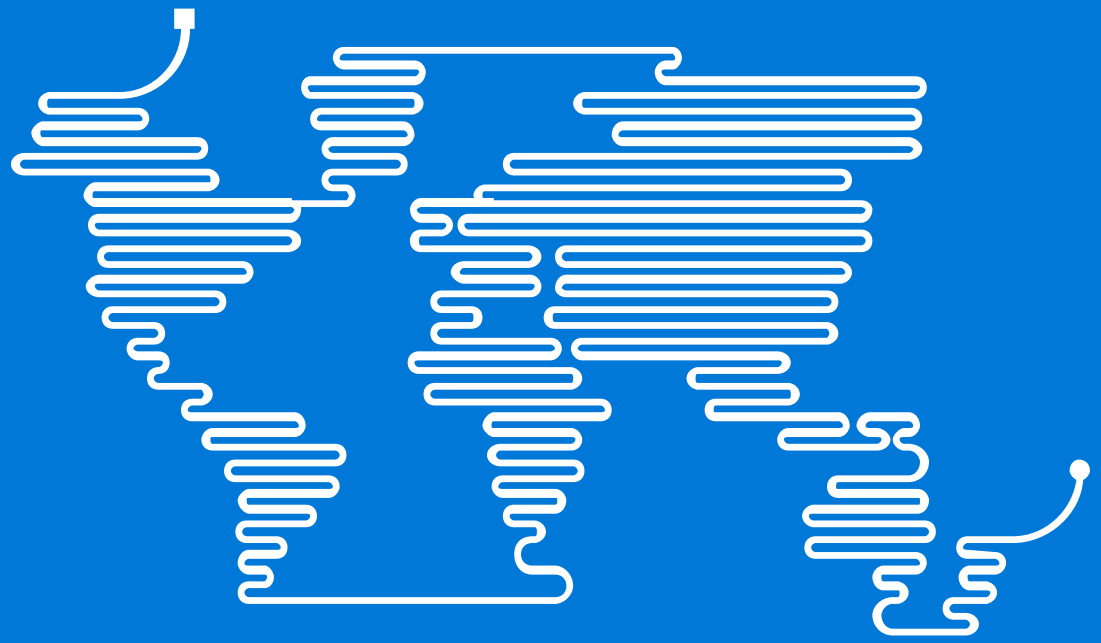
Microsoft

3. Containers Architecture

Juan Osorio
Premier Field Engineer
Apps

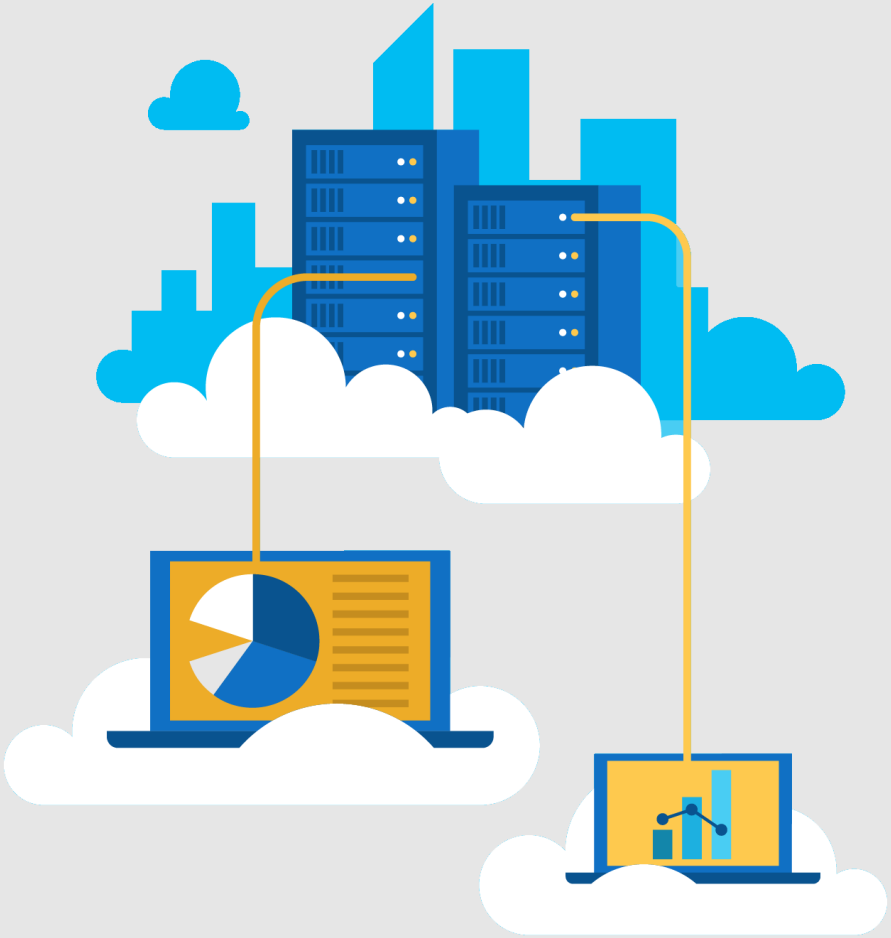


Overview

- Layers
- Process
- Docker file
- Container Registry
- Azure Container Registry

Birth of the Cloud

Virtual Machines



Why modernize?

Reasons to modernize

- **Aging infrastructure**
  - Low efficiency and reliability.
  - High operational costs and capital expenditure.
  - Growing security, audit, and compliance requirements.
  - Inflexible and unable to keep up with business growth.
- **Stagnant architecture**
  - Legacy stack and code.
  - Long deployment times and release cycles.
  - Incompatibilities with modern software systems.
  - It’s hard or impossible to add new functionality.
  - Innovation is happening outside IT, unmanaged.

Modernization benefits

- **Turn CapEx into OpEx**
- **Increased operational efficiency**
  - Get out of the data center business.
  - Meet security and compliance requirements.
  - Reduce time and budget spent on infrastructure management.
- **Rapid innovation**
  - Ship new capabilities faster.
  - Achieve scalability with confidence.
  - Better collaboration across business, Ops, IT and dev teams.

The benefits of using containers

Agility

Ship apps faster

Portability

Easily move workloads

Density

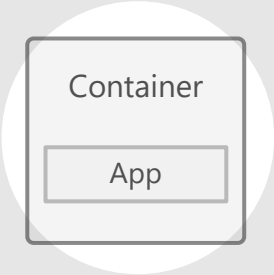
Achieve resource efficiency

Rapid scale

Scale easily to meet demand

From traditional systems to portfolio of modern apps

Existing/New Applications



Containerize Applications

Each host often runs **8** containers at a time[1]

Microservices

Seconds to deploy, churns **12x** faster[1]

Continuous innovation with DevOps

46x more frequent for code delivery[2]

Source:

1: Datadog Report: 8 Surprising Facts About Real Docker Adoption; 2: 2017 state of DevOps Report

Containers: Fast facts

Top 3 facts everyone should know and be ready to share:

STANDARIZATION (Docker)

