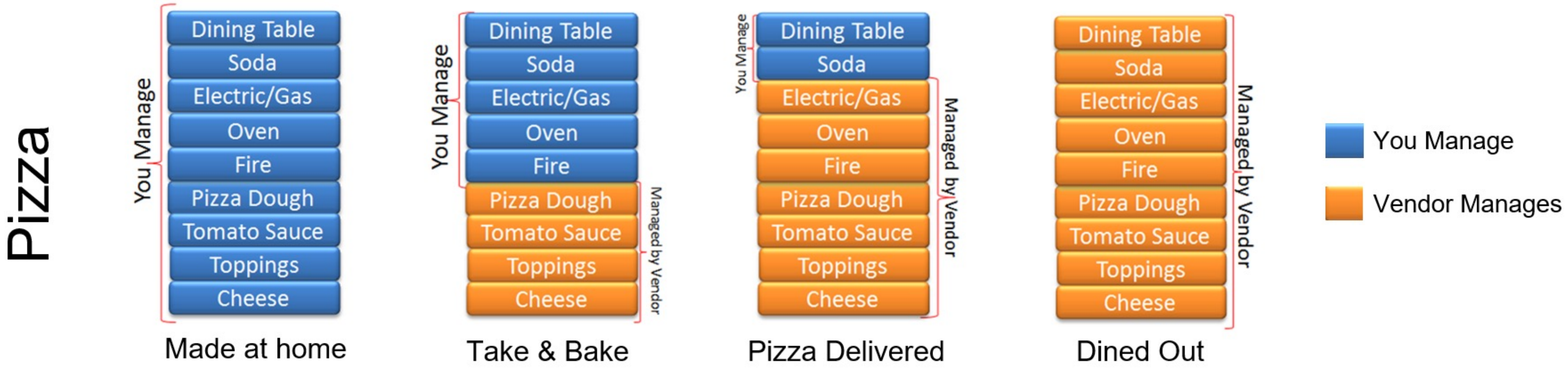
DISTRIBUTED AND CLOUD COMPUTING

LAB11 INTRODUCTION TO DOCKER

SAAS VS PAAS VS IAAS VS ON-PREMISE

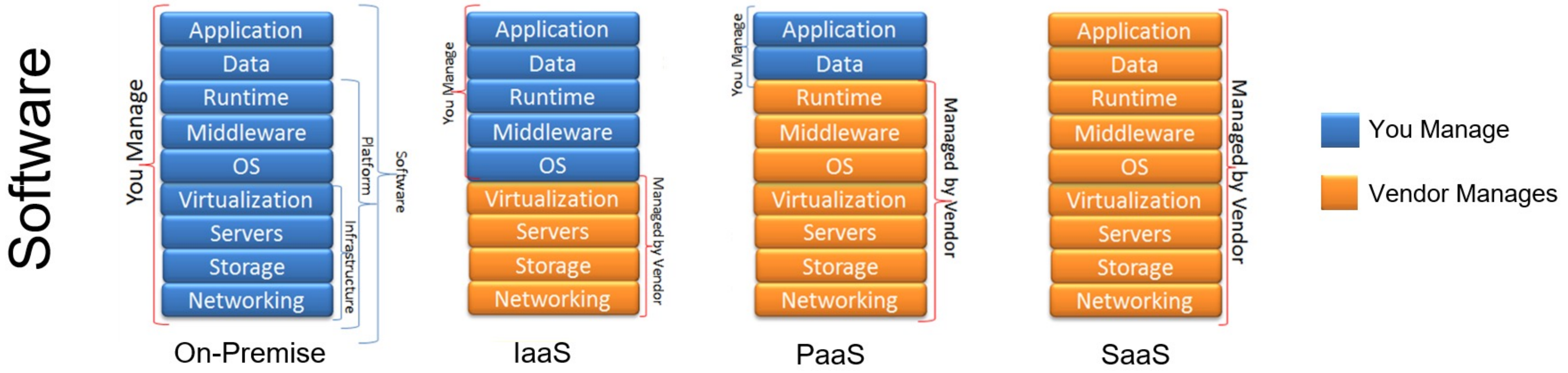
- SaaS: OneDrive, Zoom, Tencent Docs (腾讯文档) – Provide software binary and cloud service
- PaaS: Google App Engine – Provide OS and runtime
- IaaS: Amazon ECS, Google Compute Engine – Provide machine, with storage, networking, etc.
- On-Premise: Deploy everything in private machines – Manage everything

PIZZA-AS-A-SERVICE



https://blogs.sap.com/2016/06/03/its-good-to-have-your-head-in-the-clouds-part-3/

PIZZA-AS-A-SERVICE



https://blogs.sap.com/2016/06/03/its-good-to-have-your-head-in-the-clouds-part-3/

WHY VIRTUALIZATION?

