# TRAINING CONTENT

**Cloud Computing**

YOUR NEXT DESTINATION

OF SOFTWARE OUTSOURCING

# Lecture Outline

- Docker
- Container
- Containers vs Virtual Machines
- Docker Architecture Diagram
- Docker Installation
- Dockerfile
- Push Image into Registry
- Docker Networking
- Docker Compose
- Docker Compose Example
- Kubernetes Overview

# Docker Overview(1\4)

What is Docker?

Docker is a software platform that allows you to build, test, and deploy applications quickly, packaging software into standardized units called containers.

# Docker Overview (2\4)

Why Docker?

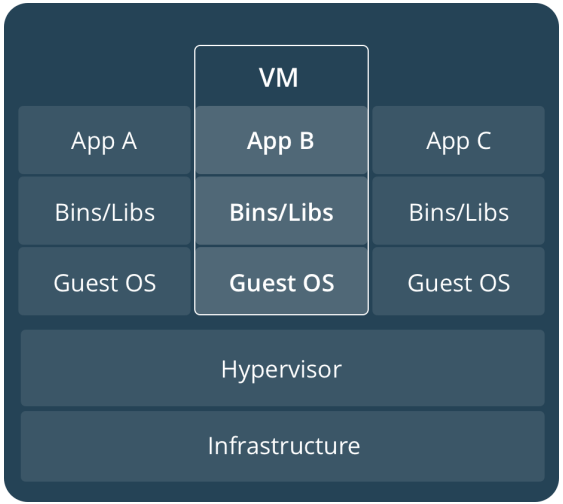
- Isolation
- Lightweight
- Simplicity
- Community

# Docker Overview (3\4)

What is a container?

Containers provide a way of creating an isolated environment, sometimes called a sandbox, in which applications and their dependencies can live.

- Container ≠ VM
- Isolated
- Share OS
- and sometimes bins/libs



# Docker Overview (4\4)

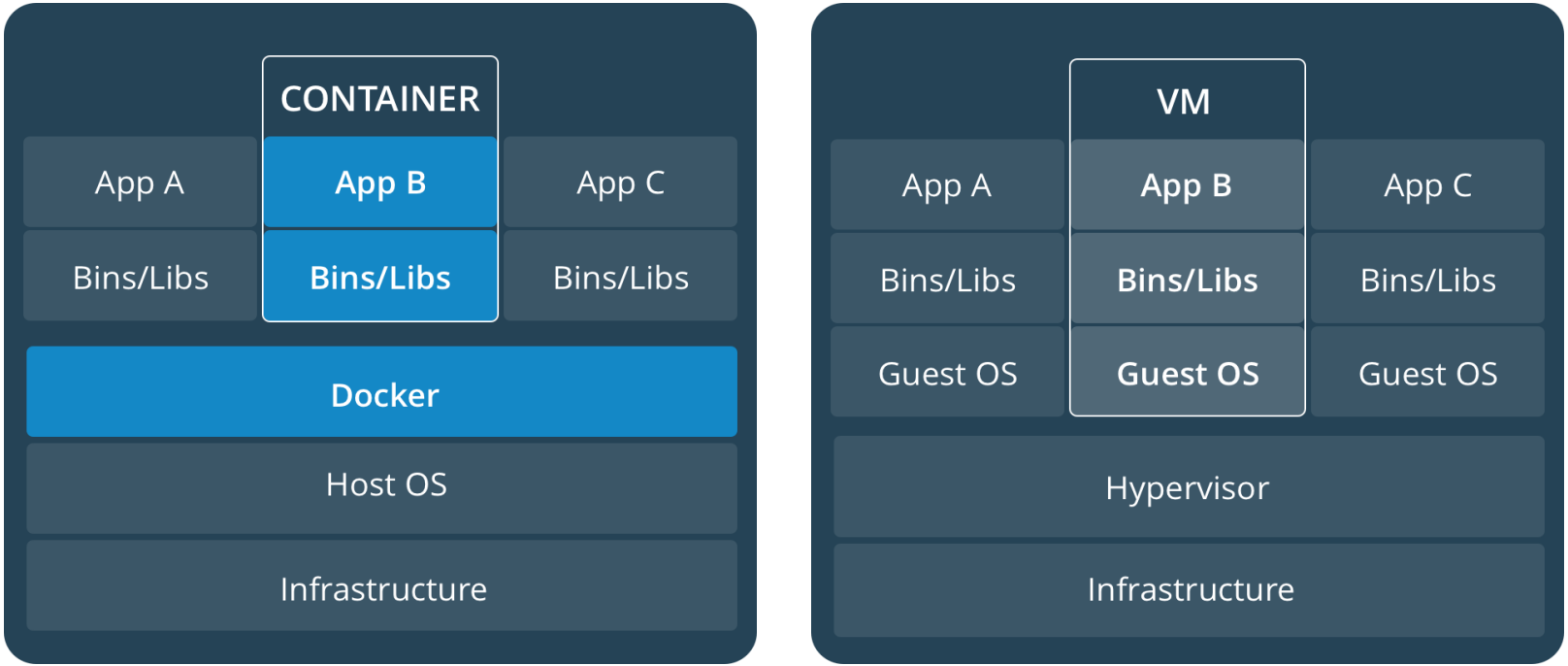
**Why Container?**

**Portability -** the isolated environment that containers provide effectively means the container is decoupled from the environment in which they run. Basically, they don't care much about the environment in which they run, which means they can be run in many different environments with different operating systems and hardware platforms.