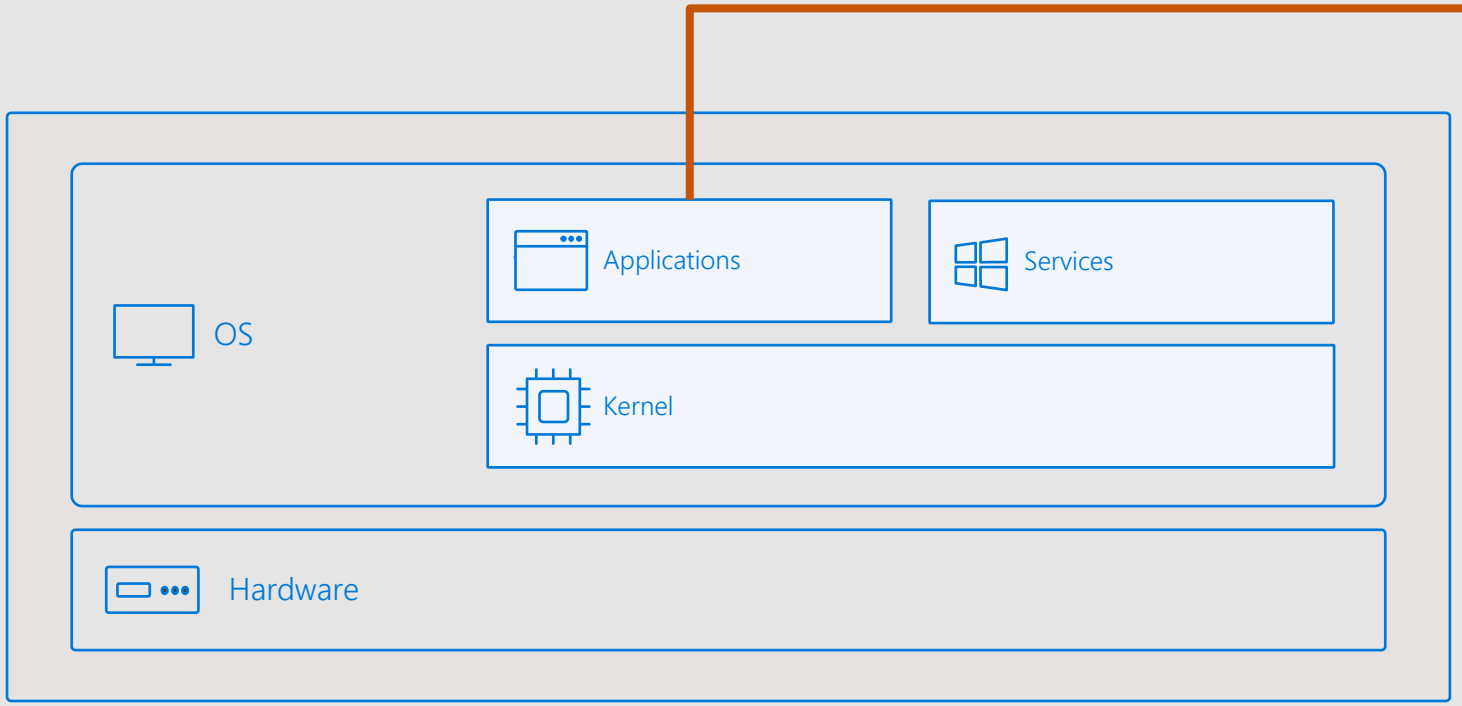
+

PORTABILTY + SPEED + CONFIGURATION

1. Containers allows you to MIMIC a Production Environment during Development, Testing and Pre-production
2. Containers and associated Tooling allows you develop and run solutions on our Laptop
3. Containers considerably reduce the time and cost of Engineering Operations

