TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

Containerization

Wes J. Lloyd
Institute of Technology
University of Washington - Tacoma

Credit: some content based on Salman A. Baset, IBM: WOC 2018 @ IC2E – Container Security

OBJECTIVES

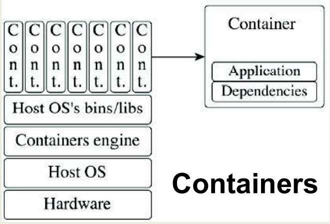
- Term project questions
- AWS Educate
- Tutorial #2
- Midterm Wednesday 5/9
- Containerization
- Tutorial #3 – Containers, cgroups, isolation

FEEDBACK

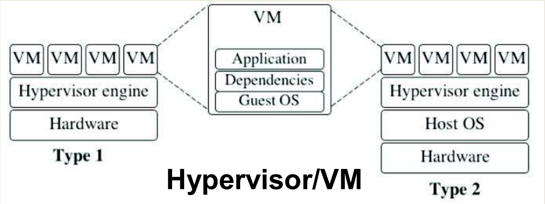
- ...

MOTIVATION FOR CONTAINERIZATION

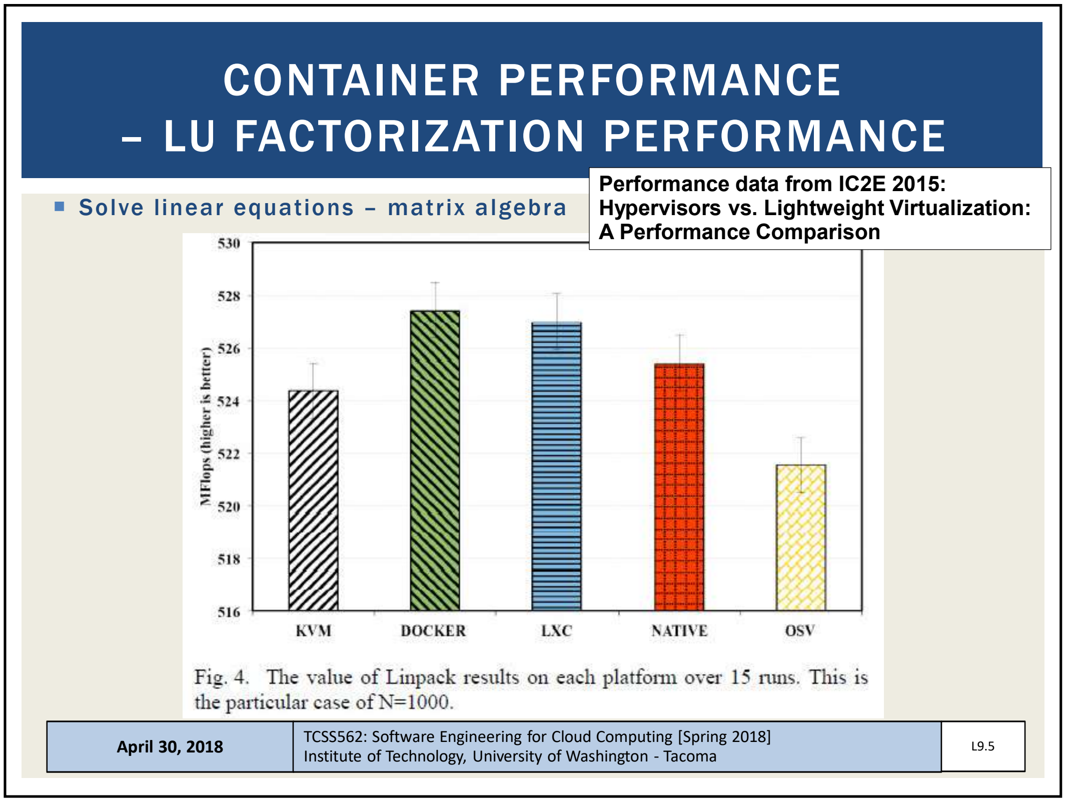
- Containers provide “light-weight” alternative to full OS virtualization provided by a hypervisor
- Containers do not provide a full “machine”
- Instead use operating system constructs to provide “sand boxes” for execution
  - Linux cgroups, namespaces, etc.
- Containers can run on bare metal, or atop of VMs



Containers



Hypervisor/VM



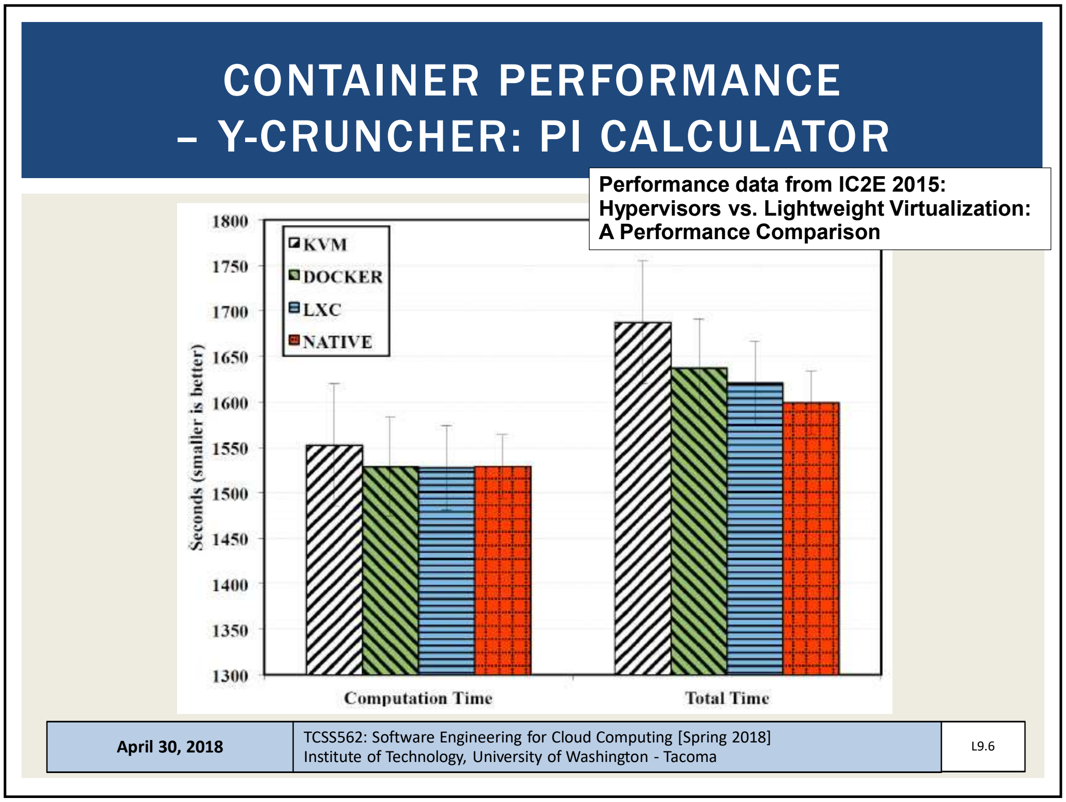
CONTAINER PERFORMANCE – LU FACTORIZATION PERFORMANCE

- Solve linear equations – matrix algebra

Performance data from IC2E 2015: Hypervisors vs. Lightweight Virtualization: A Performance Comparison

Fig. 4. The value of Linpack results on each platform over 15 runs. This is the particular case of N=1000.

