Microsoft

Module 01: A Tour of Docker

ILDC Docker Workshop

docker

1

Agenda

- Workshop Objectives and Agenda
- Microservices 101
- Why containers ?
- Docker facts
- Linux Containers
- What is a Linux Container? What is Docker
- Benefits Of Docker

2

Workshop Objectives

- Getting started with Docker containers
- Learn Docker through hands-on labs
- Understand how Docker works behind the scenes
- And most importantly, have fun with Docker!

3

Workshop Agenda

Day 1- User

- A Tour Of Docker
- Under the hood
- Docker Basic CLI

Day 2 - Builder

- Building Containers
- Windows Images
- Volumes

Day 3 - Architect

- Docker Compose
- Registries

4

The Good and Bad of Monolithic Application

The Good - when we are just getting started:

Having one code base, one development language, one runtime, one build, one everything is easy.

- Easy to develop
- Easy to build
- Easy to deploy
- Easy to find bugs
- Easy to maintain
- Easy to scale

5

Micro-services 101

- Small – do one thing and do it well
- Simple!
- Has clear domain boundaries and well defined API's
- Autonomous
- Independent development
- Independent deployment
- Build and release is automated
- Testable
- Loosely coupled

8

Fallacies of Distributed Computing

- The Network is reliable
- Latency is Zero
- Bandwidth is Infinite
- The network is Secure
- Topology doesn't change
- There is one Administrator
- Transport cost is zero
- The network is homogeneous

10

With simplicity, comes complexity

- How to deploy or update services with zero-downtime?
- How to A/B test the application?
- How to handle network failures?
- How to manage security between services?
- How to handle timeouts? Retries?
- How to rate limit? Add quotas? Back-pressure ?
- Telemetry, Logging, Monitoring?
- What about Polyglot, Legacy systems?
- Different Tech Stacks

11

Docker History

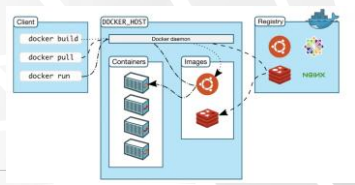
- Docker was created by dotCloud and launch in March 13, 2013 at PyCon Lighting Talk
- dotCloud was a PaaS platform company (now Docker Inc)
- Solomon Hykes is the father of Docker
- Hykes had a cofounder who's now at a partner company (mesosphere)
- Docker is CNCF Silver Member

dotCloud

12

Docker Basic Architecture

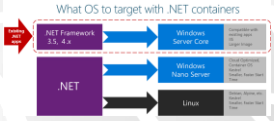
- Docker it's a basic client-server model where the docker client sends REST commands to the docker daemon and the daemon responds.
- Docker components are : Host , Registry and CLI which can all run on the same machine.



13

What about Windows Containers

- Windows Containers can only run-on Windows Host machines.
- Their Footprint is considerably larger than Linux containers
- Only if you have got absolutely no choice
- You will need to switch to Windows-Container Mode in Docker for desktop app



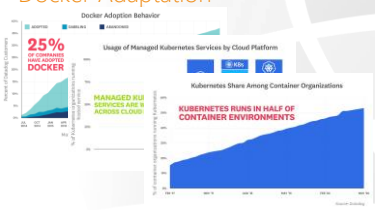
14

Container Definition

- An "Isolated" partition inside a single Operating system
- Very low footprint in comparison to VM's
- Processes inside containers are isolated (networks, storage, secrets)
- Quick deployment
- Multiple environment support : can run it on several guest OS
- Suite for microservices architecture :
  - Isolation
  - Resilience

16

Docker Adaptation



Source: 8 Surprising Facts About Real Docker Adoption | Datadog (datadoghq.com)

17

Resources for this course

- Books :
  - Docker In Action
  - Docker Up and Running
  - Docker: The Complete Introduction to Using Docker Containers Today
  - Docker Cookbook
  - Building Microservices
- Online training (free)
  - Edx -Fundamentals of Containers, Kubernetes, and Red Hat OpenS
  - Cloud Native free training (LFS158)

18

First interaction with Docker

Demo

Demo-01.md - Repos (azure.com)

19

Ex 01: Running Docker Containers

Lab

Lab-01.md - Repos (azure.com)
or
Demo-01.md - Repos (azure.com)

20

Microsoft

Module 02: Under The hood

ILDC Docker Workshop

docker

21

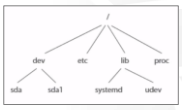
What is a Linux Container (LXC)?

- Is an operating-system-level virtualization method.
- Run multiple isolated Linux systems (containers) on a control host using a single Linux kernel.
- Is the technology on which Docker it's based.