**Consistency** – since the containers are decoupled from the environment in which they run, you can be sure that they operate the same, regardless of where they are deployed. The isolated environment that they provide is the same across different deployment environments.

**Speed to deploy** – for the same reasons as above. There is no need for considerations around how the application will operate in a production environment. If it runs in a container in one environment (say, your local machine), then it can be made to run in a container in another environment (say, in a cloud provider) very quickly

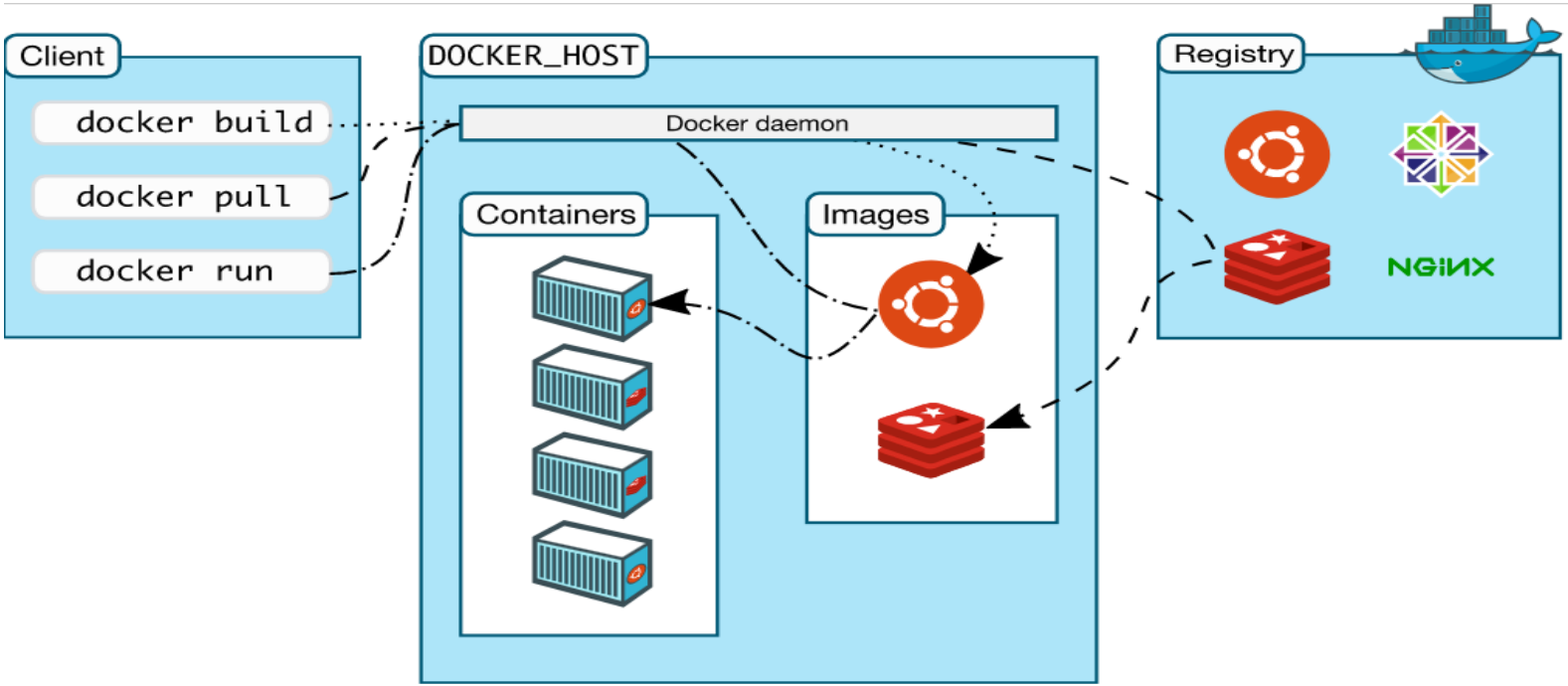
# Containers vs Virtual Machines



# Docker terminology(1/1)

- **Images:** The blueprints of our application which form the basis of containers. These contain all of the configuration settings that define the isolated environment.
- **Containers:** Are instances of a Docker image and are what run the actual application.
- **Docker Daemon:** That background service running on the host that listens to API calls (via the Docker client), manages images and building, running and distributing containers. The Daemon is the process that runs in the operating system which the client talks to – playing the role of the broker.
- **Docker Client:** The command line tool that allows the user to interact with the daemon. There are other forms of clients too.
- **Docker Hub:** A registry of Docker images containing all available Docker images. A user can have their own registry, from which they can pull images.

# Docker Architecture Diagram



# Understanding the Docker components

- Docker client and server.
- Docker image.
- Docker registry.
- Docker container.

# Thank You

Get in touch with us:

www.bjitgroup.com