Containerization Workshop Notes

Docker

Docker is a popular open-source platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. Here are some key points about Docker:

- Containers: Docker packages applications and their dependencies in containers to ensure that they run seamlessly in any environment. Containers are lightweight, share the host system's kernel, and start almost instantly.
- Images: Docker images are lightweight, stand-alone, executable software packages that include everything needed to run an application: code, runtime, libraries, environment variables, and config files.
- Dockerfile: A text document that contains all the commands a user could call on the command line to assemble an image. Docker can build images automatically by reading the instructions from a Dockerfile.
- Docker Hub: A cloud-based registry service that allows you to link code repositories, build your images, and test them, store manually pushed images, and link to Docker Cloud so you can deploy images to your hosts.
- Volumes: Docker volumes are used to persist data generated by and used by Docker containers. They are entirely managed by Docker and are the preferred mechanism for persisting data generated by and used by Docker containers.
- Networking: Docker networking allows you to link a container to one or more networks, attach an IP address to it, expose ports, etc., providing isolation and segmentation for containers.

Differences between Docker and Virtual Machines (VMs)

While both Docker and VMs serve the purpose of creating isolated environments for running applications, they operate differently and have distinct characteristics:

- Architecture:
    - Docker: Uses container technology to encapsulate an application and its dependencies into a container that can run on any Linux server. Containers share the host system’s kernel but provide virtual isolation for running applications.
    - VMs: Use hypervisor technology to fully virtualize the hardware stack, including the kernel, so each VM has its own operating system. VMs are fully isolated from the host and can run different operating systems.
- Performance:
    - Docker: Containers start faster and use less memory and CPU resources than VMs because they share the host’s kernel and don’t need to boot an OS. This makes Docker more efficient in terms of system resource usage.
    - VMs: VMs take longer to start because they need to boot the guest operating system. They also consume more system resources because each VM runs a full copy of an operating system and the application.

- Portability:
    - Docker: Docker containers are highly portable. Once a Docker container is created, it can run on any Linux machine that has Docker installed, regardless of the underlying infrastructure.
    - VMs: VMs are less portable compared to Docker containers. While VMs can be moved between different hosts, the process is more complex and time-consuming.
- Use Cases:
    - Docker: Ideal for microservices architecture, development and testing environments, and applications where scalability and fast deployment are required.
    - VMs: Better suited for applications requiring full isolation for security purposes, running applications that require different operating systems on the same host, and legacy applications that rely on specific OS environments.
- Storage and Networking:
    - Docker: Offers volumes and networks that can be configured to provide persistent storage and network isolation across containers.
    - VMs: Storage is typically attached as virtual disk files, and networking capabilities are provided through virtual switches and adapters, offering a wide range of configurations and isolation levels.