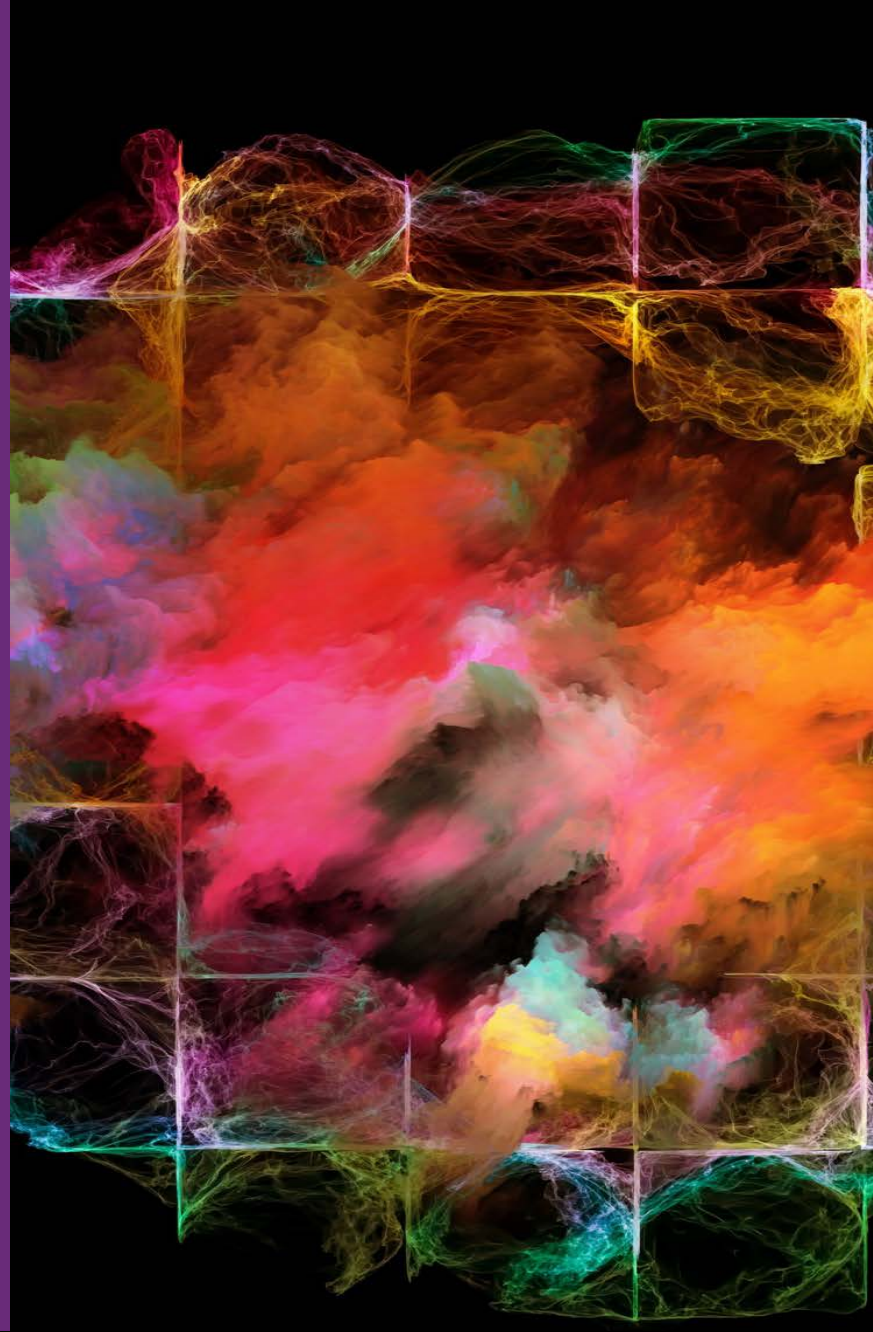
SATURN 2017

13th Software Engineering Institute Architecture Technology User Network Conference

A Hands on Introduction to Docker

Len Bass



Setting expectations

This is an introduction to Docker intended for those who have no hands on experience with Docker.

If you have used Docker you will likely not get much from this session.

The material (and hands on portion) is taken from the course that I teach at CMU called DevOps: Engineering for Deployment and Operations.

Logistics

You should have installed Docker on your laptop – either in native mode or using Docker Toolbox.