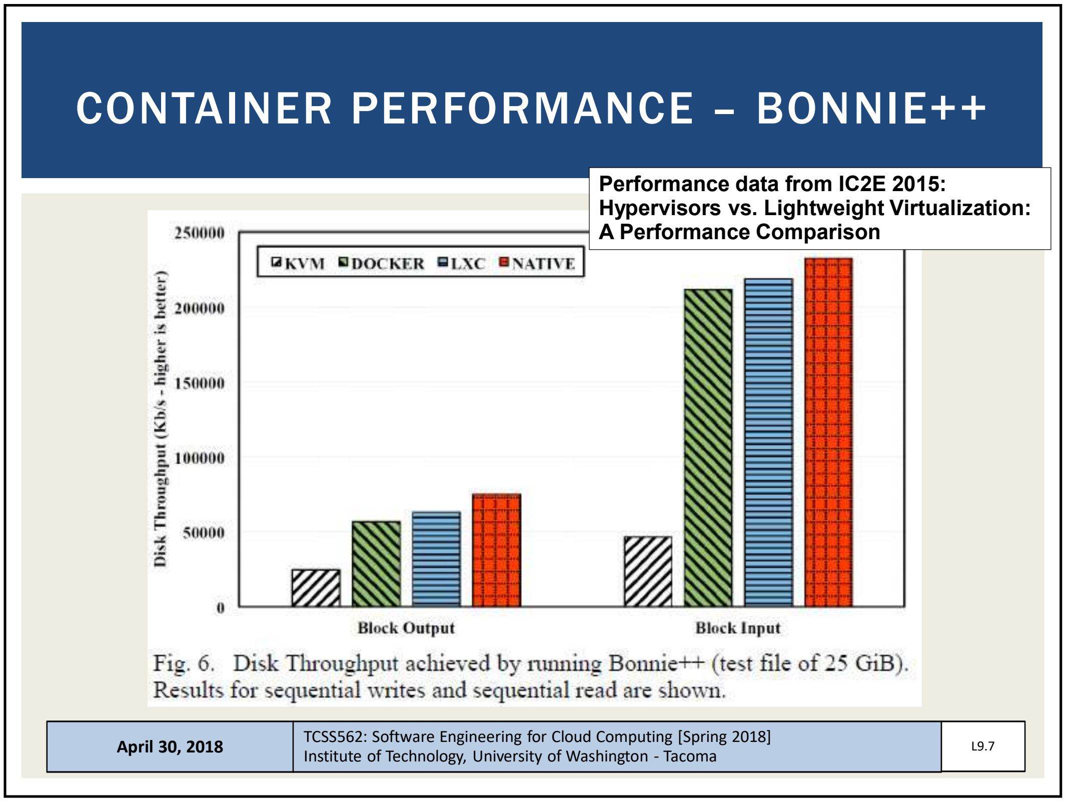
April 30, 2018 | TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma | L9.5



CONTAINER PERFORMANCE – Y-CRUNCHER: PI CALCULATOR

Performance data from IC2E 2015: Hypervisors vs. Lightweight Virtualization: A Performance Comparison

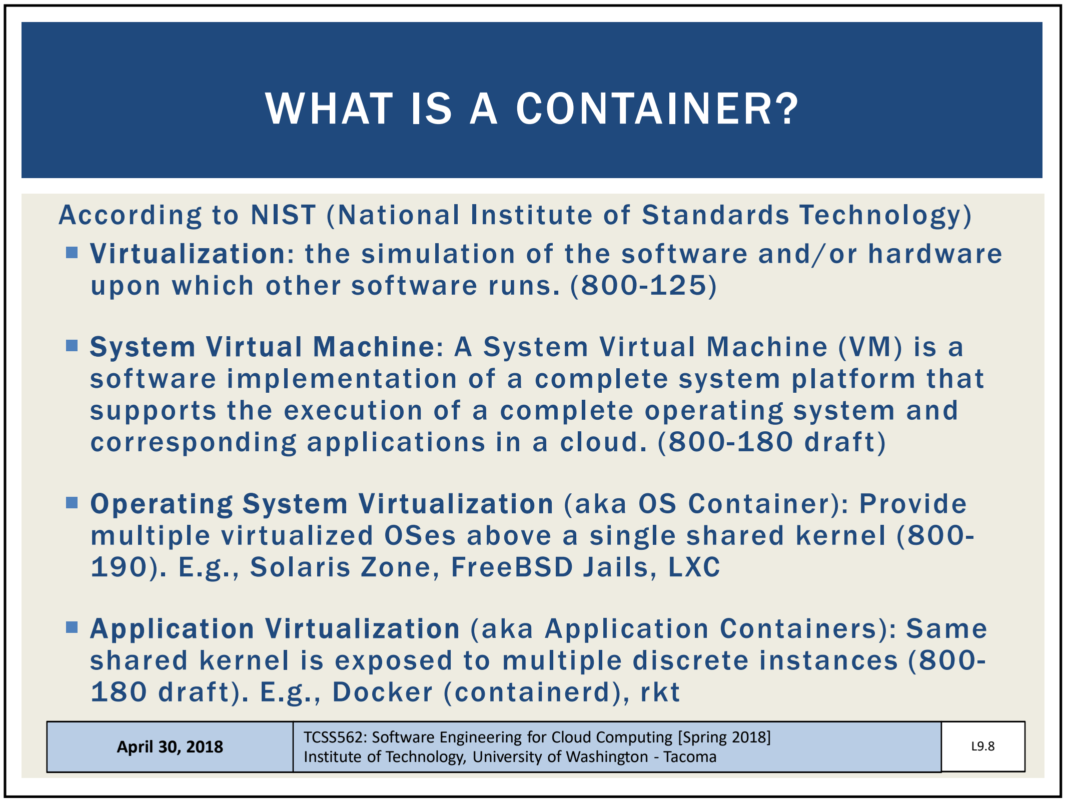
April 30, 2018 | TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma | L9.6



CONTAINER PERFORMANCE – BONNIE++

Performance data from IC2E 2015: Hypervisors vs. Lightweight Virtualization: A Performance Comparison

Fig. 6. Disk Throughput achieved by running Bonnie++ (test file of 25 GiB). Results for sequential writes and sequential read are shown.



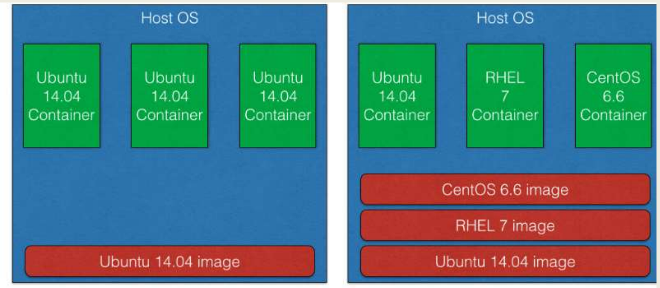
WHAT IS A CONTAINER?

