14 – Docker

CS 2043: Unix Tools and Scripting, Spring 2019 [1]

Matthew Milano

February 22, 2019

Cornell University

Table of Contents

CS2043: Docker

Friday 22, 2019

The image above is a link. Click it.

The wonderful world of virtualization

Virtualization, Containers and VMs

- Virtualization is the creation and use of a virtual (logical) version of a resource
- People often use Virtual Machines (VMs) to use multiple operating systems on a single machine without any additional hardware
    - For example, run a copy of MacOS on your Windows 10 machine
    - Relies on a piece of software called a “Hypervisor” to create and manage virtual versions of computer hardware (processor, memory, network card, etc.)
    - Create the illusion that the VM operating system has exclusive access to physical hardware

Virtualization, Containers and VMs

- Containers are a much lighter form of virtualization - they virtualize the environment in which a process runs (runtime environment)
    - Package all code and dependencies so that you can run your program on any system
    - Provides isolation from other processes running on the same machine

Problem Development to Production Delay

Imagine yourself as a web developer who is developing a new feature. During the development phrase, you created a new feature in a development environment (let's say your laptop).

Later when you want to integrate that feature into your web application, you would have to apply the changes to your production environment (let's say your server in the cloud). This can be a nightmare! You must make sure that the code will also run flawlessly in the new environment. What if package X that your feature relies on is version 2.3 locally but is version 2.4 on the clouds and your feature no longer works? Let Docker come to the rescue.

Docker!

What is Docker pictorially?

Figure 1: Docker Image

What is Docker? Overview

- “Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.”
- Concept: build once run anywhere.
- Pro: reduce the delay between development and production.

Why is Docker useful?

- Standardized environment leads to faster development.
- Great for continuous integration and continuous delivery (CI/CD) workflows.
- Highly portable workloads. Docker containers can run on a developer's local laptop, on physical or virtual machines in a data center, on cloud providers, or in a mixture of environments.
- Lightweight and Fast Compared to VMs.

Docker vs. VM

Differences

- No hypervisor (Virtual Machine Monitor).
- Run directly on Host Machine's Kernel.
- Each container run is a discrete process, takes up about the same amount of memory as any other executable.
- Light-weight

Docker Engine

Client-Server Architecture with three components. - Command Line Interface (CLI): the `docker` command. - REST API: talks to the server daemon. - Server Daemon: long-running background process.

Docker Images

An image is an executable package that includes everything needed to run a container: the code, a runtime, libraries, environment variables, and configuration files.

To build your own image, you create a Dockerfile with a simple syntax for defining the steps needed to create the image and run it. Each instruction in a Dockerfile creates a layer in the image. When you change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt. This is part of what makes images so lightweight, small, and fast, when compared to other virtualization technologies.

Docker General and Image Commands

general info

list your current images

remove a specified image

```
docker image rm [IMAGE ID]
```

build a specified image in the current directory

```
docker build -t [IMAGE NAME]
```

References

[1] Stephen McDowell, Bruno Abrahao, Hussam Abu-Libdeh, Nicolas Savva, David Slater, and others over the years. “Previous Cornell CS 2043 Course Slides”.