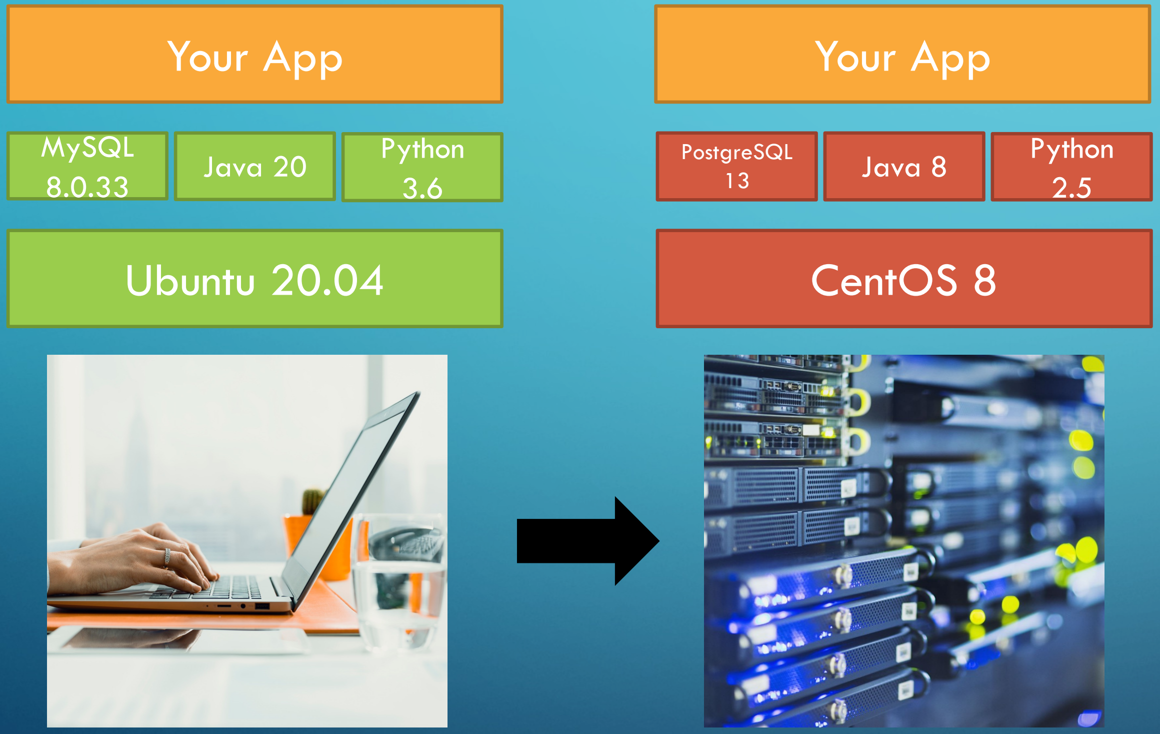
Your App

MySQL 8.0.33 | Java 20 | Python 3.6

