Virtualization and Containers

CIS 2235 Adv Linux System Administration

Agenda

- Virtualization defined
  - Types of virtualizers
- Containers
- Docker
- Container Management

Virtualization

Makes it possible to run multiple OS instances on the same physical hardware at the same time

- Each instance looks like a whole, physical server to the user

Started by IBM in the ‘60s on mainframes

Why? Virtualized servers

- Use hardware more efficiently, are portable, and can be managed programmatically.

Heavily used now for

- testing
- server farms

Terminology

Hypervisor (virtual machine monitor)

- A software layer that mediates between the VM and the underlying hardware
- Share system resources among guest OSes

Guest OS

- OS running inside a VM
- independent of each other — can be different OSes

Types of virtualization

- Full virtualization: emulate all the underlying hardware
  - Guests run without modification
  - Performance penalty for translation
  - Very complex
- Hardware-assisted virtualization
  - Intel and AMD have added features to help
  - Reduces performance penalty

Types of hypervisors

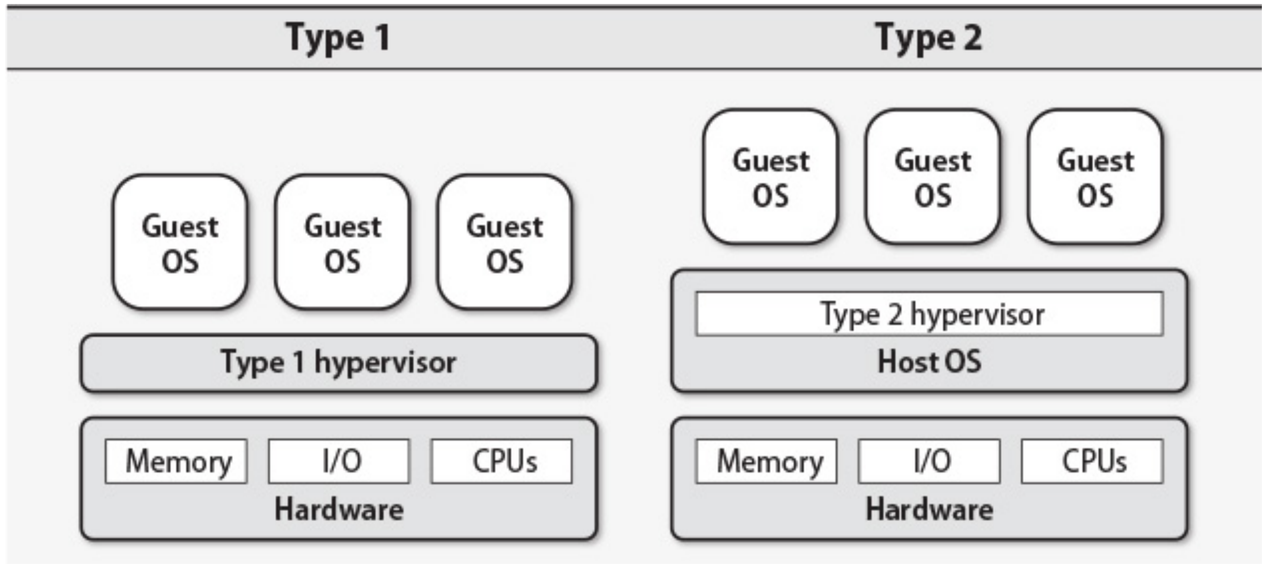
Type 1

directly on hardware - no OS between hypervisor and hardware

Examples: XenServer VMware ESXi

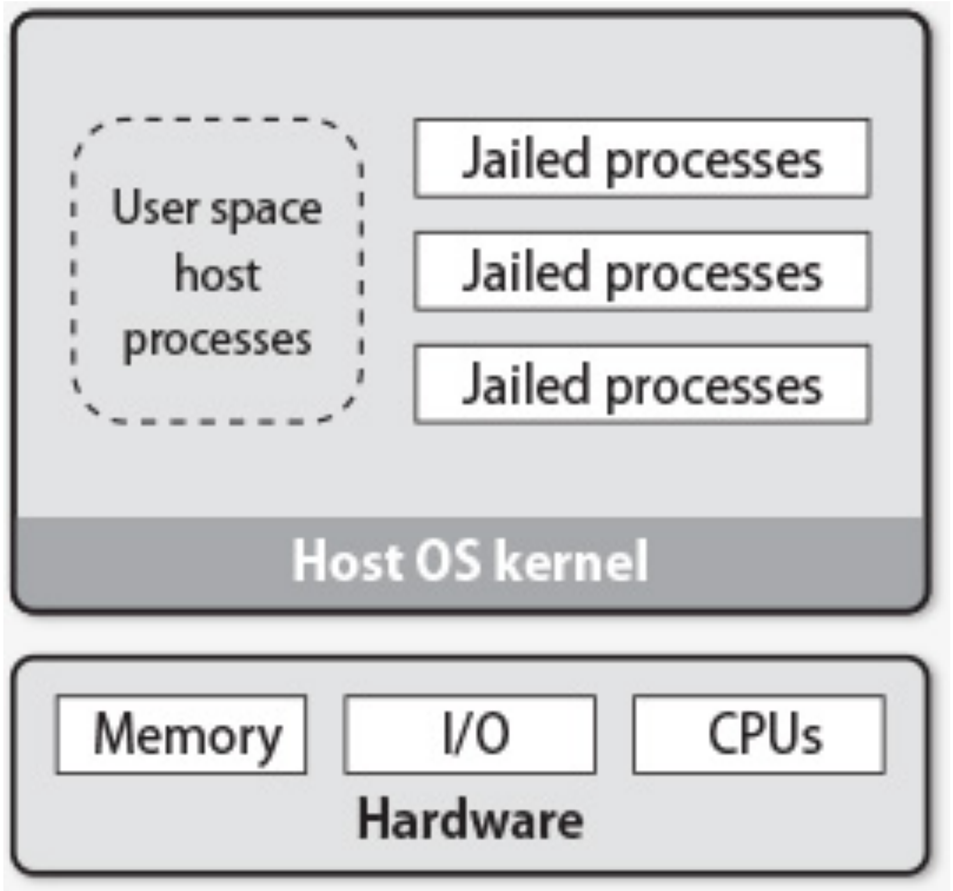
Type 2

runs on top of/as part of Host OS — ex: Virtualbox



Containerization

- OS level virtualization
  - Different approach to isolation.
  - The kernel is shared
  - Examples:
    - Docker
    - Linux XLC
    - FreeBSD jails



Virtualization versus Containerization

