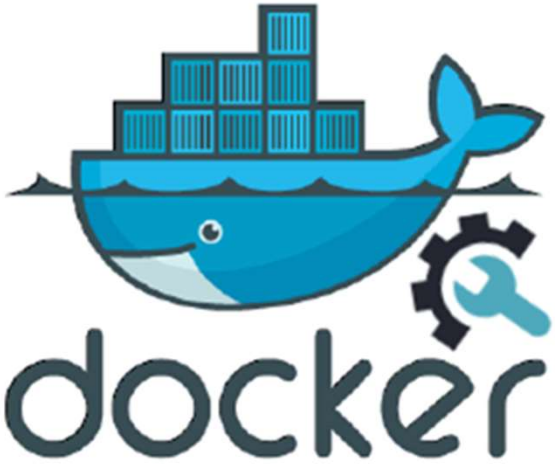
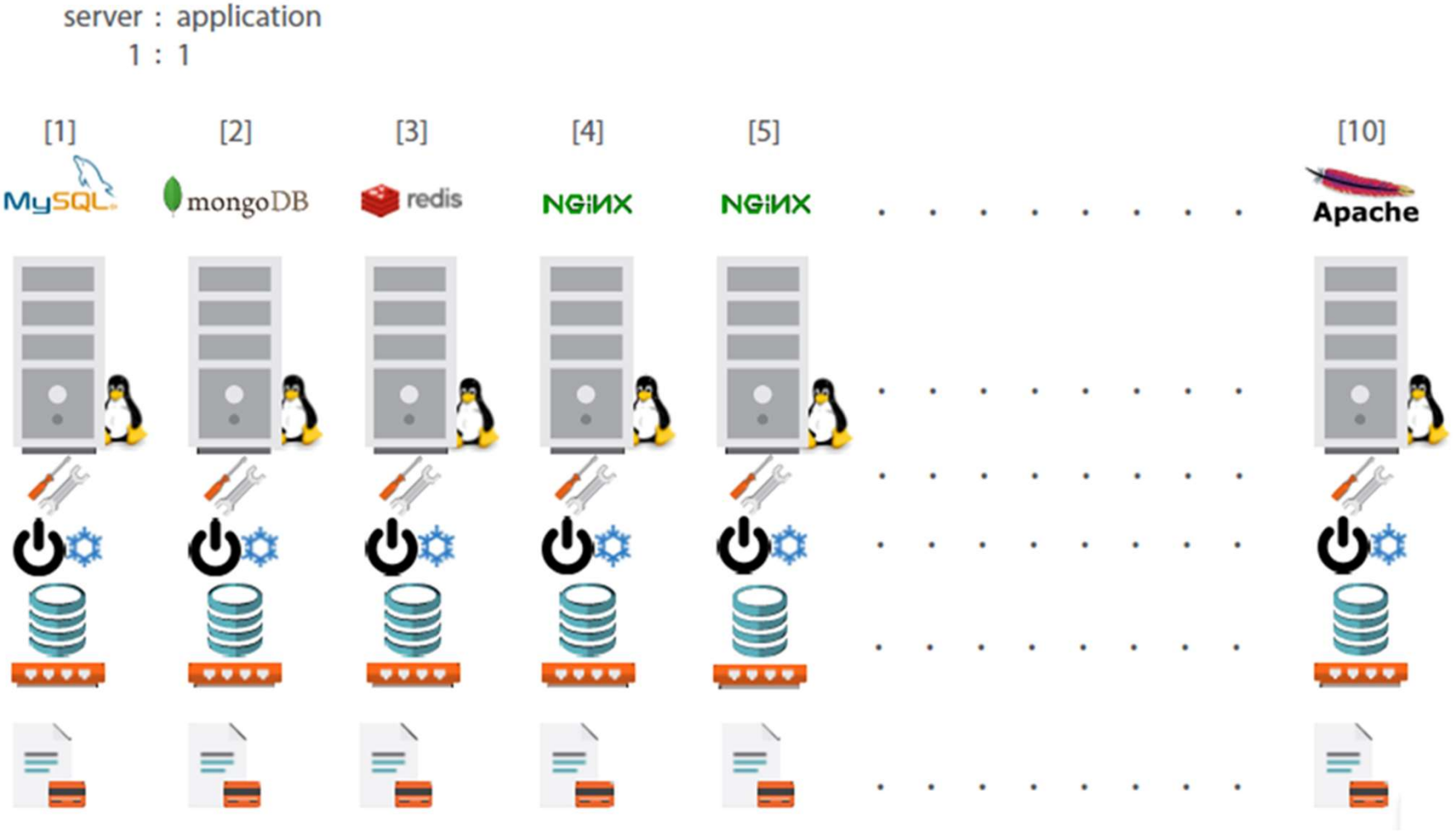
Docker Essentials