22

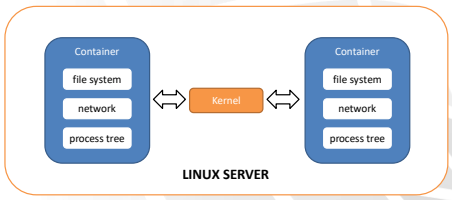
What is a Linux Container (LXC)?

- LXC have Independent:
  - File System
  - Process Tree
  - Networking Stacks
- Linux containers uses:
  - namespaces → Isolation
  - cgroups → Apply limits
  - capabilities → Manage privileges



23

What is a Linux Container (LXC)?



24

But what is this all about?

Containers are a big IT revolution in our days,

But why?

What is this all about?

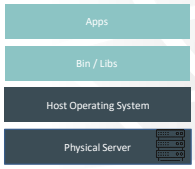
It's quite simple,

Is all about the **APPLICATIONS!**

After all, the OS only exist to facilitate the application

25

Pre-Virtualization World



Problems :

- Huge Cost
- Slow Deployment
- Hard To migrate

26

A little of history before we started…

- Then, virtual machines comes to the rescue!

10 Applications = 1 Physical Servers

1 Physical Server = 10 Virtual Machines

10 Virtual Machines = 10 Operation Systems

(The result, a waste of resources)

27

Hypervisor-based Virtualization

Benefits:

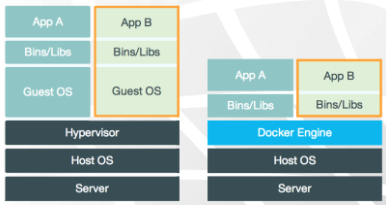
- Cost – Efficient
- Easy To Scale

Limitations:

- Kernel Resource duplication
- Application Portability

28

Docker Vs VM



29

Containers Advantages:

- It's then when **containers** come to solve the problem:
  - More lightweight than VM's
  - Containers consumes less CPU, less RAM and less diskspace.
  - Every container shares a single common Linux kernel in the host
  - Containers are faster and more portable than VM's
  - Provides a secure isolated runtime environment for each container
  - Stackable : You can stack services vertically

(No more operating system for each application)

30

What is Docker?

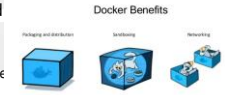
- Docker is the platform for developers and sysadmin to develop, deploy and run applications with containers.
- It brings together the kernel namespaces, cgroups, capabilities and all of that stuff into a product
- Docker provides a very uniform and standard runtime

- Docker is growing more than just a container runtime, becoming more of a platform (registry, clustering, orchestration, networking, etc.)

31

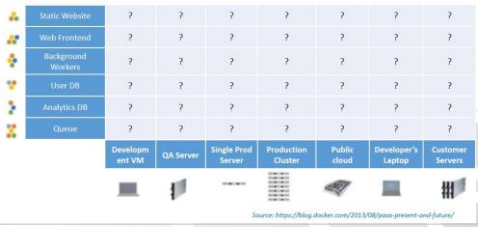
Benefits of using Docker – Development POV

- Packaging software in a way that leverages the skills developers already have
- Bundling application software and required OS filesystems together in a single standardized image format
- Using packaged artifacts to test and deliver the exact same artifact to all systems in all environments
- Abstracting software applications from the hardware without sacrificing resources



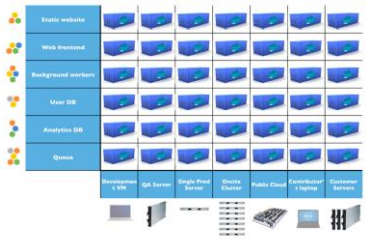
32

The Matrix from Hell Problem



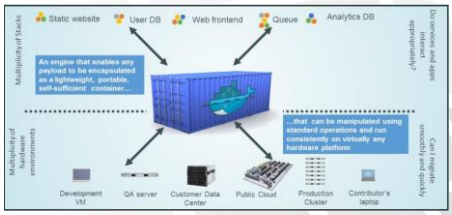
33

Docker eliminates the matrix from Hell



34

The Solution



35

What docker isn't

- Enterprise virtualization platform (VMware, KVM, etc.)
- Cloud platform (OpenStack, CloudStack, etc.)
- Configuration management (Puppet, Chef, etc.)
- Deployment framework (Jenkins,Capistrano, Fabric, etc.)
- Orchestration management tool (Mesos, Kubernetes, Swarm)
- Development environment (Vagrant, etc.)

36

Terminology

- Docker Client – is the *docker* command used to control most of docker workflows
- Docker images - consist of one or more filesystem layers and some important metadata that represent all the files required to run a Dockerized application.
- Docker container – a Linux container that has been instantiated from a Docker image.