Make sure Hello World works (from the installation instructions).

Make sure you have access to the internet since you will be downloading software.

Outline

Introduction to Docker

Hands on

What's left?

Isolation

Process
- Isolate address space
- No isolation for files or networks
- Lightweight

Virtual Machine
- Isolate address space
- isolate files and networks
- Heavyweight

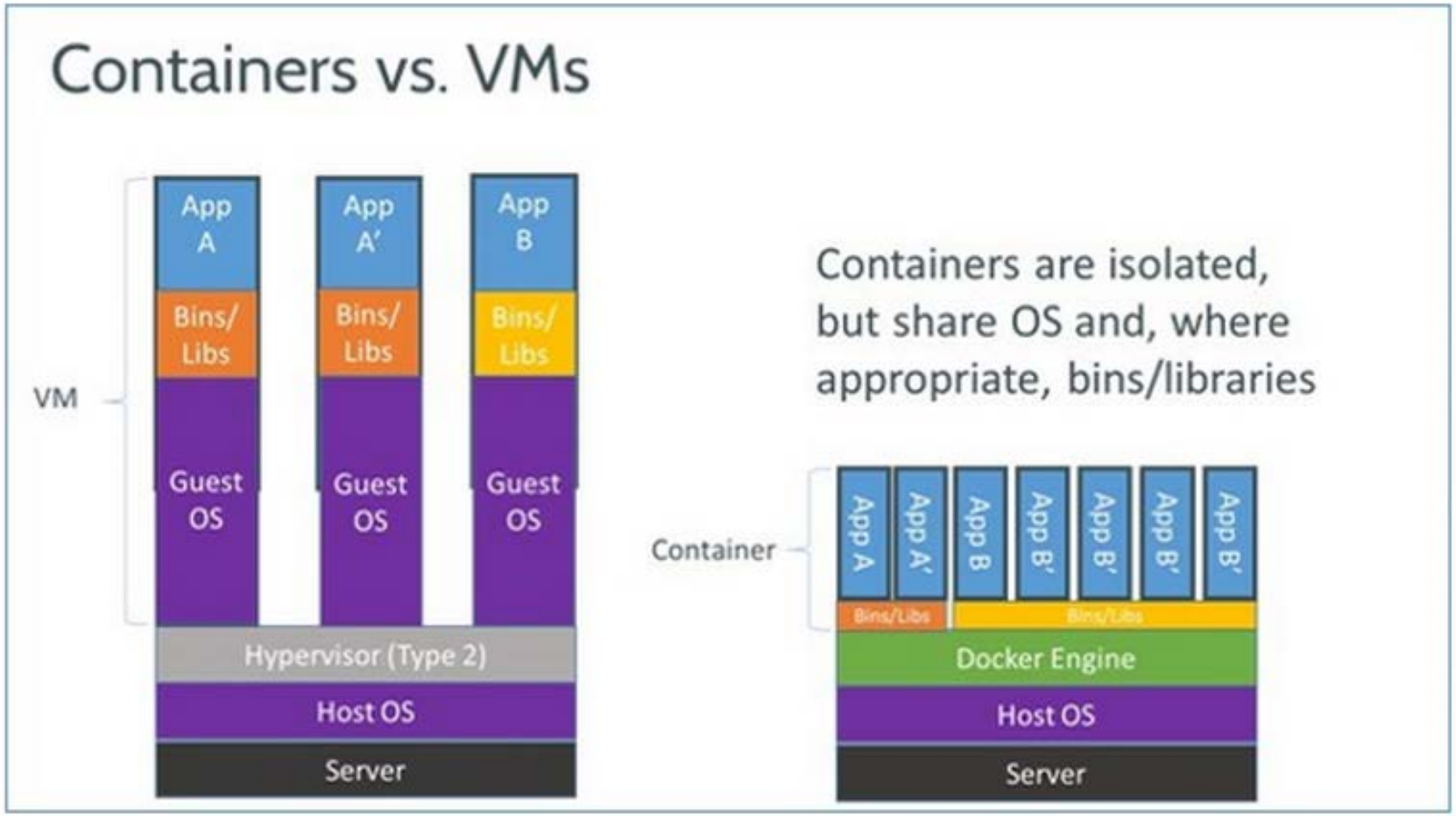
Containers

Process
- Isolate address space
- No isolation for files or networks
- Lightweight