Virtual machine	Container
A full-fledged OS that shares underlying hardware through a hypervisor	An isolated group of processes managed by a shared kernel
Requires a complete boot procedure to initialize; starts in 1-2 minutes	Processes run directly by the kernel; no boot required; starts in < 1 second
Long-lived	Frequently replaced
Has one or more dedicated virtual disks attached through the hypervisor	Filesystem view is a layered construct defined by the container engine
Images measured in gigabytes	Images measured in megabytes
A few dozen or fewer per physical host	Many per virtual or physical host
Complete isolation among guests	OS kernel and services shared with host
Multiple independent operating systems running side by side	Must run the same kernel as the host (OS distribution may differ)

Core concepts

An isolated group of processes that are restricted to a private root filesystem and process namespace

- Containers share the same OS, but are separate from each other
- Applications are not aware they are containerized — looks like they have a complete machine
- Each container can be limited to the resources it can use

Kernel support

Built on top of kernel features of the Linux Container project (LXC — Google, 2006)

- Namespaces
  - Isolate process from process mgmt, networking, filesystem mounts
- Control groups (cgroups)
  - Limit use of system resources, prioritize processes
- Capabilities
  - Allow access to sensitive kernel operations
- Secure computing mode (seccomp)
  - Restrict access to system calls. More fine-grained than capabilities

