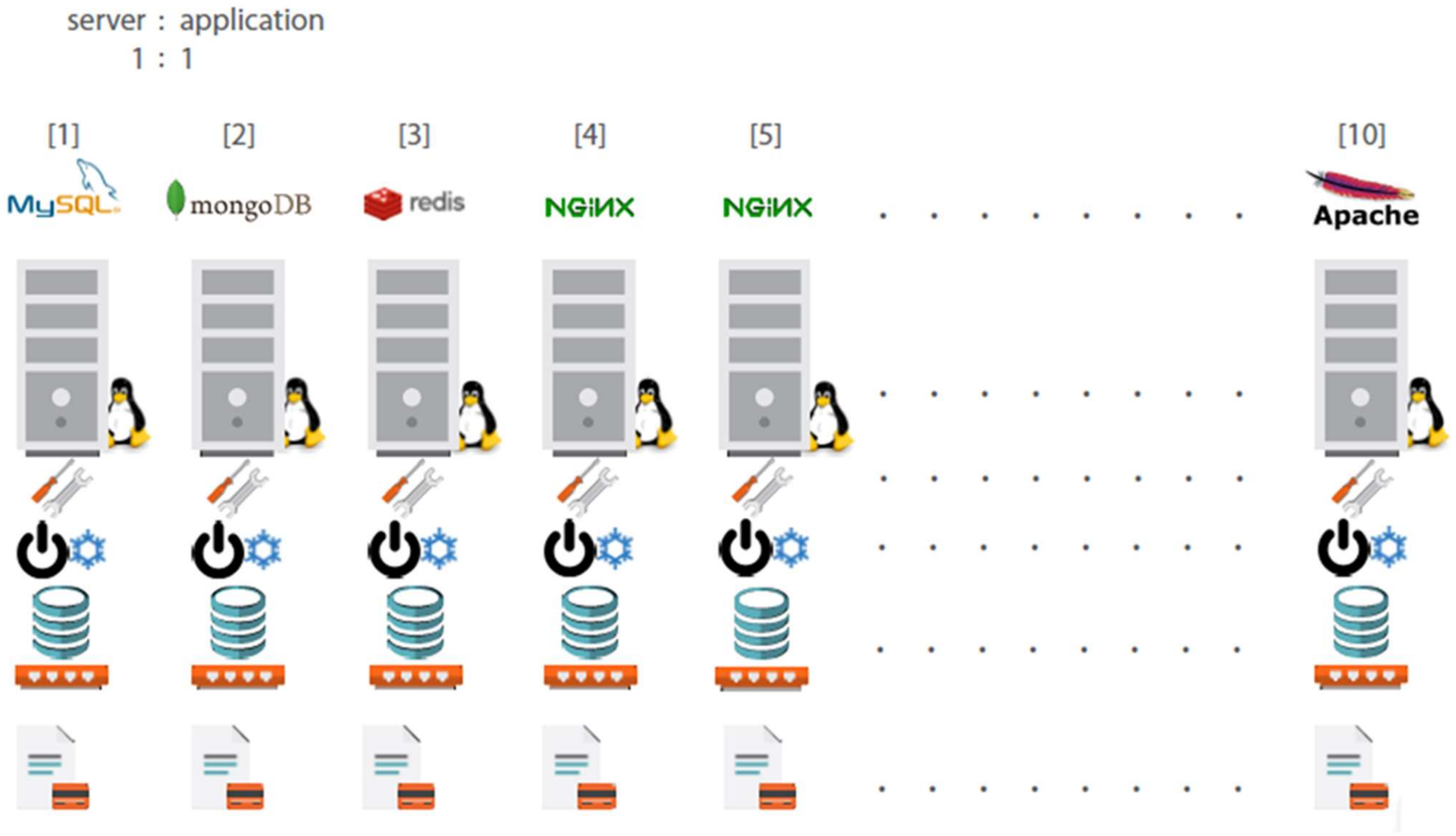
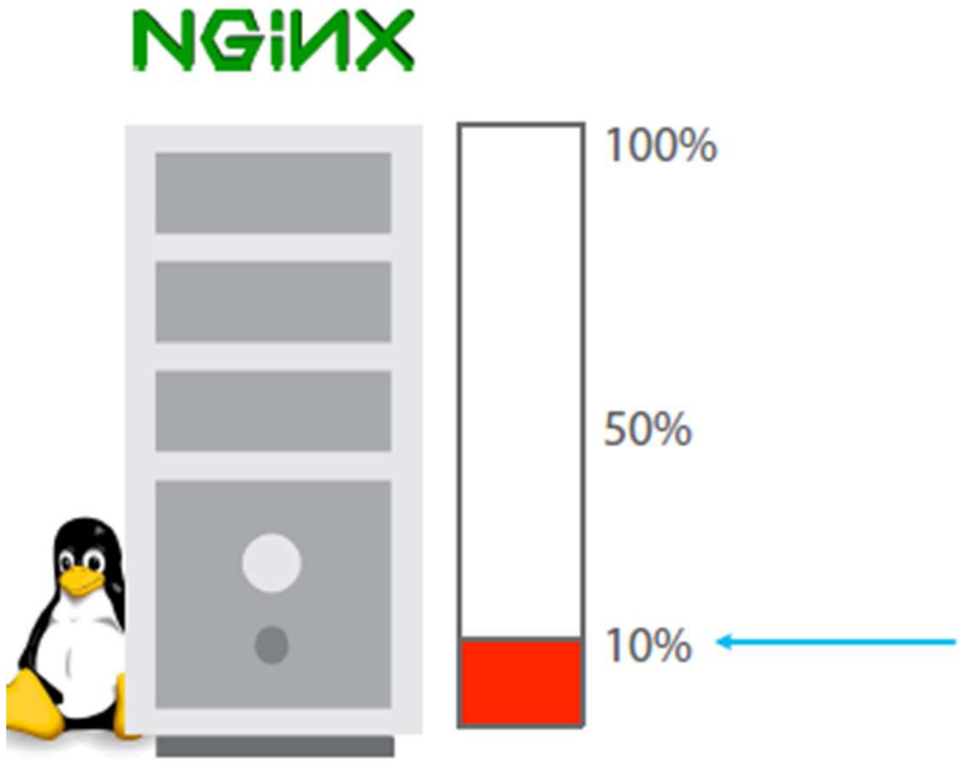
Docker

