4026531957]
lrwxrwxrwx 1 a553i967 a553i967_g 0 Apr 17 00:36 pid -> pid:[4026531836]
lrwxrwxrwx 1 a553i967 a553i967_g 0 Apr 17 00:36 user -> user:[4026531837]
lrwxrwxrwx 1 a553i967 a553i967_g 0 Apr 17 00:36 uts -> uts:[4026531838]
a553i967@cycle3 ~ $ cat /proc/24183/task/24183/children
27919 28931 %
a553i967@cycle3 ~ $
```

Containers

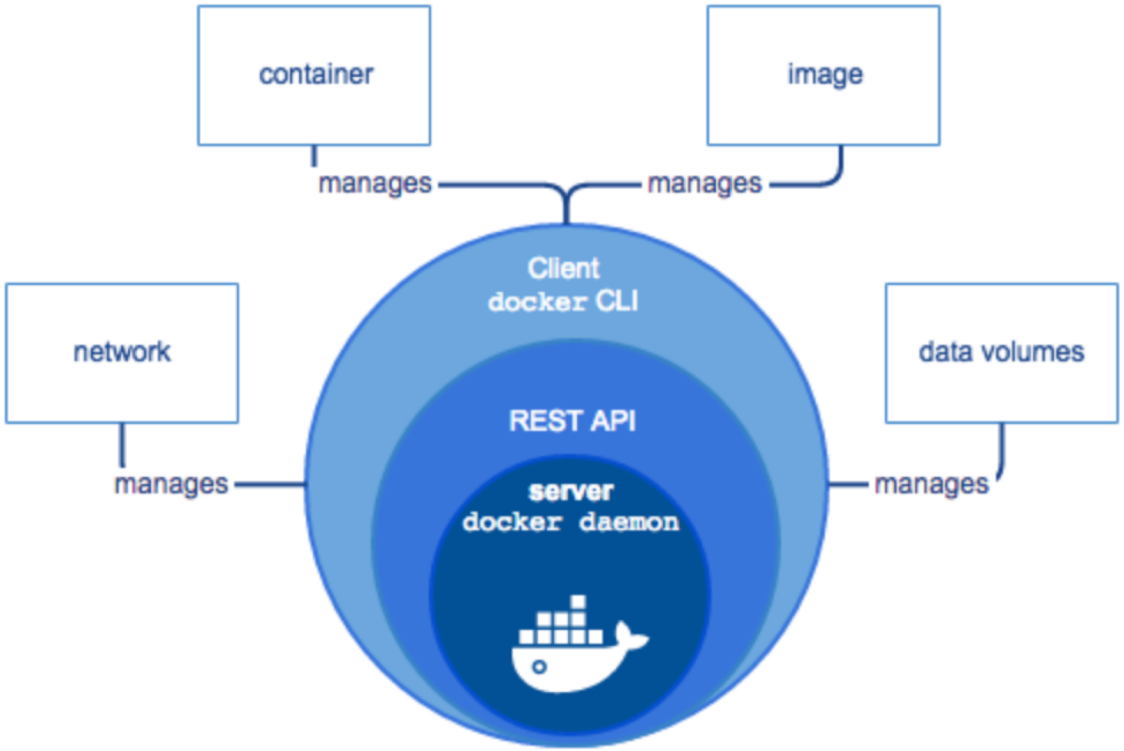
- cgroups and namespaces are orthogonal
- One can have systems
  - Use only cgroups
  - Or only name spaces
  - Or both depending on the use case
- Every process in current Linux system is containerized

Docker

- Manages lifecycle of containers