Container
- Isolate address space
- isolate files and networks
- Lightweight

Virtual Machine
- Isolate address space
- isolate files and networks
- Heavyweight

Docker containers



Docker Architecture

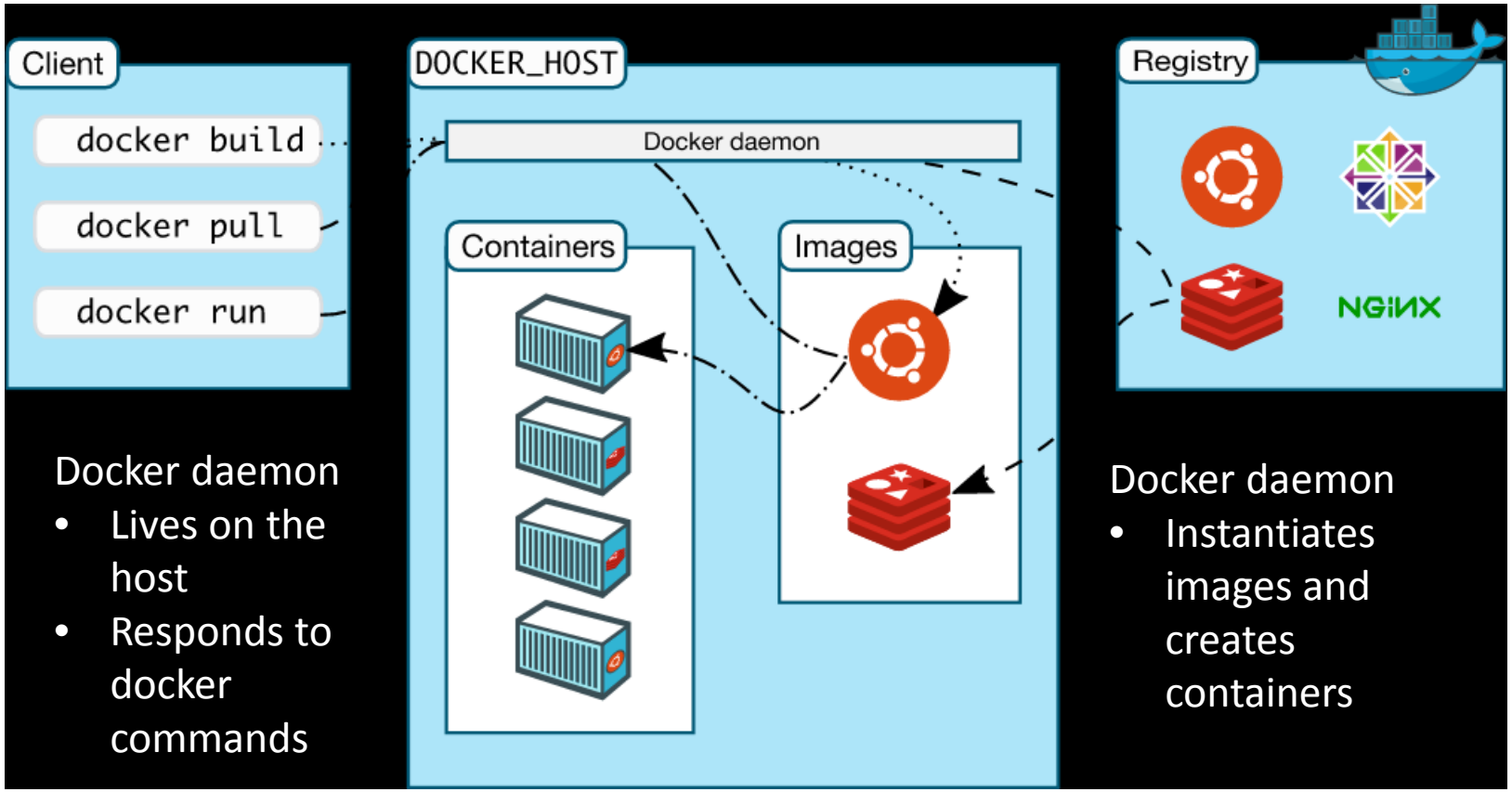


Image is instantiated to form container

Layers

A Docker container image is structured in terms of “layers”.

Process for building image

- Start with base image
- Load software desired
- Commit base image+software to form new image
- New image can then be base for more software

Image is what is transferred

Loading of software

OS is ~ 1GB(yte)

Fast network is ~ 1Gb(it) rated

Since there are 8 bits per byte, transferring an OS should take 8 seconds.

