CONTAINERS AND DOCKER

Def. Container: is a OS-level lightweight process virtualization technology to deliver software

- Virtual machine: each VM includes the application, the necessary binaries and libraries and the entire guest Operating System, all of which may be tens of GBs in size.
- Container: it include the application and all of its dependencies, but share the kernel with other containers. They run as an isolated process in user-space on the host Operating System. They're also not tied to any specific infrastructure. Containers running on a single machine all share the same OS kernel so they start instantly and make more efficient use of RAM. Images are constructed from layered filesystems so they can share common files, making disk usage and image download much more efficient.

Containers have similar resource isolation and allocation benefits as VMs but different architectural approach allows them to be much more portable and efficient.

LXC vs Docker: Both LXC and Docker are userland container managers that use kernel namespaces to provide end user containers. We also now have Systemd-Nspawn that does the same thing. The only difference is LXC containers have an an init and can thus run multiple processes and Docker containers do not have an init and can only run single processes.

Docker advantages: lightweight, portability and predictability

Docker architecture: it uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running and distributing your Docker containers. Both Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. Docker client and daemon communicate via sockets or through a RESTful API.

Docker compose: Compose is a tool for defining and running multi-container Docker applications. With Compose you use a Compose file to configure your application's services. Then using a single command, you create and start all the services from your configuration.