  - cgroups and namespace view is too low level
- Old version of docker based on LXC
- New version ships libcontainer/runc
  - Same concept different name

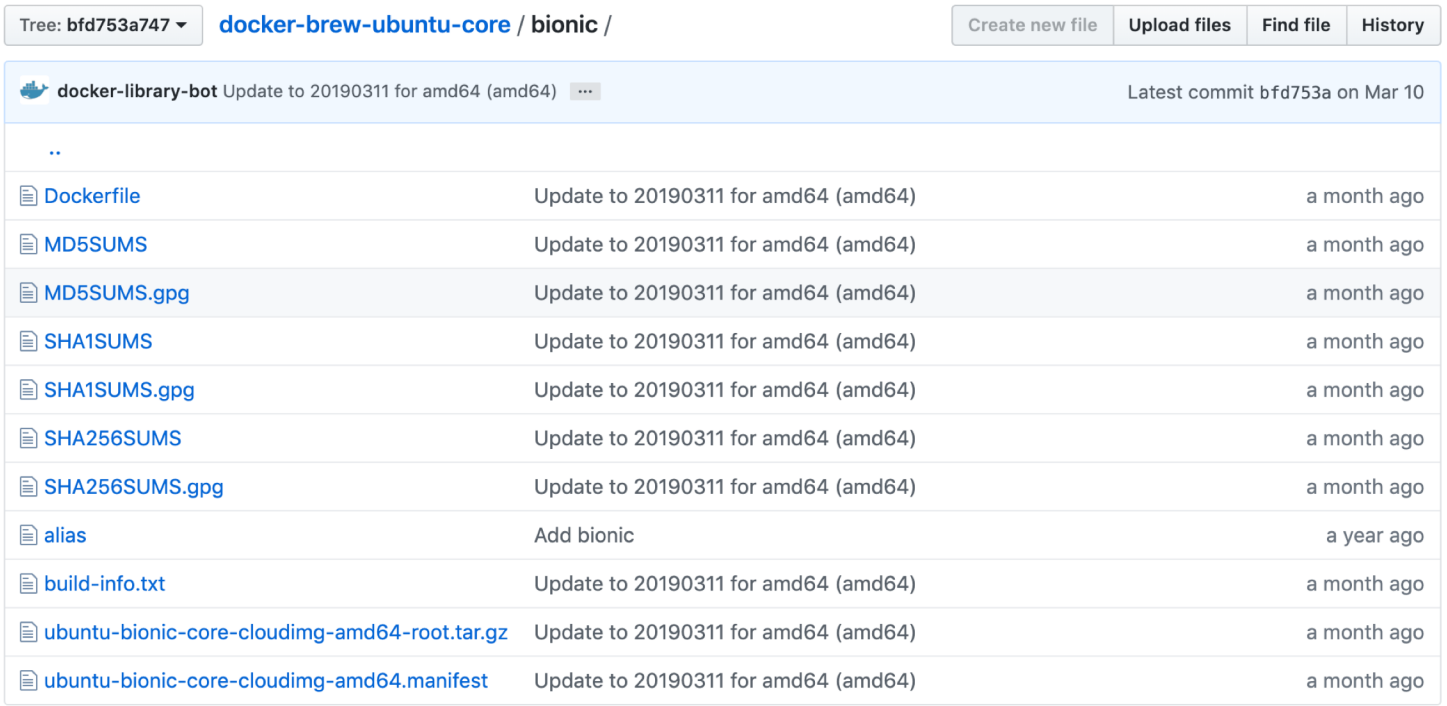
Docker

- Platform
  - dockerd – daemon server
  - Client – instructs server
  - CLI – embeds client



Docker

- Images
  - Executable – includes application binary, libraries etc.



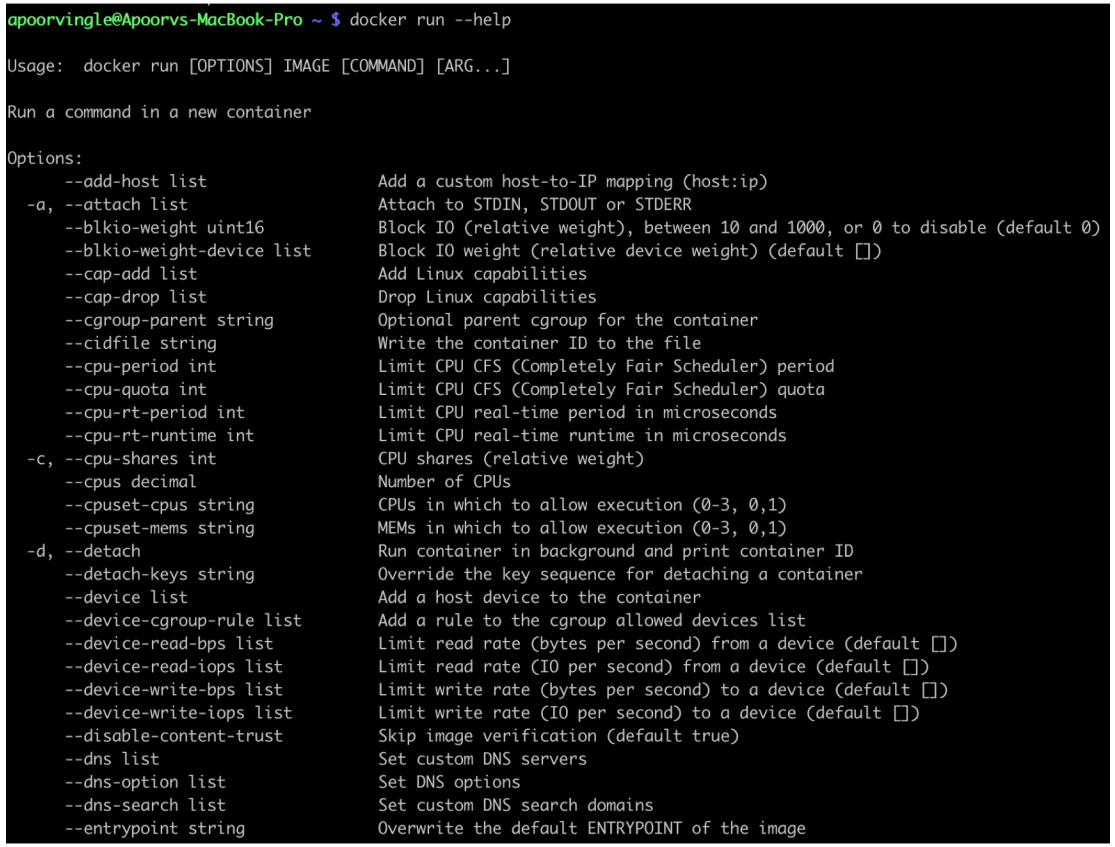
Tree: bfd753a747 ▾ docker-brew-ubuntu-core / bionic / Create new file | Upload files | Find file | History

docker-library-bot Update to 20190311 for amd64 (amd64) ··· Latest commit bfd753a on Mar 10

..		
