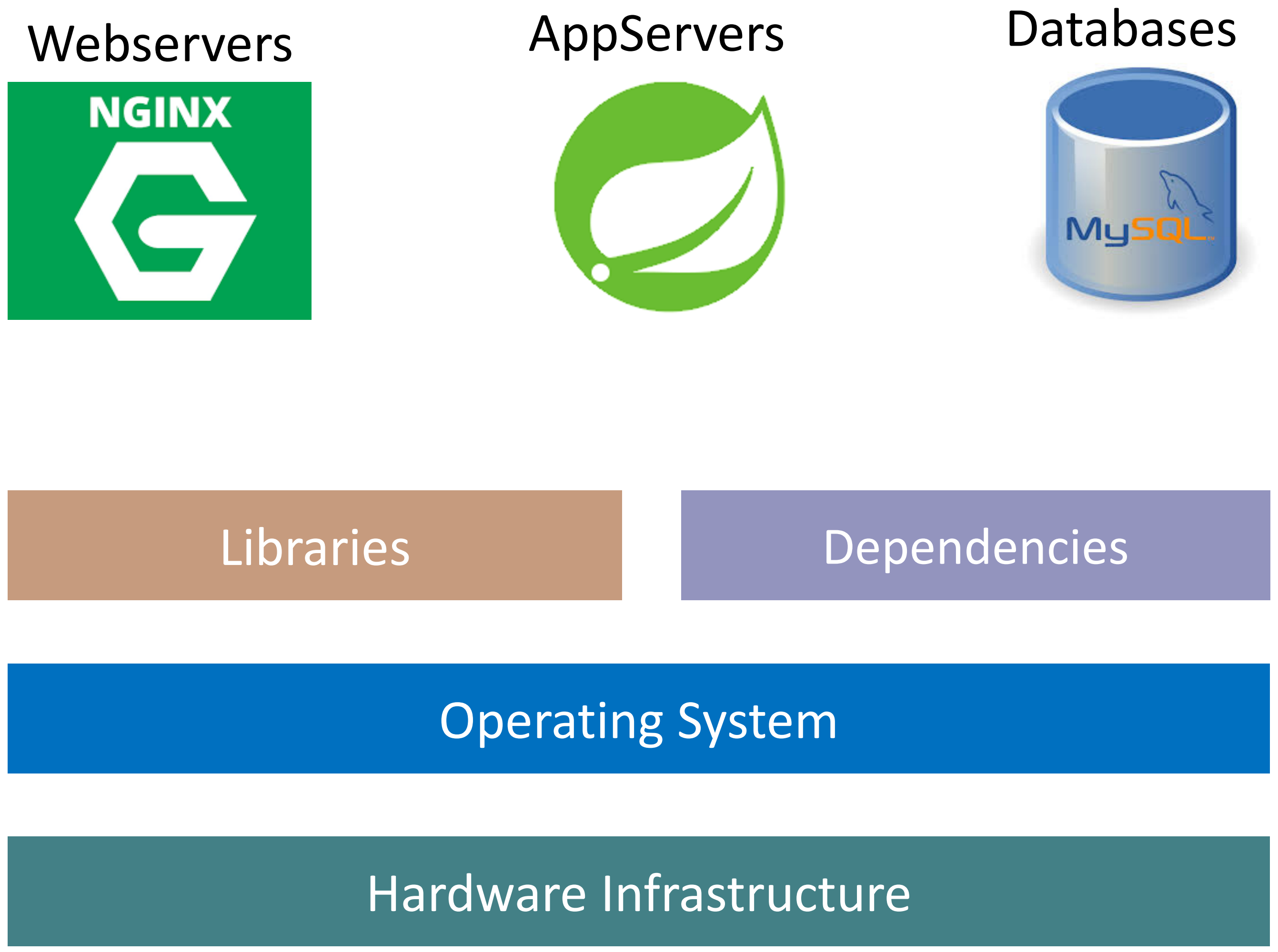
Docker Fundamentals