37

Microsoft

Module 03: Docker Basics

ILDC Docker Workshop

1

Agenda

- Docker CLI
- $ docker run
- $ docker ps
- $ docker images
- $ docker attach
- $ docker exec
- $ docker rm
- $ docker rmi
- $ docker commit
- $ docker save
- $ docker load

2

The Docker CLI

- Docker commands reference : https://docs.docker.com/engine/reference/commandline/cli/

3

$ docker run

```
$ docker run [OPTIONS] IMAGE[:TAG|@DIGEST] [COMMAND] [ARG...]
```

- Docker runs processes in isolated containers
- If the image is not found locally, it's pulled from the Docker Hub
- A container is a process which runs on a host (local/remote)
- The container exits once the command running inside of it exits
- Detached vs Foreground

4

$ docker ps

```
$ docker ps
```

- Will list the running container in your host

```
$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED   STATUS    PORTS                    NAMES
5f350b815832   registry:2   "/bin/registry /etc/d"   8 months ago   Up 3 hours   0.0.0.0:5000->5000/tcp   registry
```

- "docker ps -a" command show all containers that have run in the past but are not necessarily running now.

5

$ docker images

```
$ docker images
```

- Show all top-level images, their repository and tags, and their size.
- Intermediate layers are not shown by default.
- To see the intermediate layer as well use the flag "-a"

6

$ docker rm

```
$ docker rm <container-id>
```

- Delete a container
- The container must be stopped in order to be removed
- The flag "-f" can be used to remove running containers
- You can remove all the containers at once using the command:
  - $ docker rm $(docker ps -a -q)

7

$ docker rmi

```
$ docker rmi <image-id>
```

- Delete the specified image
- You can delete multiple images in the same commands passing them as arguments, for example:
  $ docker rmi 18wj2 as83j a92k4
- You can remove all the images at once using the command:
  $ docker rmi $(docker images -a -q)

8

Lab 02: Basic commands

Lab

Lab-02.md - Repos (azure.com)

9

$ docker attach

```
$ docker attach <container-id>
```

- Attaches to PID1 inside the container
- To detach from the container use "Ctrl + P + Q"
- Using "Ctrl + C" will stop the process in the container (and therefore stop the container itself)

10

$ docker exec

```
$ docker exec <container-id> <tool>
```

- Runs a new command (process) in a running container
- Useful when the PID1 is not a shell
- You can use the flag -it to run the command interactively

11

Lab 03: Running commands inside the container

Lab

Lab-03.md - Repos (azure.com)

12

$ docker commit

```
$ docker commit <container-id> <new-image-name>
```

- Save container status creating a new image
- By default, the container being committed, and its processes will be paused while the image is committed
- Use the flag "-p=false" to avoid this behavior

13

$ docker save

```
$ docker save -o <path/to/file.tar> <image-id>
```

- Save a container image in a file
- Useful to share containers without a container registry

14

$ docker load

```
$ docker load -i <path/to/file.tar>
```

- Load a container image from a file
- Useful to share containers without a container registry

15

Lab 04: Updating and Sharing Containers

Lab

Lab-04.md - Repos (azure.com)

16

Microsoft

Module 04: Docker Architecture

Docker Workshop

1

Agenda

- Docker Components Overview
- The Docker Engine (Daemon)
- Docker Images
- Docker Containers
- Docker Registries
- Demo 02: Understanding the Docker components

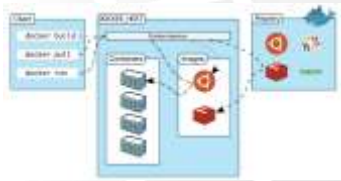
2

Docker Components Overview

- **Shipping Yard:** The infrastructure that's needed to import and export goods
- **Manifests:** List of the container content plus instructions on how to build it
- **Containers:** A package with all the goods ready to be imported/exported

3

Docker Components Overview



4

The Docker Engine (Daemon)

- It's our Docker program that we install on each Docker host to provide us with a Docker environment and access to all Docker services.
- Contains the application infrastructure and runtime dependencies (Standardized).
- Run in the same way in your laptop, datacenter or in the cloud.

5

Docker Images

- Are read only templates used to create containers
- Images are comprised of multiple layers
- The first image is called the "Base Image"
- Intermediate layers increases reusability, decrease disk usage, and speed up the build
- The higher layers win where there are conflicts

6

Docker Containers

- We can think of them as running instances of an image.
- Containers are created from images. Inside a container, it has all the binaries and dependencies needed to run the application.
- Under the hood containers are Linux processes running in the Docker host

7

Docker Registries

- Is a stateless, highly scalable server side application that stores and lets you distribute Docker images.
- A docker registry contains multiple repositories (can be private or public)
- You can pull/push images from repositories
- Docker Hub is the default Docker Registry

8

Demo 02: Understanding the Docker Components

Demo

Demo-02.md - Repos (azure.com)

9