Docker

The current most common container. Its pieces:

docker

A command-line utility for managing the Docker system

dockerd

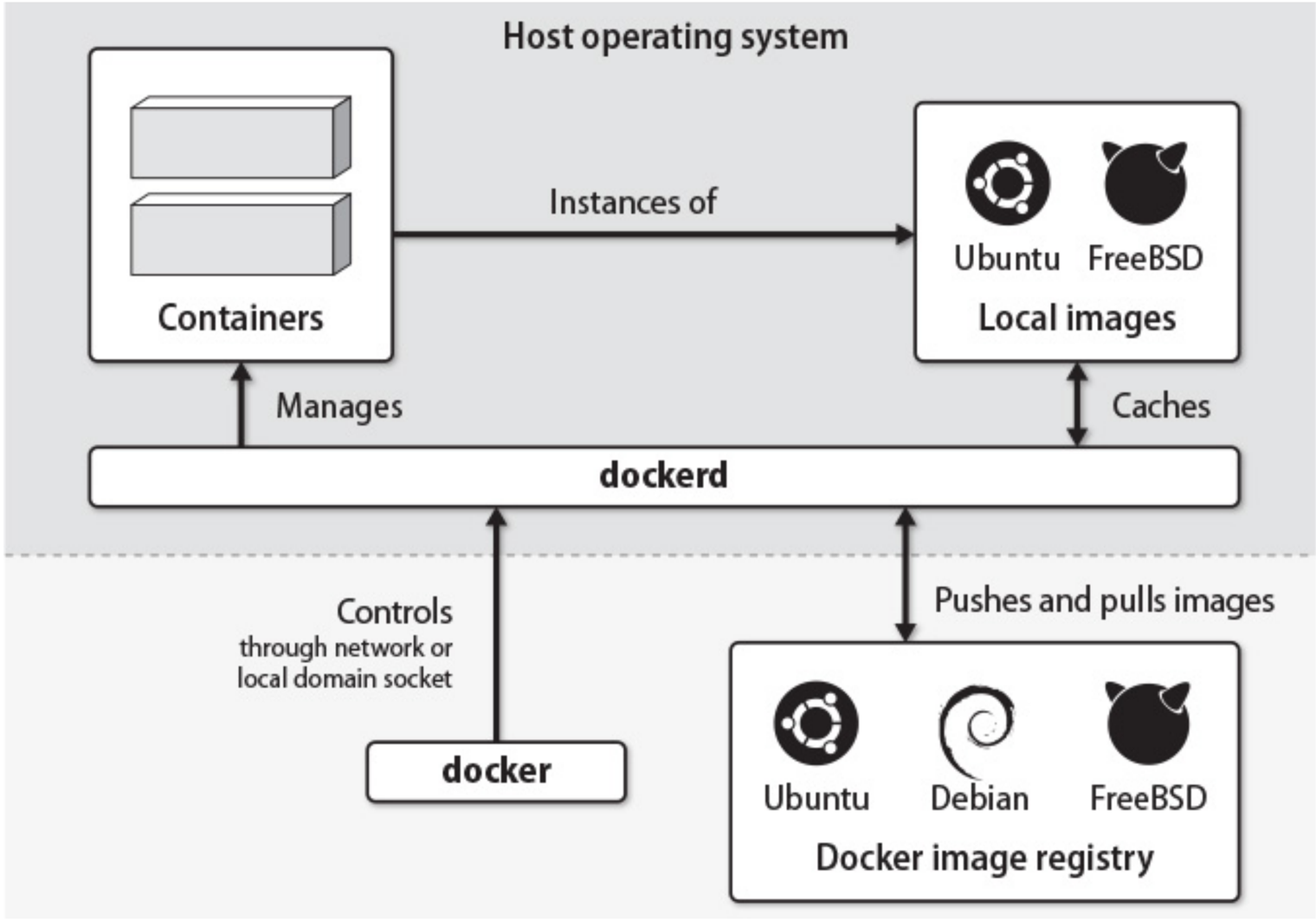
daemon process that implements container and image operations