Ubuntu 20.04

Development Phase

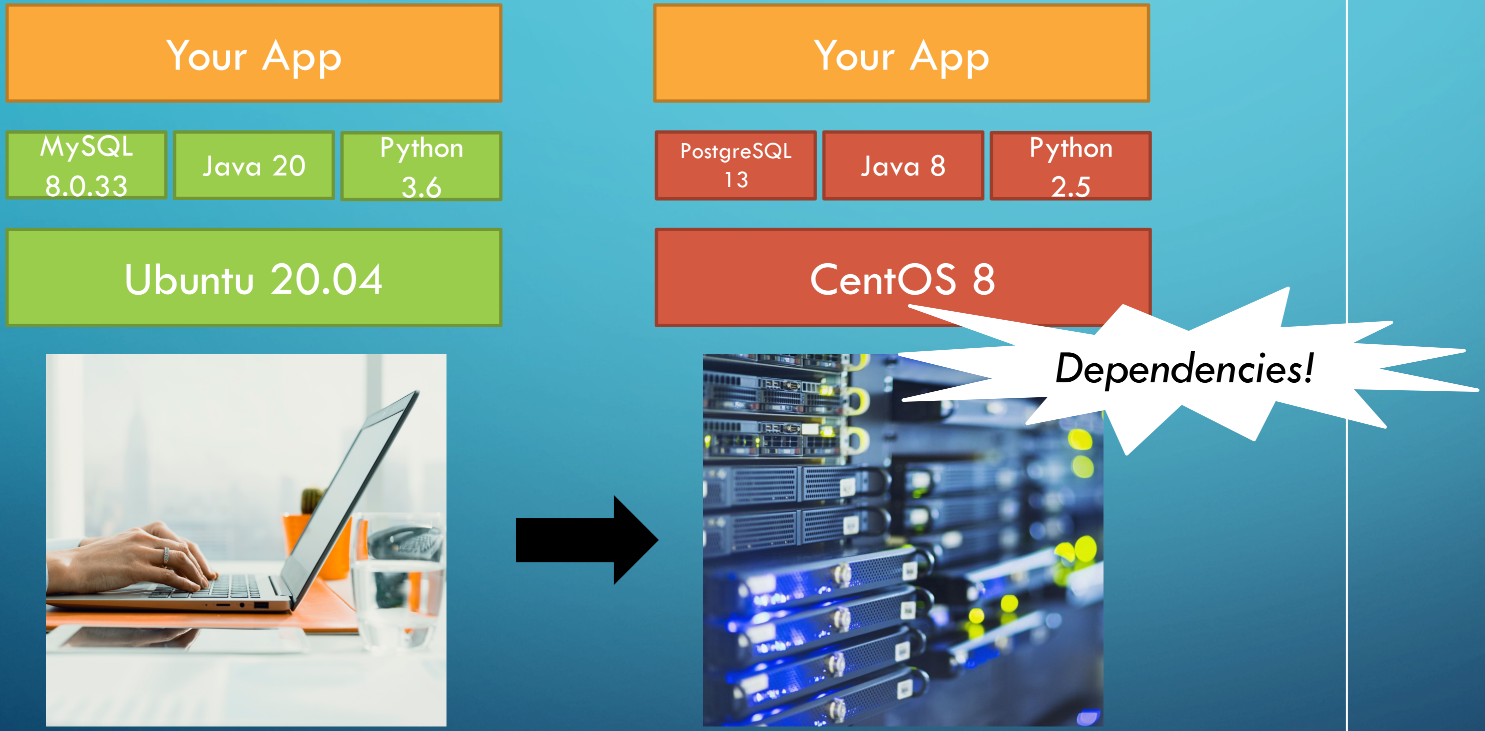
WHY VIRTUALIZATION?