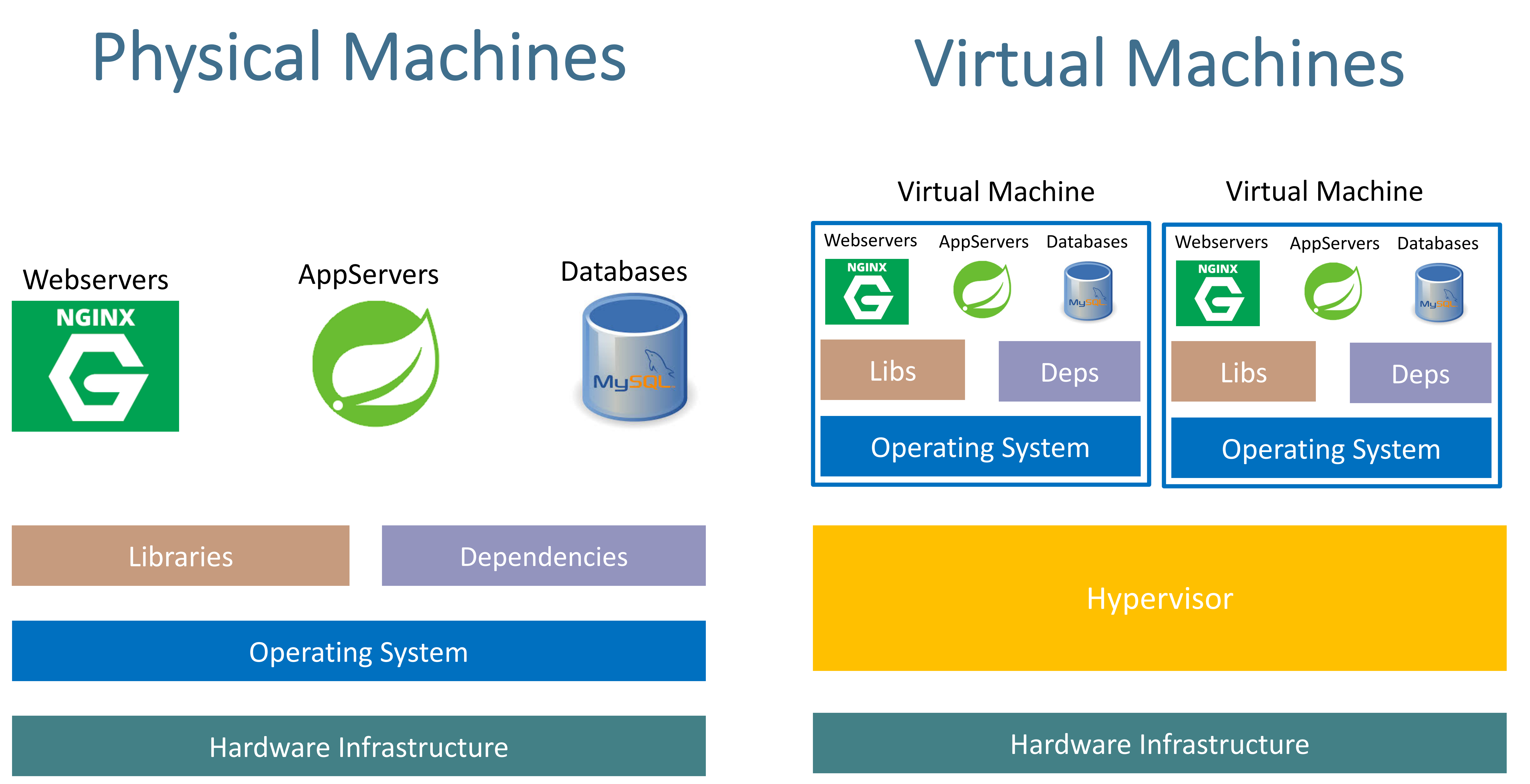
Kalyan Reddy Daida

Docker Introduction

