Docker & K8s Introduction

Welcome

Contents

Containers

- simplify and accelerate worflow when working with mutliple languages, frameworks, architectures, and discontinuous interfaces
- **Docker** is the most popular container technology

Kubernetes AKA K8s

- built by Google based on their experience running containers in production
- now an open-source project
- probably the most popular **container orchestration** technology
- To understand Kubernetes, two fundamental concepts must first be understood: Containers & Orchestration

Why use Docker and K8s?

Realistic Case Scenario:

Let's say you had a project including an end-to-end stack consisting of a web server, database, messaging system, and orchestration.

- Web server: Express JS (Node JS)
- Database: Mongo DB
- Messaging: Reddit
- Orchestration: Ansible

This application uses a lot of different components and this can cause various issues. For example,

- **Compatibility & Dependency**:
    - Each component may have compatibility issues with the current version underlying OS.

    - Each component may have library and dependency issues. Maybe your web service could require one version of dependent library, while the database or orchestration needs another version.
- **Setup time**: Every time a new developer is braught on board, it can be difficult to get them a working environment. They may have to run hundreds of commands at their CLI, using the right versions of the OS and each component.

The architecture of the project will change over time. And every time something changes, it will be inefficient if you have to go back and handle these compatibility/dependency issues to have each component interact. That situation is ocmmonly referred to in the DevOps world as the "matrix from hell".

Docker and K8s help us avoid and prevent dealing with the matrix from hell.

Docker Basic Concepts

Docker allows us to run each component in a what is called a "container". To interact with each piece of software in the stack, users will only need to make sure that they have Docker running on their machines.

What are containers?

- a standardized unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.
- Isolated environments that can run their own processes, network and interfaces, and mounts just like virtual machines, except they all share the same OS kernel.
- Containers are not unique to Docker. There are different types. Docker uses LXC containers.
- Containers have functionality that is "close to the hardware", "low-level". Docker gives a convenient way for developers to interact with the OS at a high-level.

OS concepts

OS (def):

> An interface between a computer user and computer hardware. A software which performs all the basic tasks like file management, memory management, process managemenet, handling input and output, and controlling peripheral devices such as disk drives and printers.

https://youtu.be/rmf04yII2K0?t=369

Virtual Machines:

Containers vs virtual machines:

Container vs docker image:

- Docker image: A package or template just like a VM template in the virtualization world. Used to create containers.
- Container: A running instance of a docker image that is isolated and has its own set of processes.

Advantage of using containers:

What is Docker?

Docker is

> an open-source project that automates the deployment of software applications inside containers by providing an additional layer of abstraction and automation of OS-level virtualization on Linux.

Docker Install (Windows)

Get the executable installer from Docker Hub

Open and run the intaller.

Open a command-line terminal and try out basic Docker commands:

- `docker version`: Chekcs the version
- `docker run hello-world`: Verifies that Docker can pull and run images

> Docker is available in any terminal as long as the Docker Desktop for Windows app is running. Settings are available on the UI, accessible from the Docker whale in the taskbar.

0. Running `docker run hello-world` fails.

After lots of searching around, I found someone wih a similar issue.

- The bottom-most suggestion from Zoe Tao worked when I ran `net start vmcompute` and then rebooted.

> 1. Open "Window Security"
> 2. Open "App & Browser control"
> 3. Click "Exploit protection settings" at the bottom
> 4. Switch to "Program settings" tab
> 5. Locate "C:\WINDOWS\System32\vmcompute.exe" in the list and expand it
> 6. Click "Edit"
> 7. Scroll down to "Code flow guard (CFG)" and uncheck "Override system settings"
> 8. Start vmcompute from powershell: `net start vmcompute`
> 9. Reboot your computer.

This fixed my installation and I was able to successfully use `docker run hello-world` at the Windows PowerShell.

References

- Prakhar Srivastav. *Docker for Beginners*
- Docker 101 Tutorial
- KodeKloud (2018). *Docker introduction in 15 minutes*
- Operating System Tutorial

Kubernetes Concepts

Kubernetes concepts explained in 9 minutes

If a container was to fail,

If a docker host was to fail,

large applications with 1000s of containers

Container orchestration:

- Tools for container orchestration: Docker Swarm, Kubernetes, Mesos