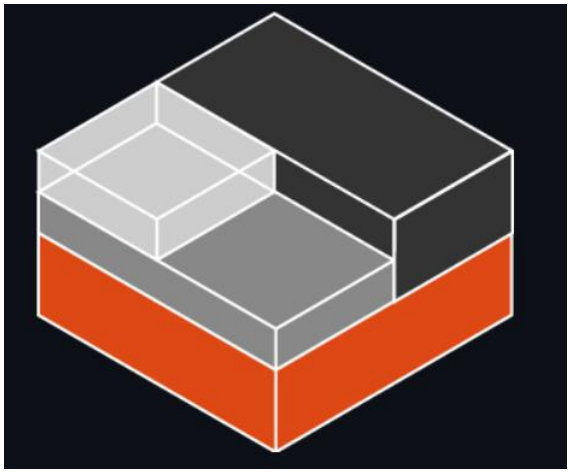
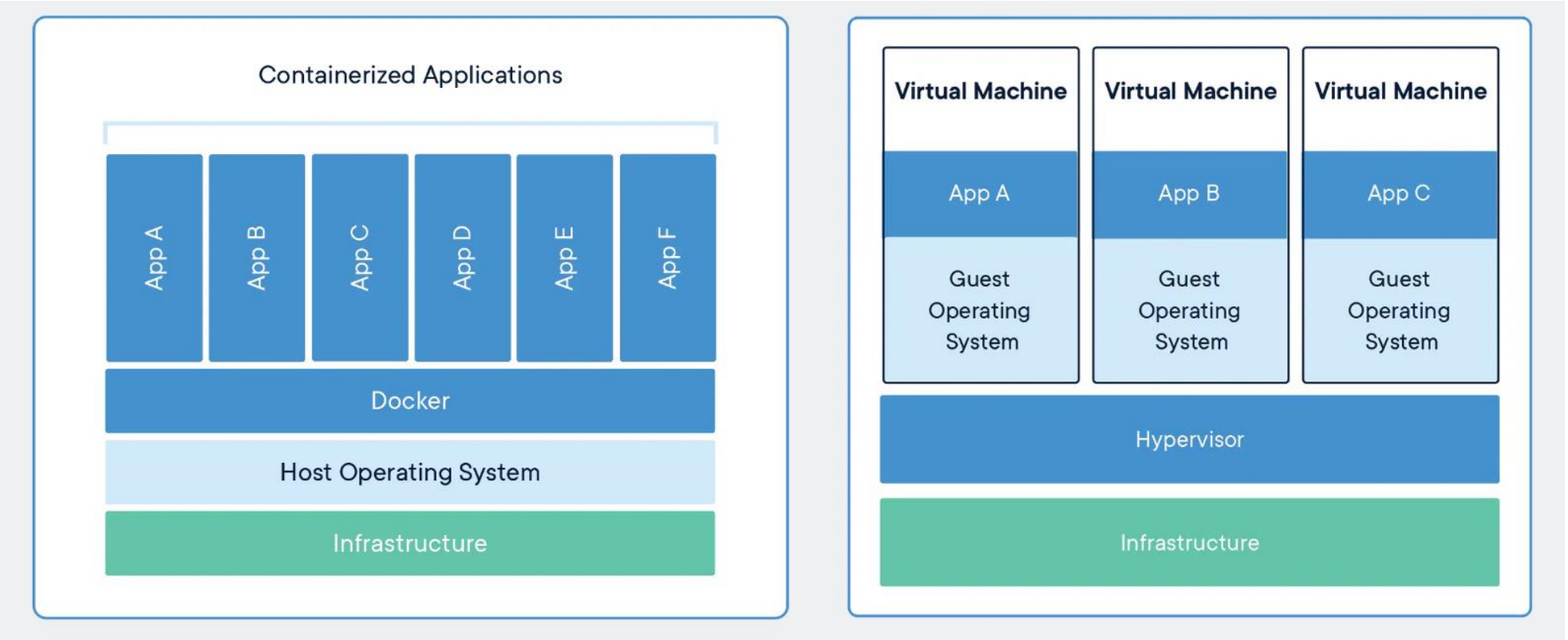


An introduction to containers

Containers

- Containers are an abstraction at the app layer.
- E.g.: Docker, Linux Containers (LXC)



Why Containers?

- Less overhead

- Containers require less system resources than traditional or hardware virtual machine environments because they don't include operating system images.

- Increased portability

- Containers can run virtually anywhere, on Linux, Windows, and Mac operating systems; on virtual machines or on physical servers; on a developer's machine or in data centers on-premises; and of course, in the public cloud.

- Greater efficiency

-allow you to use just the computing resources you need. This lets you run your applications efficiently. More rapidly deployed, patched, or scaled.

- Better application development

- Containers support agile and DevOps efforts to accelerate development, test, and production cycles. More secure.

