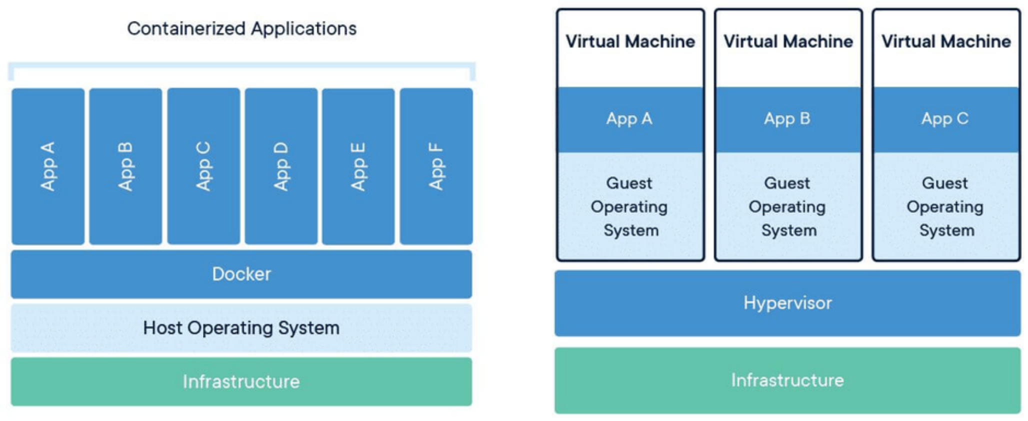
Docker

Docker:

- Have you ever have a problem of your application works on your machine and not on another machine. Virtual Machine solve this problem by replicating your machine.
- But each virtual machine is really slow because it has built-in OS in it. But docker uses the container to isolate everything at OS level so that all of our container can share an OS and be much faster and disposable.
- Using docker file it is easy to spin-up container fast and you can find huge ecosystem of existing services and applications.
- **Use docker if you want your application to be scalable, reliable and performance.**
- **Docker is just a way to package software, so it can run on any hardware.** The important factors of docker are:

1. Docker file.
2. Image.
3. Container.

- **A docker file is a blue print to build a docker image.**
- **A docker image is a template for runing docker container.**
- **A container is just a runing process.**
- For example, if we have a node application, we need a server that’s runing the same version of node and also install the dependencies.
- It will works on my machine but it might break on another machine, the whole point of docker is to solve the problem like this by reproducing environment.
- **The developer who created the software can define the environment with a docker file. So that any developer can use the docker file to rebuild the environment. Which is saved as the immutable snapshot known as the image.**
- Images can be uploaded to the cloud in both public and private registering them any developers that wants to run that software can pull the images down to create a container which is just runing the process of that image.
- In other words one image file can be used to spawn the same process multiple times in multiple places.



What are containers:

- The industry standard today is to use Virtual Machines (VMs) to run software applications. VMs run applications inside a guest Operating System, which runs on virtual hardware powered by the server’s host OS.
- VMs are great at providing full process isolation for applications: there are very few ways a problem in the host operating system can affect the software running in the guest operating system, and vice-versa. But this isolation comes at great cost — the computational overhead spent virtualizing hardware for a guest OS to use is substantial.
- Containers take a different approach: by leveraging the low-level mechanics of the host operating system, containers provide most of the isolation of virtual machines at a fraction of the computing power.

Why use containers:

- 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. This gives developers the ability to create predictable environments that are isolated from the rest of the applications and can be run anywhere.
- From an operations standpoint, apart from portability containers also give more granular control over resources giving your infrastructure improved efficiency which can result in better utilization of your compute resources.

The benefits of Docker in building and deploying applications are many:

- Caching a cluster of containers.
- Flexible resource sharing.
- Scalability - many containers can be placed in a single host.
- Running your service on hardware that is much cheaper than standard servers.
- Fast deployment, ease of creating new instances, and faster migrations.
- Ease of moving and maintaining your applications.
- Better security, less access needed to work with the code running inside containers, and fewer software dependencies.