Architecture

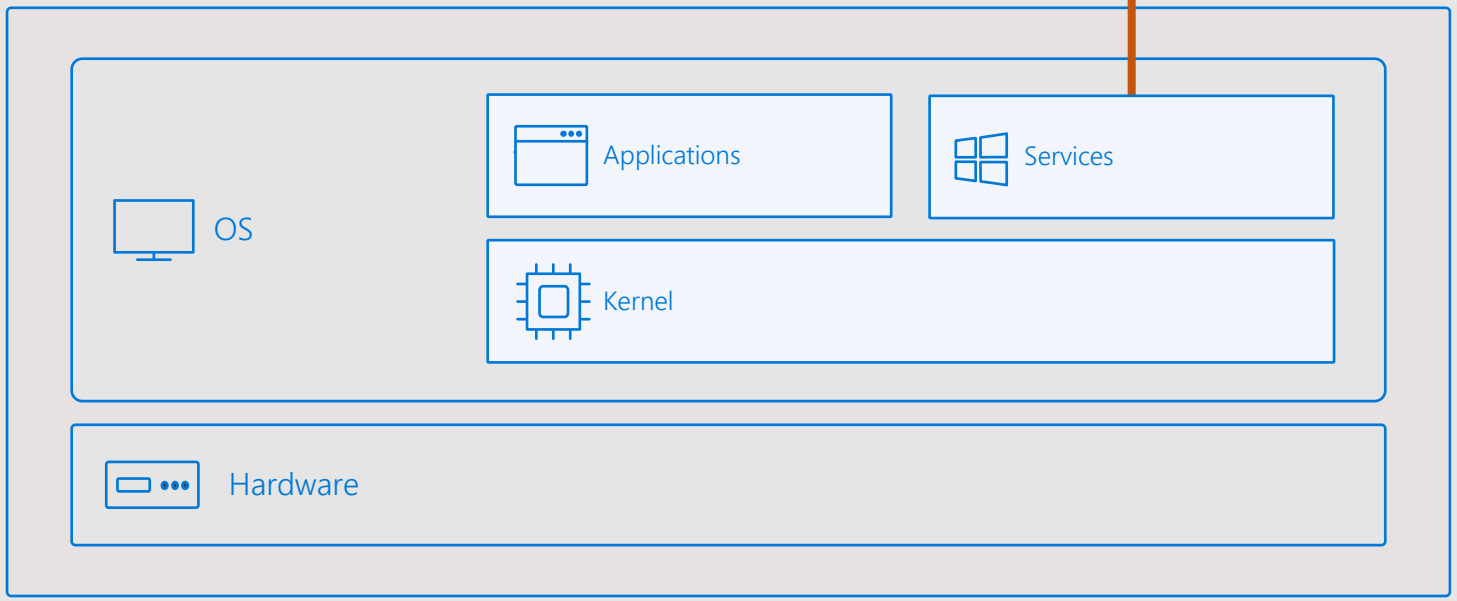
Traditionally



- Your application, could be written in a variety of languages:
  - Python
  - NodeJS
  - .NET full
  - .NET core

Architecture

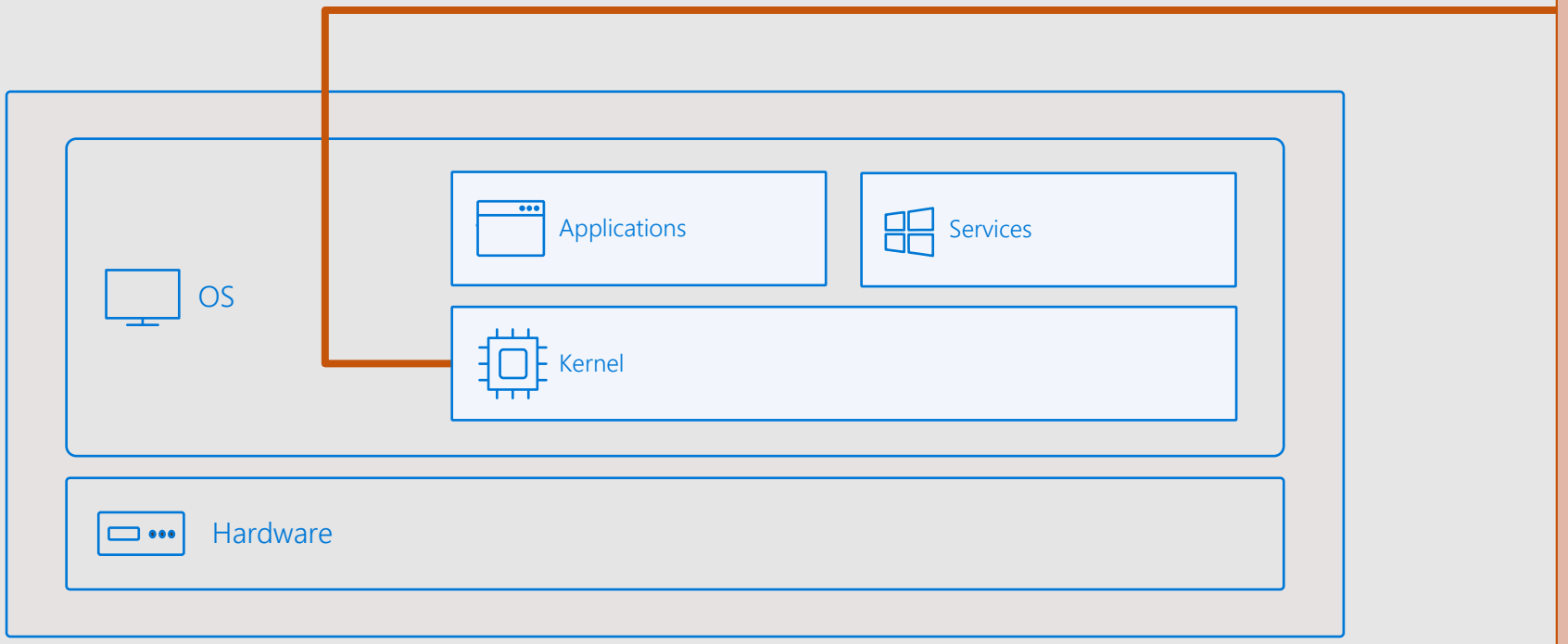
Traditionally



- Long-running processes
- Can provide additional functionality to your app
- Can assist with interacting with the kernel