Docker

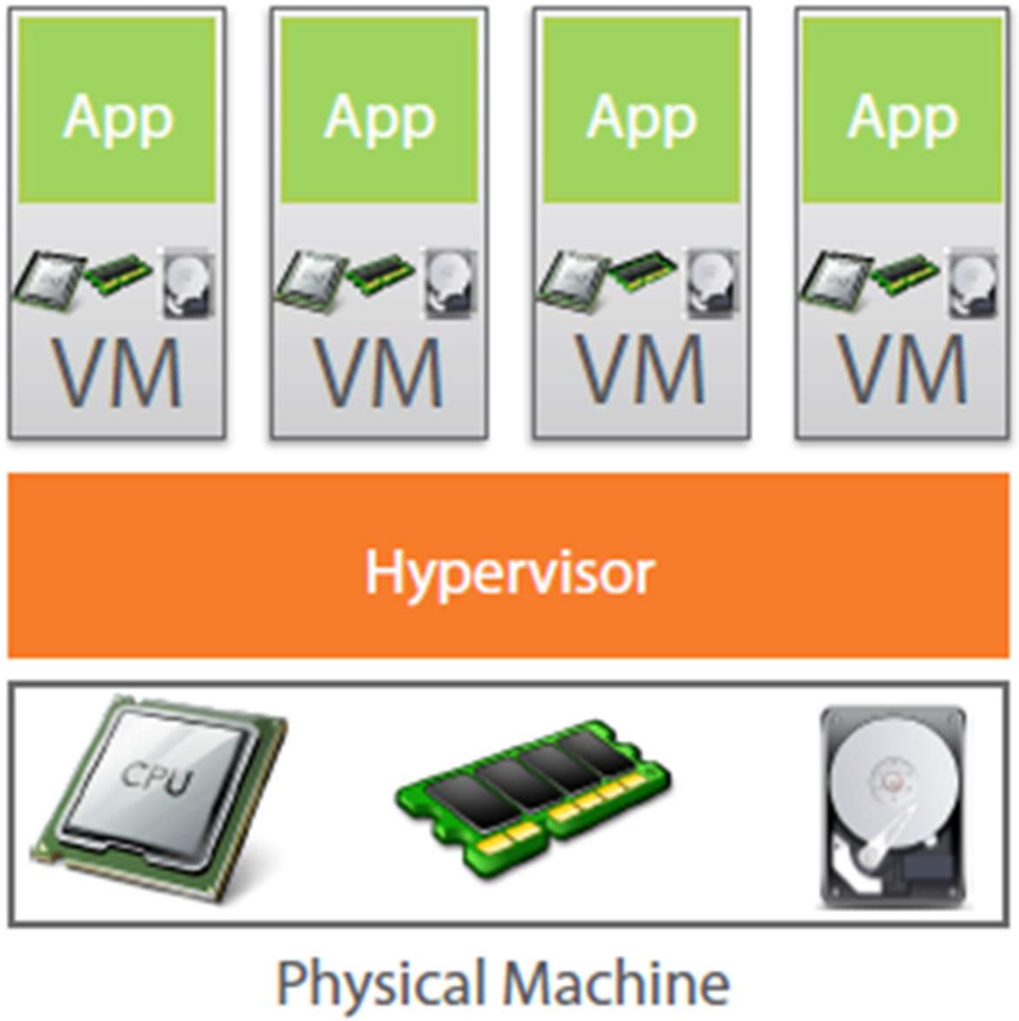
Traditional Deployment Architecture



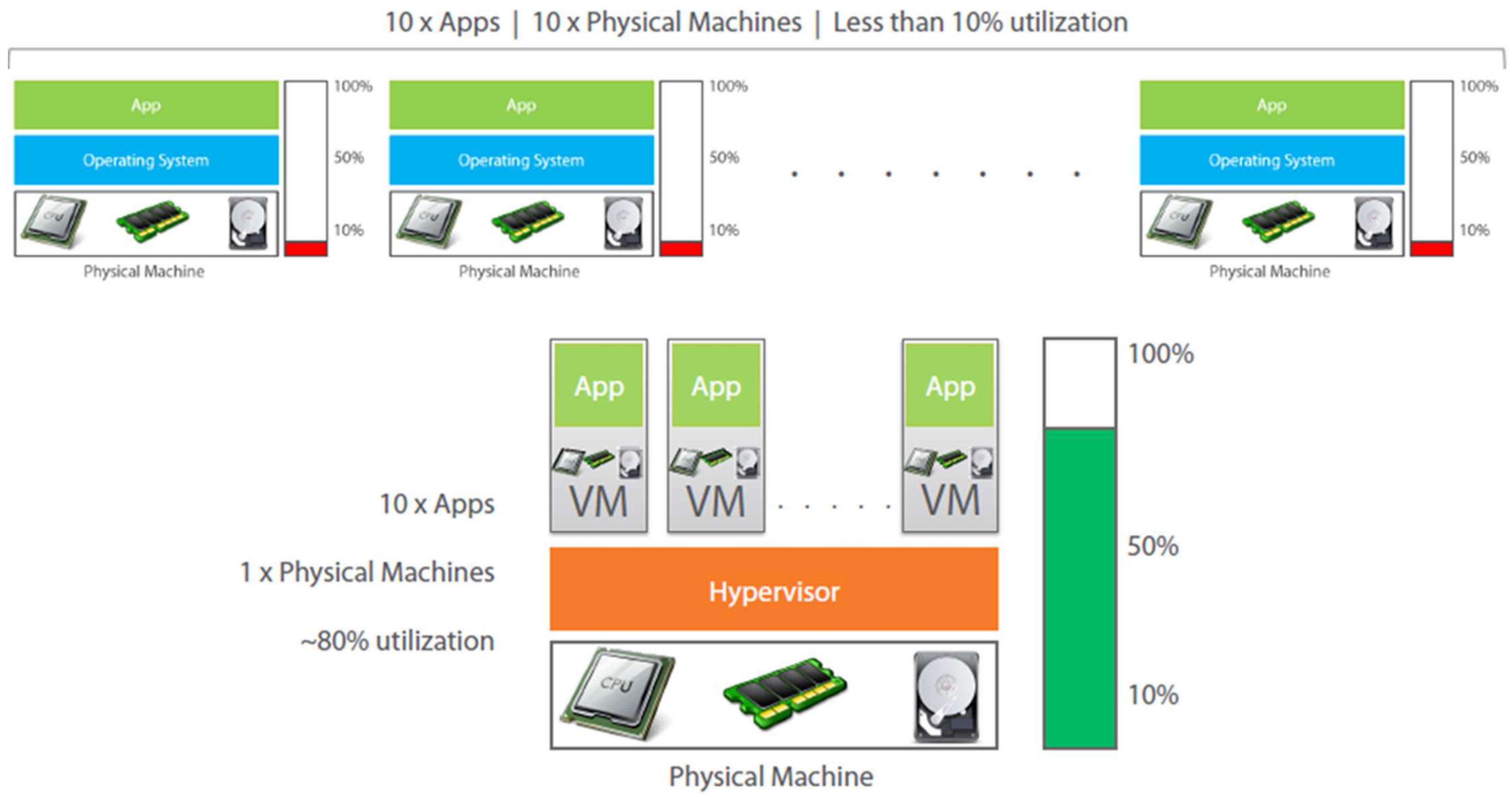
Less Utilization in Traditional Architecture



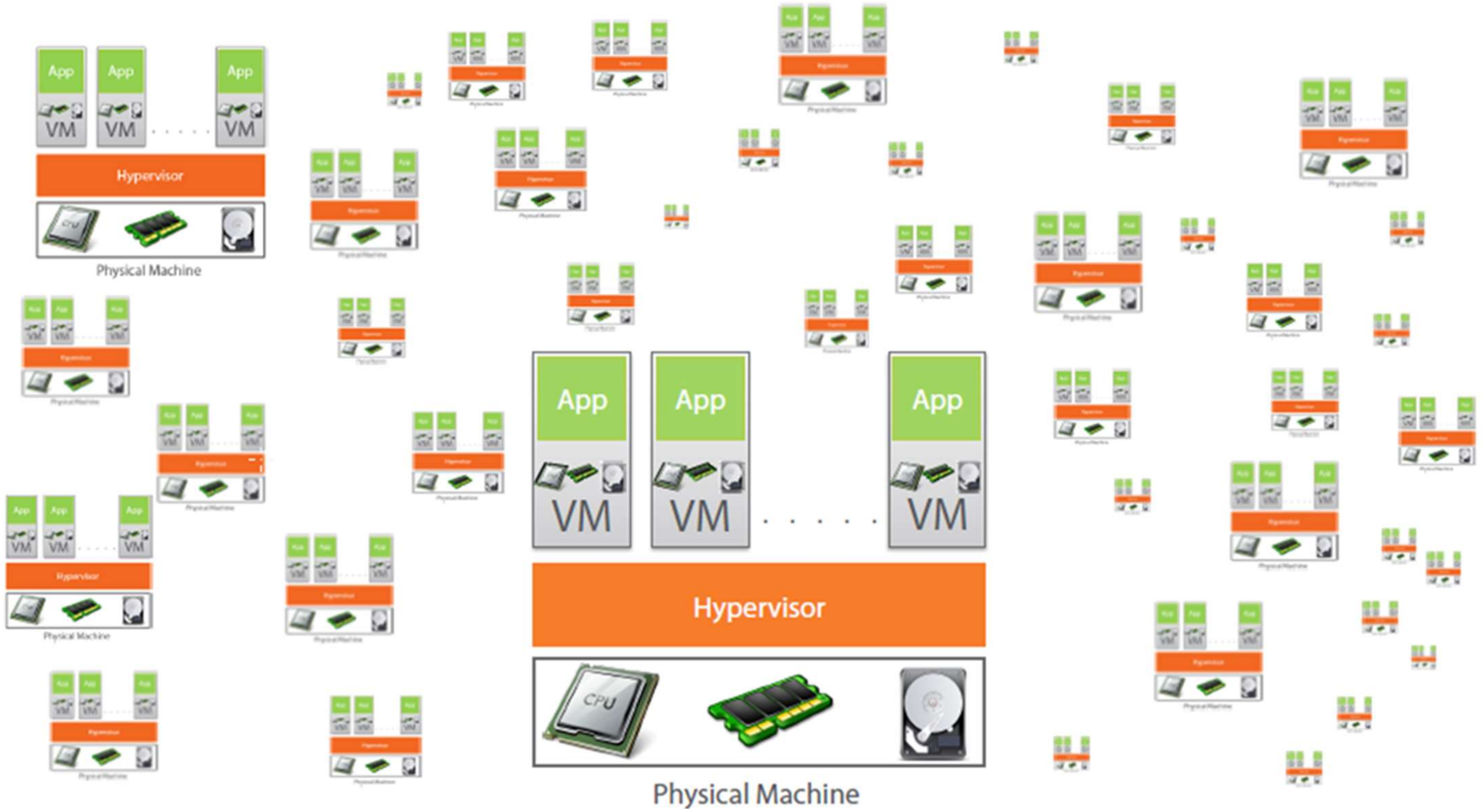
Virtual Machine to the Rescue



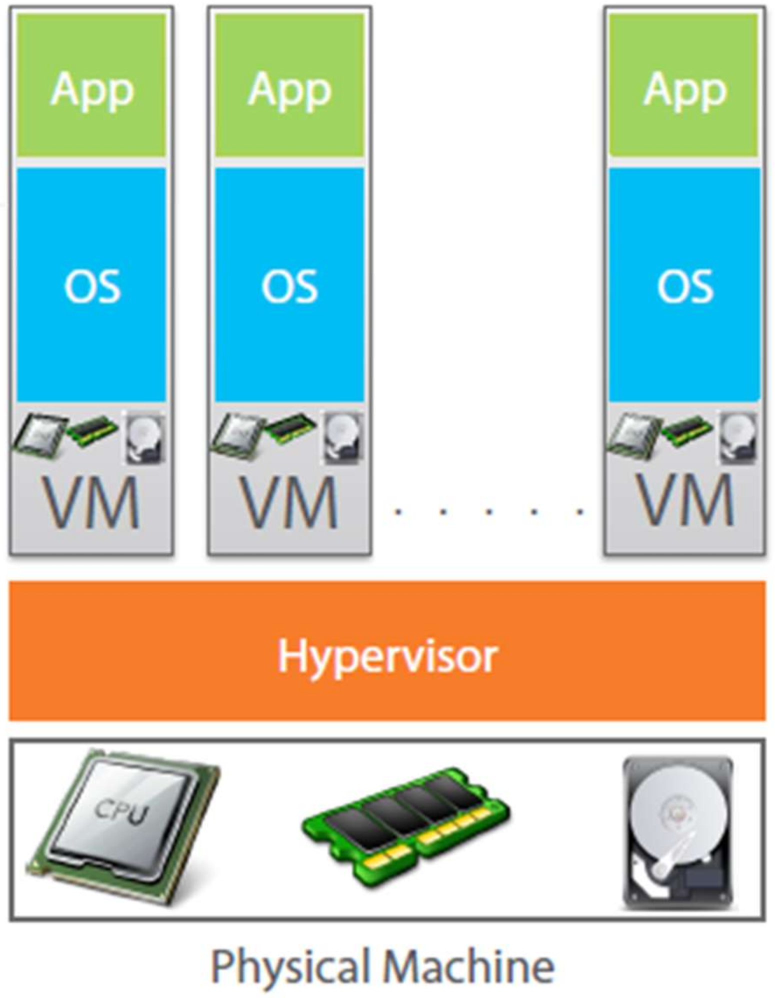
Virtual Machine provides better utilization