But a 1Gb rated network is ~35Mb in practice

This means loading an OS is >30 seconds

Consequently, sharing an OS saves >30 seconds per instance. Sharing other software saves more

*http://www.tomshardware.com/reviews/gigabit-etherne-bandwidth,2321-3.html

Exploiting layers

When an image is updated, only update new layers

Unchanged layers do not need to be updated

Consequently, less software is transferred and an update is faster.

Trade offs

Virtual machine gives you all the freedom you have with bare metal

- Choice of operating system
- Total control over networking arrangement and file structures

Container is constrained in terms of operating systems available

- Currently just Linux but soon Windows and OSX
- Provides limited networking options
- Provides limited file structuring options

Outline

Introduction to Docker

Hands on

What's left?

Hands on portion

If you have loaded Docker Toolbox, you have a copy of VirtualBox

Set port forwarding on “default” so that 8080 on host is forwarded to 8080 on VM.

docker pull ubuntu

Execute “docker pull Ubuntu”

This loads an image from the docker library

The image contains bare copy of ubuntu

docker images

Execute “docker images”

This generates a list of images known to Docker on your machine

You should see Hello World and ubuntu

docker run –i –t ubuntu

Execute docker run –i –t Ubuntu

This executes an image. An executing image is called a “container”.

You are now inside the container.

Execute “ls”.

- A directory structure is set up but only a bare bones OS has been loaded

Install software on container

Execute

apt-get update

apt-get install wget

apt-get install nodejs

apt-get install npm

<cntl d>

This installs the software you will use during this session and exits the container

docker ps –a

Execute “docker ps –a”

This generates a list of all of the containers that have been run

Output from docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

174268c64fbd ubuntu "/bin/bash" 7 minutes ago Exited (0) About a minute ago sharp_mcnulty

54ae910238b3 hello-world "/hello" 53 minutes ago Exited (0) 53 minutes ago practical_euler

docker commit sharp_mcnulty saturn

Note that the ubuntu container has a name of “sharp_mcnulty” (on my machine). It will be different on yours.

“docker commit sharp_mcnulty saturn” creates an image with the name saturn

Execute “docker images”

REPOSITORY TAG IMAGE ID CREATED SIZE

saturn latest a70567971230 13 seconds ago 456 MB

ubuntu latest 0ef2e08ed3fa 8 days ago 130 MB

hello-world latest 48b5124b2768 7 weeks ago 1.84 k

Execute “run –i –t Saturn”

You are back inside a container. Load application:

wget https://raw.githubusercontent.com/cmudevops/ipshow.js/master/initialization_script

wget https://raw.githubusercontent.com/cmudevops/ipshow.js/master/ipshow.js

Exit the container - <cntl d>

List containers

$docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

9c4b32145fa3 saturn "/bin/bash" 2 minutes ago Exited (0) 8 seconds ago reverent_lewin

174268c64fbd ubuntu "/bin/bash" 30 minutes ago Exited (0) 24 minutes ago sharp_mcnulty

54ae910238b3 hello-world "/hello" About an hour ago Exited (0) About an hour ago practical_euler

Make an image called ipshow

docker commit reverent_lewin ipshow

$ docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ipshow	latest	8f7afedea65d	6 seconds ago	456 MB
saturn	latest	a70567971230	11 minutes ago	456 MB
<none>	<none>	b348af319cbc	21 minutes ago	456 MB
ubuntu	latest	0ef2e08ed3fa	8 days ago	130 MB
hello-world	latest	48b5124b2768	7 weeks ago	1.84 k

Execute app

docker run –i –t –p 0.0.0.0:8080:8080 ipshow /bin/bash /initialization_script

In browser: localhost:8080

You should see three ip addresses in the browser:

- Ip address of local host
- 127.0.0.1 (conventially this is local host)
- Ip address of container

What have we seen

Distinction between docker images and containers

Creating a docker image in layers

Provisioning the docker image from the internet

What is left?

Scripting

Sharing of images

Scaling of images

- Swarm
- AWS container service
- Lambda

Scripting

Creating an image by hand is tedious and error prone

You can create a script to do this (Dockerfile).

Sharing image

Multiple team members may wish to share images

Images can be in production, under development or under test

Docker Hub is a repository where images can be stored and shared.