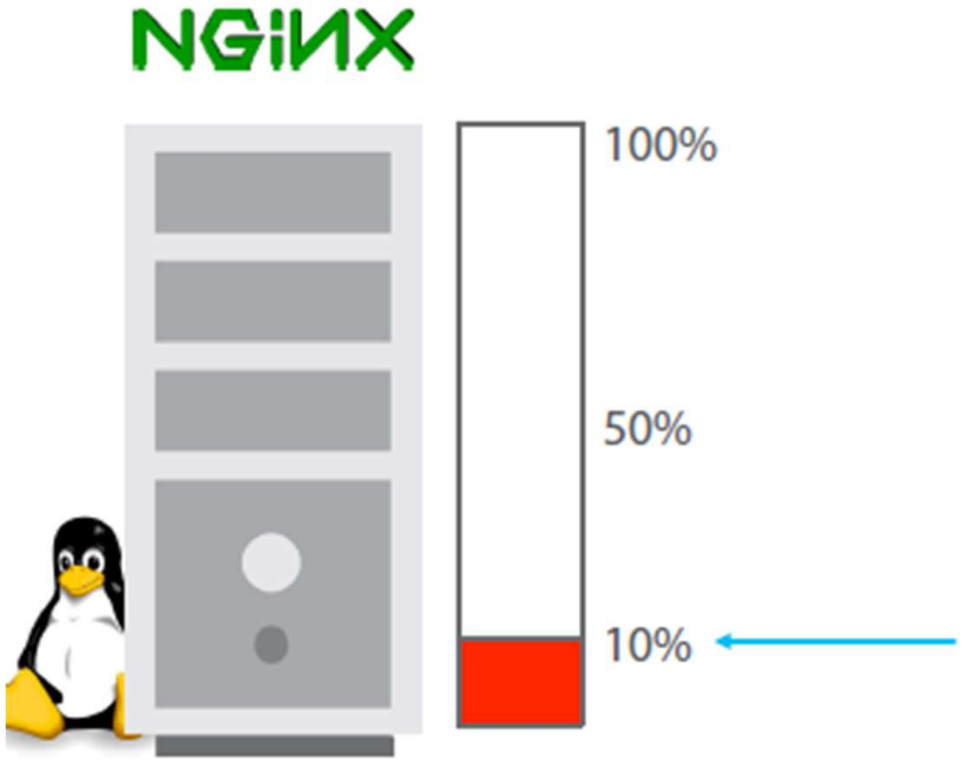
Docker



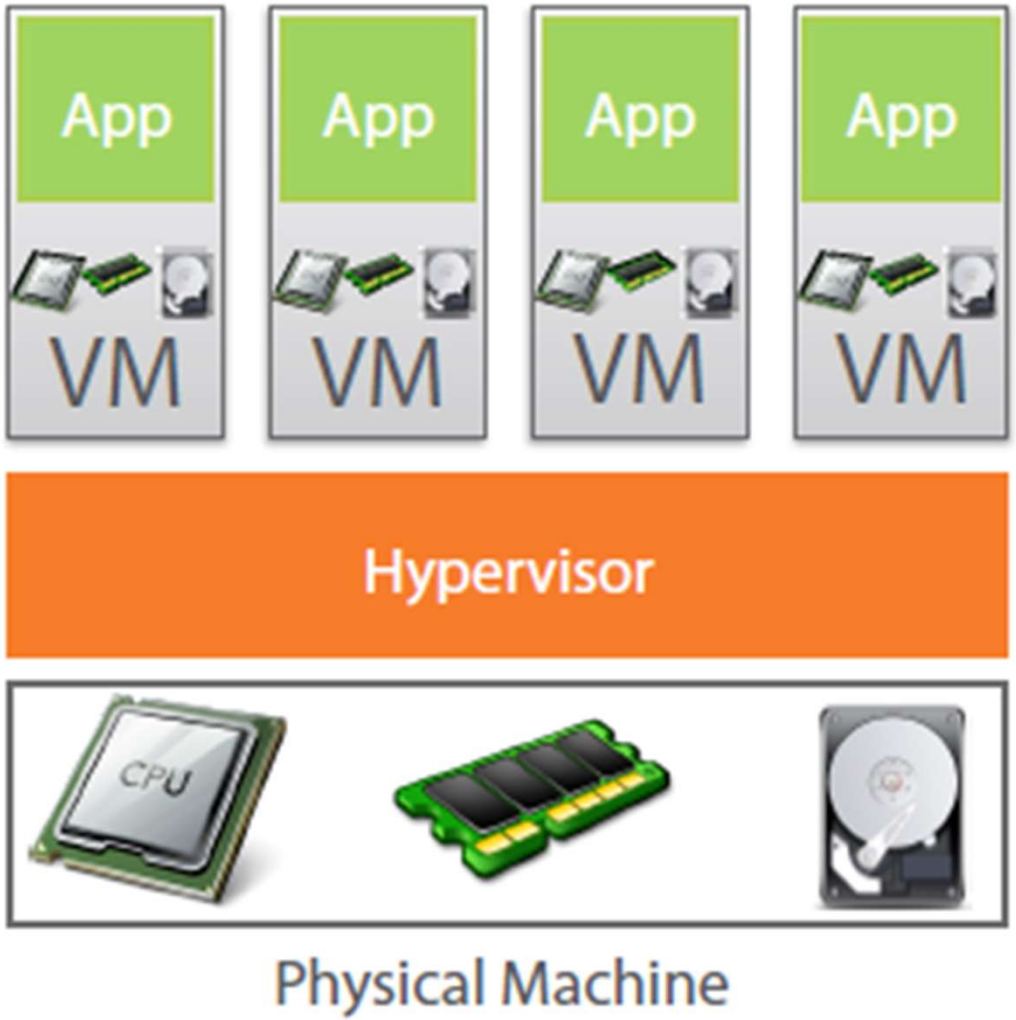
Traditional Deployment Architecture



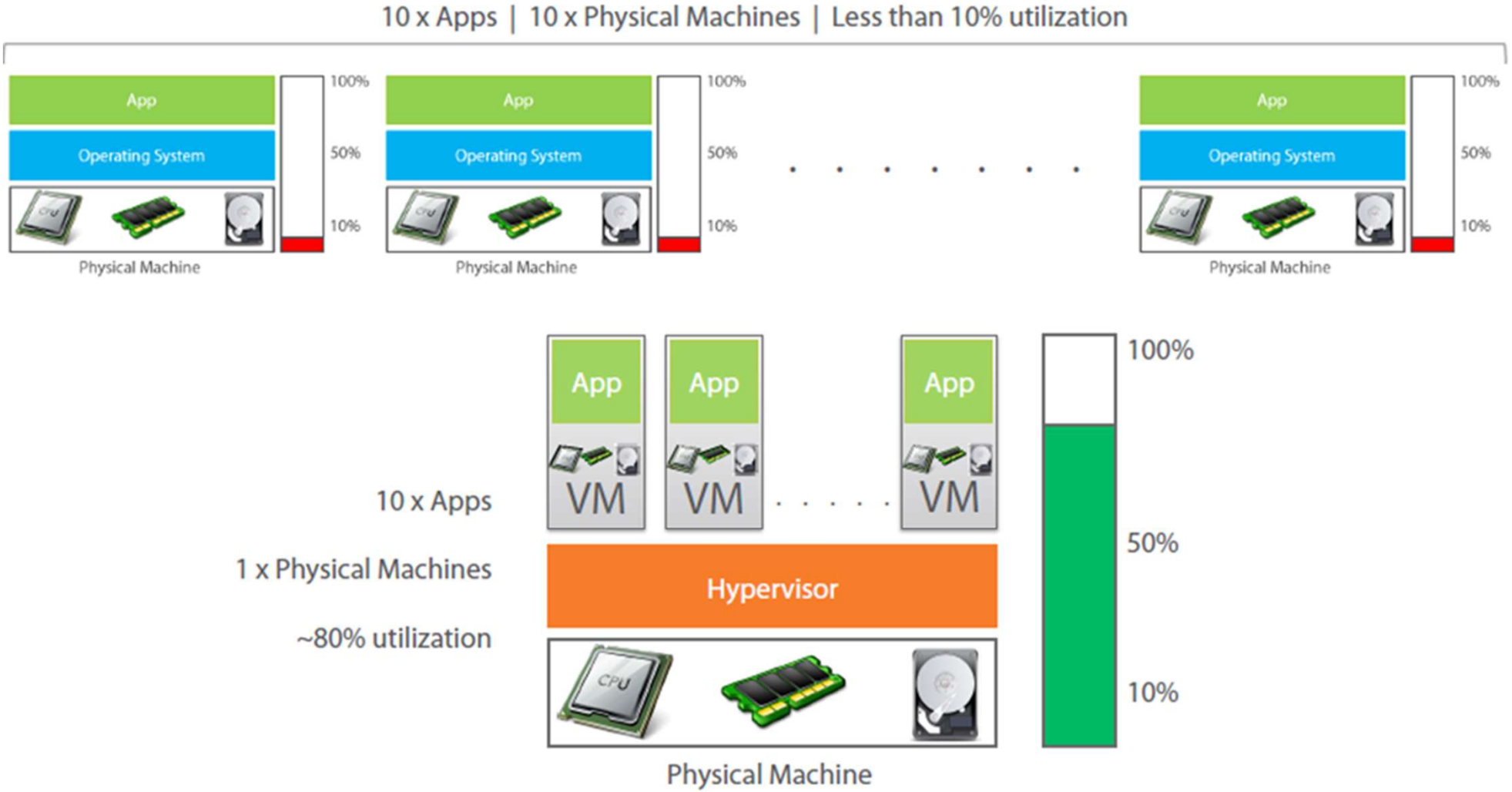
Less Utilization in Traditional Architecture