images

template of containers

You spin up an instance of an image to create a container

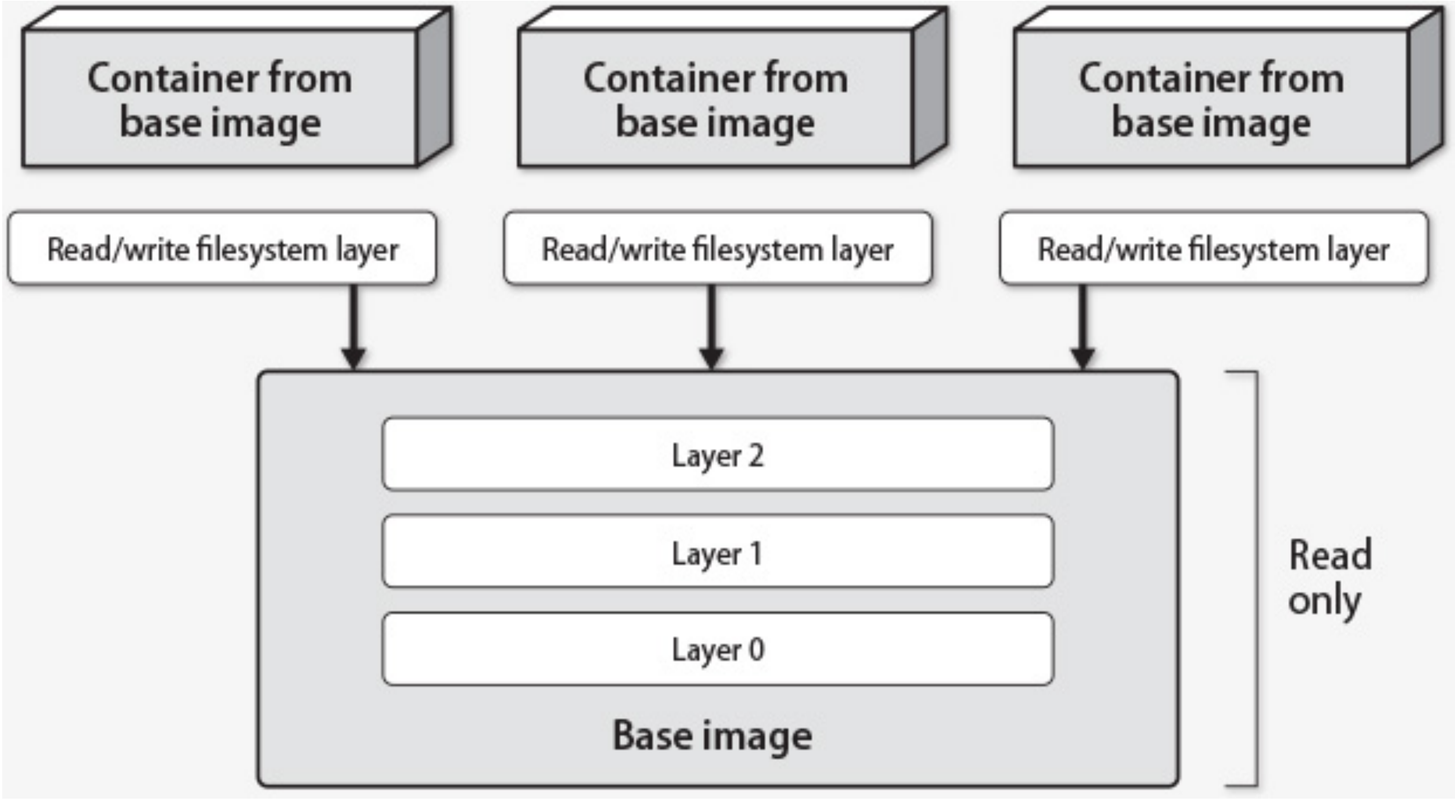
Docker architecture



Docker images

containers are built from images

start with base image, add desired features



Installation

- Docker CE (community edition) is available for free
  - Enterprise editions offer support, security, and reliability
  - https://docs.docker.com/install/linux/docker-ce/ubuntu/
- Installation involves adding a new repository for Docker, and then
  - $ sudo apt install docker-ce

Commands

Once installed, you can pull and run images:

