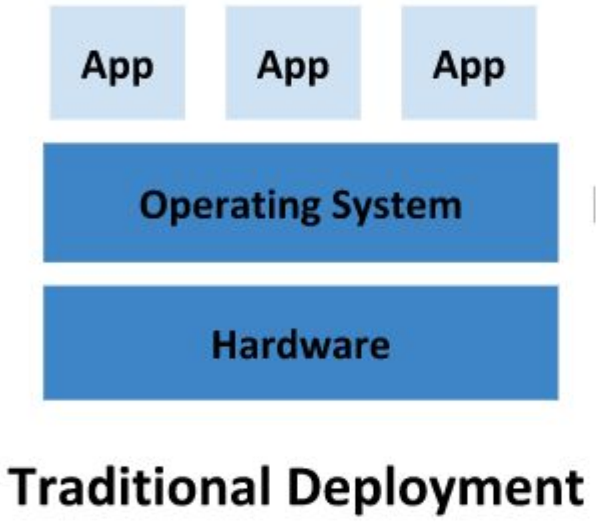
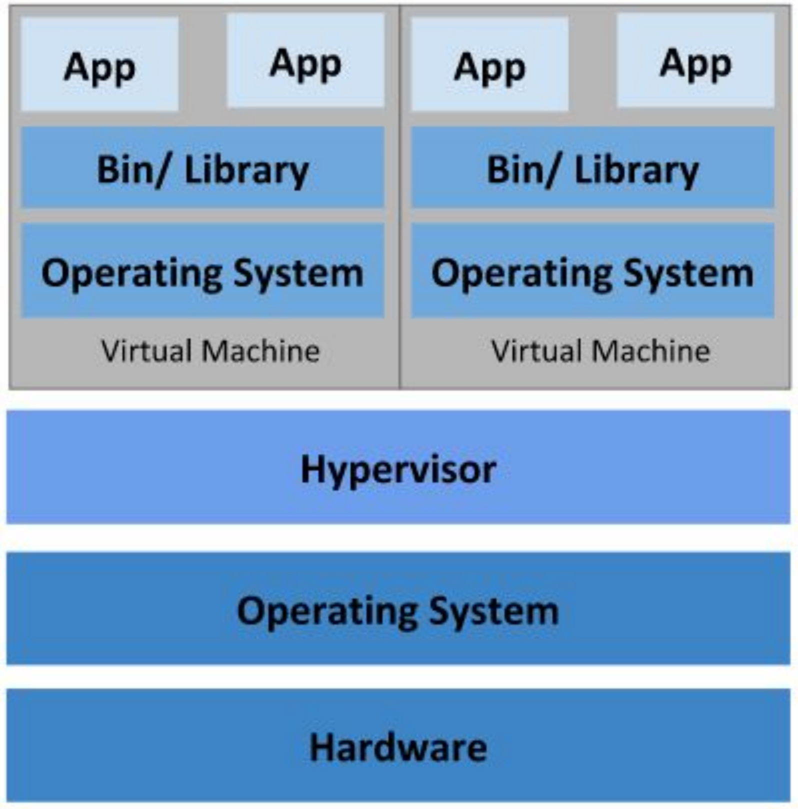
Physical Server



Drawbacks:

- CAPEx Costs
- Operational Costs
- Overpowered and massively overpriced
- if multiple applications run on a physical server, there can be instances where one application would take up most of the resources, and as a result, the other applications would underperform.
- solution for this would be to run each application on a different physical server. But this did not scale as resources were underutilized