Development Phase

Deployment Phase

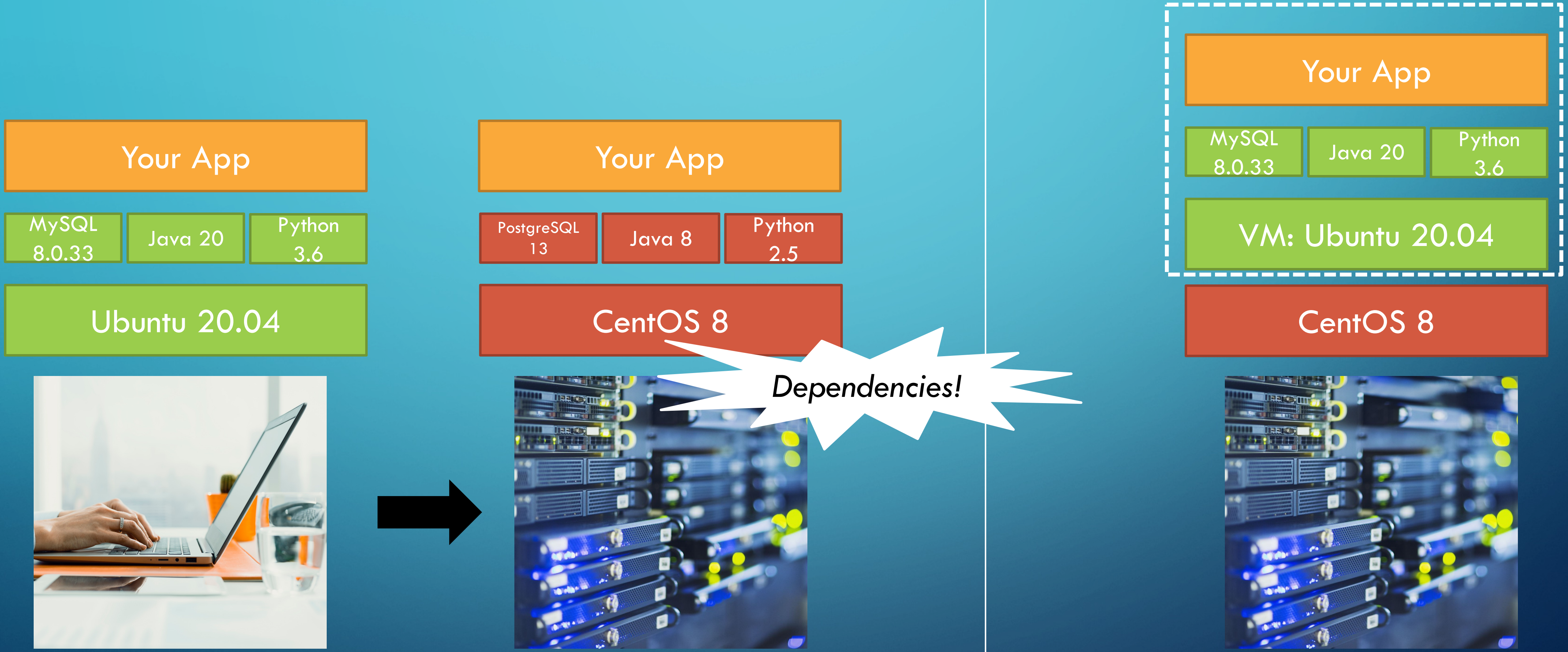
WHY VIRTUALIZATION?



Development Phase

Deployment Phase

WHY VIRTUALIZATION?