According to NIST (National Institute of Standards Technology)

- **Virtualization**: the simulation of the software and/or hardware upon which other software runs. (800-125)
- **System Virtual Machine**: A System Virtual Machine (VM) is a software implementation of a complete system platform that supports the execution of a complete operating system and corresponding applications in a cloud. (800-180 draft)
- **Operating System Virtualization** (aka OS Container): Provide multiple virtualized OSes above a single shared kernel (800-190). E.g., Solaris Zone, FreeBSD Jails, LXC
- **Application Virtualization** (aka Application Containers): Same shared kernel is exposed to multiple discrete instances (800-180 draft). E.g., Docker (containerd), rkt

OPERATING SYSTEM CONTAINERS

- Virtual environments: share the host kernel
- Provide user space isolation
- Replacement for VMs: run multiple processes, services
- Mix different Linux distros on same host
- Examples: LXC, OpenVZ, Linux Vserver, BSD Jails, Solaris zones

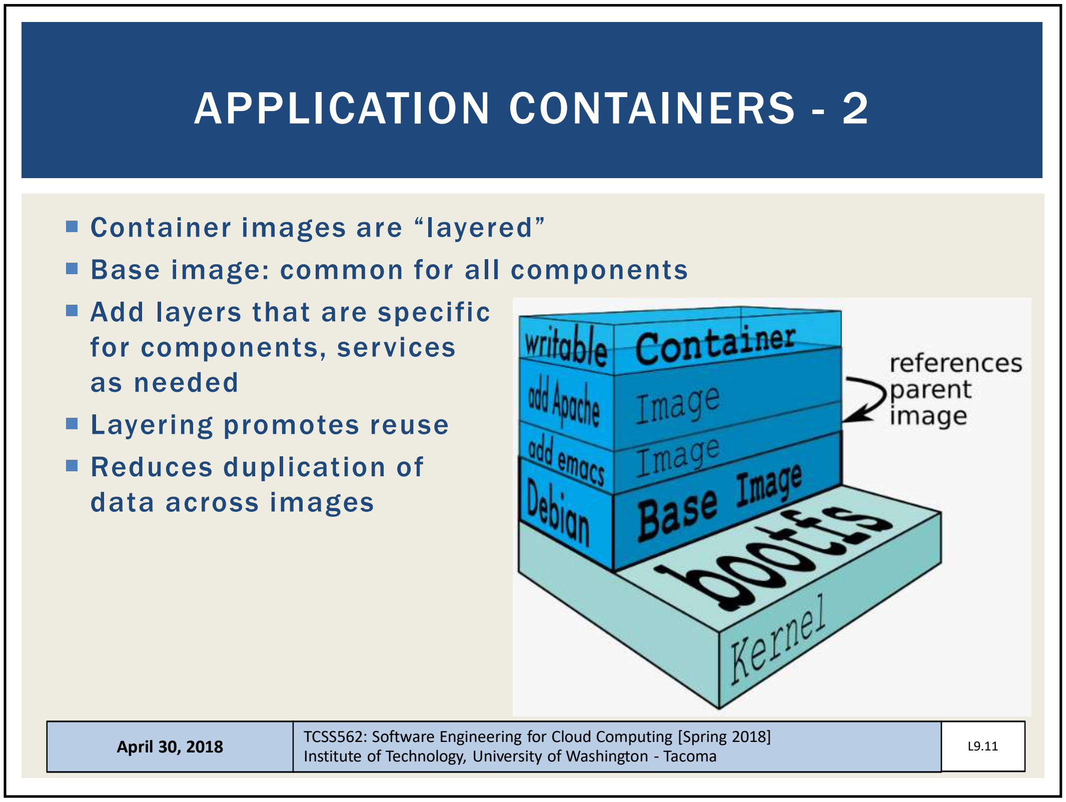


Identical OS containers   Different flavoured OS containers

- Credit: https://blog.risingstack.com/operating-system-containers-vs-application-containers/

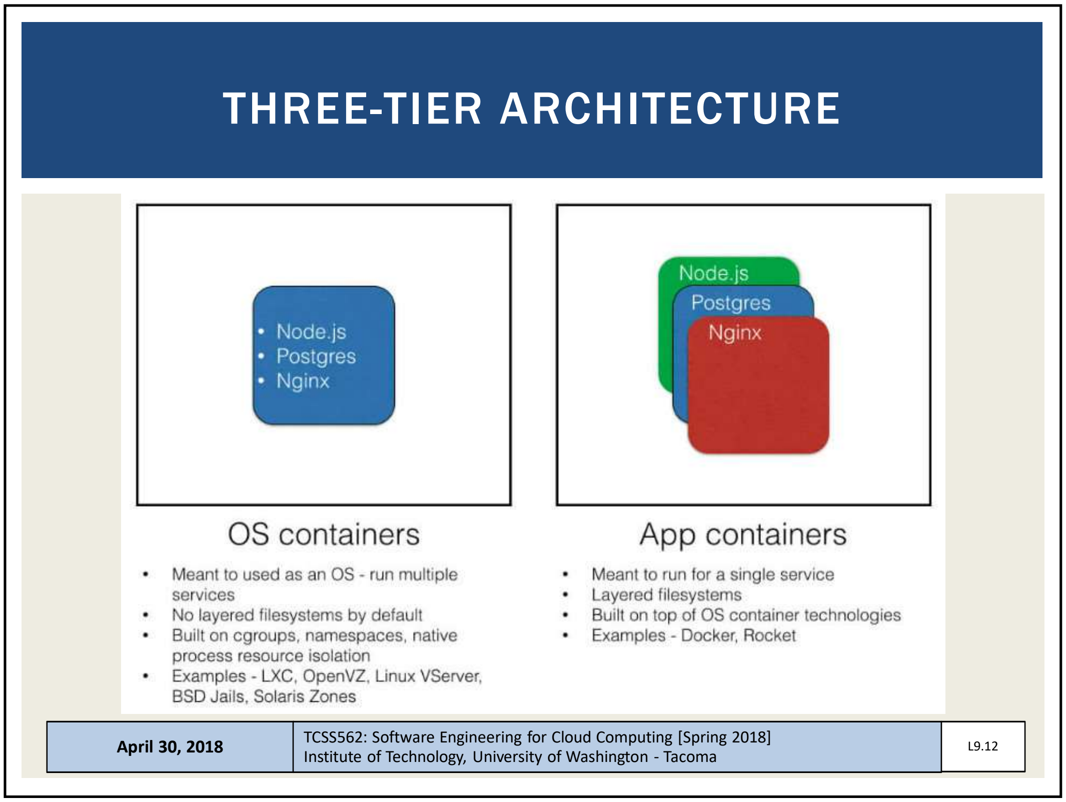
APPLICATION CONTAINERS

- Designed to package and run a single service
- All containers share host kernel
- Subtle differences from operating system containers
- Examples: Docker, Rocket
- Docker: runs a single process on creation
- OS containers: run many OS services, for an entire OS
- Create application containers for each component of an app
- Supports a micro-services architecture
- DevOPS: developers can package their own components in application containers
- Supports horizontal and vertical scaling