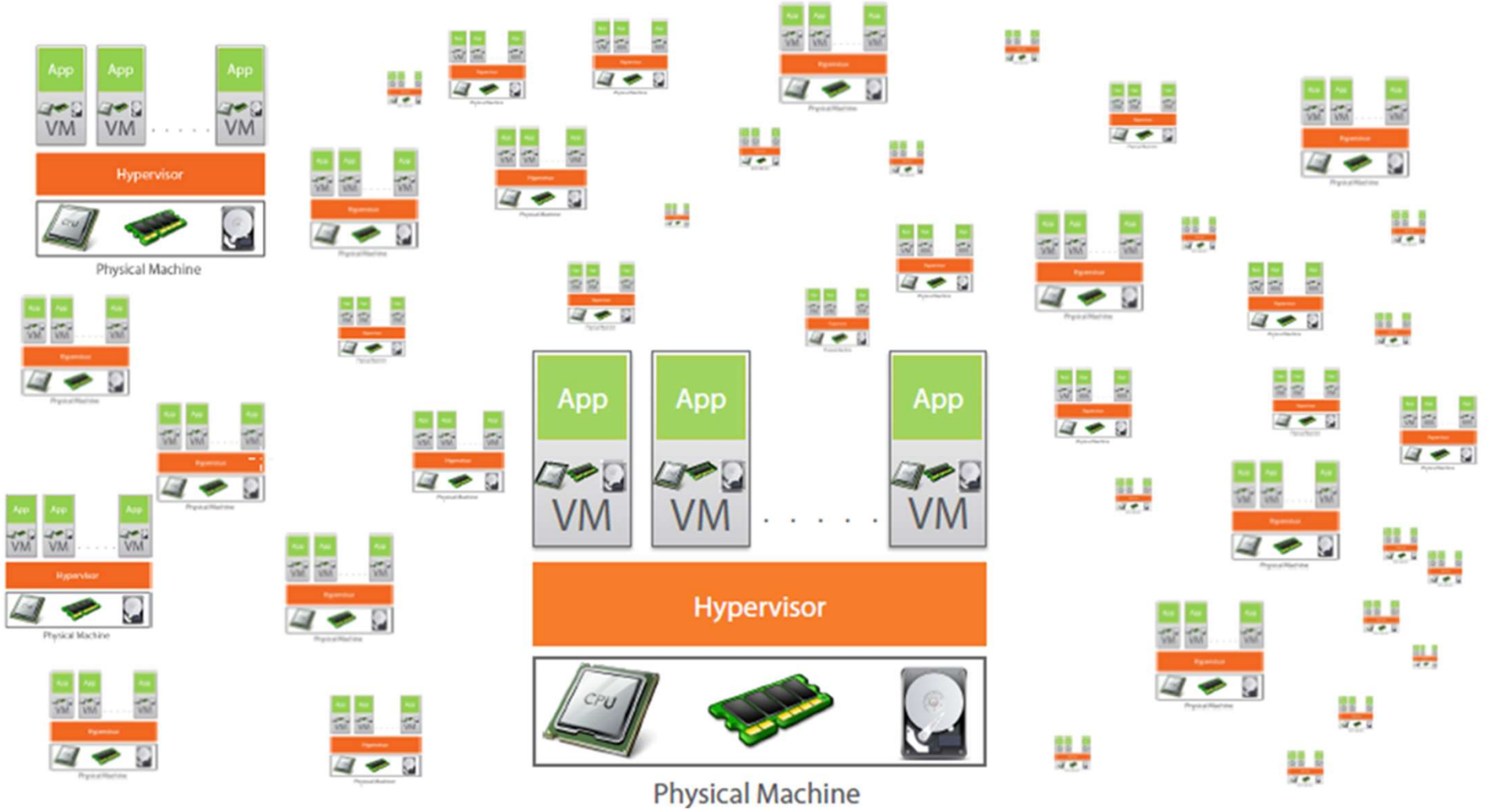
Virtual Machine to the Rescue



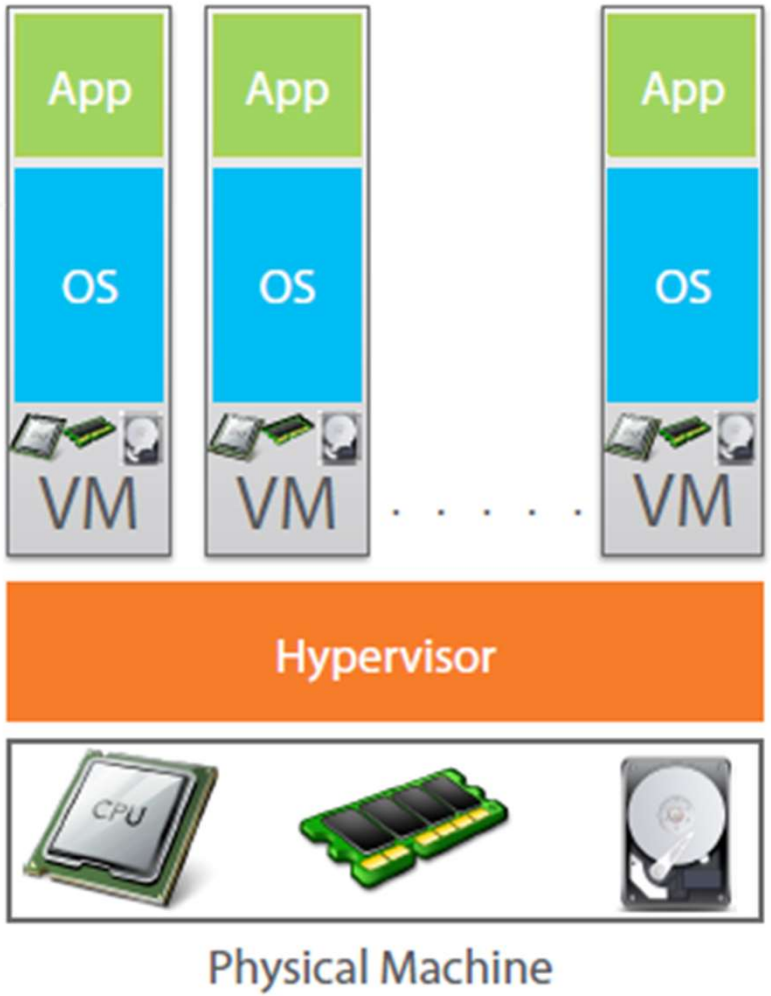
Virtual Machine provides better utilization



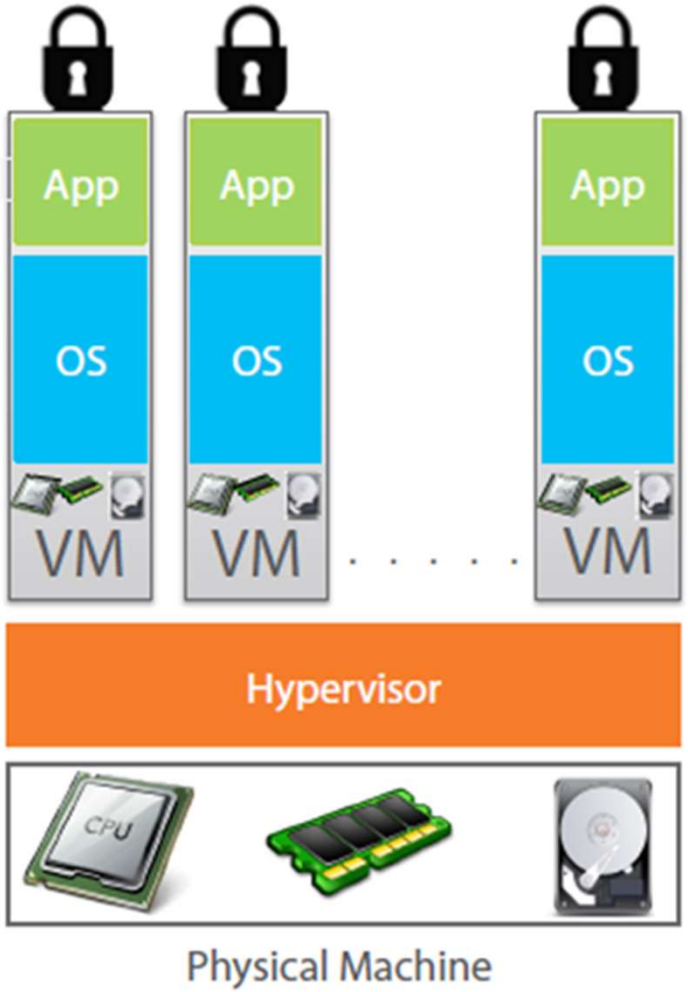
But Virtual Machine increases Licensing Cost