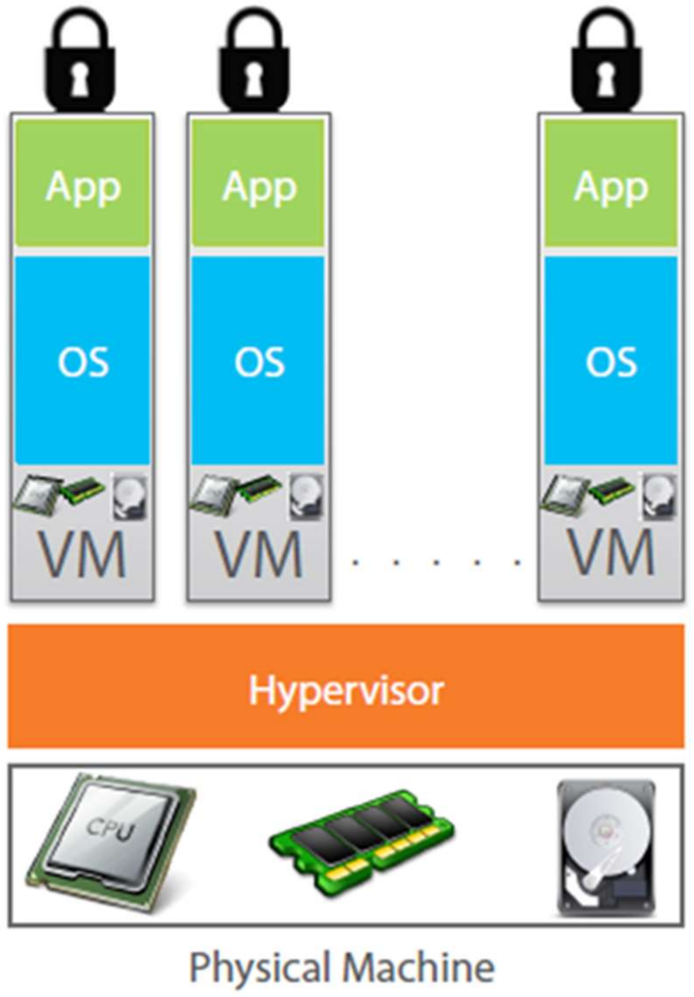
But Virtual Machine increases Licensing Cost



Each VM needs a separate OS

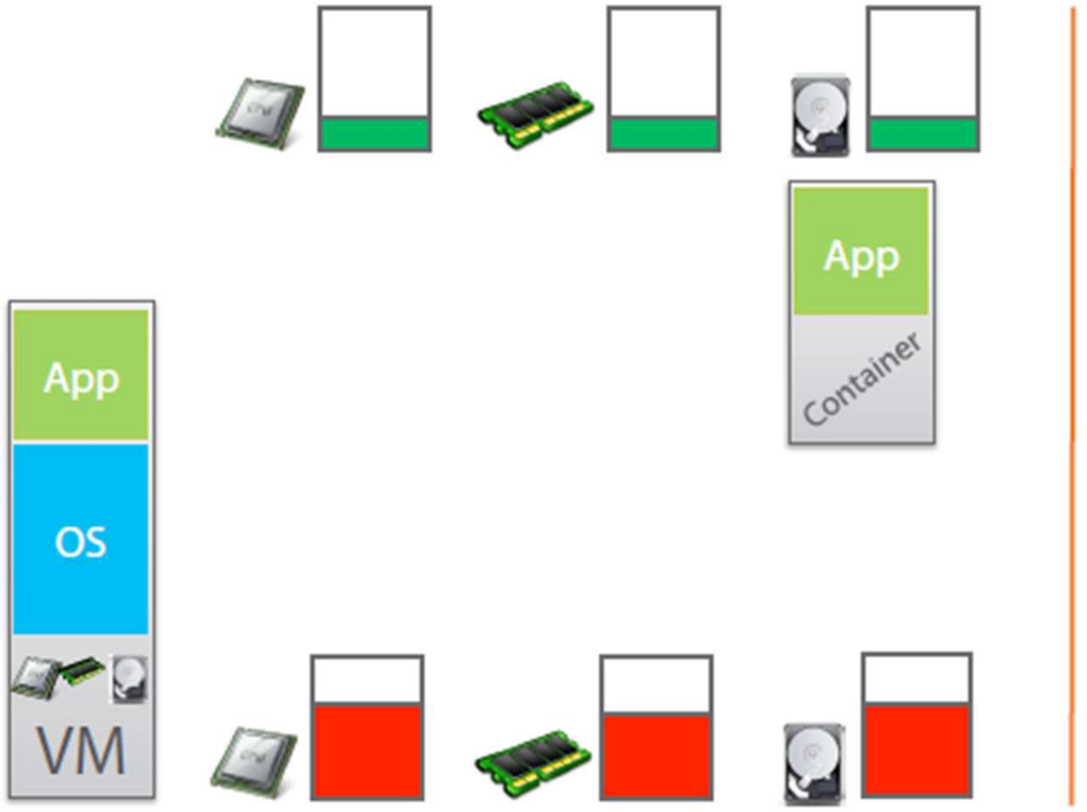


OS takes most of the Resources



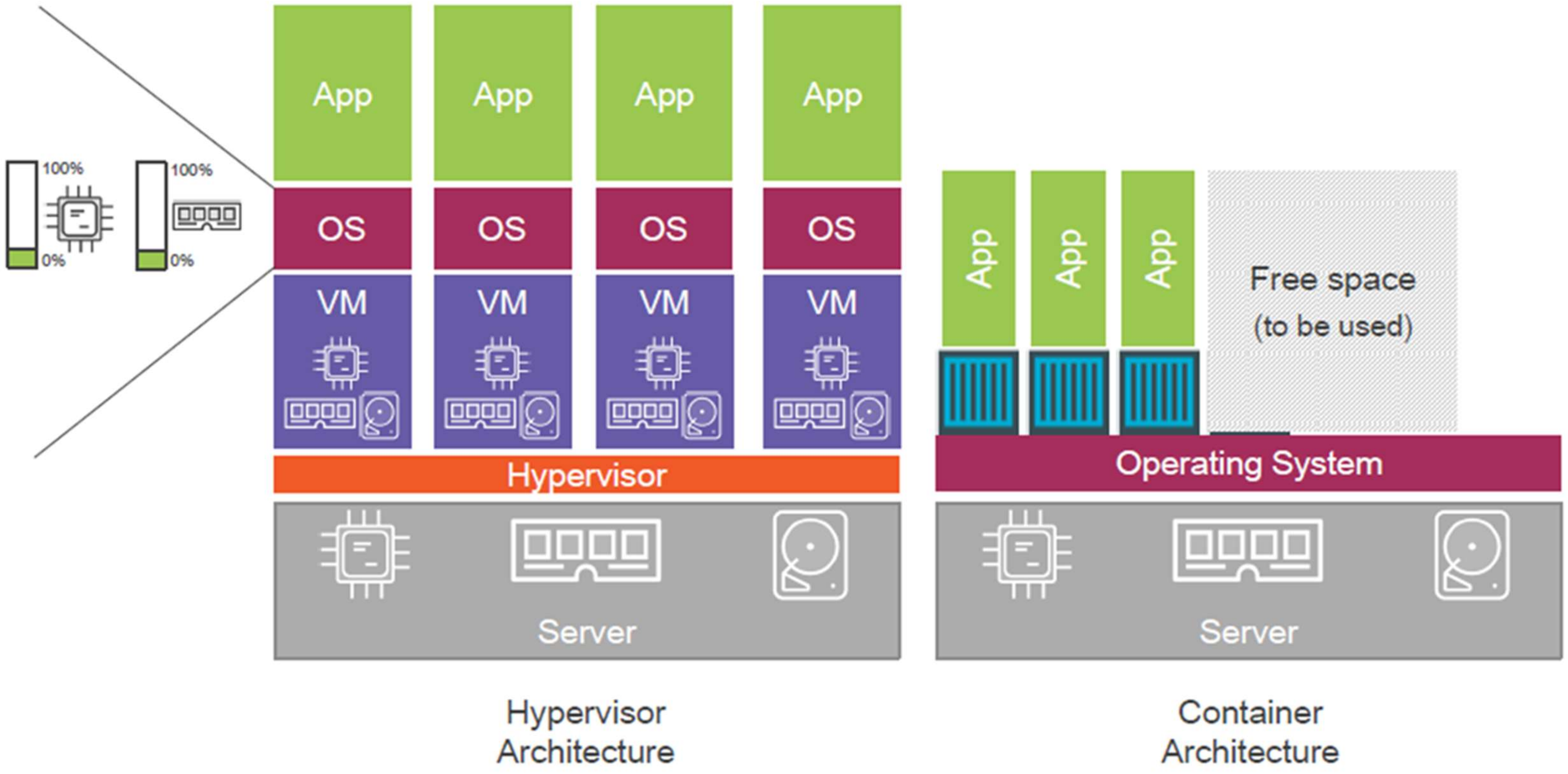
Why use separate OS for each App?

Containers to the Rescue



Containers are more lightweight than Virtual Machines

Containers vs VM



What is Docker?