APPLICATION CONTAINERS - 2

- Container images are "layered"
- Base image: common for all components
- Add layers that are specific for components, services as needed
- Layering promotes reuse
- Reduces duplication of data across images



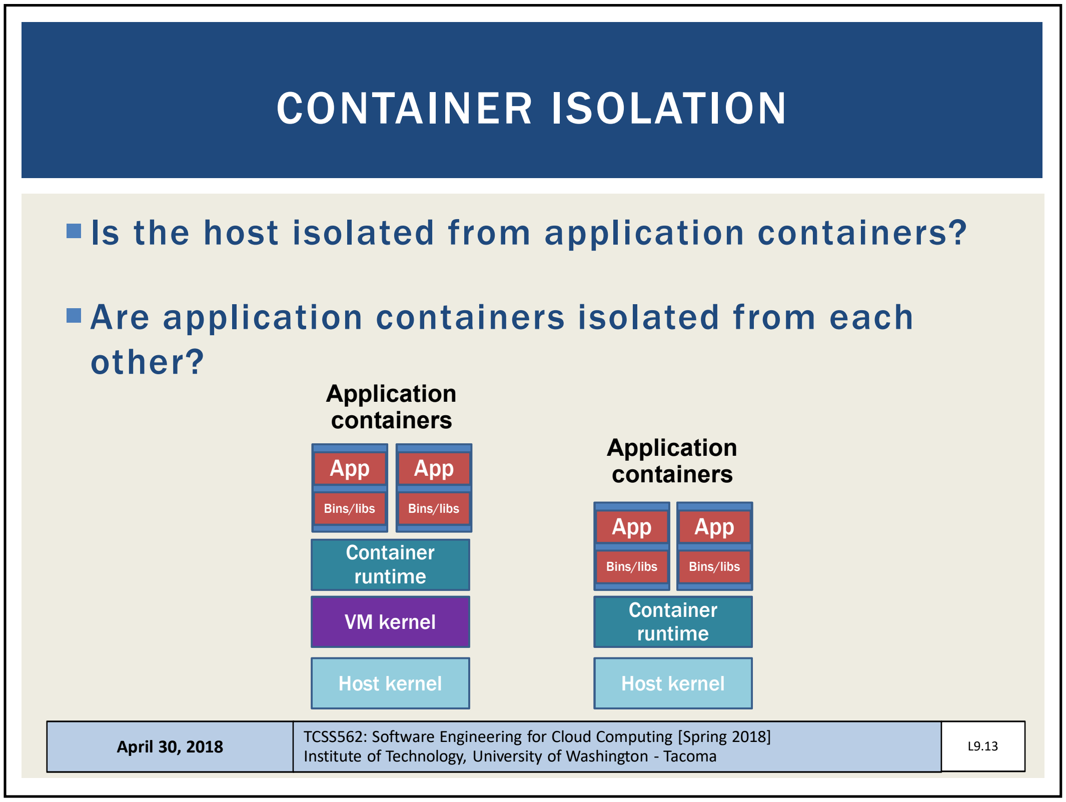
THREE-TIER ARCHITECTURE

OS containers

- Meant to used as an OS - run multiple services
- No layered filesystems by default
- Built on cgroups, namespaces, native process resource isolation
- Examples - LXC, OpenVZ, Linux VServer, BSD Jails, Solaris Zones

App containers

- Meant to run for a single service
- Layered filesystems
- Built on top of OS container technologies
- Examples - Docker, Rocket



CONTAINER ISOLATION

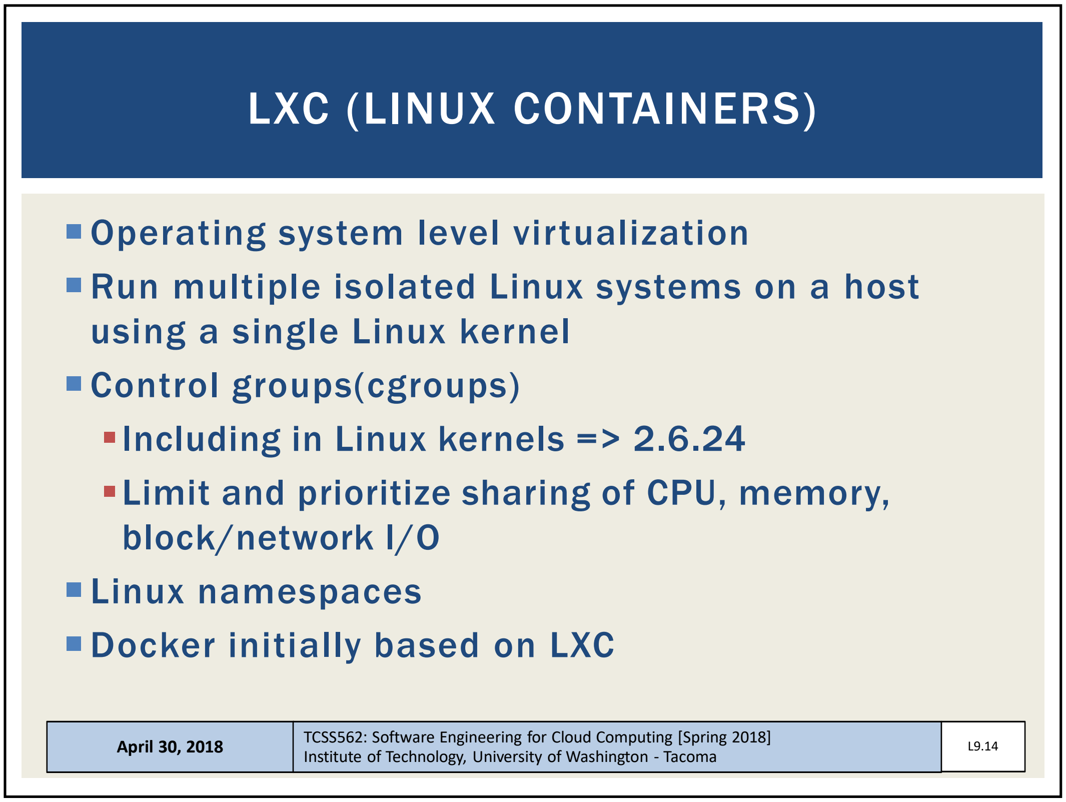
- Is the host isolated from application containers?
- Are application containers isolated from each other?