Introduction to Docker

- Docker is an open platform for developing, shipping, and running applications.
- Docker detach applications from their underlying infrastructure so one can deliver software quickly.
- Docker Image - is a read-only **template** with instructions for creating a Docker container
- A Docker container is a runnable instance of an image.
- Docker Hub
  - DOCKER OFFICIAL IMAGE

Why Docker

- Community

- Docker Hub

- Isolation

-virtualize CPU, memory, storage, and network resources at the operating system level, providing developers with a view of the OS logically isolated from other applications. Library separation.

- Lightweight

-share the machine's OS system kernel and therefore do not require an OS per application, driving higher server efficiencies and reducing server and licensing costs

- Simplicity

- Docker's friendly, CLI-based workflow makes building, sharing, and running containerized applications accessible to developers of all skill levels.

-

Cont'd

- Workflow

- Write the code.
- Build a container image.
- Push the image to the server or Docker Hub.
- Start the application, with the new image.
- Revise the (if necessary) and rerun the above workflow

Docker Commands

```
#List docker images

docker image ls

#Docker image search

docker search <image name>

#Download Docker image

docker pull <image name>

#List docker contaniers that are currently running

docker container ls

#Run a docker image

docker run -d --name <name> -p <port:port> -d <image name>

#Stop a docker container

docker stop <container name/ID>
```

Dockerfile

- Used to setup a Docker image
- A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.
- Dockerfile format

```
# Comment
INSTRUCTION arguments
```

- The instruction is not case-sensitive. However, convention is for them to be UPPERCASE to distinguish them from arguments more easily.

Cont'd

- Generally, a Dockerfile must begin with a FROM instruction.
- Commonly used instructions with formats
  - FROM <parent Docker image name>
  - RUN <command>
  - CMD <command>

  The main purpose of a **CMD** is to provide defaults for an executing container.

  RUN actually runs a command and commits the result; **CMD** does not execute anything at build time, but specifies the intended command for the image.

  - COPY <src>... <dest>
  - EXPOSE <port> [<port>/<protocol>...]
  - VOLUME <["/data"]>

  The **VOLUME** instruction creates a mount point with the specified name and marks it as holding externally mounted volumes.

Dockerfile Example

- Simple Dockerfile content

```
FROM php:8.0-apache
COPY index.php /var/www/html/
EXPOSE 80
CMD apachectl -D FOREGROUND
```

Docker image commands

```
#docker build image

docker build . -t <docker hub username>/<respository name>:v1

#share docker image

docker login -u <docker hub username>

docker push <docker hub username>/<respository name>:v1
docker logout
```

Docker Compose

- Compose is a tool for defining and running multi-container Docker applications.
- With Compose, you use a YAML file to configure your application's services.
- Then, with a single command, you create and start all the services from your configuration.
- Can install as a plugin

Introduction to LXC

- LXC - Linux Containers.
- LXC **command** is a client interface for the Linux kernel containment features.
- Here, LXD is the lightweight container hypervisor.
- Through a powerful API and simple tools, it lets Linux users easily create and manage containers.
- The goal of LXC is to create an environment as close as possible to a standard Linux installation but without the need for a separate operating system.

LXC Features

- LXC can be used in two distinct ways.
  - **Privileged** - by running the **lxc** commands as the root user.
  - **Unprivileged** - by running the **lxc** commands as a non-root user.

- Templates

-Creating a container generally involves creating a root filesystem for the container. **lxc-create** delegates this work to templates, which are generally per-distribution.

- Autostart

- LXC supports marking containers to be started at system boot

- Apparmor

-LXC ships with a default Apparmor profile - to protect from accidental misuses of privileges.

Cont'd

- Snapshot /Cloning

- For rapid provisioning, you may wish to customize a container according to your needs and then make multiple copies of it.
- This can be done with the **lxc-clone** program.
- Clones are either snapshots or copies of another container.
- To more easily support the use of snapshot clones for iterative container development, LXC supports snapshots.
- When working on a container C1, before making a potentially dangerous or hard-to-revert change, you can create a snapshot

LXC Components

- The liblxc library
- Several language bindings for the API:
  - python3
  - lua
  - Go
  - ruby
  - Haskell
- A set of standard tools to control the containers
- Distribution container templates

LXC Commands

```
#Before running lxc for the first time, have to initiate it.

lxd init

#Check remote repositories

lxc remote list

#Check local repositories

lxc image list

#Check remote images.

lxc image list images:
lxc image list images:ubuntu

#Create a Ubuntu Container.

lxc launch ubuntu:22.04 test-ct
```

Cont'd

```
#List LXC

lxc list

#Login to container

lxc exec test-ct bash

#Delete a container

lxc stop test-ct
lxc delete --force test-ct

#Make a snapshot

lxc snapshot test-ct test-ct1
```

Need Help ?

- Support channels
  - Docker Forums
  - Stack Overflow
  - GitHub

Thank You