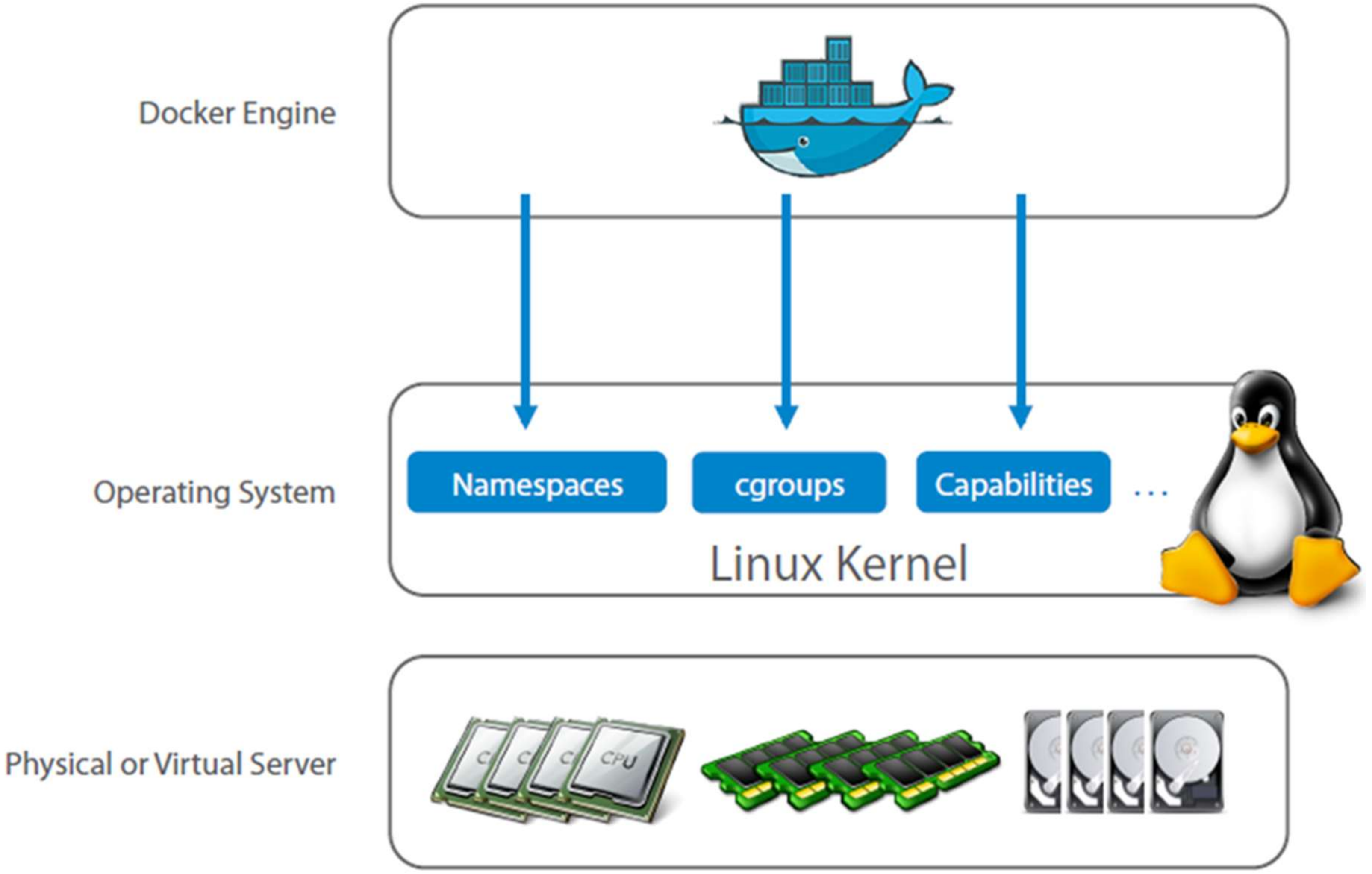
- Docker is an open-source project
  - that automates the deployment of applications inside software containers,
  - by providing an additional layer of abstraction and
  - automation of operating system–level virtualization on Linux.

Practical

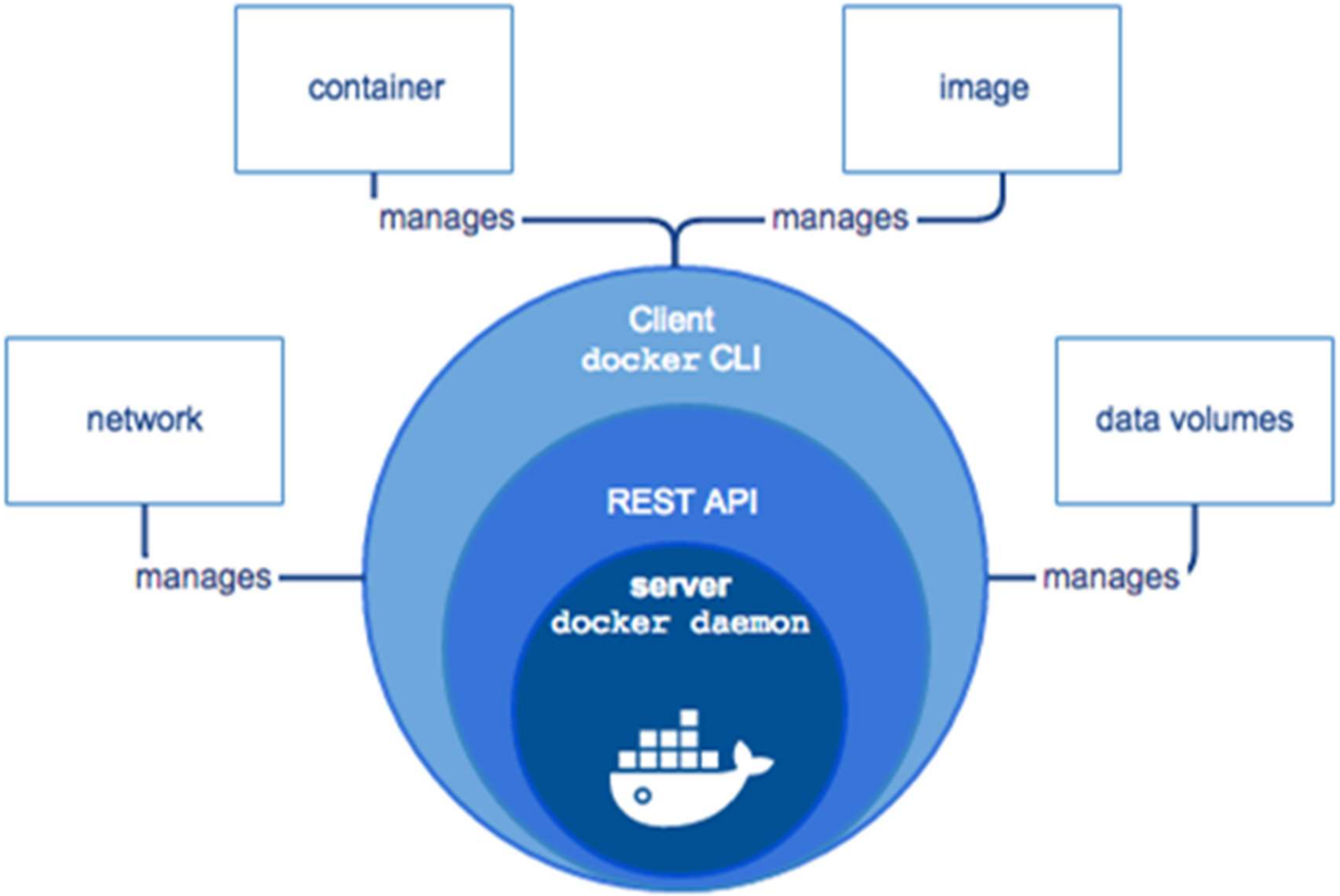
Practical Guide

- Docker Installation on Ubuntu:
  - sudo groupadd docker
  - sudo usermod -aG docker $USER
  - curl -fsSL https://get.docker.com -o get-docker.sh
  - sh get-docker.sh
- Refer to the Practical Guide on:
  - 3-docker.sh