$ sudo apt install docker-ce

- Then run:

```
$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
9bb5a5d4561a: Pull complete
Digest: sha256:f5233545e43561214ca4891fd1157e1c3c563316ed8e237750d59bde73361e77
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Grabbing a full ubuntu

```
$ sudo docker run -it ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
a48c500ed24e: Pull complete
1e1de00ff7e1: Pull complete
0330ca45a200: Pull complete
471db38bcfbf: Pull complete
0b4aba487617: Pull complete
Digest: sha256:d2518289e66fd3892c2dae5003218117abeeed2edbb470cba544aef480fb6b3a
Status: Downloaded newer image for ubuntu:latest
root@77de03c08867:/#
```

Notice the prompt — I'm now running a bash inside the container

Or an nginx

```
$ sudo docker run -p 8080:80 --hostname nginx --name nginx -d nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
2a72cbf407d6: Pull complete
04b2d3302d48: Pull complete
e7f619103861: Pull complete
Digest:
sha256:80e2f223b2a53cfcf3fd491521e5fb9b4004d42dfc391c76011bcdd9565643df
Status: Downloaded newer image for nginx:latest
57edd0c567bf88175ea25168a4186001de4002eac2b69cd09ab3d1107ea13855
```

Once running, I can contact the webserver:

```
$ curl localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
```

and get the index page back

Official repositories

Where are these packages coming from?

- docker repositories
- can be official, or community supplied

Official list: https://hub.docker.com/explore/?page=1

Docker commands

Subcommand	What it does
docker info	Displays summary information about the daemon
docker ps	Displays running containers
docker version	Displays extensive version info about the server and client
docker rm	Removes a container
docker rmi	Removes an image
docker images	Displays local images
docker inspect	Displays the configuration of a container (JSON output)
docker logs	Displays the standard output from a container
docker exec	Executes a command in an existing container
docker run	Runs a new container
docker pull/push	Downloads images from or uploads images to a remote registry
docker start/stop	Starts or stops an existing container
docker top	Displays containerized process status

Creating your own container

- A dockerfile is a recipe for building an image
  - series of instructions and shell commands
  - docker build
    - reads dockerfile
    - executes instructions
    - commits result as an image
- base image is first line of a dockerfile
  - each line after creates a new layer on top of previous image
  - union filesystem merges layers together to make the root filesystem

Creating a dockerfile

Take the nginx image, and change the index.html

Assuming file in the local directory named index.html:

```
$ cat dockerfile
FROM nginx
# add a new index.html to the document root
ADD index.html /usr/share/nginx/html