Each VM needs a separate OS

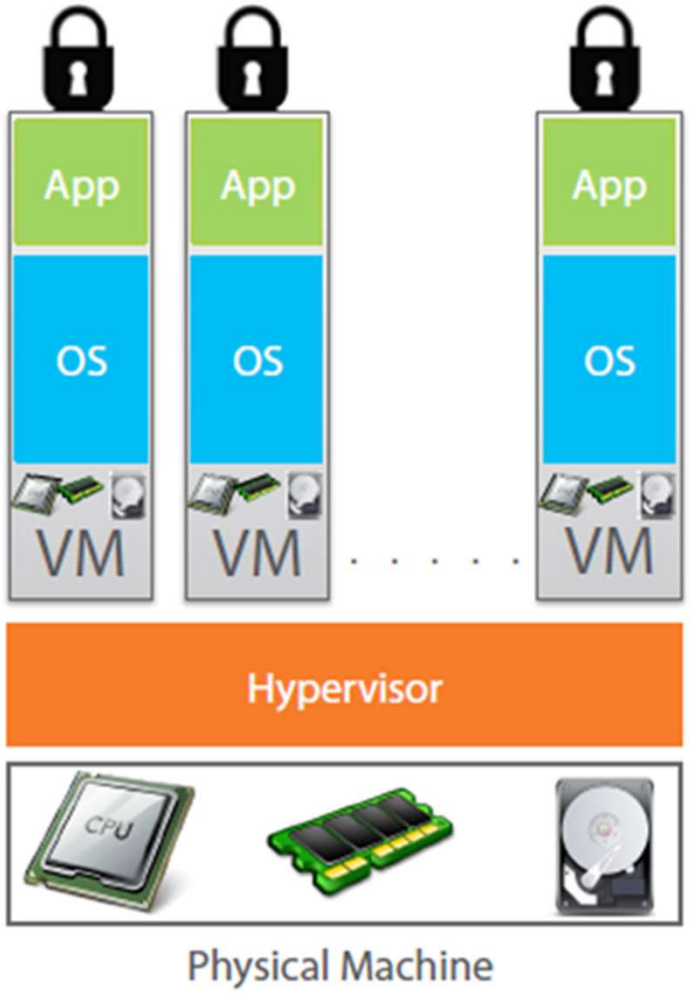


More OSes doesn’t increase Business Value



> OS != Business Value

OS takes most of the Resources

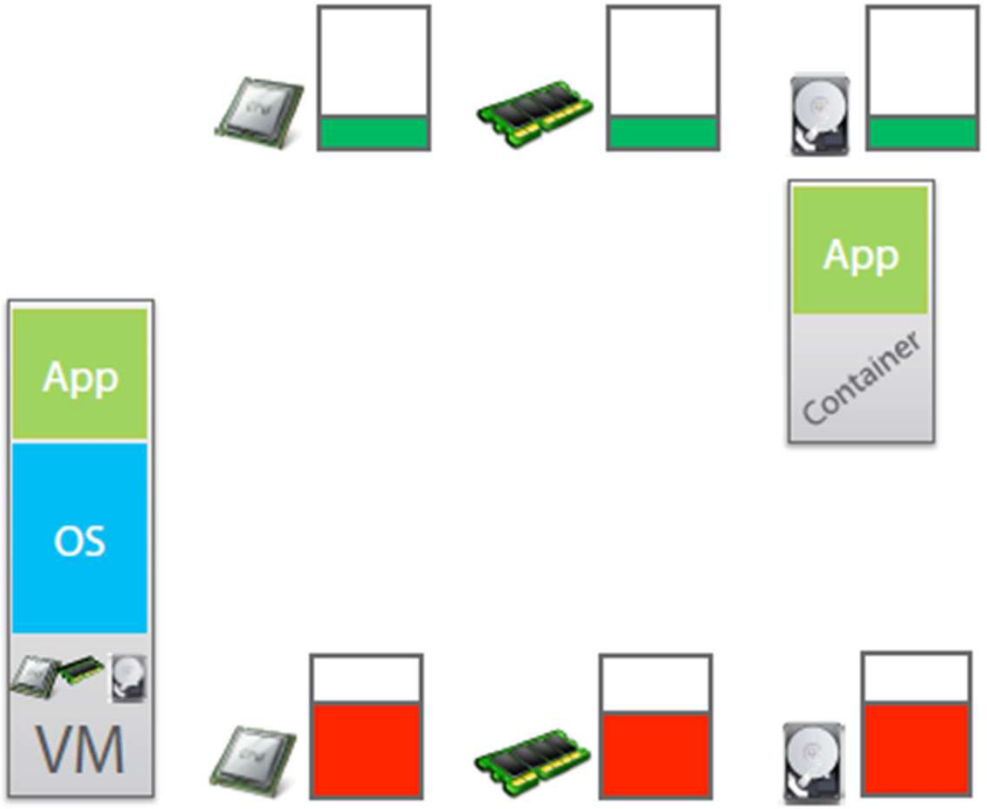


Why use separate OS for each App?

Containerization

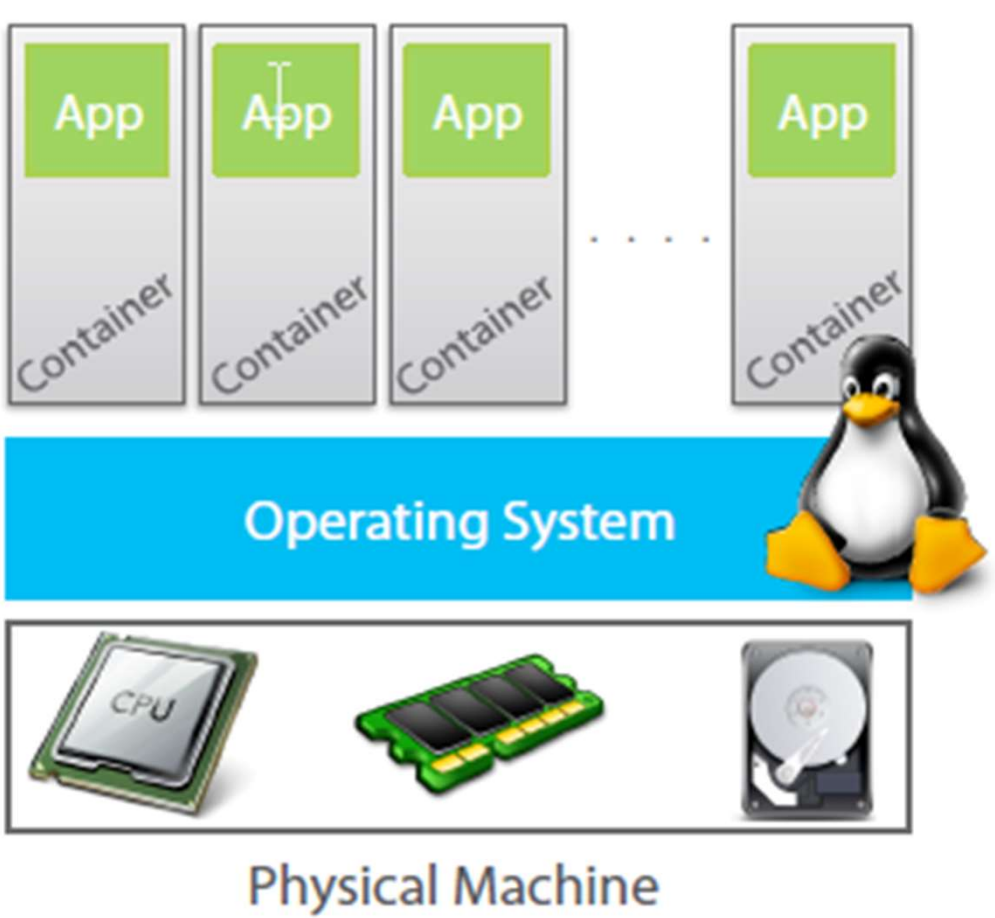
- Encapsulation of an application and its required environment.
- The process of packaging an application along with its required libraries, frameworks, and configuration files together so that it can be run in various computing environments efficiently.

