Fourth Brain Homework

**Rubric Q & A**

**Dr. Shai**

**MLE 10 Cohort**

**Q1. What does it mean to create a Docker image and why do we use Docker images?**

* A Docker image is a read-only template that contains instructions for creating a Docker container.
* It enables you to create, deploy, and run applications conveniently with the help of containers.
  + It is used to package up application code and its dependencies in order to be deployed in a reproducible and isolated environment.
    - Docker images can be created by either building them from scratch using a Dockerfile, or by using existing images as a base for a new image.
* Creating a Docker image allows for applications to be packaged up and deployed in a predictable and efficient manner.
  + This helps to ensure that applications will work the same regardless of where they are deployed.
* Advantages of using docker:
  + it provides you with a consistent and isolated environment. It takes the responsibility of isolating and segregating your apps and resources in such a way that each container becomes able to access all the required resources in an isolated manner i.e., without disturbing or depending on another container. It eventually allows you to run multiple containers simultaneously on the same host.
  + It efficiently organises the entire development lifecycle by providing a standardised working environment to the developers.
  + Docker images can be easily sorted across multiple servers.
  + Docker containers can run on any platform whether it be Amazon EC2, Google Cloud Platform, VirtualBox, Rackspace server, or any other – though the host OS should support Docker. As the application and all its dependencies are packaged together in a Docker container – you can deploy it to any system that supports Docker and the application will perform similarly.

**Q2. Please explain what is the difference between a Container vs a Virtual Machine?**

* A virtual machine (VM) is a virtual physical computer.
  + VMs can run what appear to be multiple machines, with multiple operating systems, on a single computer.
  + Virtual machines are heavy software packages that provide complete emulation of low level hardware devices like CPU, Disk and Networking devices. Virtual machines may also include a complementary software stack to run on the emulated hardware. These hardware and software packages combined produce a fully functional snapshot of a computational system.
  + They interact with physical computers by using lightweight software layers called ***hypervisors***.
    - Hypervisors can separate VMs from one another and allocate processors, memory, and storage among them.
* VMs are also known as virtual servers, virtual server instances and virtual private servers.
* Containers are light-weight, agile ways of coordinating virtualization — and don't use a hypervisor > so resource provisioning is faster
* Containerization packages together everything needed to run a single application or microservice (along with runtime libraries they need to run).
* The container includes all the code, its dependencies and the OS itself, enabling apps to run almost anywhere — a desktop computer, a traditional IT infrastructure or the cloud.
* Containers use a form of OS virtualization leveraging features of the host operating system to isolate processes and control the processes’ access to CPUs, memory and desk space.
* While containers virtualize the OS, each individual container contains only the application and its libraries and dependencies.
* Containers are small, fast, and portable because, unlike a virtual machine, containers do not need to include a guest OS in every instance and can, instead, simply leverage the features and resources of the host OS.
* Just like virtual machines, containers allow developers to improve CPU and memory utilisation of physical machines.
  + Containers go even further, however, because they also enable microservice architectures, where application components can be deployed and scaled more granularly.
* This is an attractive alternative to having to scale up an entire monolithic application because a single component is struggling with load.
* It is entirely possible to use containers and virtual machines in unison although the practical use-cases may be limited. A virtual machine can be created that emulates a unique hardware configuration. An operating system can then be installed within this virtual machine's hardware. Once the virtual machine is functional and boots the operating system, a container runtime can be installed on the operating system.
* One practical use for this configuration is experimentation for system on chip deployments.
  + Popular systems on chip computational devices like the Raspberry Pi, or BeagleBone development boards can be emulated as a virtual machine, to experiment with running containers on them before testing on the actual hardware.

**Q3. What are 5 examples of container orchestration tools (please list tools)?**

Container orchestration is a process for managing the deployment, integration, scaling, and life cycles of containerized software and applications in complex, dynamic environments. As an automation tool, it analyses, organises, and integrates applications and services at the base operating level.

The orchestration tool then manages the container's lifecycle based on specifications in the configuration file. The orchestration tool will also use system parameters like CPU and memory capacity, and file parameters like proximity and file metadata, to help inform its decisions.