Architecture

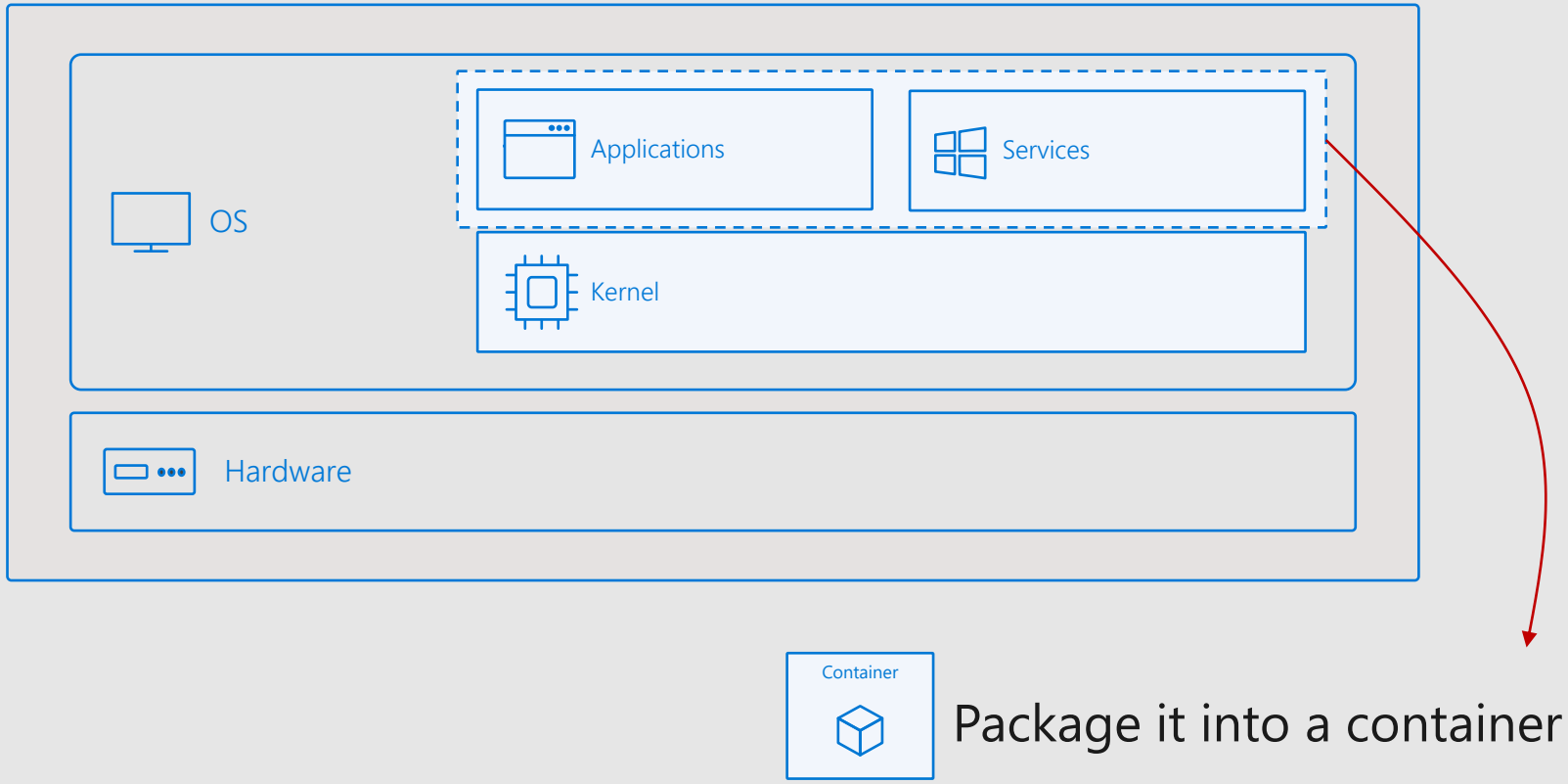
Traditionally



- Provides low-level operations for your app.
- Talks to the hardware on behalf of your app
- Manages system resources
- Drivers