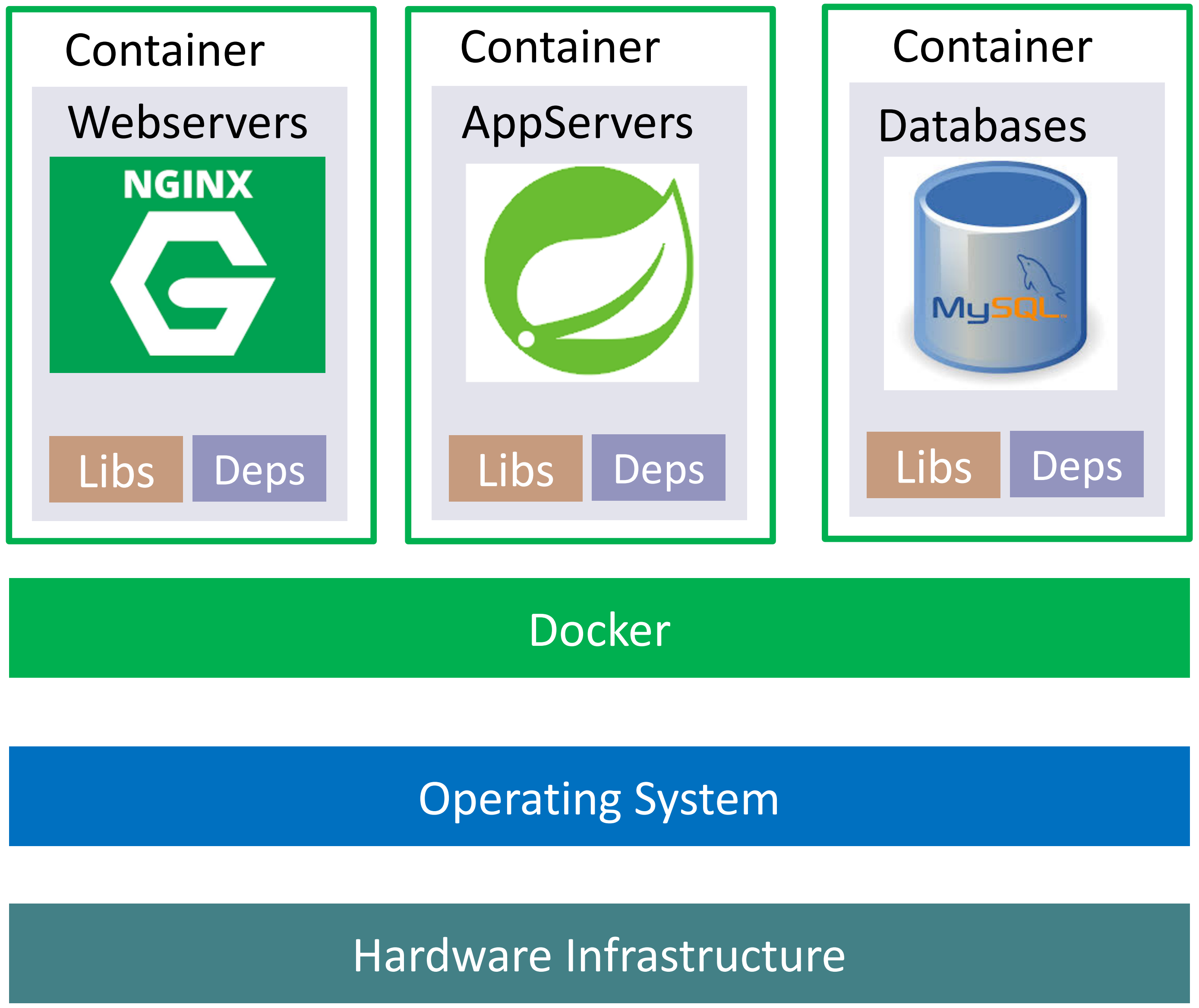
What problems we have with Traditional Infra?