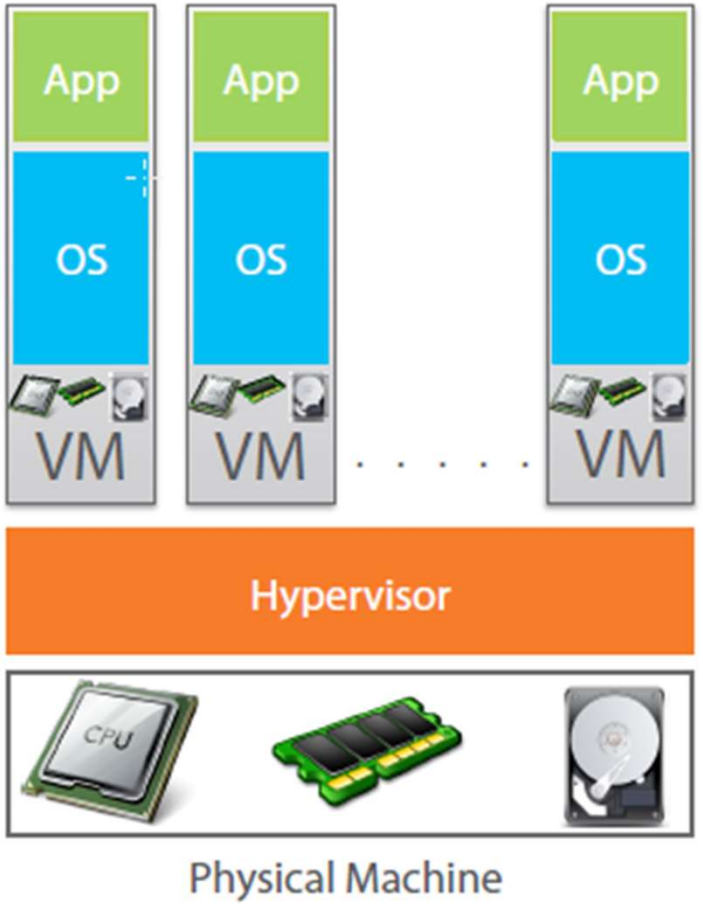
Containers to the Rescue



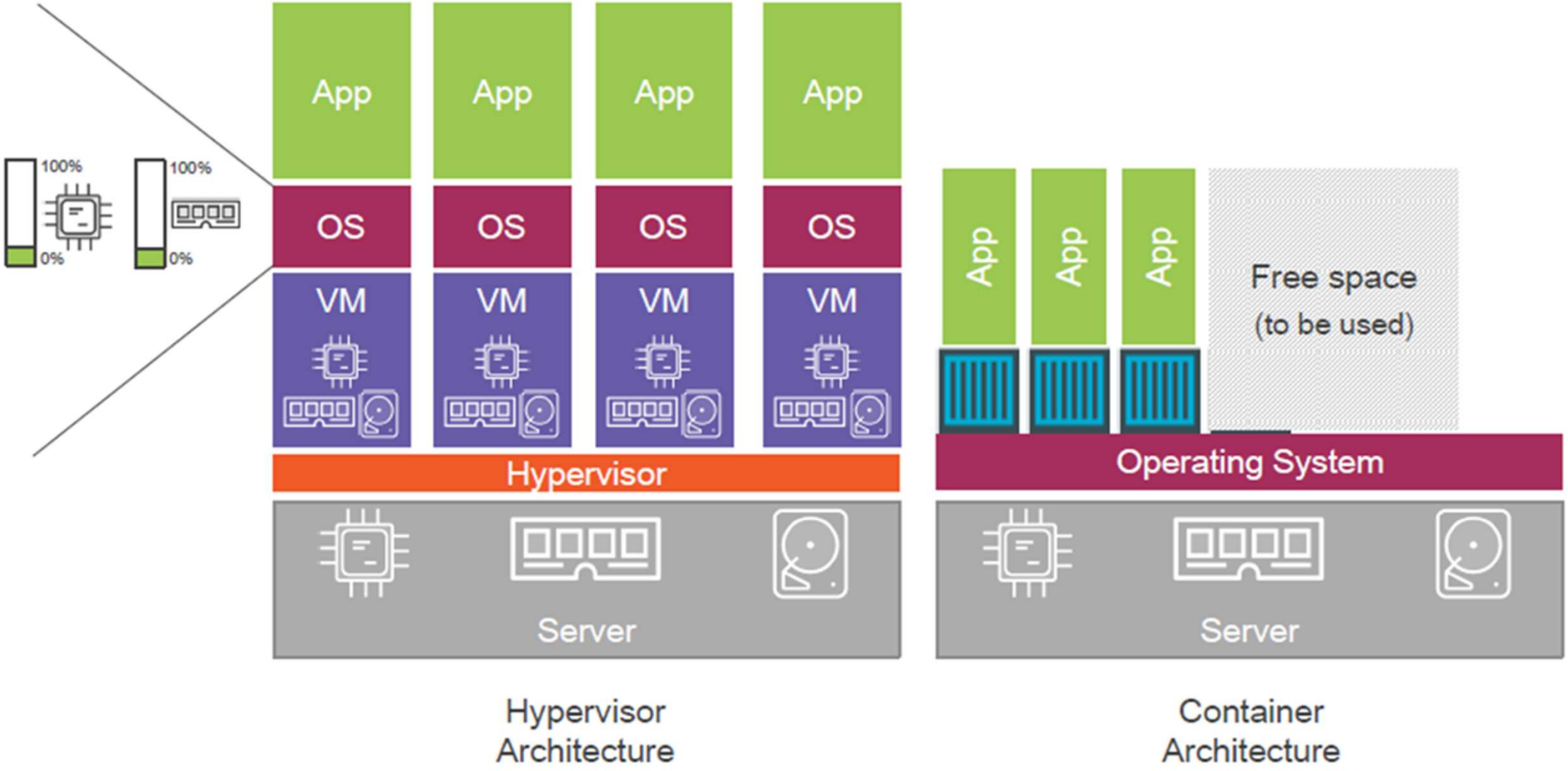
Containers are more lightweight than Virtual Machines