Development Phase

Deployment Phase

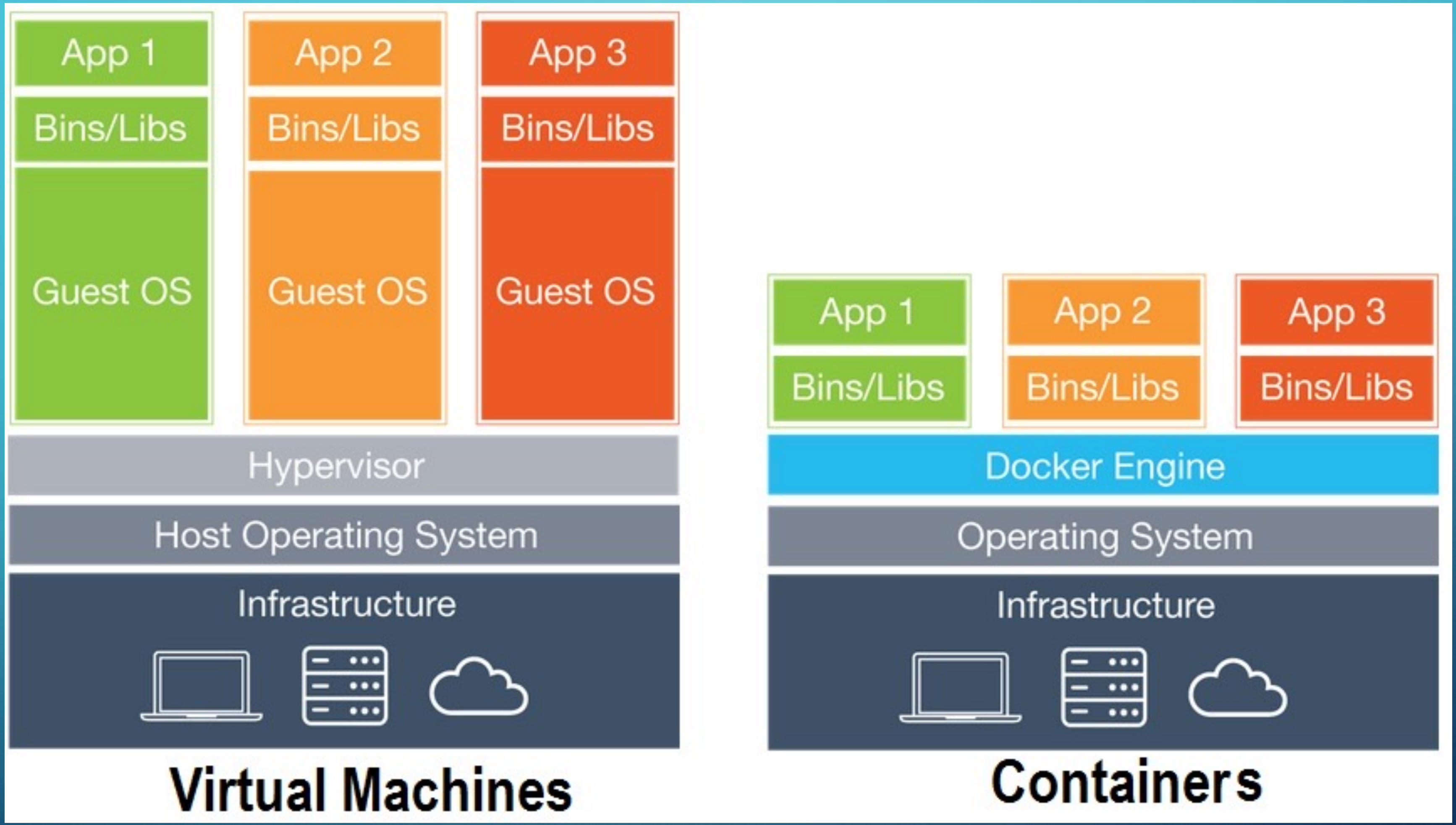
One possible solution:
use a Virtual Machine

WHY VIRTUALIZATION?

- Provide the same runtime environment for software
  - e.g. Java version, Python package, environment variables
- Solution: Deliver the software with its runtime environment
- Other concerns: security, isolation, management, snapshots etc.

VIRTUALIZATION TECHNOLOGIES

- Virtual Machine
  - Provides full virtualization – CPU, disk, memory etc.
  - Install Guest OS
  - Software: VMWare, QEMU
- Container
  - Provides process-level virtualization – runtime library, storage etc.
  - No full OS/hardware control
  - Software: Docker, Apptainer (*Singularity*), sandboxie


App 1
Bins/Libs
Guest OS
App 2
Bins/Libs
Guest OS
App 3
Bins/Libs
Guest OS
Hypervisor
Host Operating System
Infrastructure
Virtual Machines
App 1
Bins/Libs
App 2
Bins/Libs
App 3
Bins/Libs
Docker Engine
Operating System
Infrastructure
Containers

CONTAINER VS VM (IN THE CONTEXT OF RUNTIME)

- Container:
  - Fast to start – Just like starting a new process, ~seconds
  - Less resource consumption – Only overhead for process isolation
  - Smaller size – Kernel etc. is shared
- VM
  - Slow to start – Complete boot sequence
  - Resource consuming – Need to run OS, ~minutes
  - Redundancy – OS, user management, network stack

CONTAINER TECHNOLOGIES

- Container technology is not a new technology
- 1979, Linux chroot – Isolated storage
- 2005, OpenVZ (OS-level virtualization) – Shared Kernel
- 2006, cgroups – resource management and isolation
- 2008, LXC (Linux Container) – based on cgroup, namespace and chroot
- 2013, Docker (Container standard today)