- Traditional Approach
- Installation & Configuration
  - Time consuming
  - Need to perform install/configs on every server and every environment (dev, qa, staging, production)
- Compatibility & Dependency
  - Need to keep resolving issues related to libraries and dependencies
- Inconsistencies across Environments
  - Very hard to track changes across Dev/QA/Staging and Prod environments and they end up with inconsistencies
- Operational Support
  - Need more resources to handle operational issues on day to day basis
    - Server Support (hardware, software)
    - Patching releases
- Developer Environments
  - When a new developer joins the team, time it takes to provision his development environment in traditional approach is time taking.

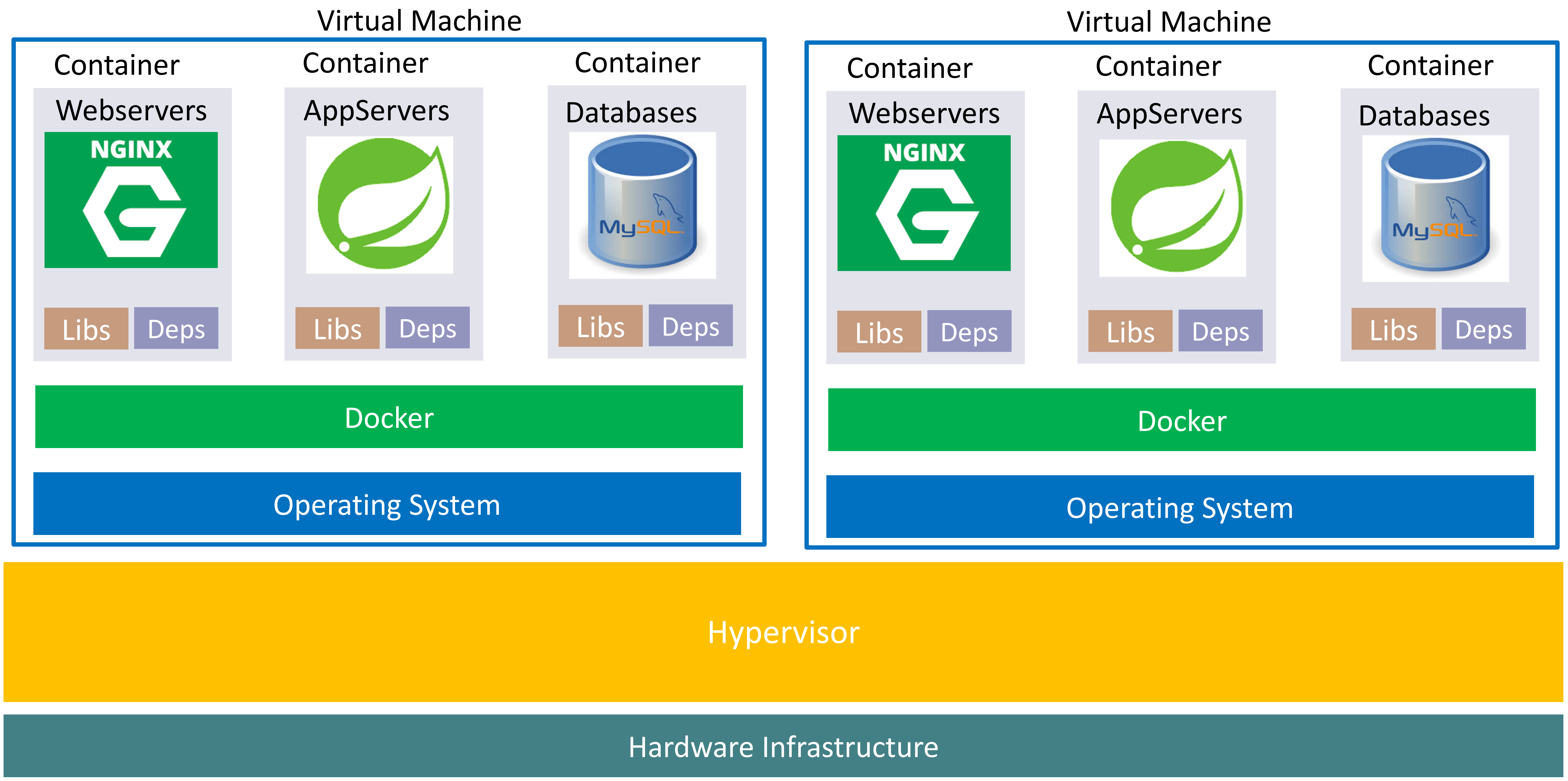



Physical Machines
Virtual Machines
Webservers
AppServers
Databases
NGINX
MySQL
Libraries
Dependencies
Operating System
Hardware Infrastructure
Virtual Machine
Virtual Machine
Webservers
AppServers
Databases
NGINX
MySQL
Libs
Deps
Operating System
Webservers
AppServers
Databases
NGINX
MySQL
Libs
Deps
Operating System
Hypervisor
Hardware Infrastructure