- Each image is tagged to allow versioning
- Any image can be “pulled” to any host (with appropriate credentials)
- Tagging as “latest” allows updates to be propagated. Pull <image name>:latest gets the last image checked into repository with that name.

Allocation of images to hosts

images

To run an image, the image and the host must be specified

hosts

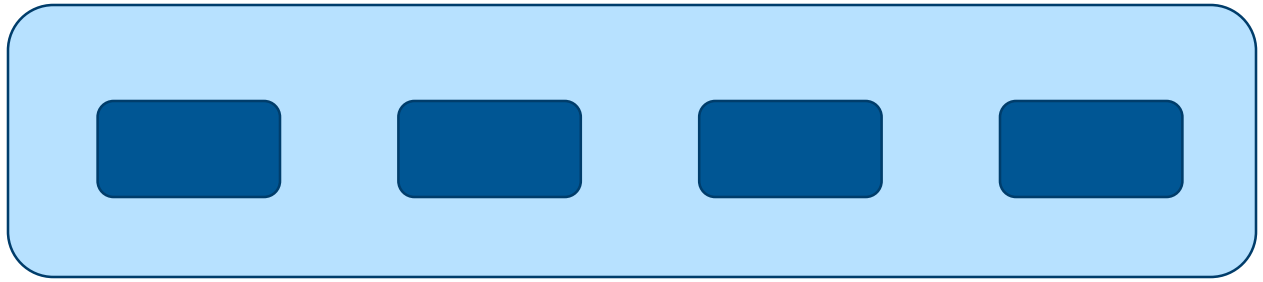
With basic Docker this allocation must be done manually

Docker Swarm

image

To run an image, the image but not the host must be specified

Swarm encapsulates hosts

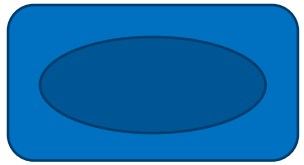


A swarm looks like a single host from the point of view of allocation but actually consists of multiple hosts

Swarm Master

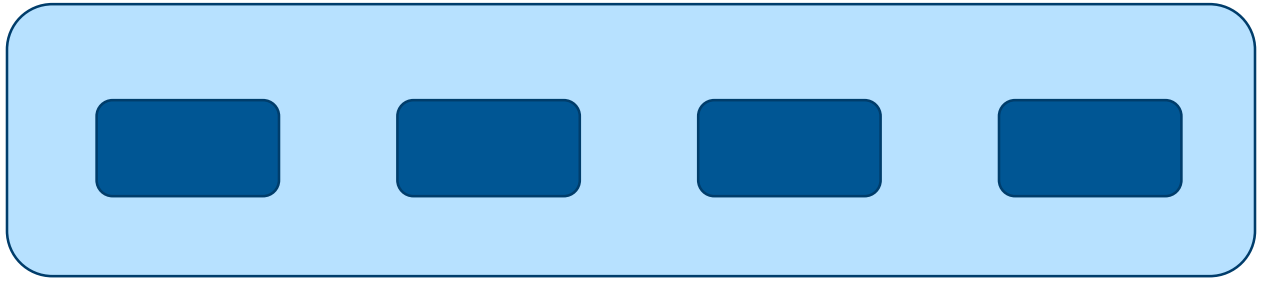
image

Run request is sent to swarm master which selects host



Swarm Master is a specific container on a host not in the swarm

Swarm



How do containers get to hosts?

Three options

- Containers can be copied at each invocation.
  - Copying time is overhead
  - Makes hosts flexible with respect to which containers they run
- Containers can be preloaded on hosts
  - No copying time at invocation
  - When there are multiple different containers, allocator is constrained to allocate to hosts with appropriate containers.
- Some layers can be preloaded on hosts
  - Only copying time for additional layers
  - Allocator is constrained to allocate to appropriate preloaded software