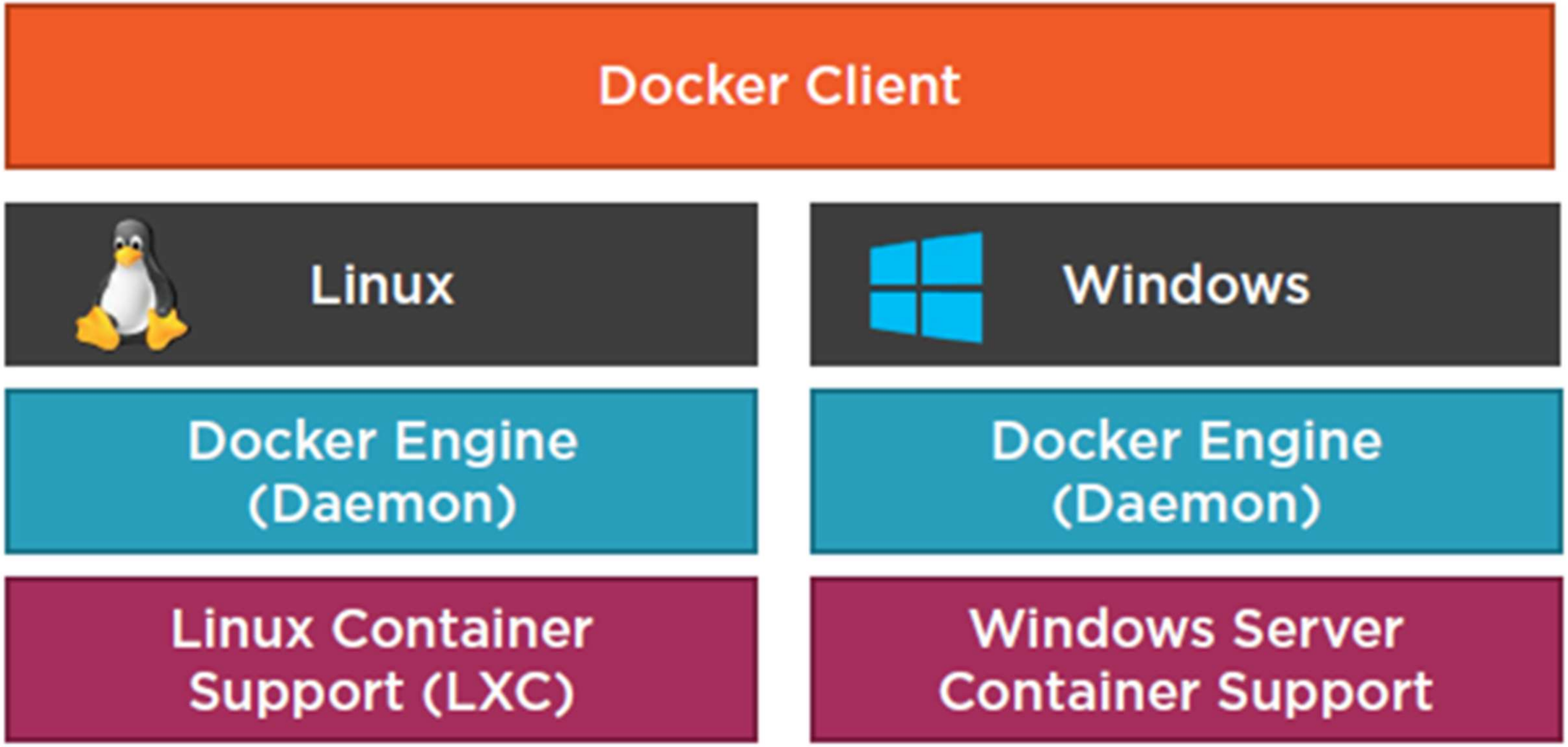
Docker Engine



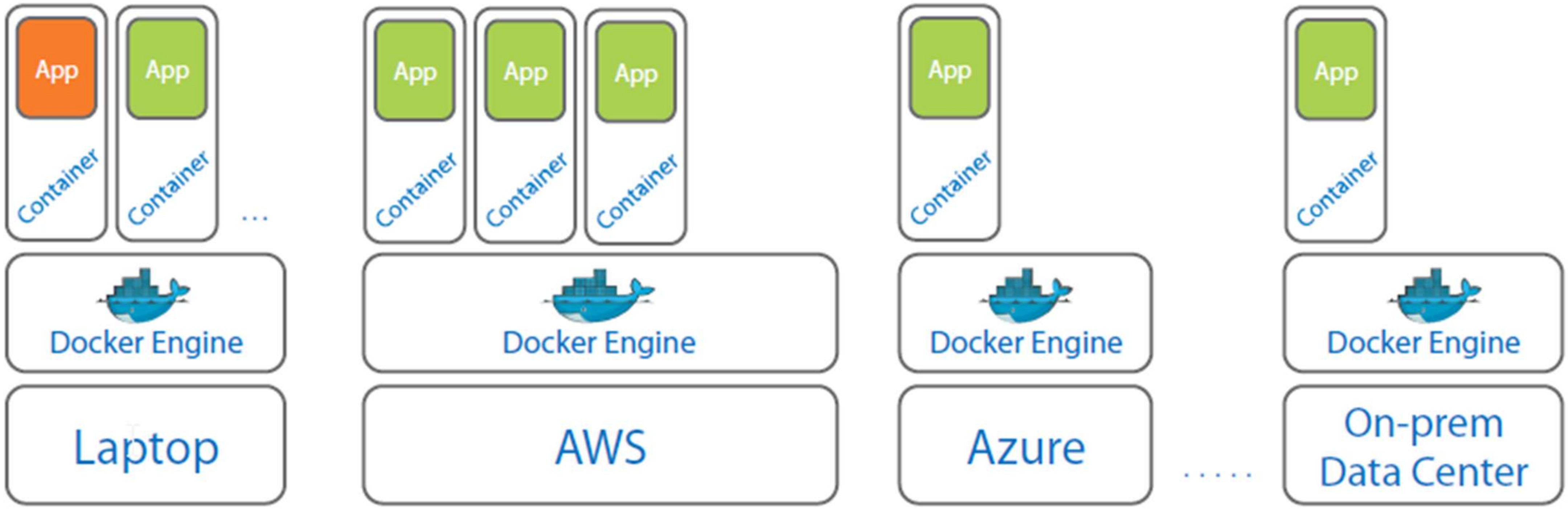
Docker Engine



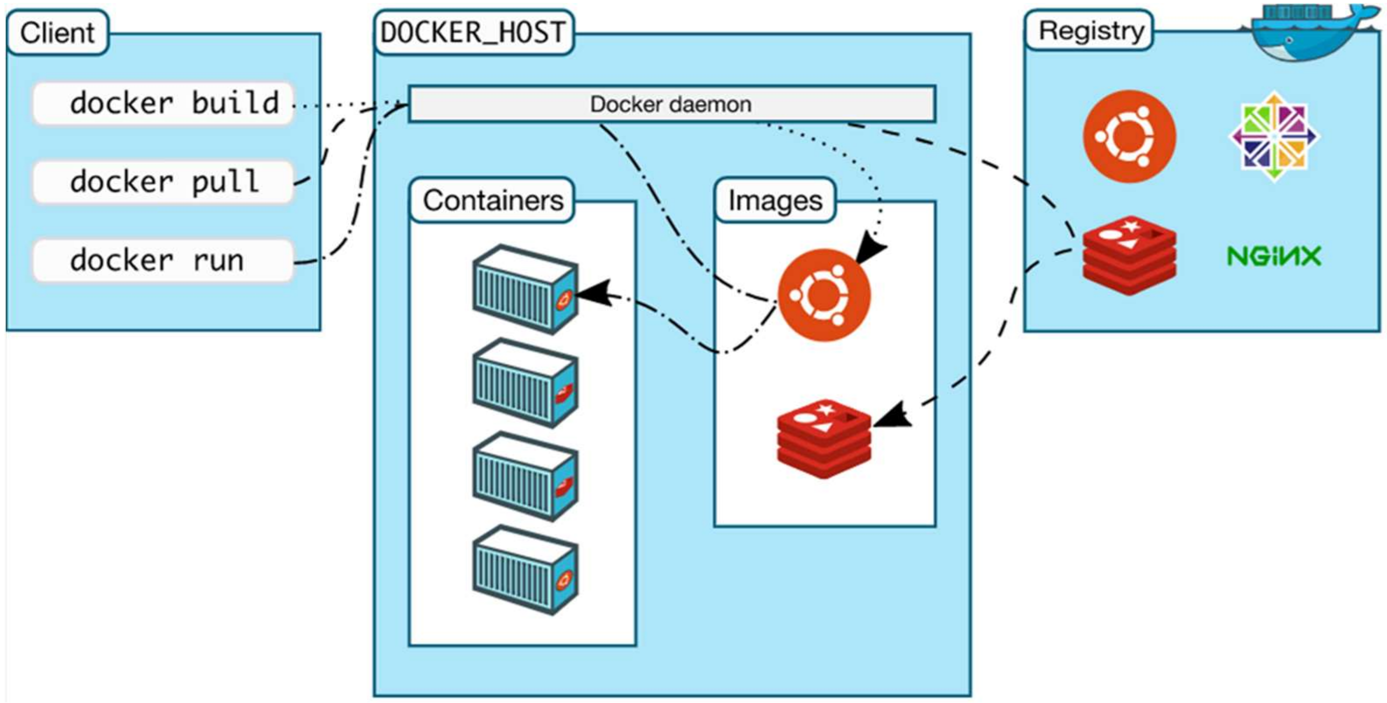
Where does Docker Run?



Docker can run anywhere



Docker Architecture



- Docker uses a client-server architecture.
- Docker client talks to the Docker daemon
- The Docker client and daemon can run on the same system, or can connect a client to a remote Docker daemon.
- The Docker client and daemon communicate using a REST API