Containers vs VM

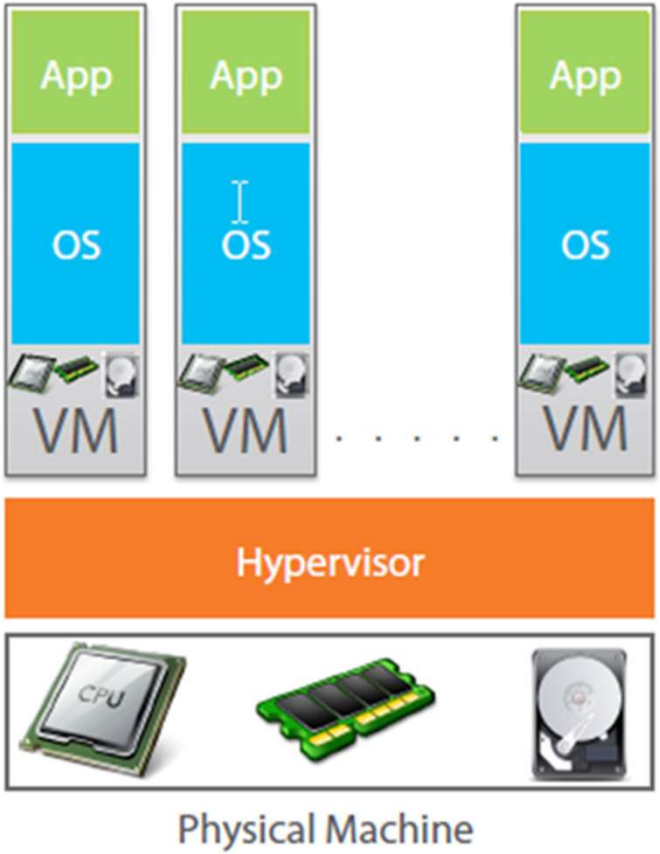


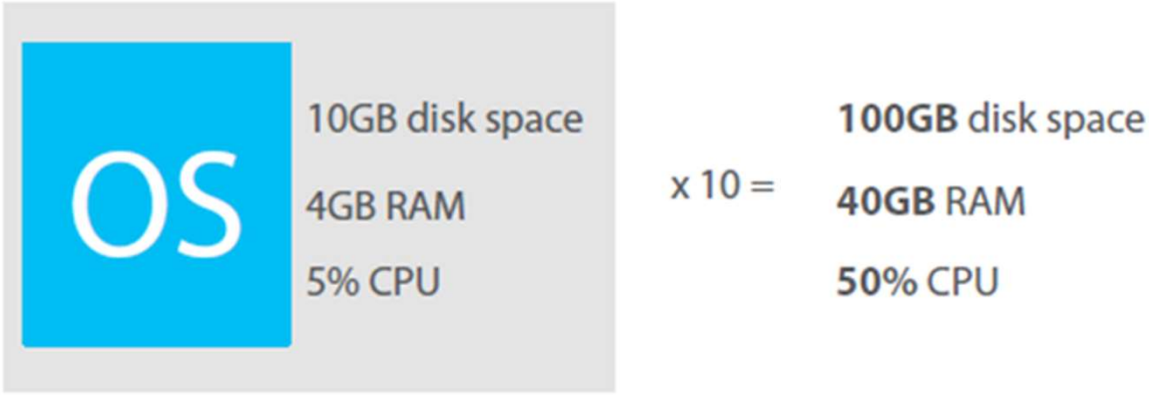


Containers vs VM

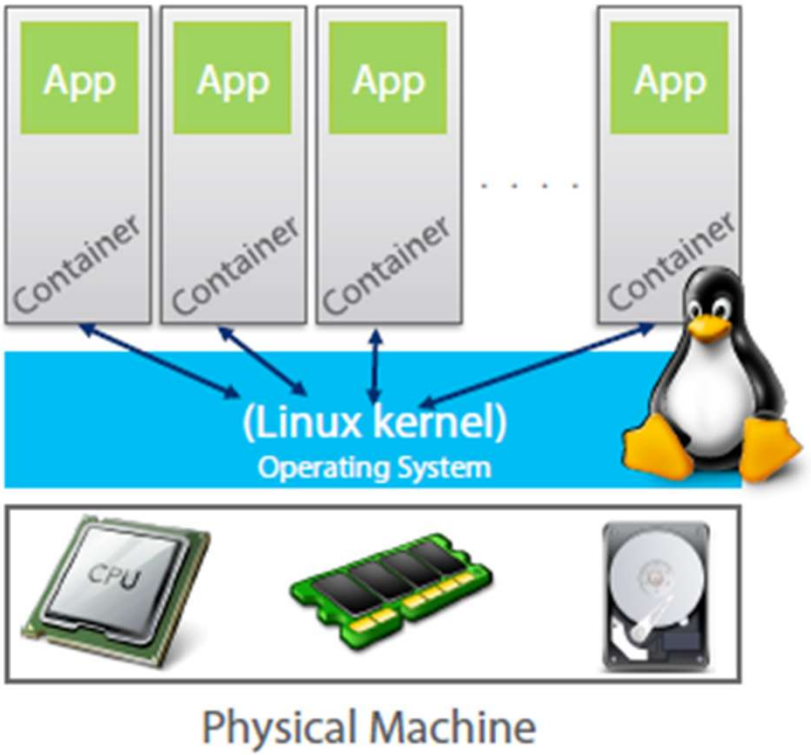


OS takes more resources and Licensing cost





Containers takes less resources



Containers consume less CPU, RAM and disk resource than Virtual Machines

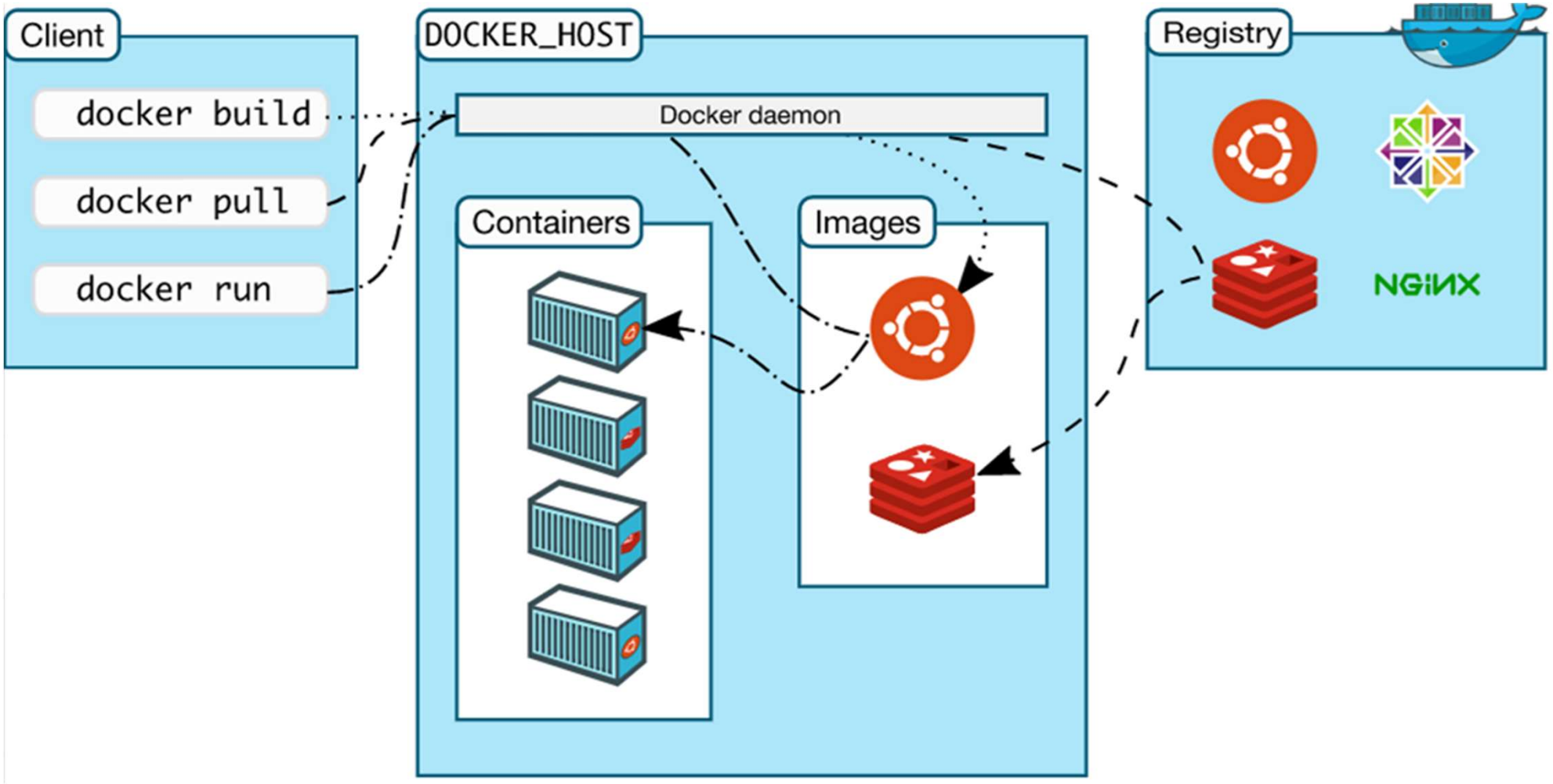
What is Docker?