Multiple swarms

It is possible to have multiple swarms simultaneously active

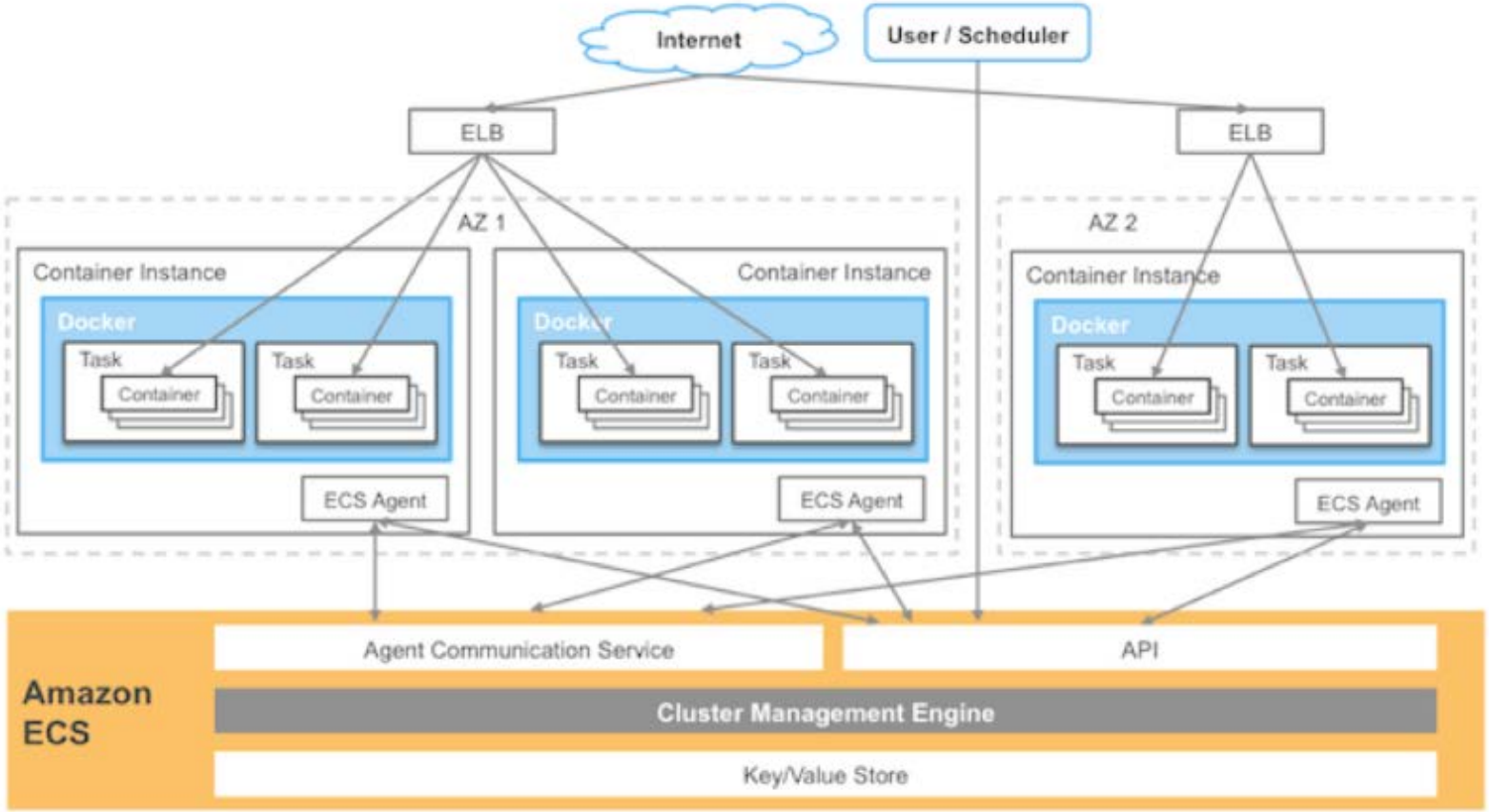
Swarm discovery token is used to identify which swarm each host belongs to

Scaling Swarms

Having an instance in a swarm be automatically replicated depending on workload is accomplished by utilizing autoscaling facilities of cloud provider

AWS has an EC2 container management facility that combines features of Docker Swarm and autoscaling.

AWS EC2 container management



AWS Lambda

AWS also has a facility called “Lambda” that consists of preloaded OS + execution engines. Exists for

- Java
- Node.js
- Python
- C#

AWS maintains pool of partially loaded containers that only require app specific layer.

- Load in micro secs.
- Only one request per Lambda instance

Summary

A container is a lightweight virtual machine that provides address space, network, file isolation

Docker allows building images in layers and deployment of a new version just requires deploying layers that have changed.

Containers can be managed either on VMs through autoscaling or on preallocated pool for short duration, quick loading

Development workflow is supported through an image repository.

Questions and book pitch