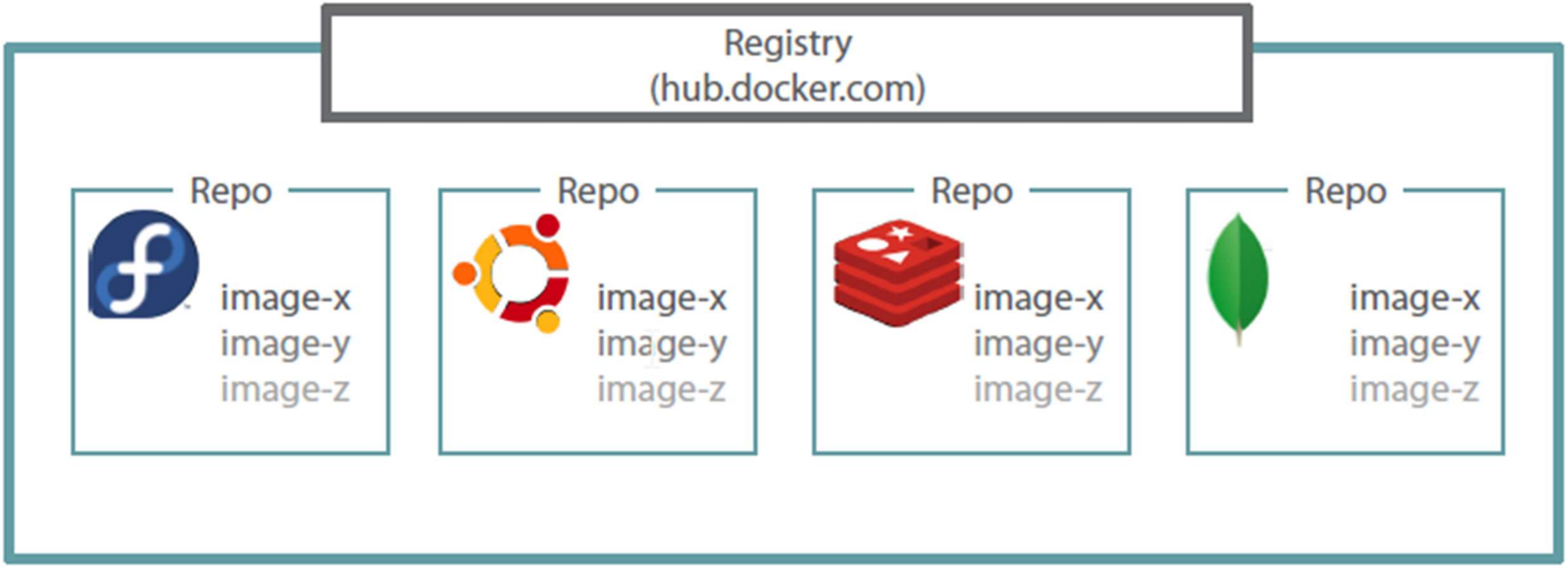
Image

- 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



Hands-On

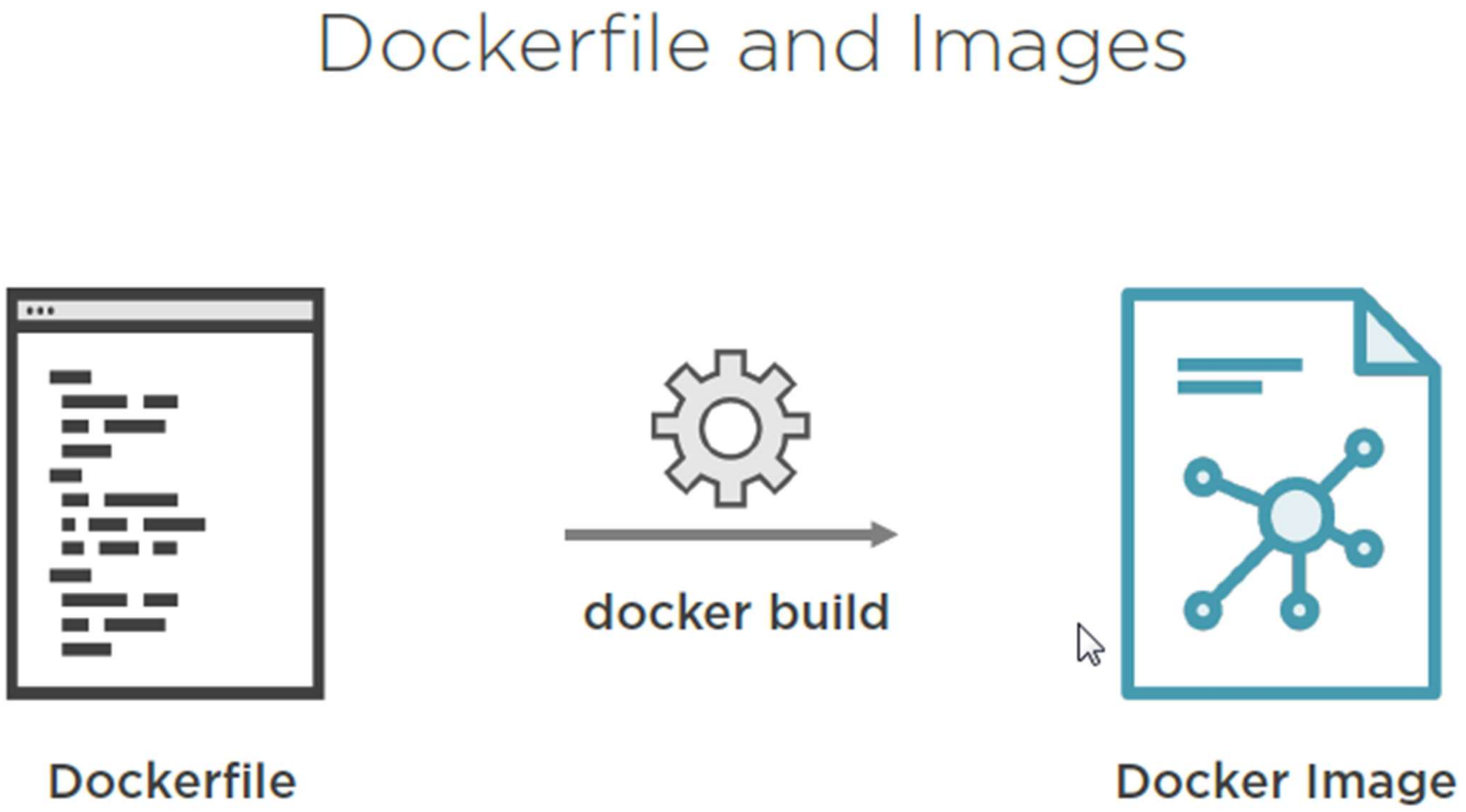
- We need to do the below hands-on:
  - ssh to Ubuntu server
  - Install Docker on Ubuntu
  - Validate docker engine is successfully installed
  - Launch a docker container
  - Login to container
  - Work in a container
  - List containers
  - Delete container
- Refer to “3-docker.sh” in Commands guide for instructions

Container Images and Dockerfile

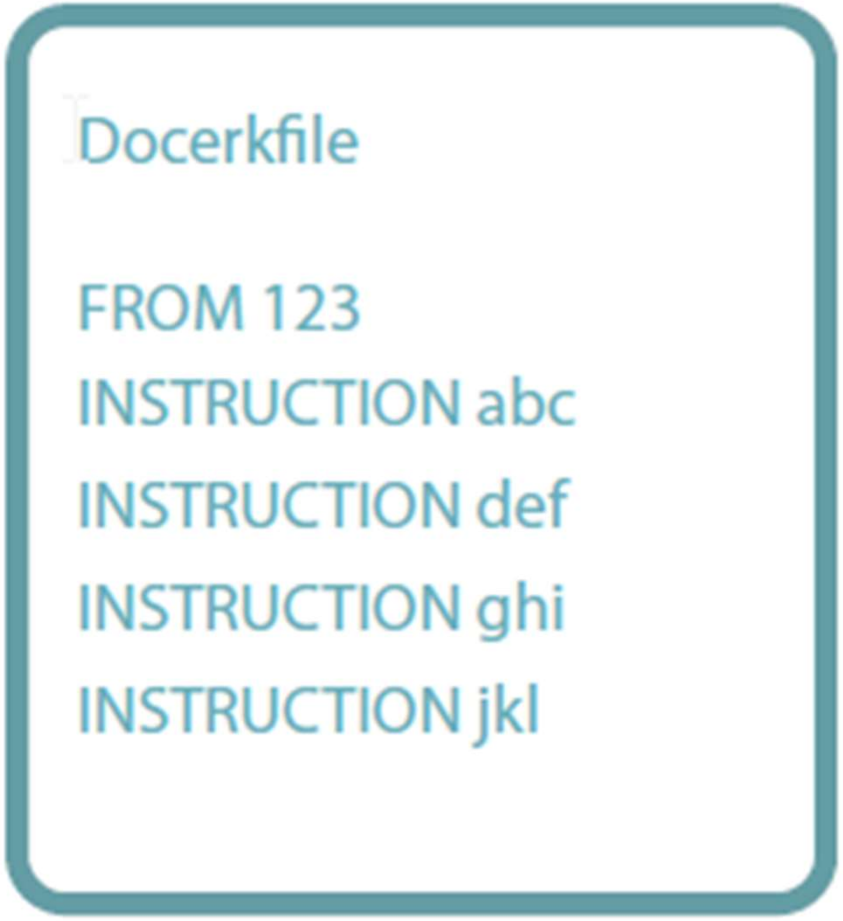
Create Dockerized Application

- 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



Docerkfile

```
FROM 123
INSTRUCTION abc
INSTRUCTION def
INSTRUCTION ghi
INSTRUCTION jkl
```

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/

Clean Up

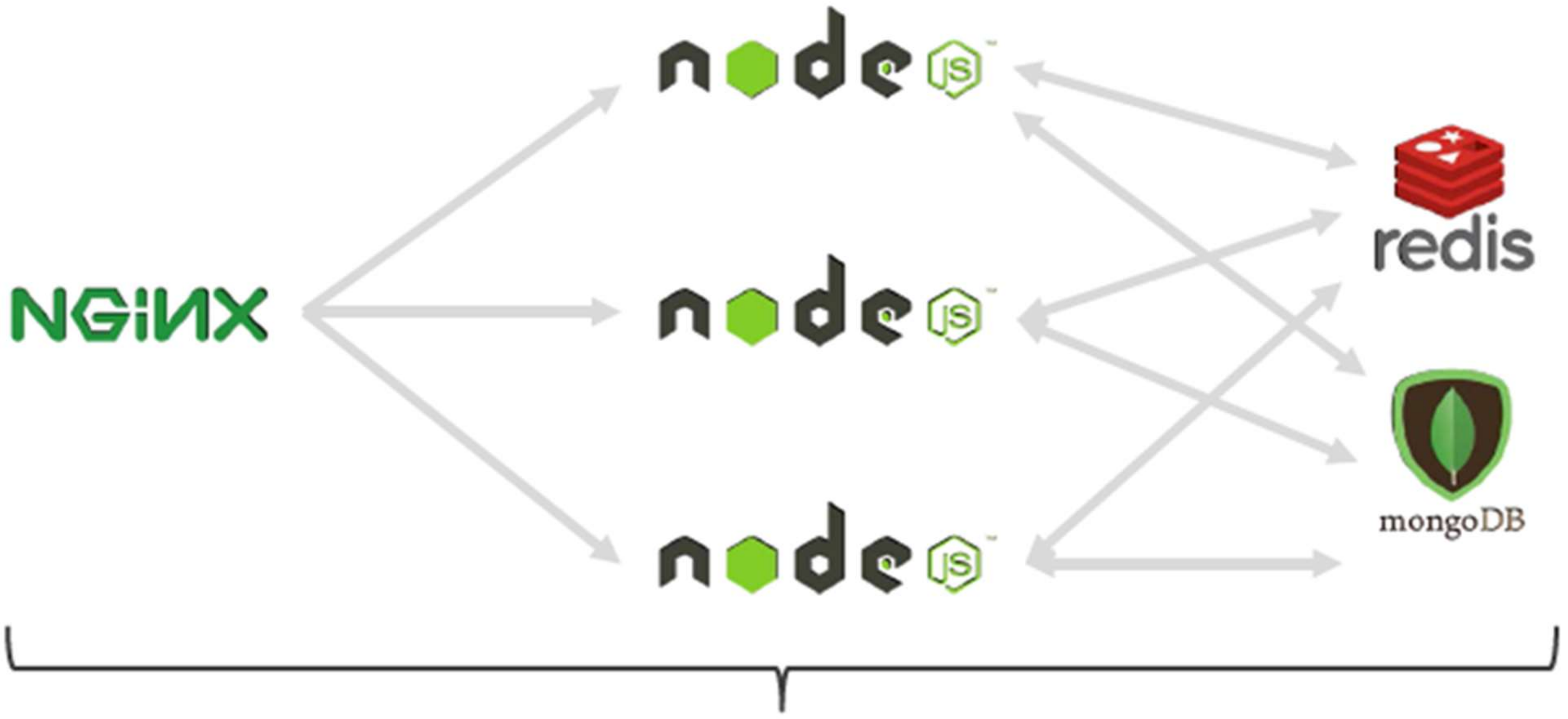
- docker stop $(docker ps -a -q) #stop ALL containers
- docker rm -f $(docker ps -a -q) # remove ALL containers