$ docker build -t nginx:newIndex .
Sending build context to Docker daemon  3.072kB
Step 1/2 : FROM nginx
 ---> b175e7467d66
Step 2/2 : ADD index.html /usr/share/nginx/html
 ---> Using cache
 ---> 698984424614
Successfully built 698984424614
Successfully tagged nginx:newIndex
```

Containers can be saved in private repositories at docker, or kept centrally convenient for your site.

Using containers

Containers are used differently than full machines

Need a scheduled job?

- Don't run cron in the container
- the host machine uses cron to run a container that runs the job and then exits.
- Containers are disposable

Don't run sshd in a container

- use docker exec to open an interactive shell to your container

Using containers

Containers are used differently than full machines

create a container for each logical grouping of services
- webserver in one container
- db in another
- only related apps in same container

don't log in to change things in a container.
- fix the issue in the dockerfile, and
- create a new container

More using containers

Deploy software by replacing containers, not patching them

deployments consist of replacing containers

security: dockerd is what enforces security

- restrict access. Either require sudo, or restrict who you add to docker group
- limit capabilities within container
- ensure you are starting from a trusted base image

Container clustering and management

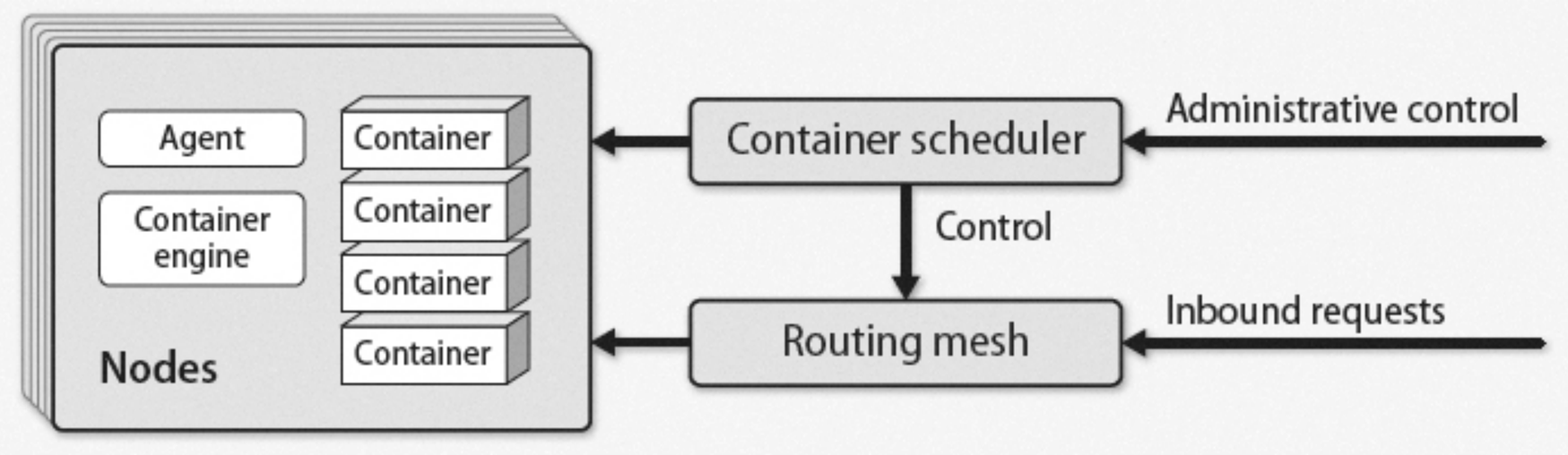
- containers isolate services from each other, and are easily replaceable
- CM systems support docker — to make sure hosts run a set of containers with known configurations
  - doesn't take advantage of standardizing container deployment
- imagine a world where all the host machine has is the OS, plus docker
  - servers become compute farms
    - offer cpu, memory, disk, network
    - when scheduler gets request, finds node with sufficient spare resources, and places container there

Container management

algorithms select best node given job's requested resources and utilization of cluster

placement can accommodate geographic needs

adding/subtracting nodes is easy



Kubernetes

aka k8s is leader in this space

originally designed by google, now maintained by cloud native computing foundation

services

- API server - for incoming requests
- scheduler - place tasks
- controller manager - tracks state of cluster
- the Kubelet - agent that runs on all nodes
- cAdvisor - monitors container metrics
- proxy - routes incoming requests to appropriate container

Kubernetes

- Containers are deployed in "pods"
  - pod contains 1 or more containers
  - pod always co-located on single node
  - short lived
  - pods get cluster wide unique ip
- Services are collections of pods with a fixed address
  - if pod within service dies, it can be replaced
  - built in DNS can assign resolvable names to services
- Google Container Engine implemented on k8s

Docker Swarm

- Docker container cluster manager built into Docker
  - response to Kubernetes, Mesos, others
  - easier to get started with
  - any node running docker can join swarm
  - docker swarm init create a swarm
- services are collections of containers
  - declare: “Three containers with web app”
  - and swarm schedules tasks onto cluster
- built in load balancer
- deploys secure containers by default

AWS EC2 Container Service

- ECS is designed for use with EC2 instances
  - cluster manager components operated by AWS
  - users run instances with docker and ECS agent installed
  - agent connects to central ECS to register resource availability
  - tasks given in JSON format through ECS API
    - ECS schedules
  - integrates with other AWS services
    - load balancing, etc.

Choosing

- Docker
  - ease of use
  - totally integrated inside docker — but limited to docker features
- Kubernetes
  - higher barrier to entry
  - solves more complex situations
  - integrated with Google Container Services
- ECS
  - integrated inside AWS
  - best choice for AWS users often