Day3

Thursday, September 8, 2022 8:14 AM

Containers vs Virtual Machines

Both provide the ability to isolate processes

Virtual Machine: simulates an entire computer on top of a PC
- They virtualize an entire OS
- Enabled by hypervisors

Pros:
- Complete isolation
- Provides virtualization
- ensure any application runs reliably regardless of the host

Cons:
- very bulky, and expensive in context to resources

Containers: bundle together applications and all needed dependencies runs them in isolation
- Containers share the underlying OS kernel
- lighter weight than VM
- be provided by tools such as docker

Pros:
- considered lighweight
- you can enable layers of isolation
- provides a virtualized view of certain resources
- package an application in an isolated environment
- ensure the app will run reliably

Cons:
- having layers of isolation could also cause issues with communication

Containerization:
- the process of bundling an app into a container
- makes sure we can run the app reliably
- the container should not be able to modify or interact directly with anything it doesn't need to
- the container should have no effects on the host

Linux Containers are the foundation of modern containerization
Started with cgroups and namespaces
- cgroups allow control over certain resources
- namespaces allow for encapsulation of resources

Containers are:
- built from an image
- run on a container engine
- Ideally stateless
- Virtualized and isolated

More benefits:
- Security
- Standardized and portable
- Lighweight
- flexible and loosely coupled
- Scalable

Intro to Docker

Open source platform for developing, shipping, and running application using containers

Dockers uses a Client-Server Architecture
- You run commands in the CLI that sends requests to the docker daemon

Docker CLI (Comand Line Interface)
- Interacts with the daemon
- The client in the client server architecture
- Uses a rest API to communicate with the daemon

Docker Daemon
- the long running process that does the heavy lifting for docker
- manage docker objects
  - containers
  - images
- core of the running dockerized applications

Dockerhub Registry
- Stores images for others to use with docker

Docker Objects:

The building blocks that are managed by the docker daemon
- images and containers

Docker images:
- Blue print for a container
- Outlined in a dockerfile

Docker Containers:
- running isolated instance of a set of processes and their dependencies
- outlined by an image
- managed by the docker engine

Dockerfiles: step by step instructions on how to configure and run a container
- Made up of commands which are listed below

Docker Keywords:

FROM image name
- Specify the parent image from which to create this image

RUN <command> / RUN ["executable", "params"]
- Used to setup your image, the state of the image after each command is run forms a new layer

ADD <src> <dest>
- Add files to the image

EXPOSE
- Expose ports that can be used from outside the container

VOLUME [/dirname]
- Create a mount point inside the image

WORKDIR
- Set the working directory in the image and eventual container

CMD
- Another way to run commands inside of the containers

Building a Docker Image

Two choices:
- docker build command
- [illegible]

Two choices:
- docker build command
  - creates an image from a docker file
- docker commit:
  - commits the changes from a container you specify
  - creates an image based off that container

Creating the container

Two ways to do this
- docker create command
  - creates the container from the image in the ready to run state
- docker run command
  - build a container from an image either locally or from the registry
  - and run the container

Comands to manage Containers

Docker container ls
- Lists all running containers

Docker ps –a
- List all containers even none running

Docker container kill
- Stop a container which you specify

Docker container pause
- Pause a container which you specify

Docker container start
- Start a container you specify

Docker container rm
- Remove a specified container

Docker volume rm
- Remove a specified volume

Docker Best Practices

The containers should ephemeral
- easily tear down and start up

Be mindful what directory you are building in

Try to leverage multistage and image cache

Each container should serve only one purpose

Make your dockerfile commands readible by adding new lines

Use volumes to persist any data

Use secrets for sensitive data