Application containers

App | App
Bins/libs | Bins/libs
Container runtime
VM kernel
Host kernel

Application containers

App | App
Bins/libs | Bins/libs
Container runtime
Host kernel

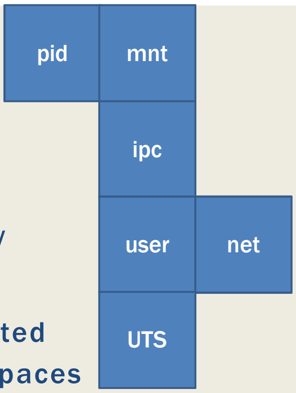


LXC (LINUX CONTAINERS)

- Operating system level virtualization
- Run multiple isolated Linux systems on a host using a single Linux kernel
- Control groups(cgroups)
  - Including in Linux kernels => 2.6.24
  - Limit and prioritize sharing of CPU, memory, block/network I/O
- Linux namespaces
- Docker initially based on LXC

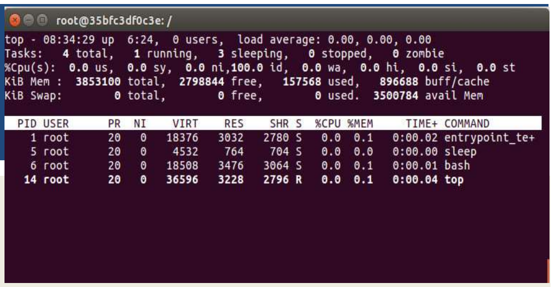
LINUX KERNEL NAMESPACES

- Partitions kernel resources
- Processes see only their set of resources
- Provides isolation
- Namespaces are hierarchical
- Parent processes can see down the hierarchy
- 7 namespaces in Linux (cgroups not shown)
- Each process can only see resources associated with the namespace, and descendent namespaces



pid
mnt
ipc
user
net
UTS

NAMESPACES - 2



```
top - 08:34:29 up  6:24,  0 users,  load average: 0.00, 0.00, 0.00
Tasks:   4 total,   1 running,   3 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  3853100 total,  2798844 free,   157568 used,   896688 buff/cache
KiB Swap:        0 total,        0 free,        0 used.  3500784 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
    1 root      20   0   18376   3032   2780 S   0.0  0.1   0:00.02 entrypoint_te+
    5 root      20   0    4532    764    704 S   0.0  0.0   0:00.00 sleep
    6 root      20   0   18508   3476   3064 S   0.0  0.1   0:00.01 bash
   14 root      20   0   36596   3228   2796 R   0.0  0.1   0:00.04 top
```

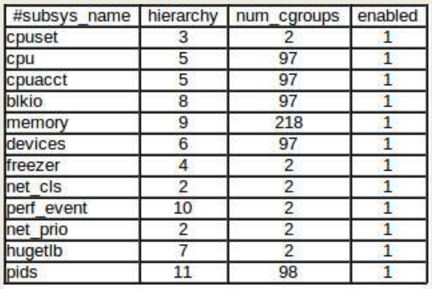
- Provides isolation of OS entities for containers
- mnt: separate filesystems
- pid: independent PIDs; first process in container is PID 1
- ipc: prevents processes in different IPC namespaces from being able to establish shared memory. Enables processes in different containers to reuse the same identifiers without conflict.
  … provides expected *VM like isolation…*
- user: user identification and privilege isolation among separate containers
- net: network stack virtualization. Multiple loopbacks (lo)
- UTS (UNIX time sharing): provides separate host and domain

CONTROL GROUPS (CGROUPS)

- Collection of Linux processes
- Group-level resource allocation: *CPU, memory, disk I/O, network I/O*
- Resource limiting
  - Memory, disk cache
- Prioritization
  - CPU share
  - Disk I/O throughput
- Accounting
  - Track resource utilization
  - For resource management and/or billing purposes
- Control
  - Pause/resume processes
  - Checkpointing → Checkpoint/Restore in Userspace (CRIU)
  - https://criu.org

CGROUPS - 2

- Control groups are hierarchical
- Groups inherent limits from parent groups
- Linux has multiple cgroup controllers (subsystems)
- ls /proc/cgroups
- “memory” controller limits memory use
- “cpuacct” controller accounts for CPU usage

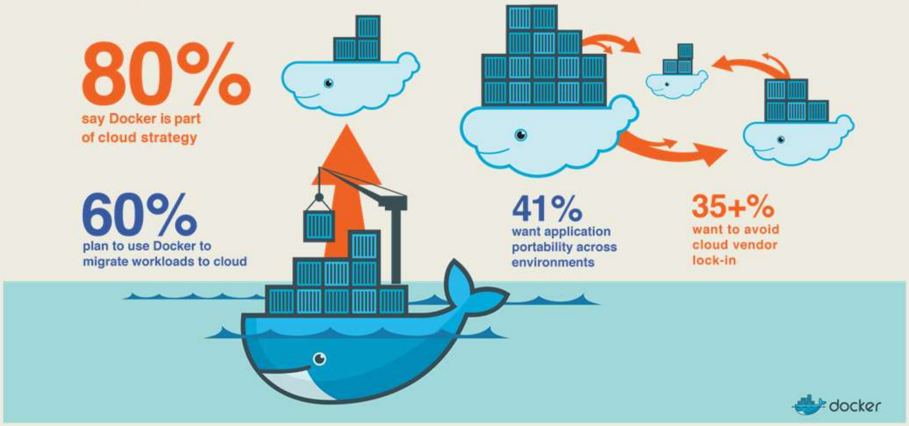


#subsys_name	hierarchy	num_cgroups	enabled
cpuset	3	2	1
cpu	5	97	1
cpuacct	5	97	1
blkio	8	97	1
memory	9	218	1
devices	6	97	1
freezer	4	2	1
net_cls	2	2	1
perf_event	10	2	1
net_prio	2	2	1
hugetlb	7	2	1
pids	11	98	1

- cgroup filesystem:
- /sys/fs/cgroup
- Can browse resource utilization of containers…

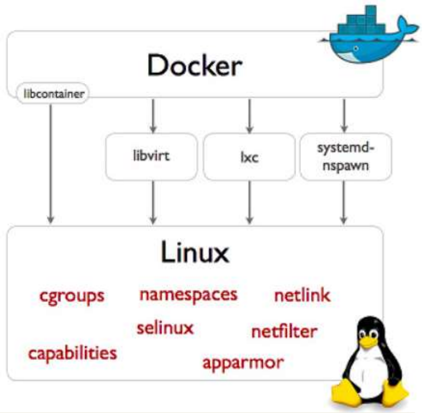
2016 DOCKER SURVEY

- Docker application containers
  - Leading containerization vehicle



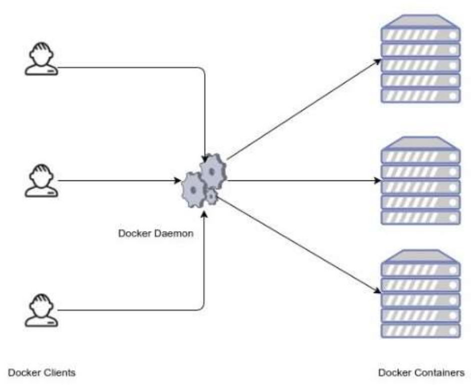
DOCKER EXECUTION ENVIRONMENTS

- (1) Original default Docker execution enviornment: LXC
- (2) Docker v0.9: libcontainer introduced (~2014)
- (3) Now runc (2015)
- Provides Docker access to Linux container APIs
- Execution drivers concept:
- Enable docker to leverage many OS containers as the exec environment
- OpenVZ, system-nspawn, libvirt-lxc, libvirt-sandbox, qemu/kvm, BSD Jails, Solaris Zones, and chroot



