Introduction to Docker containers

Container

* An isolated environment that allows us to build and run software.
* It includes code and dependencies to run apps quickly without any pc environment aka ‘images’
* The container image becomes what we use for docker

Software containerization

* A method of virtualizing to deploy and run containers without a VM. It can have a physical hardware, cloud, and across multiple VMs

Docker

* Platform used to develop, run, and ship containers
* Desktop version supports multiple OS such as linux, winows, and MacOs
* Consists of several components to build

Docker hub

* Container registry, basically a repository that is used to store and distribute created container images
* Default registry for images created using docker

Container image

* Portable package that contains software
* Once created it cannot be changed unless you create a new one

What is host OS

* The OS in which the docker engine runs on.
* Linux shares OS kernel and doesn’t require container OS
* Windows however does

Base image

* Image that uses docker from ‘scratch’. Scratch is an empty container that does not create a file system layer
* More custom and more control over the image

Parent image

* Container image from which you create your images
* Immutable as you need to create a new image to make changes as it is a parent image

Dockerfile

* Basically a set of instructions on how to create the container image
* Includes things such as OS, parent image, services, commands to run

Container image

A container image is a read-only portable package that contains software and may include an operating system.

Docker build

* You use the docker build command to rebuild a container image. Once you've built an image, the image can't be changed. The only way to change an image is to create a new image.

Docker Desktop is an app for building and sharing containerized apps and microservices available on which of the following operating systems?

Windows, macOS, and Windows Subsystem for Linux (WSL)

While Docker Desktop is only available for Windows and macOS, it does support using a Linux command line via Windows Subsystem for Linux (WSL). For more information and set up, see the WSL + Docker documentation.

Container

* Container names are unique meaning you can’t reuse it to create a new container unless you remove the previous one

Common docker commands

|  |  |
| --- | --- |
| Docker ps -a | View all containers |
| Docker run -d tmp-ubuntu | Run a container, -d runs the container with the website in the background |
| Docker pause <container name> | Pause container |
| Docker unpause | Resumes paused container |
| Docker restart | Receives stop command, then restart |
| Docker stop | Stops container |
| Docker rm | Removes container |
| Docker volume create | Creates a storage option |
| Docker stats | Returns information about container such as cpu usage, memory, I/O written on disk |

Docker storage

* Container storage is temporary
* Two options for storage called ‘volumes’ and ‘bind mounts’

Docker volume

* Docker storage option
* Multiple containers can use the same volume
* Stored within directories on the host filesystem
* Can be created during the container creation process in a dockerfile
* Stored at a specific location

Bind Mount

* Same as volume but can be stored anywhere
* Limited capability compared to volume

Container network configuration

* Bridge
* Host
* None

Bridge

* Default config used for container when created
* Internal, private network, and isolates the environment

Host

* Enables to run container on the host directly and removes isolation between the host and the container
* Only use ports not already used by the host

Sample Questions

Which storage option is the best choice that allows the host and container to share a file to manage name server resolution, for example the resolve.conf file on Linux?

A bind mount, like a volume, is stored on the host filesystem at a specific folder location. However, bind mount data is expected to be updated by the host. The resolve.conf contents is expected to change by the host and used by both the container and host.

Docker benefits

* Run without use of virtual machines, relies on host kernel
* Can run multiple containers without affecting each other due to configuring data storage and network configuration to isolate containers or to share data between each other
* Can run almost anywhere making it lightweight and easy to move between different environments
* Each container can be managed easily due to being isolated making it easy
* Docker containers are supported by many cloud deployments

**Choose a data storage approach in Azure**

Choosing data storage varies for different needs. We need to consider how data will be used, classified, and how to get the best performance for it.

Structure Data

* Data that follows a strict schema, so most data share the same fields or properties
* Easy to search using SQL
* Usually used in business or inventory management
* Data that neatly fits and

Semi-Structured

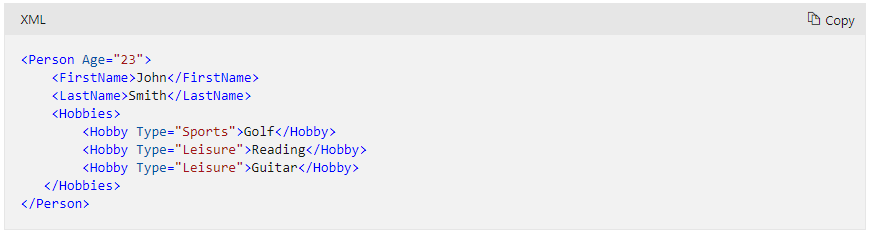
* Less organized and is not stored in a relational format
* Aka non-relational data
* Still organized and has clear properties and vales, with variety to the data

Data serialization is used to write data stored in memory to file, sent to another system, parsed, and read.

Common formats include

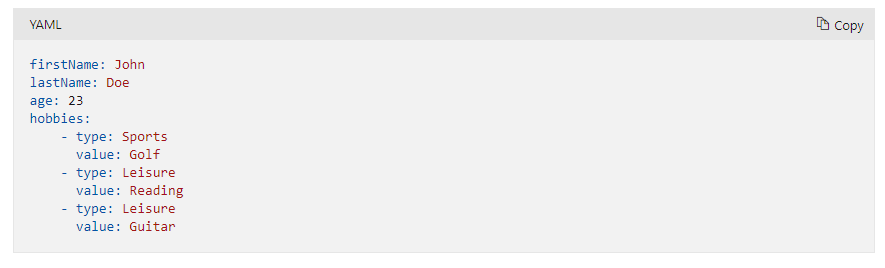
XML

* Can have parent and child elements to represent data
* Structured data



YAML

* New data language that’s known for its user friendliness



JSON

* Javascript, relies on curly brackets for data structure that’s mostly used in web services to return data
* Semi structured

Unstructured data

* Files such as photos and videos because the data that comprises files are unstructured

Transaction

* Logical group of database operations that execute together

ACID

Atomicity, executes exactly once

Consistency, ensures data is consistent before and after transaction

Isolation, not impacted by another transaction

Durability, changes made in the transaction is permanent

OLAP

* Online analytical processing
* Supports few users, longer response times
* Large and complex transactions

OLTP

* Online transactional processing
* Lots of users, quick response times, and handle larger volumes of data
* Handles small and simple transactions

Possible scenarios

Product catalog data

* Semi structure data, need to extend or modify data for new products
* Transactional support because data is historical but changing
* Azure cosmos DB

Photos and Videos

* Unstructured data
* No transactional support
* Azure blog storage

Business data

* Structured data due to complex transactions across many databases