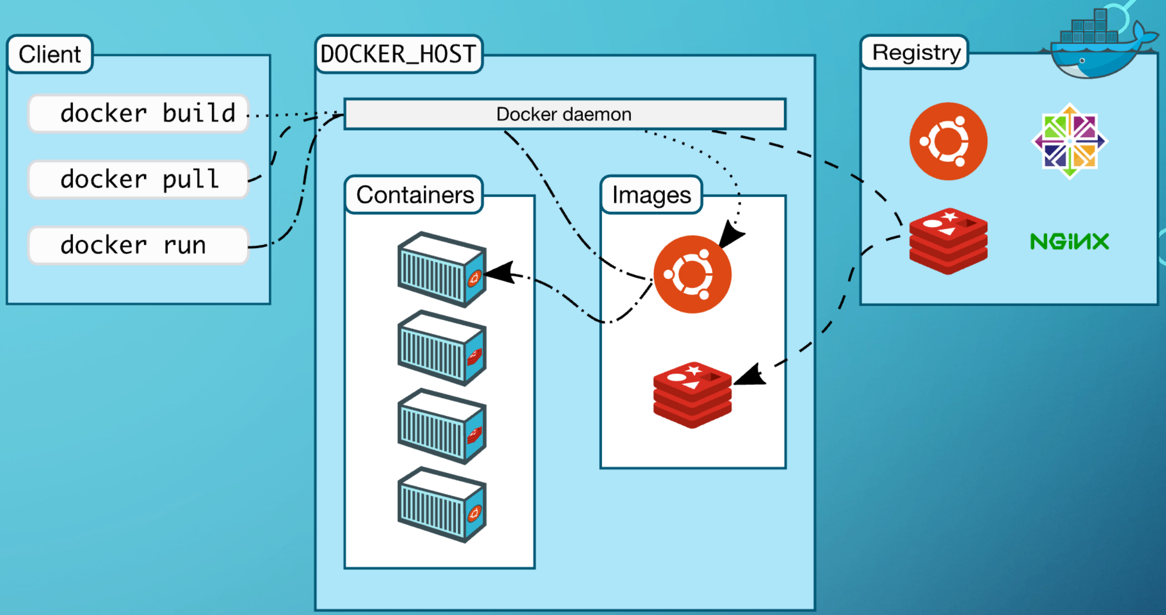
https://blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016

DOCKER

- Docker is a kind of encapsulation to Linux container, and provides a set of easy-to-use API
- It packs application and its dependencies to a file – called docker image
- It provides APIs for users to create, run, manage docker images/applications
- Common usage:
  - Temporary environment – e.g. Testing software
  - Elastic cloud service – On demand scaling
  - Microservice

DOCKER ARCHITECTURE

- Client-server architecture
  - Docker daemon (dockerd)
  - Docker client (docker)
  - Docker registry: stores docker images



CORE CONCEPTS

- Image: A special kind of file system, provides resources and configuration for application. Once created, content will not change
- Container: A running process in docker, created from docker image
- Registry: A repository with which users can store and share docker images with others
- Docker Hub: Docker's official repository service

INSTALLING DOCKER

- Easy to do with apt under Ubuntu
- Follow guides at https://docs.docker.com/engine/install/ubuntu/
- After installing, you can add the current user to docker group, since running docker needs sudo permission

```
sudo groupadd docker
sudo usermod -aG docker $USER
```

DOCKER USAGE (COMMAND-LINE)

- Docker is a C/S architecture, so start docker service first

```
sudo systemctl start docker
```

