|  |  |
| --- | --- |
| Course Name | ITD 2313 – Script Programming |
| Instructor | Michael Schnell |
| Student Name | Monteith Wright |
| Due date | 3/19/2023 |
| Grade | Put grade earned here |
| Grading Comments | Put instructor comments here |

Container Programming

Containers are lightweight packages of your application code together with dependencies such as specific versions of programming language runtimes and libraries required to run your software services. Containers are a stream way to build, test and deploy applications in multiple environments from a developers laptop. The greatest benefit of containers is the great degree of modularity they offer. You can break the entire complex application into several modules and make different containers for each of these modules. This is known as a microservices approach which offers simple & easy manageability.

An advantage to containers is that they are lightweight compared to virtual machines. Also, they can be started in a just in time fashion with out having to wait until bootup. A container is only tens of megabytes in size whereas a virtual machine may be several gigs in size. (Rubens). The benefits of containers are streamlined a way to build. The have less overhead Containers require less system resources than traditional or hardware virtual machine environments because they don’t include operating system images. Applications in containers can be deployed easily on different operating systems and hardware platforms. Development teams know applications on containers will run the same regardless of where they are deployed. The offer greater efficiency containers allow applications to be deployed rapidly, patched, and scaled. They also offer better application development. Containers support agile and DevOps efforts to accelerate development, test, and production cycles.

Docker is a popular runtime environment used to create and build software inside containers. It uses Docker images (copy-on-write snapshots) to deploy containerized applications or software in multiple environments, from development to test and production. Docker was built on open standards and functions inside most common operating environments, including Linux, Microsoft Windows, and other on-premises or cloud-based infrastructures. Kubernetes orchestrates the operation of multiple containers in harmony together. It manages areas like the use of underlying infrastructure resources for containerized applications such as the amount of compute, network, and storage resources required. Orchestration tools like Kubernetes make it easier to automate and scale container-based workloads for live production environments. (https://www.netapp.com/devops-solutions/what-are-containers/)

People sometimes confuse container technology with virtual machines (VMs) or server virtualization technology. Although there are some basic similarities, containers are very different from VMs.

Virtual machines run in a hypervisor environment where each virtual machine must include its own guest operating system inside it, along with its related binaries, libraries, and application files. This consumes a large amount of system resources and overhead, especially when multiple VMs are running on the same physical server, each with its own guest OS.

In contrast, each container shares the same host OS or system kernel and is much lighter in size, often only megabytes. This often means a container might take just seconds to start (versus the gigabytes and minutes required for a typical VM).

Containers are becoming increasingly prominent, especially in cloud environments. Many organizations are even considering containers as a replacement of VMs as the general-purpose compute platform for their applications and workloads. But within that very broad scope, there are key use cases where containers are especially relevant. Containers are small and lightweight which makes them a match for microservices architectures applications are constructed of independently deployable smaller services. (https://www.ibm.com/topics/containers). The combination of microservices as an architecture and containers as a platform is a common foundation for many teams that embrace DevOps as the way they build, ship, and run software.

Containers offer a logical packaging mechanism in which applications can be abstracted from the environment in which they actually run. This decoupling allows container-based applications to be deployed easily and consistently, regardless of whether the target environment is a private data center, the public cloud, or even a developer’s personal laptop. (https://cloud.google.com/learn/what-are-containers).

Some of the benefits of using containers are the separation of responsibility Containers provide a separation of responsibility for developers focus on logic and dependencies. IT teams can focus on deployment and management. That way they don’t have to focus on details like software versions and configurations. Containers virtualize CPU, memory, storage, and network resources at the operating system level, providing developers with a view of the OS logically isolated from other applications. (Michael Hines)

Containerization works well for Linux operating systems. However, while Windows has a container environment, it is not supported nearly as much as the Linux environments. Containers share the kernel of the OS, so if the kernel becomes vulnerable, all the containers will be vulnerable, as well. Containerization is usually used to build multi-layered infrastructure, with one application in one container. This means you must monitor more things than you would if running all your applications on one virtual machine. (Stephen Guyton)

# Works Cited

https://cloud.google.com/learn/what-are-containers. n.d. March 2023.

https://www.ibm.com/topics/containers. 1 January 2023.

https://www.netapp.com/devops-solutions/what-are-containers/. January 2023.

Rubens, Paul. https://www.cio.com/article/247005/what-are-containers-and-why-do-you-need-them.html. 27 June 2017.

Stephen Guyton. *Containers & Containerization – Pros and Cons* . 24 May 2019.

(Michael Hines)