Dockerfile	Update to 20190311 for amd64 (amd64)	a month ago
MD5SUMS	Update to 20190311 for amd64 (amd64)	a month ago
MD5SUMS.gpg	Update to 20190311 for amd64 (amd64)	a month ago
SHA1SUMS	Update to 20190311 for amd64 (amd64)	a month ago
SHA1SUMS.gpg	Update to 20190311 for amd64 (amd64)	a month ago
SHA256SUMS	Update to 20190311 for amd64 (amd64)	a month ago
SHA256SUMS.gpg	Update to 20190311 for amd64 (amd64)	a month ago
alias	Add bionic	a year ago
build-info.txt	Update to 20190311 for amd64 (amd64)	a month ago
ubuntu-bionic-core-cloudimg-amd64-root.tar.gz	Update to 20190311 for amd64 (amd64)	a month ago
ubuntu-bionic-core-cloudimg-amd64.manifest	Update to 20190311 for amd64 (amd64)	a month ago

Docker

- Containers
  - Runtime instances of images
  - Just a process running on host OS
    - cgroups and namespaces



Docker

- $ docker run -it ubuntu /bin/bash
  - Runs image name ubuntu
  - Start point bash

- $ docker run -it ubuntu -u nobody /bin/bash
  - User is nobody instead of root
  - Checks from passwd file

- Run command pulls image from repository if not locally stored
- Runs the image

Kubernetes

- Orchestration of containers
  - Dynamic load balancer?
  - OSS by Google in 2014
- Think of application rather than machines
- Stores information about which service is located where

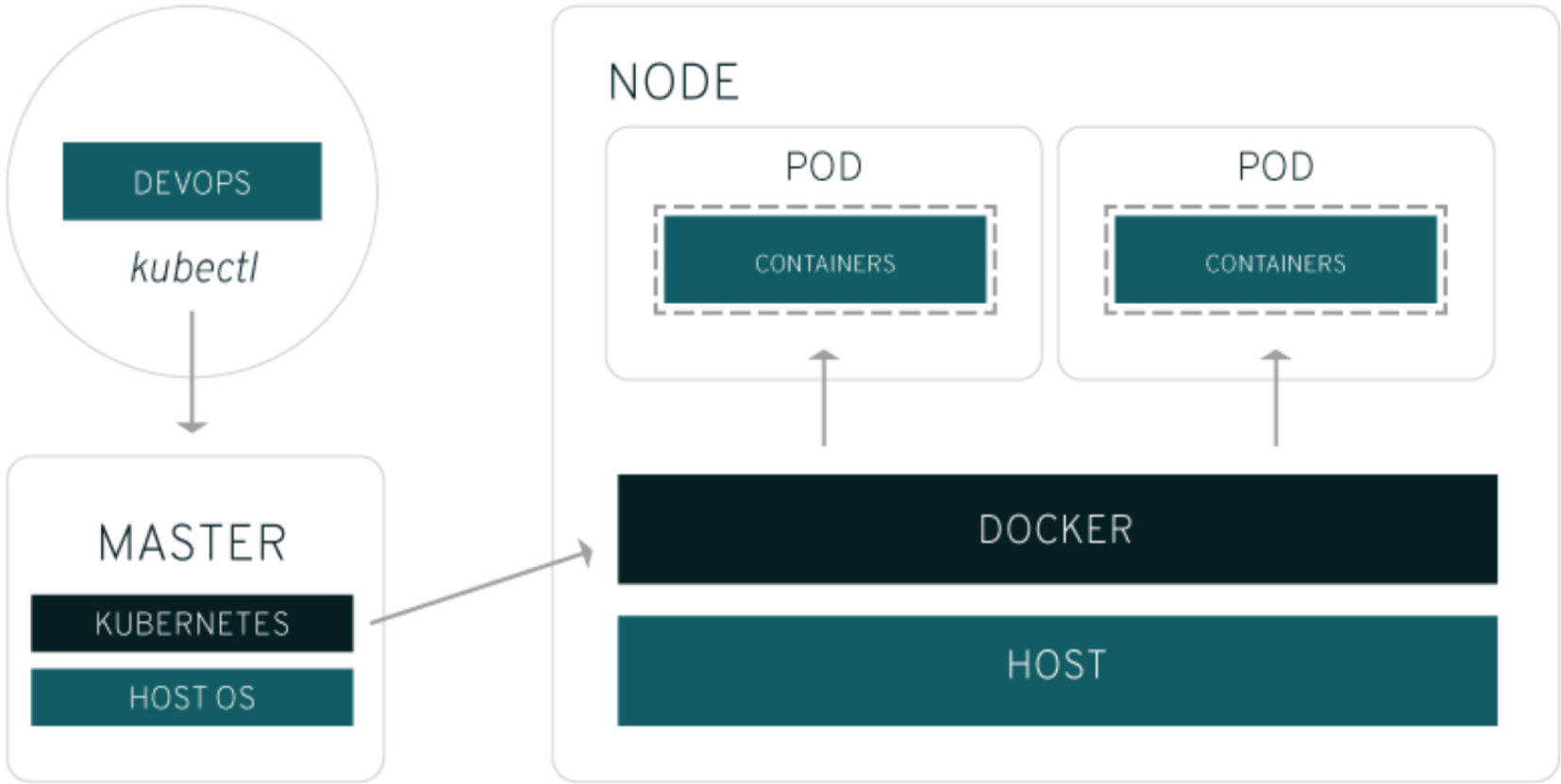
Kubernetes

- Microservice architecture
  - Roughly each service handles a business logic
  - Service may consist of multiple processes on different hosts
- Scaling
  - Add/reduce containers per application
- Healing
  - Restart on failure
- Monitoring at different levels
  - Container, service

Kubernetes

- Glossary
- Master: Main Orchestrator machine
- Node: Worker machines
- Pod: Group of containers on a node. Abstraction over network/fs
- Replication controller: Controls how many identical copies of a pod should be running
- Kubelet: Monitoring. Runs on nodes to ensure the necessary containers are started and running.