Docker Compose

Docker Compose

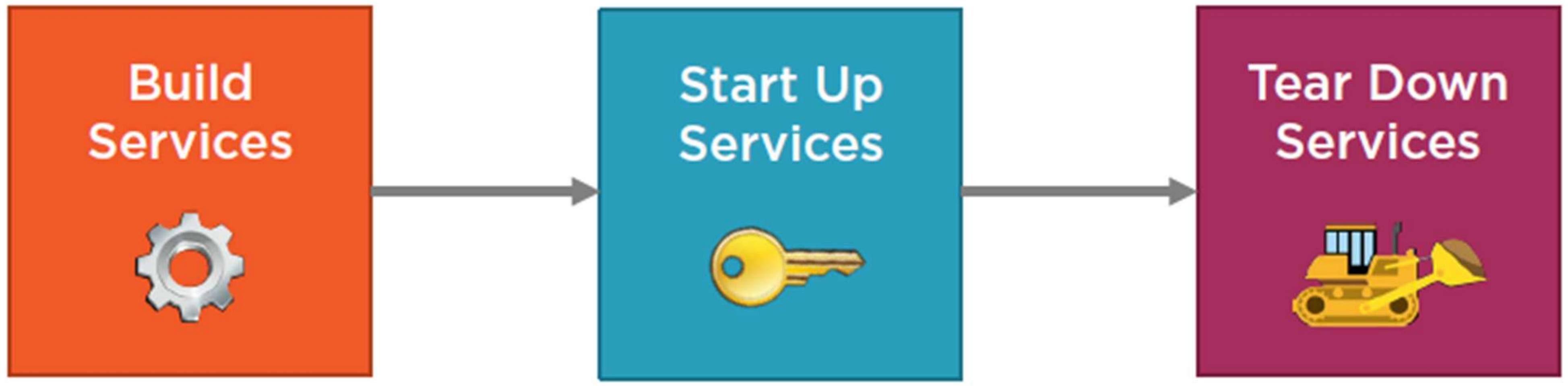
- Manages the whole application lifecycle:
    - Start, stop and rebuild services
    - View the status of running services
    - Stream the log output of running services
    - Run a one-off command on a service

The need for Docker Compose

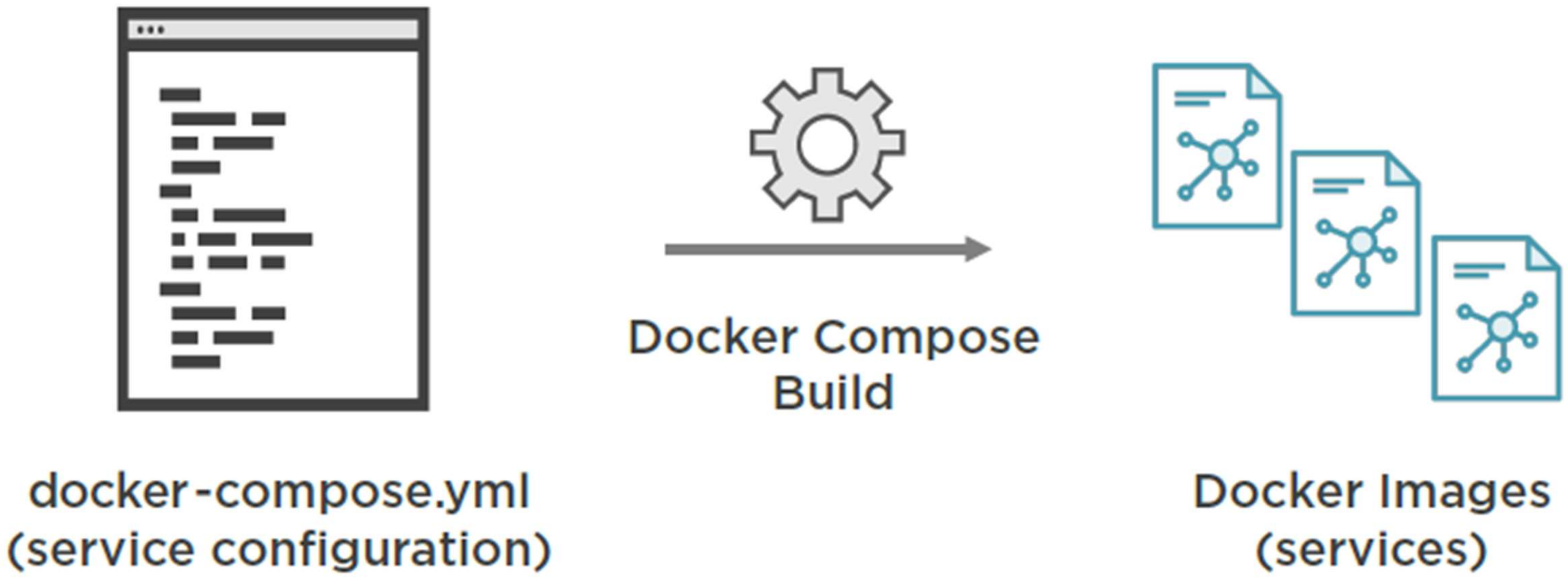


Docker Compose
(docker-compose.yml)

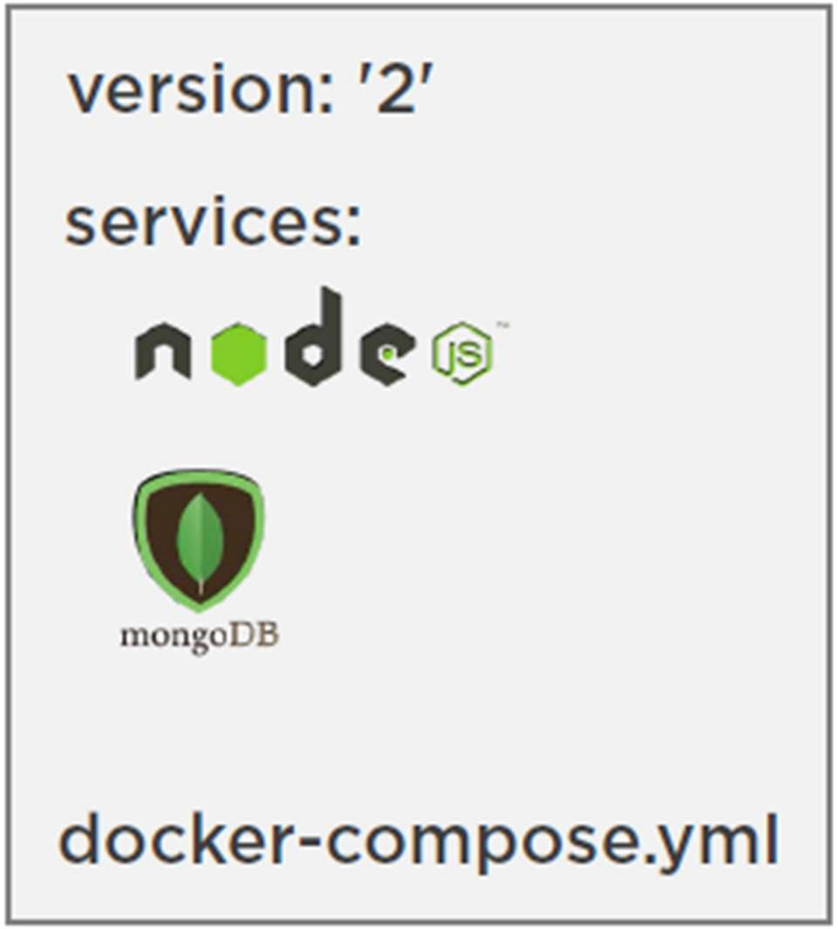
Docker Compose Workflow



The Role of the DockerCompose File



Docker Compose and Services



docker-compose.yml Example

```
•   version: '3'
•   services:
•     web:
•       build: .
•       ports:
•         - "101:5000"
•     redis:
•       image: "redis:alpine"
```

Key Docker Compose Commands

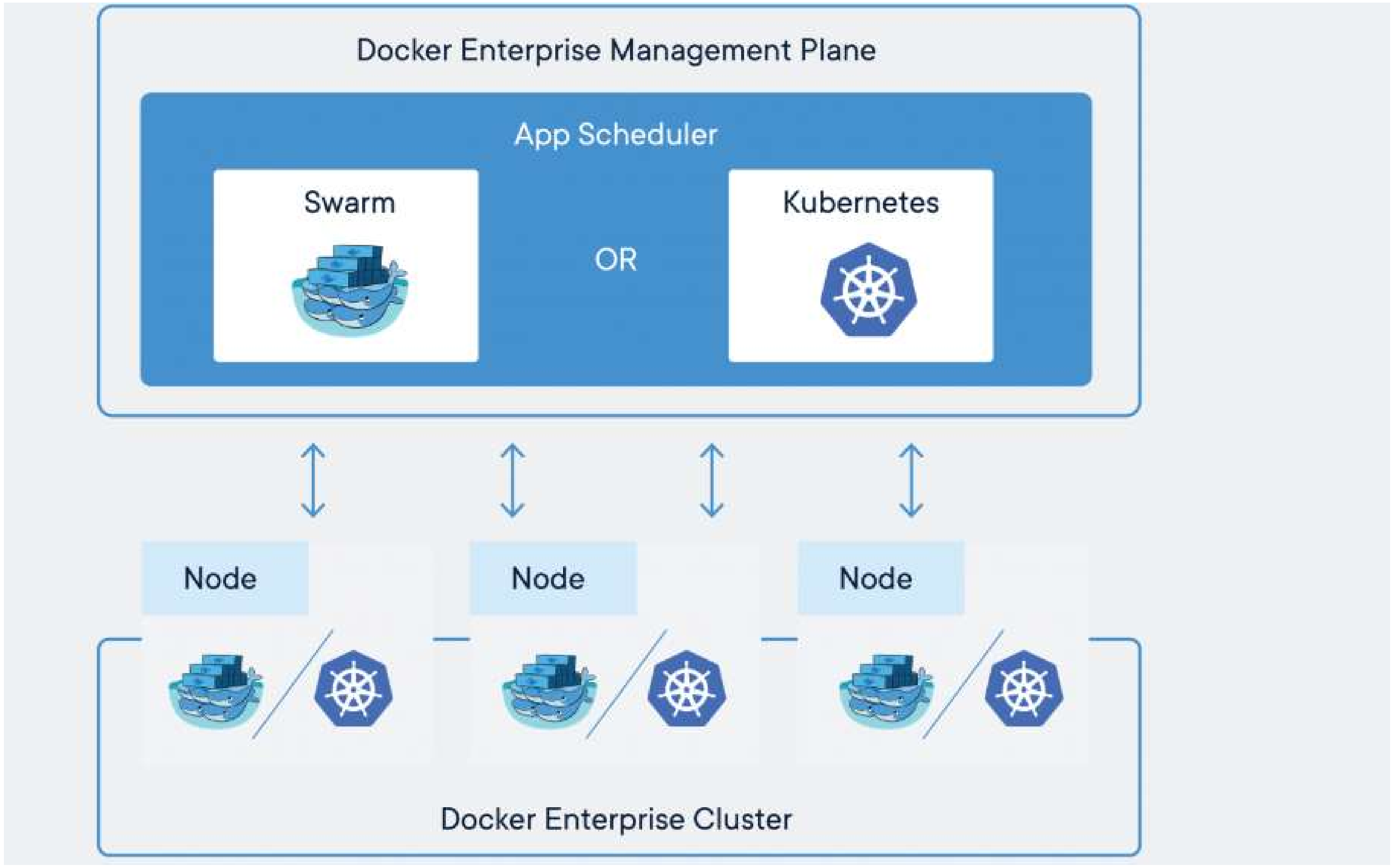
- docker-compose build
- docker-compose up
- docker-compose down
- docker-compose logs
- docker-compose ps
- docker-compose stop
- docker-compose start
- docker-compose rm