DOCKER

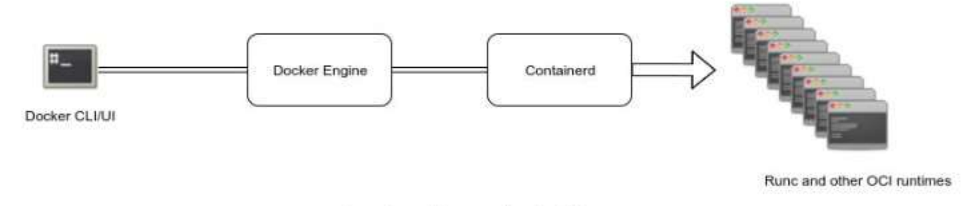
- Docker daemon "dockerd"
  - Provides docker services to Linux
- Docker 1.11+
- Open Container Initiative
- June 2015: Industry standard for container runtimes and formats
- Ensure containers are portable among different execution environments (engines)



Docker Client-Server Architecture

- Credit: https://hackernoon.com/docker-containerd-standalone-runtimes-heres-what-you-should-know-b834ef155426

DOCKER - 2

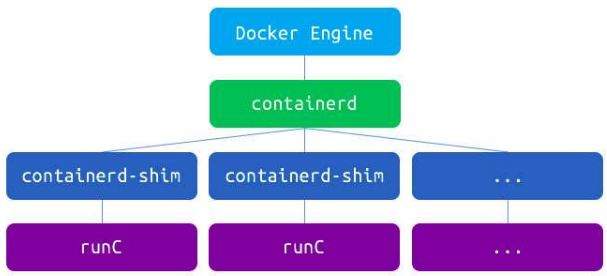


Containerd Integration Architecture

- Docker CLI: interfaces with dockerd daemon
- Docker engine: dockerd daemon, interfaces with Containerd
- Containerd: simple daemon, interfaces with runc to manage containers; CRUD interface for containers, images, volumes, networks, builds; HTTP API → Google RPC (gRPC) interface;
- runc: lightweight command-line tool for running containers; Interfaces with Linux cgroups, namespaces; Runs an OCI container

DOCKER - 3

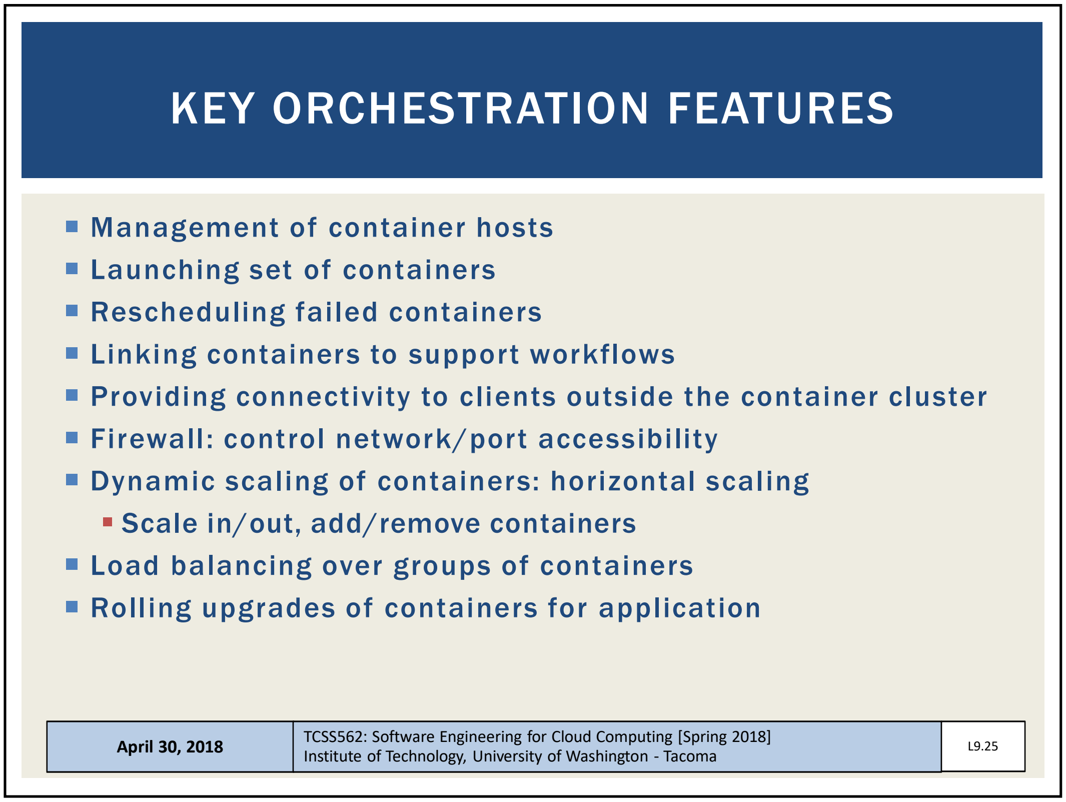
- Docker architecture:



- Other Docker tools:
- **Docker Machine:** automatically provision and manage sets of docker hosts to form a cluster
- **Docker Swarm**: Clusters multiple docker hosts together to manage as a cluster.
- **Docker Compose**: Config file (YAML) for multi-container application; Describes how to deploy and configure multiple containers