Virtualized Deployment

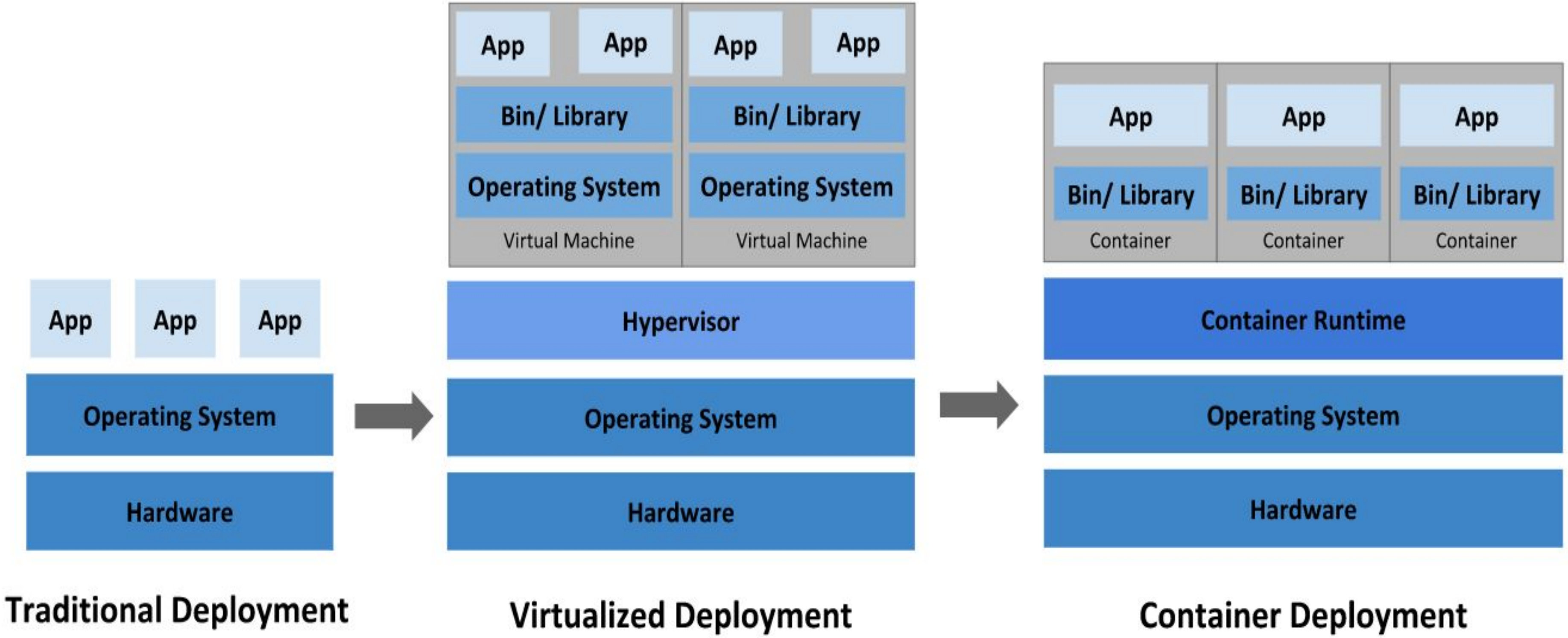
Virtual Machines

Benefits:

- Multiple VMs on Single Machine
- Consolidate apps into single physical machine
- Cost savings
- Faster server provisioning

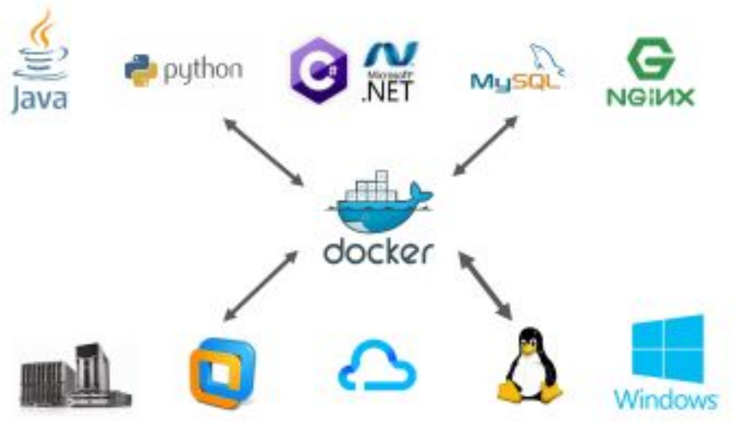
Drawbacks:

- Requires compute and storage
- OS Licenses
- CAPEx costs


App
App
App
Operating System
Hardware
Traditional Deployment
App
App
App
App
Bin/ Library
Bin/ Library
Operating System
Operating System
Virtual Machine
Virtual Machine
Hypervisor
Operating System
Hardware
Virtualized Deployment
App
App
App
Bin/ Library
Bin/ Library
Bin/ Library
Container
Container
Container
Container Runtime
Operating System
Hardware
Container Deployment