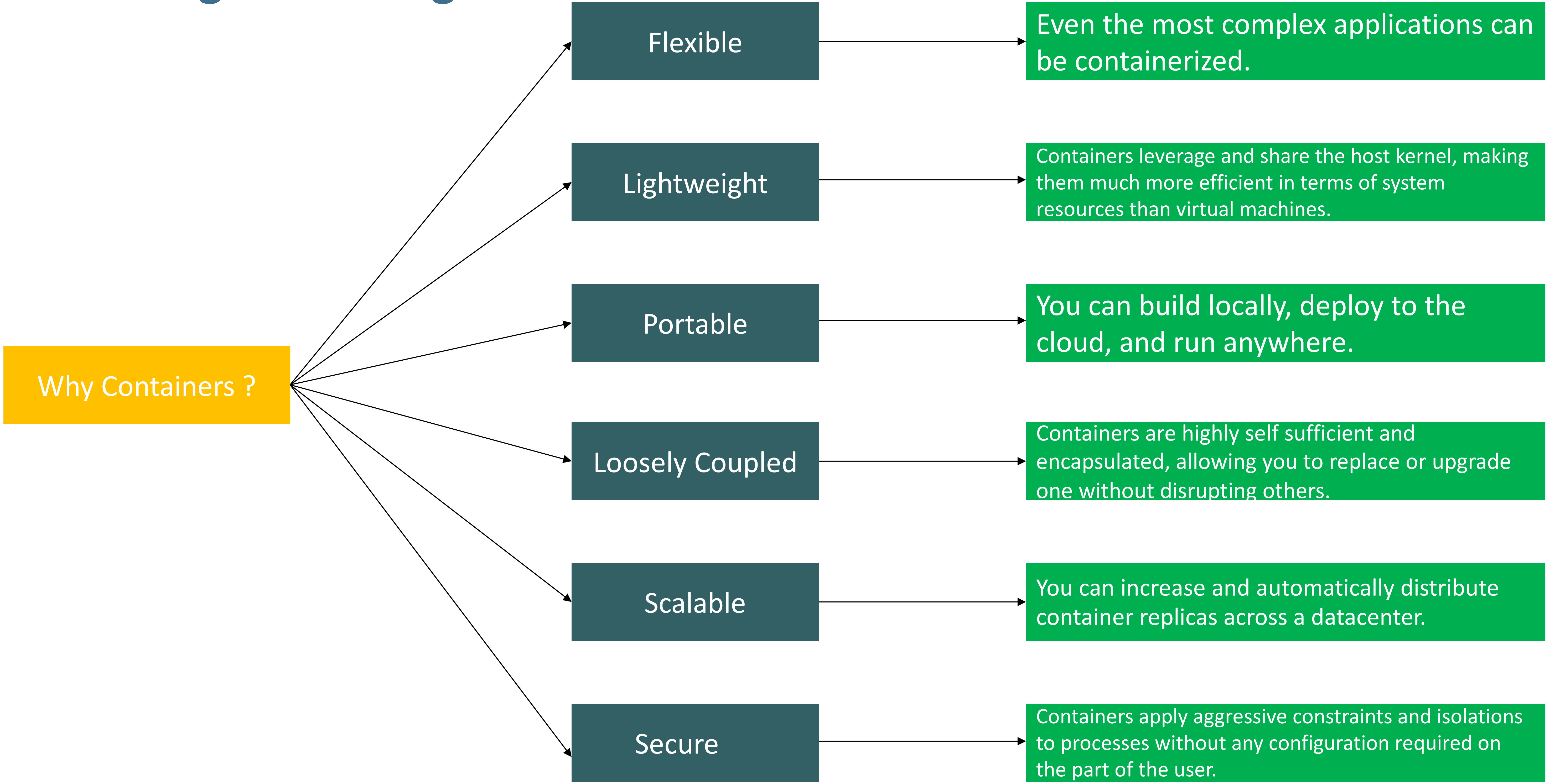
Physical Machines with Docker



Virtual Machines with Docker



Advantages of using Docker



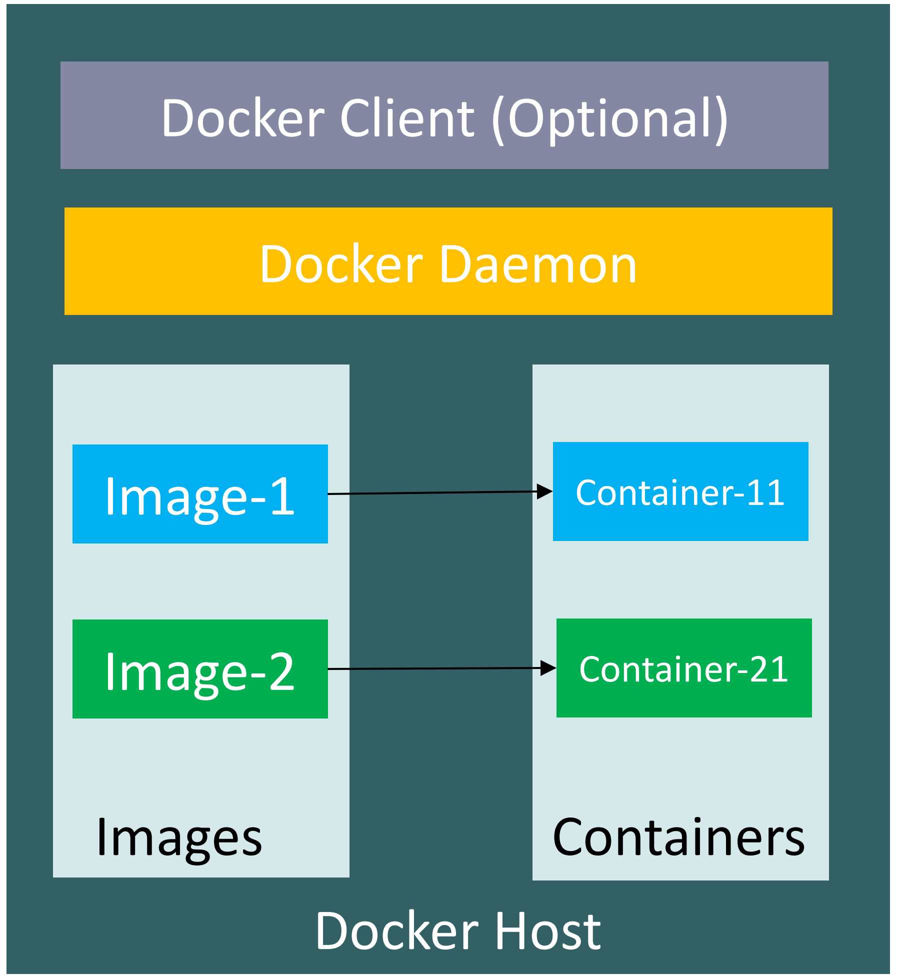
Why Containers ?

- Flexible → Even the most complex applications can be containerized.
- Lightweight → Containers leverage and share the host kernel, making them much more efficient in terms of system resources than virtual machines.
- Portable → You can build locally, deploy to the cloud, and run anywhere.
- Loosely Coupled → Containers are highly self sufficient and encapsulated, allowing you to replace or upgrade one without disrupting others.
- Scalable → You can increase and automatically distribute container replicas across a datacenter.
- Secure → Containers apply aggressive constraints and isolations to processes without any configuration required on the part of the user.