Kubernetes


DEVOPS
kubectl
MASTER
KUBERNETES
HOST OS
NODE
POD
CONTAINERS
POD
CONTAINERS
DOCKER
HOST

Summary

- Containers
  - cgroups and namespaces
  - Uses same kernel
- Docker
  - Abstraction over low-level cgroups and ns
- Kubernetes
  - Container orchestrator for infrastructure

Questions?

References

- Anatomy of a Container: Namespaces, cgroups & Some Filesystem Magic, https://www.slideshare.net/jpetazzo/anatomy-of-a-container-namespaces-cgroups-some-filesystem-magic-linuxcon
- Soltesz, Stephen, Herbert Pötzl, Marc E. Fiuczynski, Andy Bavier, and Larry Peterson. 2007. “Container-Based Operating System Virtualization: A Scalable, High-Performance Alternative to Hypervisors.” In *Proceedings of the 2Nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, 275–287. EuroSys ’07. New York, NY, USA: ACM. https://doi.org/10.1145/1272996.1273025.
- Bernstein, D. 2014. “Containers and Cloud: From LXC to Docker to Kubernetes.” *IEEE Cloud Computing* 1 (3): 81–84. https://doi.org/10.1109/MCC.2014.51.
- Burns, Brendan, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. 2016. “Borg, Omega, and Kubernetes.” *Queue* 14 (1): 10:70–10:93. https://doi.org/10.1145/2898442.2898444.
- “Everything You Need to Know about Linux Containers, Part I: Linux Control Groups and Process Isolation | Linux Journal.” n.d. Accessed April 16, 2019. https://www.linuxjournal.com/content/everything-you-need-know-about-linux-containers-part-i-linux-control-groups-and-process.
- “Everything You Need to Know about Linux Containers, Part II: Working with Linux Containers (LXC) | Linux Journal.” n.d. Accessed April 16, 2019. https://www.linuxjournal.com/content/everything-you-need-know-about-linux-containers-part-ii-working-linux-containers-lxc.
- “Everything You Need to Know about Containers, Part III: Orchestration with Kubernetes | Linux Journal.” n.d. Accessed April 16, 2019. https://www.linuxjournal.com/content/everything-you-need-know-about-containers-part-iii-orchestration-kubernetes.