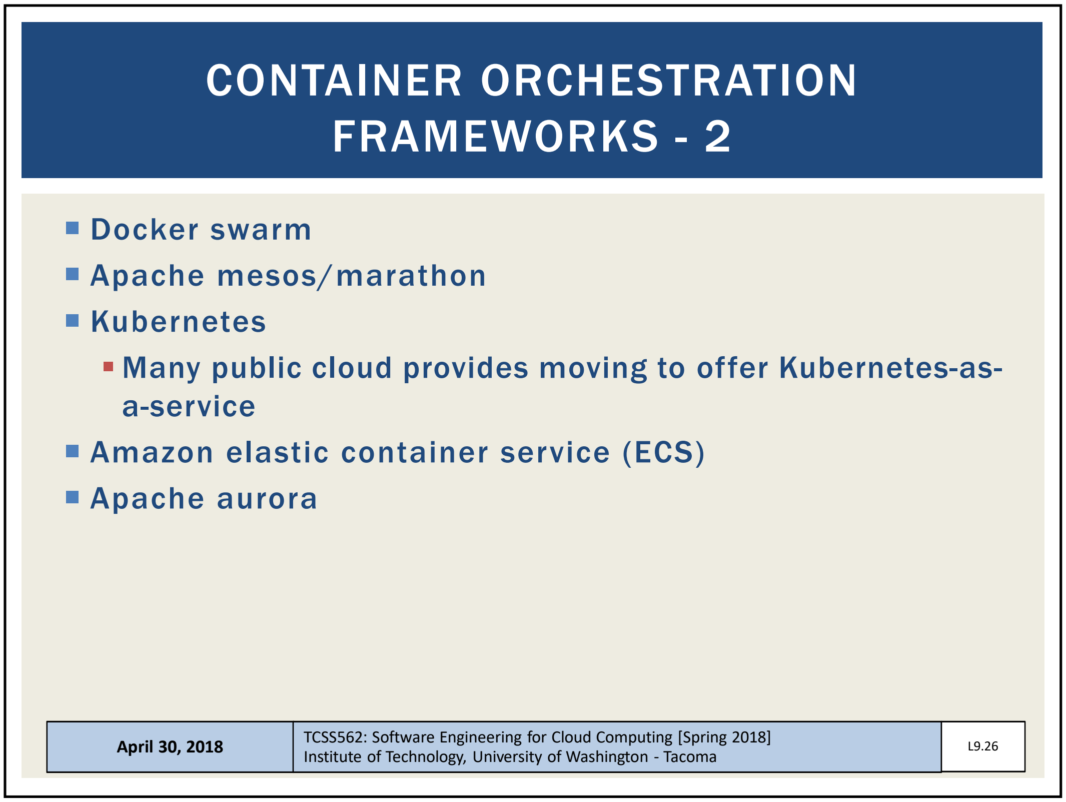
CONTAINER ORCHESTRATION FRAMEWORKS

- Framework(s) to deploy multiple containers
- Provide container clusters using cloud VMs
- Similar to “private clusters”
- Reduce VM idle CPU time in public clouds
- Better leverage “sunk cost” resources
- Compact multiple apps onto shared public cloud infrastructure
- Generate to cost savings
- Reduce vendor lock-in



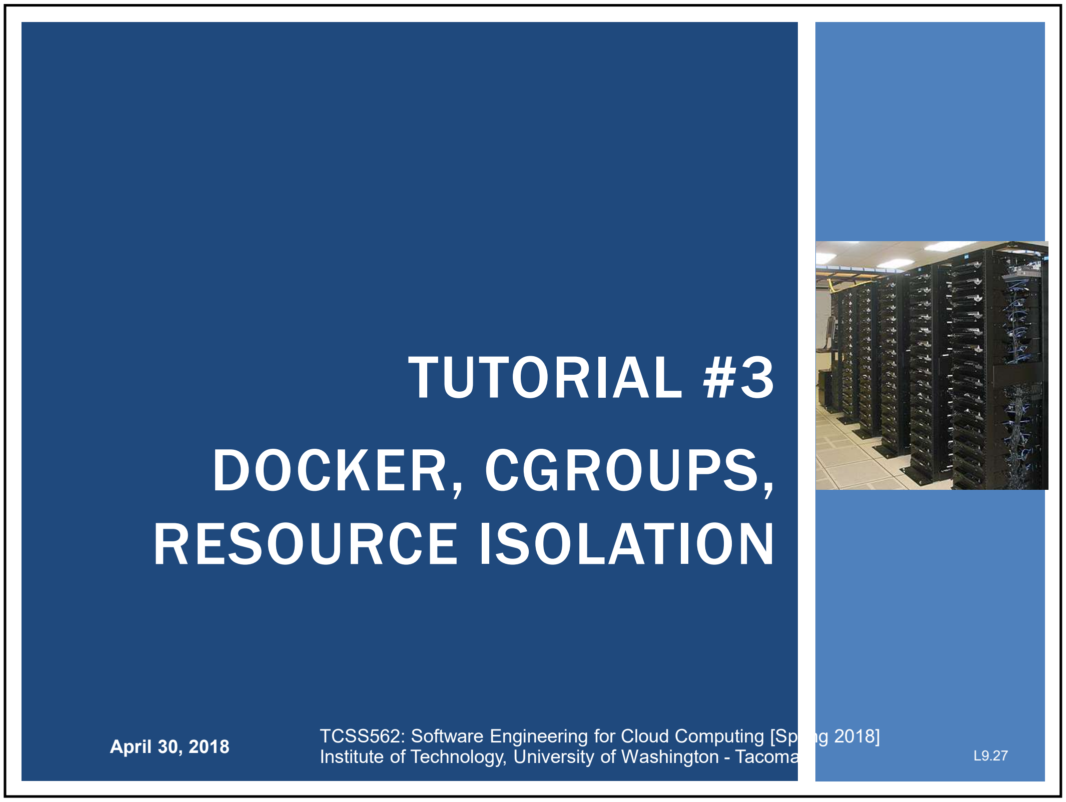
KEY ORCHESTRATION FEATURES

- Management of container hosts
- Launching set of containers
- Rescheduling failed containers
- Linking containers to support workflows
- Providing connectivity to clients outside the container cluster
- Firewall: control network/port accessibility
- Dynamic scaling of containers: horizontal scaling
  - Scale in/out, add/remove containers
- Load balancing over groups of containers
- Rolling upgrades of containers for application



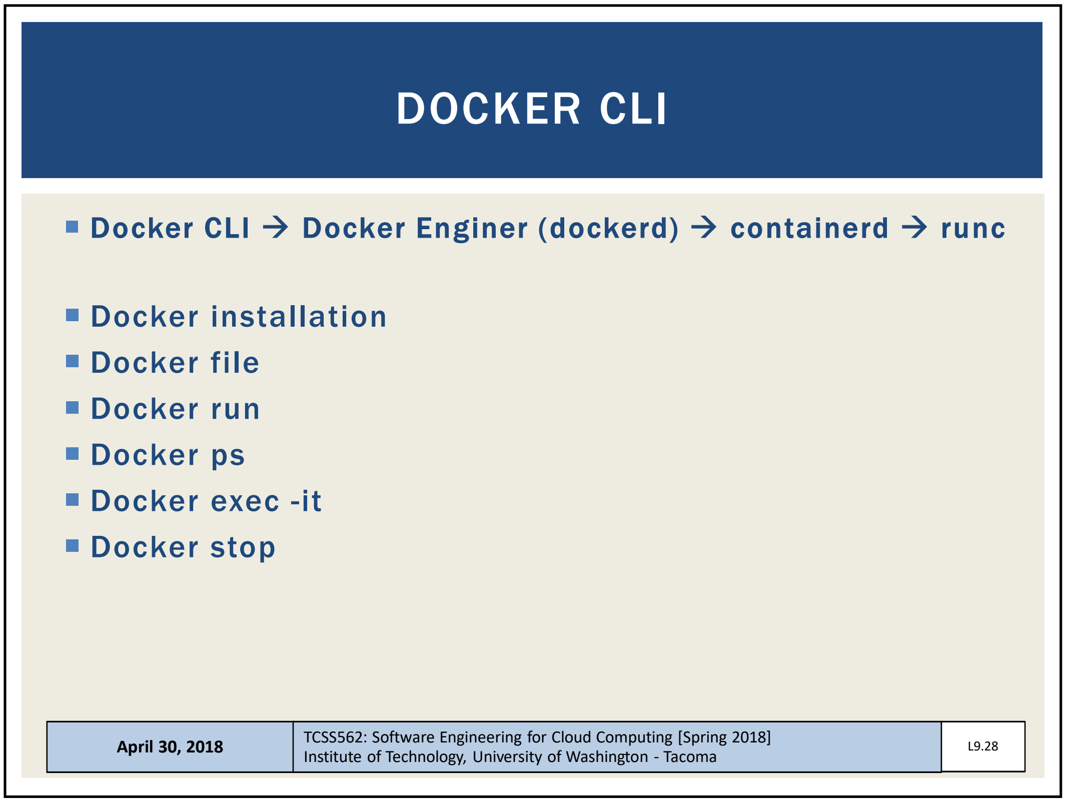
CONTAINER ORCHESTRATION FRAMEWORKS - 2

- Docker swarm
- Apache mesos/marathon
- Kubernetes
  - Many public cloud provides moving to offer Kubernetes-as-a-service
- Amazon elastic container service (ECS)
- Apache aurora