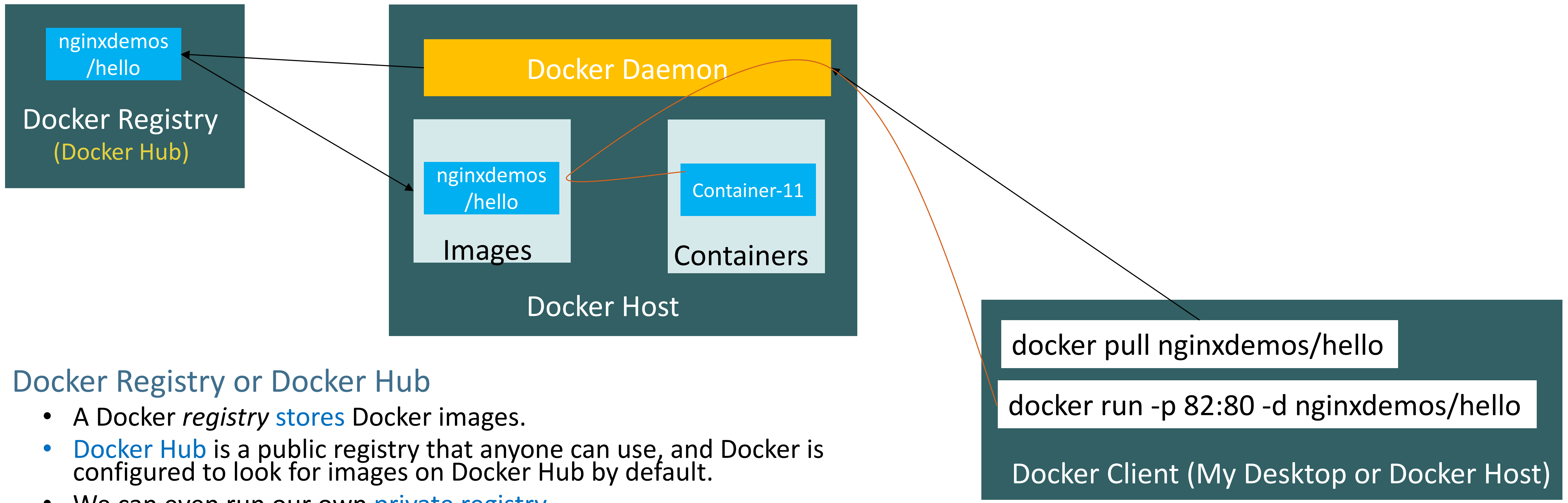
Docker Architecture

Docker - Terminology

- Docker Daemon
  - The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes.
- Docker Client
  - Docker client can be present on either Docker Host or any other machine.
  - The Docker client (docker) is the primary way that many Docker users interact with Docker.
  - When you use commands such as docker run, the client sends these commands to dockerd (Docker Daemon), which carries them out.
  - The docker command uses the Docker API.
  - The Docker client can communicate with more than one daemon.
- Docker Images
  - An *image* is a read-only template with instructions for creating a Docker container.
  - Often, an image is *based on* another image, with some additional customization.
  - For example, we may build an image which is based on the ubuntu image, but installs the Apache web server and our application, as well as the configuration details needed to make our application run.
- Docker Containers
  - A container is a runnable instance of an image.
  - We can create, start, stop, move, or delete a container using the Docker API or CLI.
  - We can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.
  - When a container is removed, any changes to its state that are not stored in persistent storage disappear.



Docker - Terminology



```
docker pull nginxdemos/hello
docker run -p 82:80 -d nginxdemos/hello
```

- Docker Registry or Docker Hub
  - A Docker *registry* stores Docker images.
  - Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default.
  - We can even run our own private registry.
  - When we use the docker pull or docker run commands, the required images are pulled from our configured registry.
  - When we use the docker push command, our image is pushed to our configured registry.

Thank You