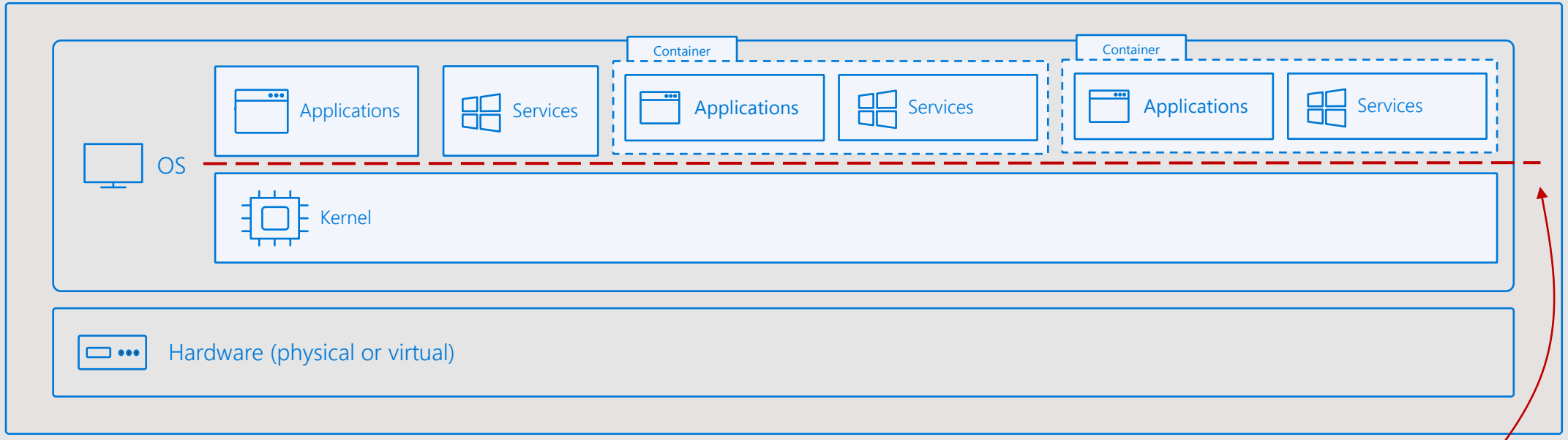
Architecture

Traditionally -> Containers



Architecture

With Containers

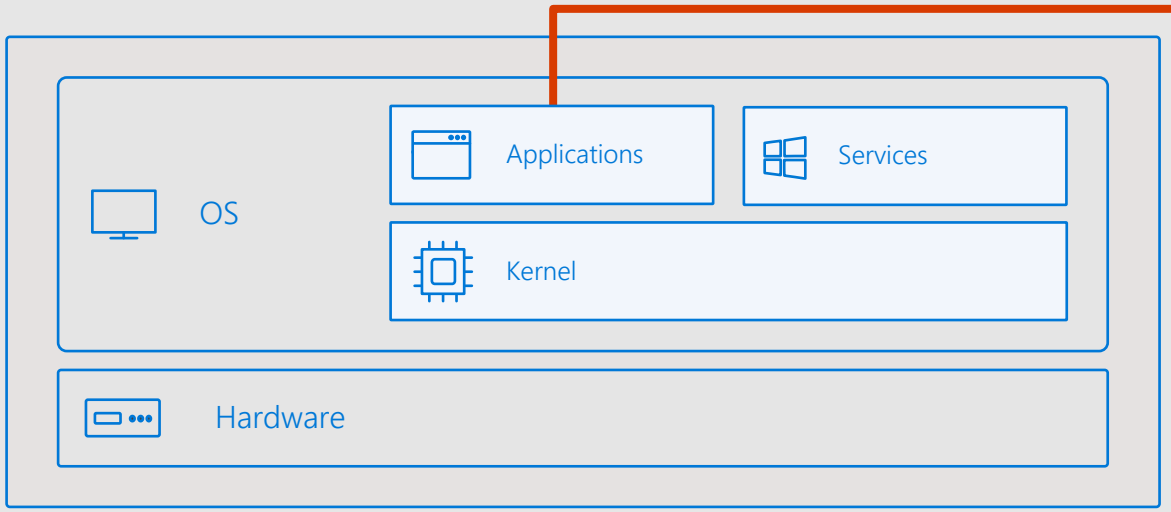


This is our virtualization boundary.