- Docker is an open-source project
  - that automates the deployment of applications inside software containers,
  - by providing an additional layer of abstraction and
  - automation of operating system–level virtualization on Linux.

Practical

Practical Guide

Docker Architecture



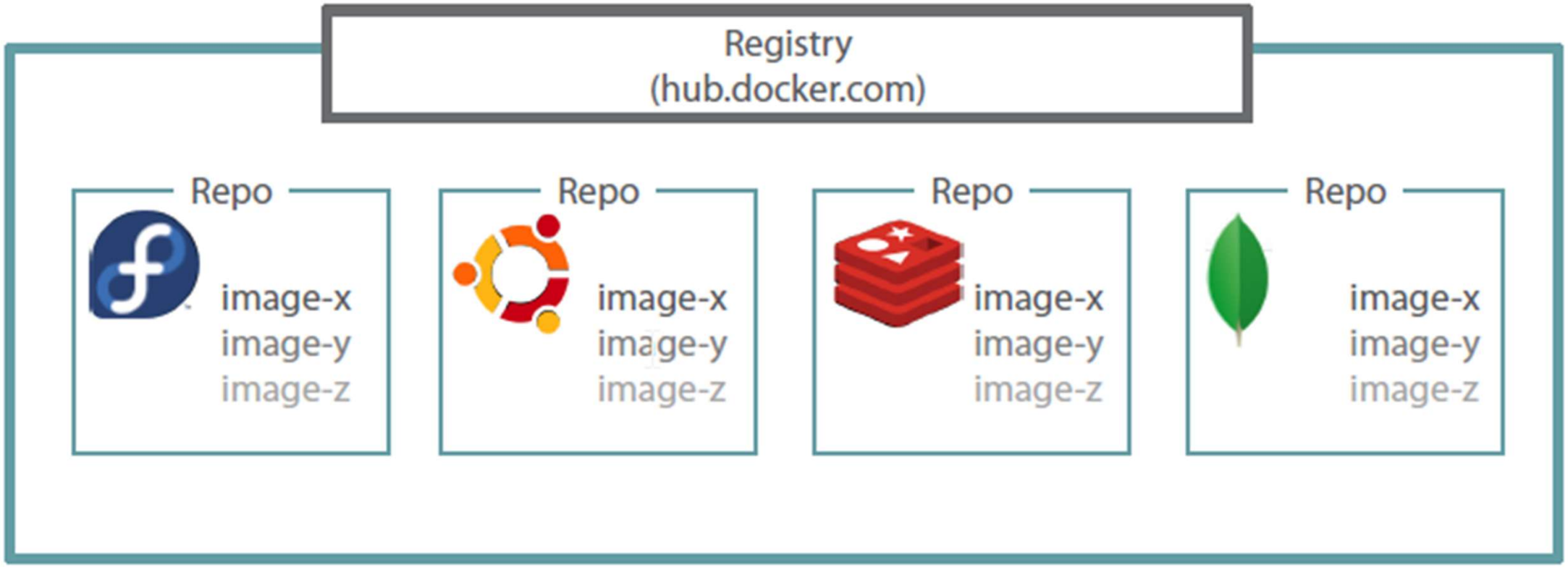
Image

- Persisted snapshot that can be run
- Common Docker Commands:
  - images: List all local images
  - run: Create a container from an image and execute a command in it
  - tag: Tag an image
  - pull: Download image from repository
  - rmi: Delete a local image

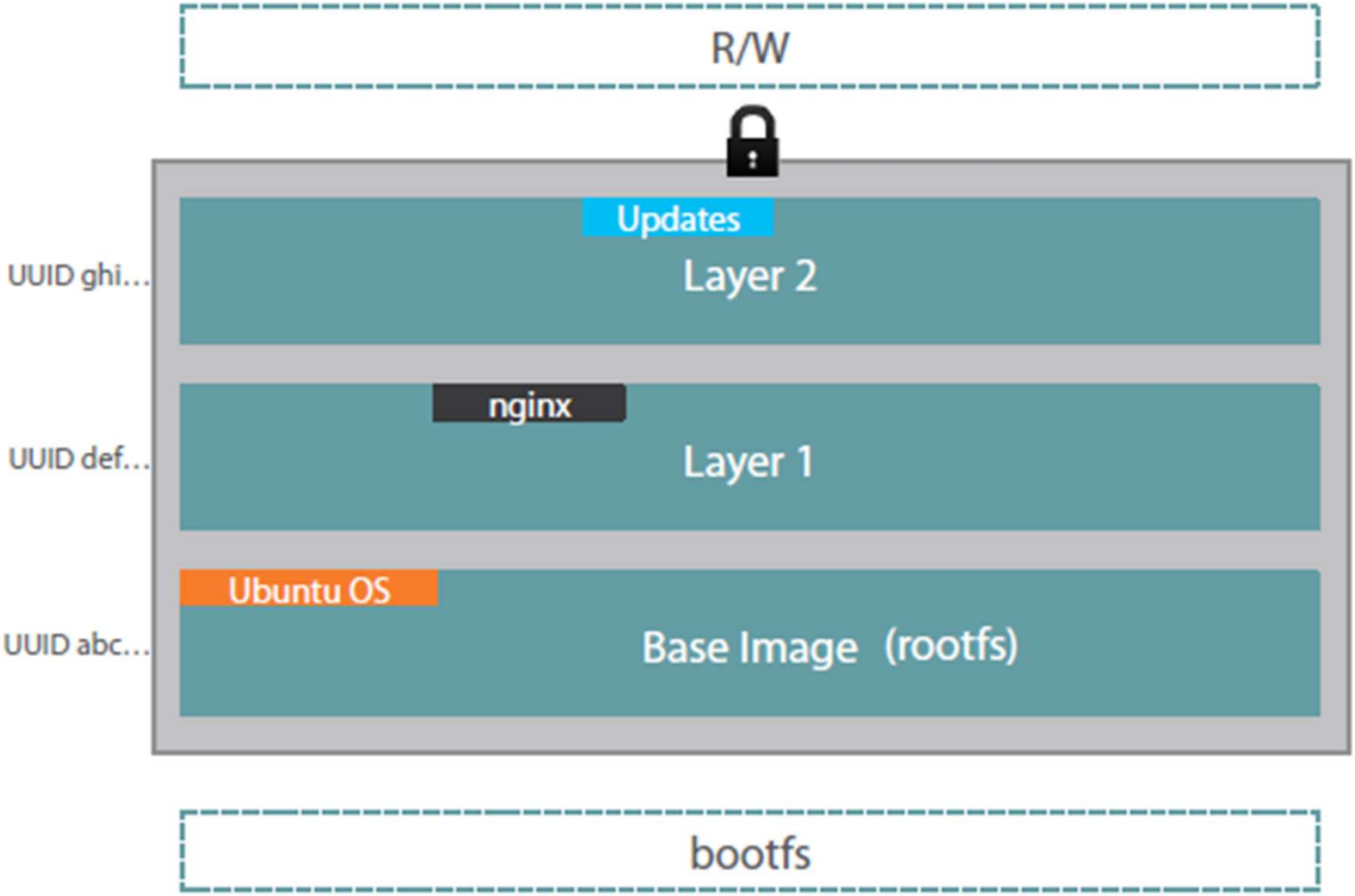
Container

- Runnable instance of an image
- Common Docker Commands
  - ps: List all running containers
  - ps –a: List all containers (incl. stopped)
  - top: Display processes of a container
  - start: Start a stopped container
  - stop: Stop a running container
  - pause: Pause all processes within a container
  - rm: Delete a container
  - commit: Create an image from a container