To support scaling and help maintain productivity, orchestration tools automate many of these tasks. Repeatable patterns in Kubernetes are used as building blocks by developers to create complete systems.

i) Kubernetes (K8s)

* Kubernetes is an open-source platform that was originally designed by Google and now maintained by the Cloud Native Computing Foundation. Kubernetes supports both declarative configuration and automation. It can help to automate deployment, scaling, and management of containerized workload and services.
* Kubernetes API helps to establish communication between users, cluster components, and external third-party components. Kubernetes control plane and Nodes run on a group of nodes that together form the cluster. Application workload consists of one or more Pods that runs on Worker node(s). The control plane manages Pods and worker nodes.
* Companies like Babylon, Booking.com, AppDirect extensively use Kubernetes.
* **Key Features:**
  + Service discovery and load balancing
  + Storage orchestration
  + Automated rollouts and rollbacks
  + Horizontal scaling
  + Secret and configuration management
  + Self-healing
  + Batch execution
  + IPv4/IPv6 dual-stack
  + Automatic bin packing
  + Automated deployment, rollouts, and rollbacks.
  + Automatic scalability and controllability
  + Isolation of containers.
  + Ability to keep track of service health
  + Service discovery and load balancing
  + It works as a platform providing service.

**Specific examples include:**

* IBM Cloud Kubernetes Service
* Azure Kubernetes Service (AKS)
* Amazon Elastic Kubernetes Service (EKS)

ii) Amazon Elastic Container Service (ECS)

* Amazon ECS is a container orchestration tool that runs applications in a managed cluster of Amazon EC2 instances. ECS powers Amazon services like Amazon.com’s recommendation engine, AWSBatch, and Amazon SageMaker and this setup ensures the credibility of its security, reliability, and availability. Therefore ECS can be considered as suitable to run mission-critical applications.
* **Key Features**
  + More effective load balancing.
  + Similar to EKS, ECS clusters run in serverless AWS Fargate.
  + Run and manage Docker containers.
  + Integrates with AWS App Mesh and other AWS services to bring out greater capabilities. For example:
  + Amazon Route 53,
  + Amazon CloudWatch
  + Access Management (IAM)
  + AWS Identity,
  + Secrets Manager
  + Support for third party docker image repository.
  + Support Docker networking through Amazon VPC.
  + Payment is based on resources per application.
  + Provision and managed servers are not needed.
  + Updated resource locations ensure higher availability.
  + End to end visibility through service mesh
  + Networking via Amazon VPC ensures container isolation and security.
  + Scalability without complexity.

iii) Docker Platform

* Docker Orchestration tools facilitate the SDLC from development to production while Docker swarm takes care of cluster management. It provides fast, scalable, and seamless production possibilities for dispersed applications. A proven way to best handle Kubernetes and containers.
* It enables building and sharing Docker images within teams as well as large communities. Docker platform is extremely popular among developers. According to a Stack Overflow survey, it ranked as the most “wanted,” “loved,” and “used” platform.
* **Key Features**
* It supports both Windows and Linux OS
* It provides the ability to create Windows applications using the Docker Engine (CS Docker Engine) and Docker Datacenter.
* It uses the same kernel as Linux, which is used in the host computer.
* Supports any container supported infrastructure.
* Docker Datacenter facilitates heterogeneous applications for Windows and Linux.
* Docker tools can containerize legacy applications through Windows server containers.
* It provides a perfect platform to build, ship, and run distributed systems faster.
* Docker provides a well-equipped DevOps environment for developers, testers, and the deployment team.
* Improved performance with cloud-like flexibility.
* Smaller size as it uses the same kernel as the host.
* It provides the ability to migrate applications to the cloud without a hassle.