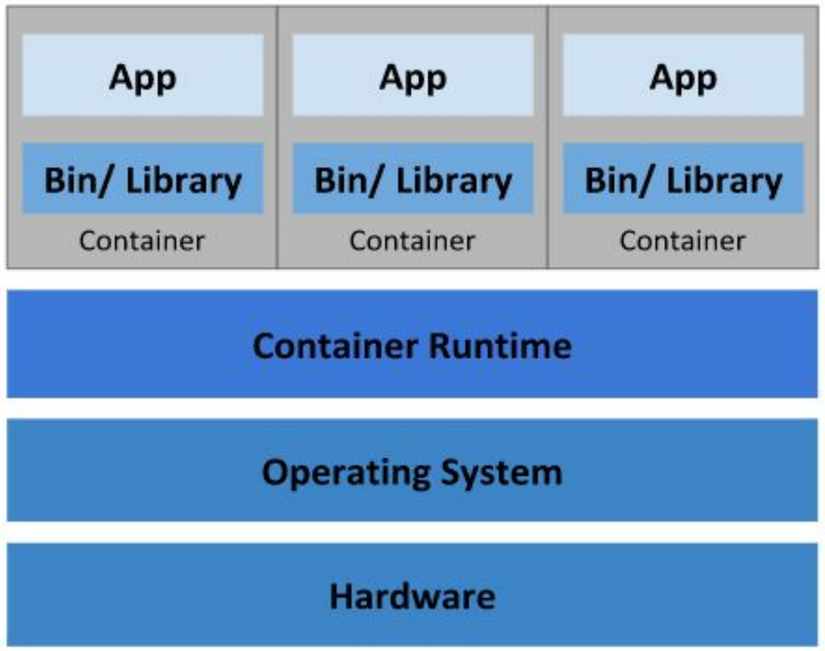
Containers in general

What are Containers?



Do you know Google runs its Gmail, YouTube, Search and most of its application inside containers?

Google runs average of 2 Billion Containers every week.



Container Deployment

Containers

- Containers are similar to VMs, but they have relaxed isolation properties to share the Operating System (OS) among the applications.
- a container has its own filesystem, CPU, memory, process space, and more. As they are decoupled from the underlying infrastructure, they are portable across clouds and OS distributions.
- Containers become very light weight -They don't need separate os.
- It boots up in matter of seconds
- Takes fraction of disk and memory space -Unlike vm where each virtual machine requires separate os which min requires approx 1gb.

Container Advantages

1. Portability : Cloud and OS distribution portability: Runs on Ubuntu, RHEL, CoreOS, on-premises, on major public clouds, and anywhere else.
2. Consistency: Environmental consistency across development, testing, and production: Runs the same on a laptop as it does in the cloud.
3. Resource isolation: predictable application performance because all containers will have their own memory and cpu space..
4. Resource utilization: high efficiency and density.
5. Speed of app development :Agile application creation and deployment: increased ease and efficiency of container image creation ,replication, destroyment compared to VM image use.
6. Scaling: Scaling up of a identical containers with in a cluster is very easy based on real time traffic.

Google runs average of 2 Billion Containers every week

- How are these containers created and managed at such large scale?
- How do all these containers connect and communicate?
- How do you scale these containers as per the traffic demands?

Container Orchestration Engine

Container Orchestration Engine automates deploying, scaling, and managing containerized applications on a group of servers.

1. Containers are a good way to bundle and run your applications. In a production environment, you need to manage the containers that run the applications and ensure that there is no downtime.
2. If a container goes down, another container needs to start. It wil be easier if this behavior was handled by a system or Tool.

Kubernetes

- **Service discovery and load balancing** Kubernetes can expose a container using the DNS name or using their own IP address. If traffic to a container is high, Kubernetes is able to load balance and distribute the network traffic so that the deployment is stable.
- **Storage orchestration** Kubernetes allows you to automatically mount a storage system of your choice, such as local storages, public cloud providers, and more.
- **Automated rollouts and rollbacks** You can describe the desired state for your deployed containers using Kubernetes, and it can change the actual state to the desired state at a controlled rate. For example, you can automate Kubernetes to create new containers for your deployment, remove existing containers and adopt all their resources to the new container.
- **Automatic bin packing** You provide Kubernetes with a cluster of nodes that it can use to run containerized tasks. You tell Kubernetes how much CPU and memory (RAM) each container needs. Kubernetes can fit containers onto your nodes to make the best use of your resources.
- **Self-healing** Kubernetes restarts containers that fail, replaces containers, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.
- **Secret and configuration management** Kubernetes lets you store and manage sensitive information, such as passwords, OAuth tokens, and SSH keys. You can deploy and update secrets and application configuration without rebuilding your container images, and without exposing secrets in your stack configuration.
- **Clustering, Scheduling, Scalability.**