Hands-on

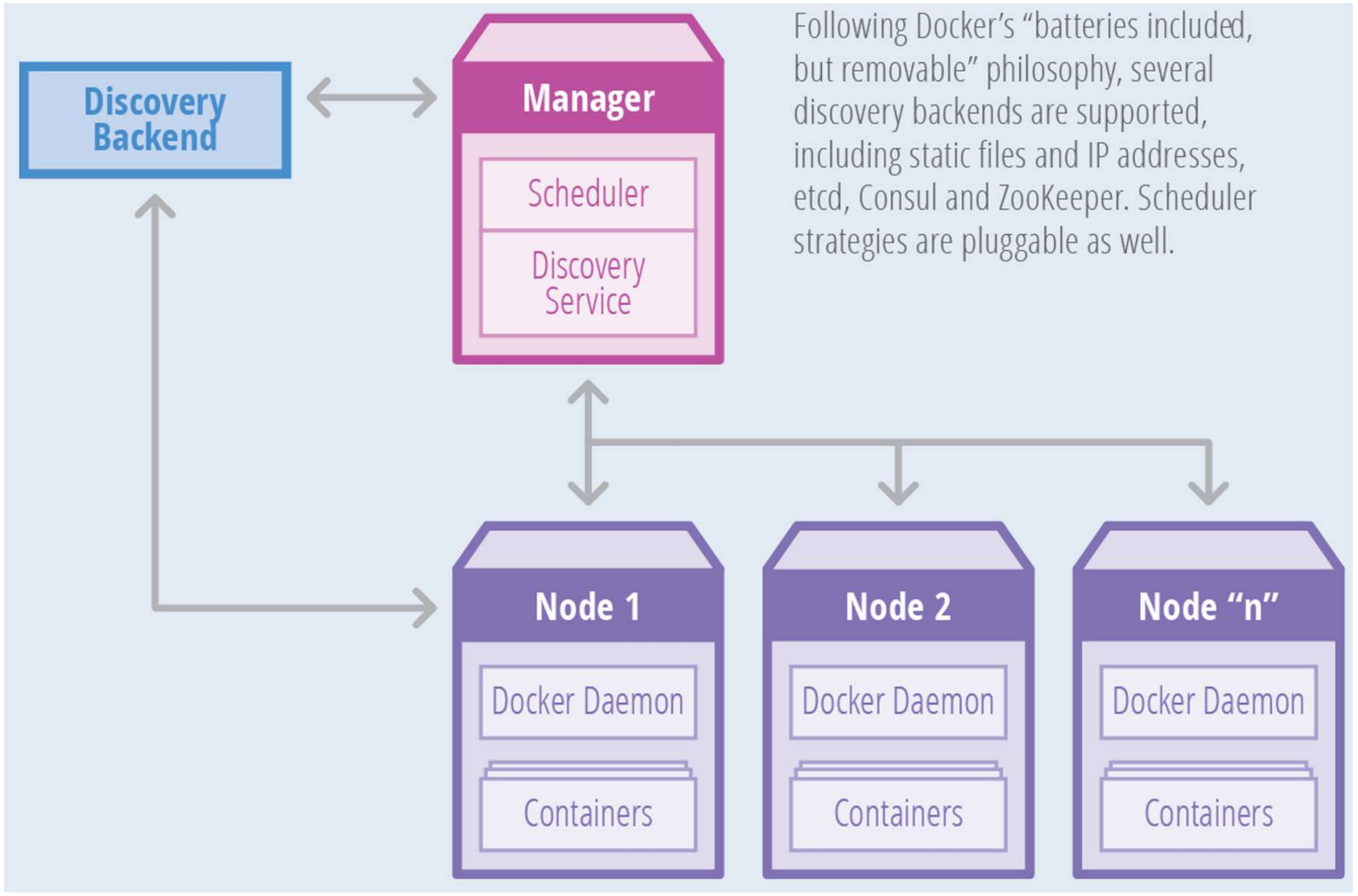
- Refer
    - 4-docker_compose.sh

Docker Swarm

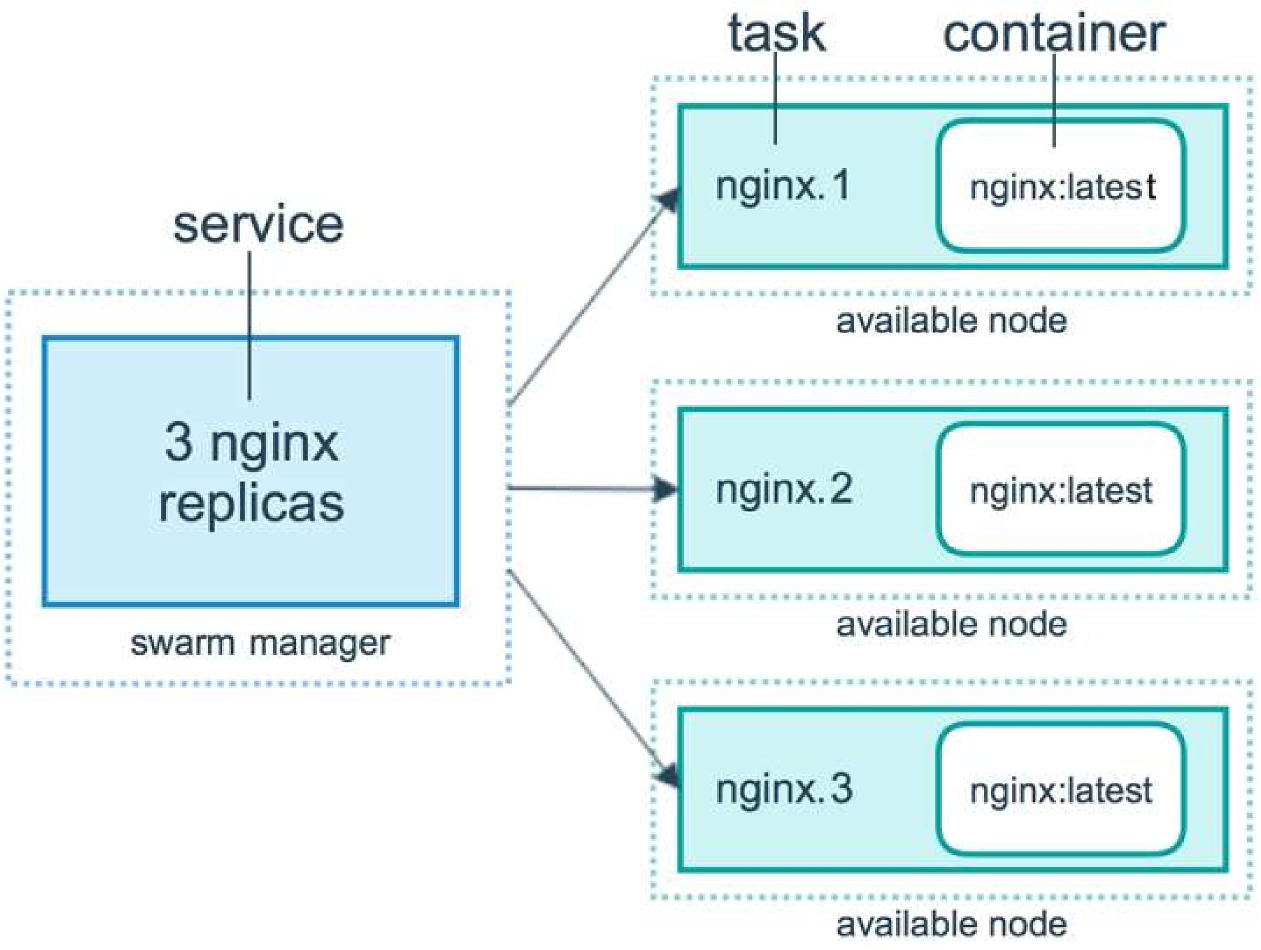
Container Orchestration



Docker Swarm



Following Docker's "batteries included, but removable" philosophy, several discovery backends are supported, including static files and IP addresses, etcd, Consul and ZooKeeper. Scheduler strategies are pluggable as well.


service
3 nginx replicas
swarm manager
task
container
nginx.1
nginx:latest
available node
nginx.2
nginx:latest
available node
nginx.3
nginx:latest
available node

Thanks