**iv) Helios**

* Helios is an open-source platform for Docker by Spotify. It enables running containers across many servers. Further, it avoids a single point of failure since it can handle many HTTP requests at the same time. Helios logs all deploys, restarts, and version changes. It can be managed through its command-line and via HTTP API.
* **Key Features**
  + Fits easily into the way you do DevOps.
  + Works with any network topology or operating system.
  + It can run many machines at a time or a single machine instance.
  + No prescribed service discovery.
  + Apache Mesos is not a requirement to run Helios. However, JVM and Zookeeper are prerequisites.
  + Advantages
  + Pragmatic
  + Works at scale
  + No system dependencies
  + Avoid single points of failure

v) Azure Service Fabric

* ASF is a distributed service framework for managing container-based applications or microservices. It can be either cloud-based or on-premise. Its scalable, flexible, data-aware platform delivers low latency and high throughput workloads, addressing many challenges of native cloud-based applications.
* A “run anything anywhere” platform, it helps to build and manage Mission-critical applications. ASF supports Multi-tenant SaaS applications. IoT data gathering and processing workloads are its other benefits.
* **Key Features**
  + Publish Microservices in different machines and platforms.
  + Enabling automatic upgrades.
  + Self-repair scaling in or scaling out nodes.
  + Scale automatically by removing or populating nodes.
  + Facilitates the ability to have multiple instances of the same service.
  + Support for multi-language and frameworks.
  + Low latency and improved efficiency.
  + Automatic upgrades with zero downtime
  + Supports stateful and stateless services
  + It can be installed to run on multiple platforms.
  + Allows more dependable resource balancing and monitoring
  + Full application lifecycle management with CI/CD abilities.
  + Perform leader election and service discovery automatically.

**Q4. How does a Docker image differ from a Docker container?**

* A Docker image is a read-only immutable template that defines how a container will be realised.
* A Docker container is a runtime instance of a Docker image that gets created when the $ docker run command is implemented.
* Before the docker container can even exist docker templates/images are built using $ docker build CLI.
* Docker image templates can exist in isolation but containers can't exist without images.
* So docker image is an integral part of containers that differs only because of their objectives which we have already covered.
* Docker images can’t be paused or started but a Docker container is a run time instance that can be started or paused.
* When a Docker user runs an image, it becomes one or multiple container instances. The container’s initial state can be whatever the developer wants — it might have an installed and configured web server, or nothing but a bash shell running as root. In practice, though, most images include some preconfigured software and configuration files.
* Docker images are immutable, so you cannot change them once they are created. If you need to change something, create another container with your changes, then save those as another image. Or, just run your new container using an existing image as a base and change that one.

A continuous integration solution like CircleCI enables developers to automate builds, tests, and deployments. CircleCI can use Docker containers to make deploying your applications to multiple environments easier. For example, CircleCI can build Docker images and push them to a container image registry like Docker Hub. From there, it can instantiate the images into containers in Kubernetes, OpenShift, or elsewhere.

The flow works like this:

* You commit changes to your Git repo.
* This commit triggers a CircleCI build job that checks out the source code from Git and runs unit tests on the code.
* If the unit tests pass, CircleCI pushes the built image to Docker Hub.
* If the unit tests fail, CircleCI alerts the developer and stops the workflow.

**References**

Docker overview (2022). <https://docs.docker.com/get-started/overview/>

Kisller, E. (2023). A Beginner’s Guide to Understanding and Building Docker Images <https://jfrog.com/knowledge-base/a-beginners-guide-to-understanding-and-building-docker-images/>

Gillis, A. (2023). Docker image <https://www.techtarget.com/searchitoperations/definition/Docker-image>

Geeksforgeeks. (2022). Why to use docker. <https://www.geeksforgeeks.org/why-should-you-use-docker-7-major-reasons/>

Containers vs. Virtual Machines (VMs): What’s the Difference? (2022). <https://www.ibm.com/cloud/blog/containers-vs-vms>

Containers vs. virtual machines. (2023). <https://www.atlassian.com/microservices/cloud-computing/containers-vs-vms#:~:text=The%20key%20differentiator%20between%20containers,above%20the%20operating%20system%20level>.

Cisco. (2022). What Is Container Orchestration? <https://www.cisco.com/c/en/us/solutions/cloud/what-is-container-orchestration.html>

Chandrayan P. (2021). Docker Image vs Container: The Key Differences <https://www.knowledgehut.com/blog/devops/docker-vs-container>

Schmitt J. (2022). Docker image versus container: What are the differences? <https://circleci.com/blog/docker-image-vs-container/>