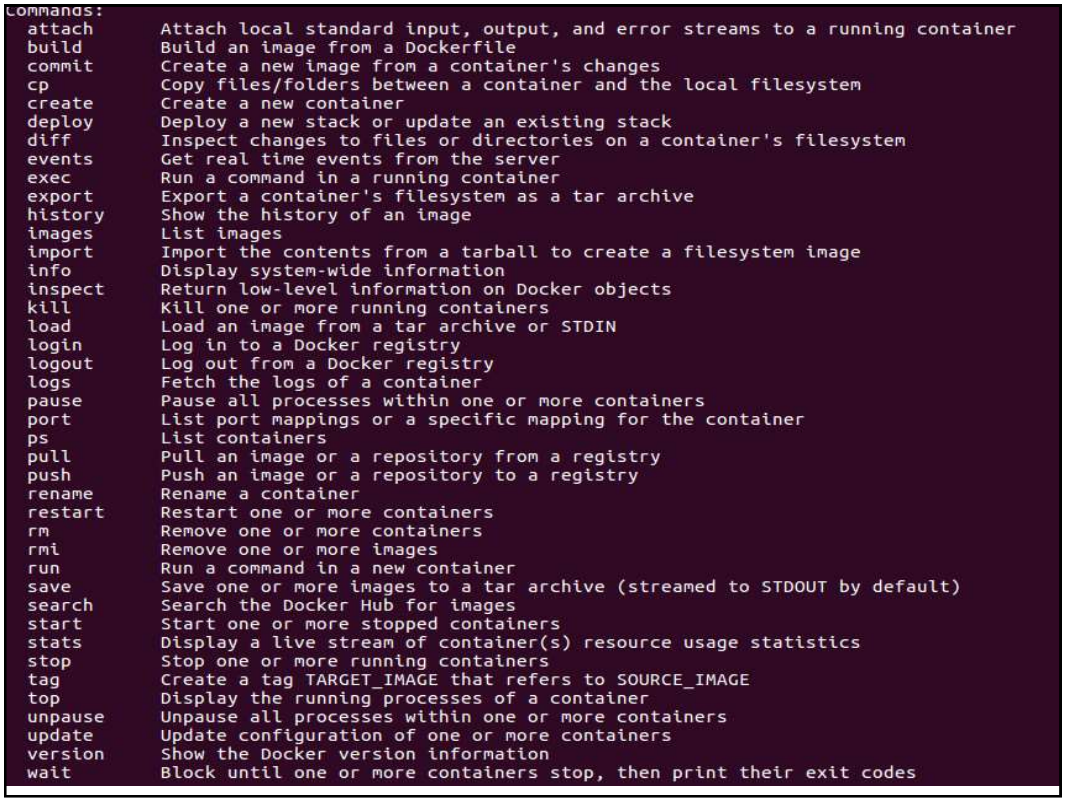
TUTORIAL #3

DOCKER, CGROUPS, RESOURCE ISOLATION



DOCKER CLI

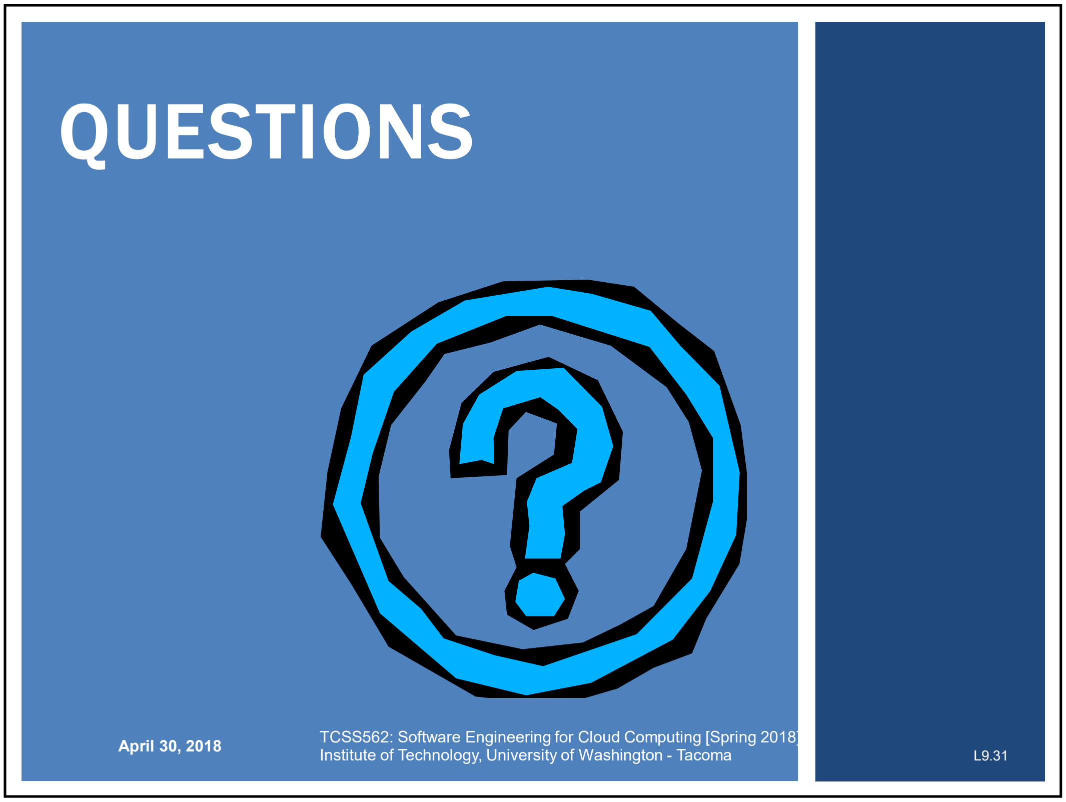
- Docker CLI → Docker Enginer (dockerd) → containerd → runc
- Docker installation
- Docker file
- Docker run
- Docker ps
- Docker exec -it
- Docker stop



```
Commands:
  attach      Attach local standard input, output, and error streams to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  deploy      Deploy a new stack or update an existing stack
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  run         Run a command in a new container
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  search      Search the Docker Hub for images
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  version     Show the Docker version information
  wait        Block until one or more containers stop, then print their exit codes
```

TUTORIAL 3

- Linux performance benchmarks
- stress-ng
- 100s of CPU, memory, disk, network stress tests
- Sysbench
- Used in tutorial for memory stress test



QUESTIONS

April 30, 2018

TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L9.31