Docker Registry



Layers in Images



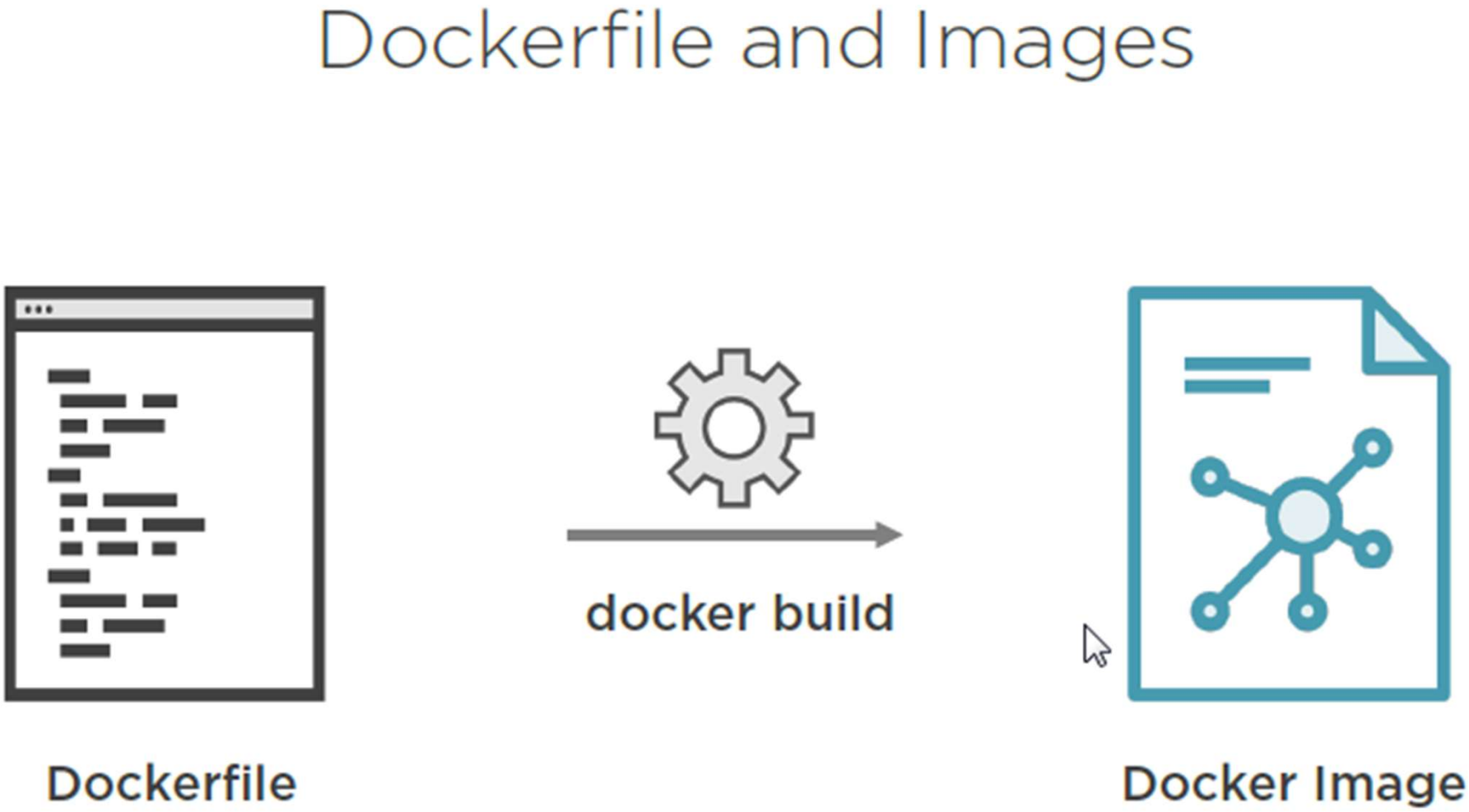
Hands-On

- We need to do the below hands-on:
  - ssh to Ubuntu server
  - Install Docker on Ubuntu 18.04
  - Validate docker engine is successfully installed
  - Launch a docker container
  - Login to container
  - Work in a container
  - List containers
  - Pause a container
  - Un-pause a container
  - Delete container
- Refer to the command guide for instructions

Create Dockerized Application

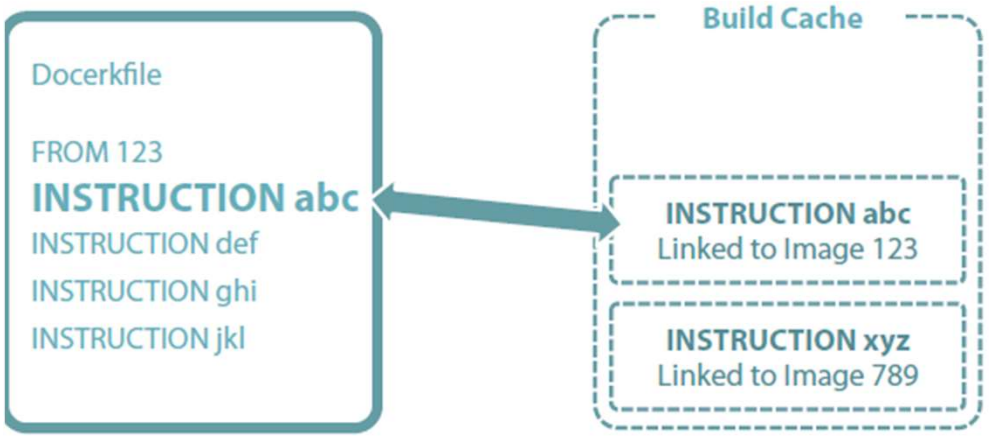
- We can dockerize our application using dockerfile
  - Dockerfile Create images automatically using a build script: «Dockerfile»
  - It Can be versioned in a version control system like Git
  - Docker Hub can automatically build images based on dockerfiles on Github
- This is a basic Dockerfile we need to dockerize a node application
  - FROM node:4-onbuild
  - RUN mkdir /app
  - COPY . /app/
  - WORKDIR /app
  - RUN npm install
  - EXPOSE 8234
  - CMD ["npm", "start"]

Dockerfile


Dockerfile and Images
docker build
Dockerfile
Docker Image

Dockerfile Template

Dockerfile Build Cache



FROM node:4-onbuild

- Pulls/downloads a base image from docker hub which is a public hub for docker images.
- For running a node appication you need to install node in your system

RUN mkdir /app

- In this command we make create an empty directory which will be our working directory with the code files.

COPY . /app/

- Copies all files in current directory to the newly created app directory.
- Your Dockerfile should be in the parent directory of your project.

WORKDIR /app

- To switch from current directory to the app directory where we will run our application.

RUN npm install

- This npm command is related to node application.
- When we copied all dependencies, our main file - package.json would have been copied.
- So running above command installs all dependencies from the file and creates a node_modules folder with mentioned node packages.

EXPOSE 8234

- This command is to expose a port we want our docker image to run on.

CMD ["npm", "start"]

- This is a command line operation to run a node application.
- It may differ based on projects.

Build Image

- Now once we have our Dockerfile ready lets build an image out of it.
- Assuming you all have docker installed on your system lets follow some simple steps:-
  - Navigate to directory containing Dockerfile.
  - Run the following command on your terminal:-
    - docker build -t myimage .
- docker images
- docker run -p 8234:8234 'your image name'

Publish Port

- docker run –t –p 8080:80 ubuntu
  - Map container port 80 to host port 8080

Docker Hub

- Public repository of Docker images
  - https://hub.docker.com/
  - docker search [term]

- Use my own registry
  - To pull from your own registry, substitute the host and port to your own:
    - docker login localhost:8080
    - docker pull 164.52.197.86 :5000/test-image

Clean Up

- docker stop $(docker ps -a -q) #stop ALL containers
- docker rm -f $(docker ps -a -q) # remove ALL containers


thanks