Containers

Concepts

Containers vs. VM

- VM (virtual machine)
  - Virtualization or emulation of a physical hardware computer
  - OS (operating system) runs on a VM instead of a physical hardware computer
  - Multiple VMs can run on a single physical hardware computer
- Container
  - Virtualization or emulation of an OS
  - Some say lightweight VM
  - Most modern computing is done in containers
    - Or rather clusters of containers which we will cover next week

Multiples

- Multiple containers can run in the same VM
- Multiple VMs can run on the same physical hardware
  - On top of a hypervisor on top of the physical hardware
    - Typical of cloud-based VMs
  - On top of an OS on top of the physical hardware
    - Typical of desktop VMs

Containers Can Run Several Places

- Containers can run in a VM running on top of a hypervisor
  - Typical for cloud computing
- Containers can run on an OS on a physical hardware computer
  - Containers running on MS Windows on your laptop or desktop
  - Containers running on Mac OS on your laptop or desktop
- Containers can run in a VM on an OS on a physical hardware computer
  - Containers running in a VM on MS Windows on your laptop or desktop
  - Containers running in a VM on Mac OS on your laptop or desktop

Same Everywhere

Containers are the same, regardless of where they are run.

- Major benefit of container vs. VM
- Emulation at the OS level removes dependencies of where they run

Overhead

- Containers add a software layer on top of the VMs
- Containers run slower than VMs without containers
- General consensus
  - Overhead of containers is worth it
  - Benefits outweigh the performance hit
- Performance hit is nowhere near the performance hit of a desktop GUI

The End

Containers

Business Cases

Learning

- We want to learn a new, complicated product that takes hours to install and configure
- Overhead of installation and configuration is greater than overhead of actually learning the product
- Solution
  - Use a container for the product which we can create in minutes without a complicated installation and configuration

POC: Proof of Concept

- Company wants to build a POC
- Does not want to invest in learning installation, configuration, and administration until they are sure they want to use the product
- Solution
  - Use a container for the product which we can create in minutes without an installation or configuration, and in which administration not needed

Development

- Company wants each developer to have a full environment of their own so they do not interfere with other developers
- Solution
  - Each developer uses their own private container

QA: Quality Assurance

- Likewise, each QA tester needs their own private environment similar to developers
- Solution
  - Each QA tester uses their own private container

Production

- Company has been doing development and QA using containers
- Ready for production
- Containers run slower than VMs
- Production more complicated with VMs
- Solution
  - Use a container for production, since the small overhead of the container is worth it to simplify production

The End

Microservices

Concepts

SOA: Service-Oriented Architecture

- Most modern apps use SOA
- SOA typically involve large monolithic services due to prior technology limitations on hardware and software
- Modern technologies allow us to use smaller services
- Will cover more when we get to web APIs

Microservices

- Smaller services
- Better performance
  - Run faster
  - Less CPU
  - Less memory
- Easier to develop and maintain

Bug Worst Case

- What is the worst case for a software bug?
  - Crash? Actually, not the worst case
- Worst case is a bug that causes memory corruption
  - Not bad enough to crash
  - Introduces errors
    - Current user
    - Other users
  - Data corruption errors might not be found for hours, days, or even weeks
    - Hard to go back and figure out what is wrong and how to correct it

Rebooting

- Ever had a time when your phone or tables or laptop was acting flaky, you rebooted, then everything worked fine?
- Memory corruption errors from a bug that was not enough to crash
- Rebooting cleared out the memory corruption errors

Microservices Using Containers

- For each transaction (or login), give each customer their own private container
- When transaction ends (or logout), destroy the container
- Equivalent to a reboot for each transaction or user
- Containers give a wall of separation between current users—added security
- Fresh container ensures that future users will not be impacted by past memory corruption

Container Microservice Overhead

- Creating a new container for each transaction or user login costs more money
  - Few cents per container
- General consensus
  - Worth the cost because a single data corruption could cost millions or more to fix

The End

Microservices

Business Cases

Bank Website

- Bank has a website
- Customers login, access their accounts, perform transactions, logout
- Solution
  - Create a private container for each user every time they login and destroy the container when they logout

Similar

- Banks
- Credit cards
- Brokerage
- Utility companies
- Insurance
- Online stores
- Etc.

POS: Point of Sale

- Transaction: computerized cash register scans items, looks up prices, applies coupons, computes subtotal, computes taxes, computes final total, processes customer payment
- Solution
  - For each transaction, create a container, execute the transaction in the container, then destroy the container when the transaction is completed

The End

Container Images

Concepts

Analogy to Object-Oriented

- Object-oriented
  - Object is an instance of a class
  - A class can have unlimited number of objects
- VMs
  - VM is an instance of a VM image
  - An image can have unlimited number of VMs
- Containers
  - Container is an instance of a container image
  - An image can have unlimited number of containers

Analogy to Class Hierarchies

- Class hierarchy in object-oriented
  - Class can be the parent of other classes
  - Class can be the child of other classes
- VM image hierarchy
  - VM image can be the parent of other VM images
  - VM image can be the child of other VM images
- Container image hierarchy
  - Container image can be the parent of other container images
  - Container image can be the child of other container images

Container Image Building

- Scripted process
  - Start with a container image
  - Create a container from the image
  - Customize the container
    - Install more software
    - Change configuration
    - Add users, groups, directories, files, permission, etc.
  - Create a new image from our customizations
- Images of images of images… allowed
- Base images are pure OSs

The End

Container Images

Business Cases

Software Vendor

- Software product has a complicated install and configuration that takes hours and pages of instructions
- Solution
  - Take a well-known OS container image
  - Create a container
  - Install and configure their software into that container
  - Make a new image
  - Distribute new image to customers
  - Customers can create containers from the new image

Customizing Vendor Software

- Software vendor has given us a container image with everything installed and configured
- We have some custom configuration parameters we want to change and some custom files to add
- Solution
  - Take the vendor’s image, create a container, add our customizations, make a new image that we will use to create containers

The End