- Getting the docker hello world image

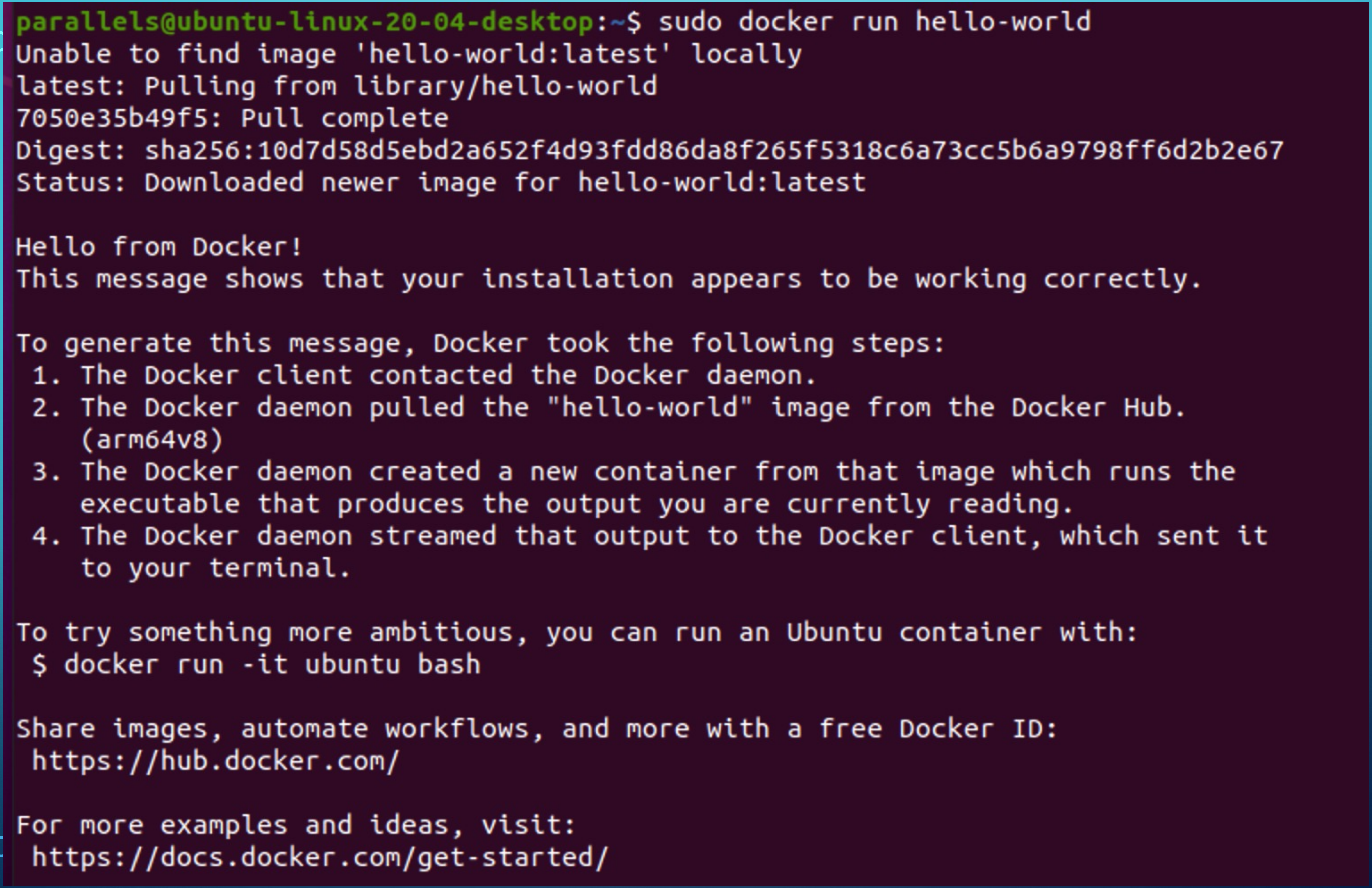
```
docker image pull hello-world
```

- List all image files locally

```
docker image ls
```

- Run

```
docker run hello-world
```



```
parallels@ubuntu-linux-20-04-desktop:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
7050e35b49f5: Pull complete
Digest: sha256:10d7d58d5ebd2a652f4d93fdd86da8f265f5318c6a73cc5b6a9798ff6d2b2e67
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

DOCKER CLI (2)

```
docker ps          # list all running containers

docker ps -all     # list all containers

docker stop <id> # stop container process

docker rm <id>     # remove stopped container

docker run -it ubuntu bash
```

Instantiate a container from the ubuntu image, and execute bash in interactive mode

TIPS

- Installation: https://docs.docker.com/engine/install/ubuntu/
- Tutorial: https://docs.docker.com/get-started/
- Configure Docker Hub mirror to speed up accessing from mainland China
  - Modify /etc/docker/daemon.json

```
{
        "registry-mirrors": ["https://hub-mirror.c.163.com/"]
}
```

  - Then restart: `sudo systemctl restart docker`