Let’s Visualize the Architecture with tasklist

Architecture

Example: Application




```
top - 04:22:17 up  4:38,  0 users,  load average: 0.12, 0.16, 0.15
Tasks:   2 total,   1 running,   1 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.2 us,  0.5 sy,  0.0 ni, 98.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  8155284 total,  5018192 free,   661224 used,  2475868 buff/cache
KiB Swap:        0 total,        0 free,        0 used.  7162536 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
     1 root      20   0   18124   2008   1516 S   0.0  0.0   0:00.13 bash
   253 root      20   0   41020   1796   1332 R   0.0  0.0   0:00.00 top
```

top command at linux bash prompt

ps –ef at host bash prompt command showing **top** process

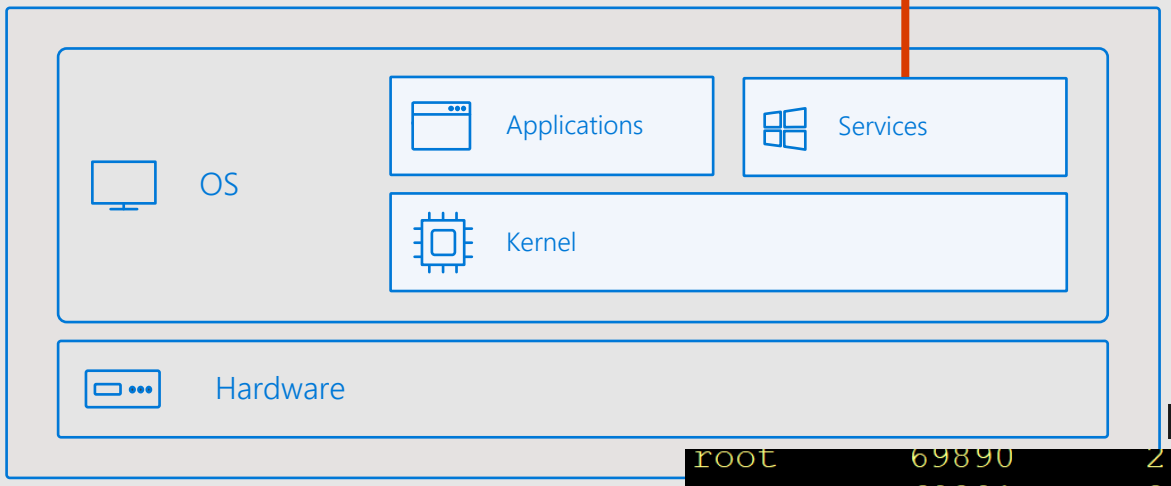
```
root      69892     2  0 04:46 ?        00:00:00 [kworker/1:16]
root      69893     2  0 04:46 ?        00:00:00 [kworker/1:17]
root      69894     2  0 04:46 ?        00:00:00 [kworker/1:18]
root      69895     2  0 04:46 ?        00:00:00 [kworker/1:19]
root      69896     2  0 04:46 ?        00:00:00 [kworker/1:20]
sujit     69951 64558  1 04:46 pts/1    00:00:00 top
root      69963     1  1 04:46 ?        00:00:00 /usr/libexec/fprintd
root      69964  2070  0 04:46 ?        00:00:00 /usr/lib/systemd/systemd-udevd
root      69978 46385  0 04:46 pts/0    00:00:00 ps -ef
```

Architecture

Example: Services

```
[root@labs api]# curl http://localhost:80
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
    <head>
        <title>Test Page for the Nginx HTTP Server on Fedora</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <style type="text/css">
            /*<![CDATA[*/
            body {
                background-color: #fff;
                color: #000;
                font-size: 0.9em;
                font-family: sans-serif,helvetica;
```



nginx started on the host Linux

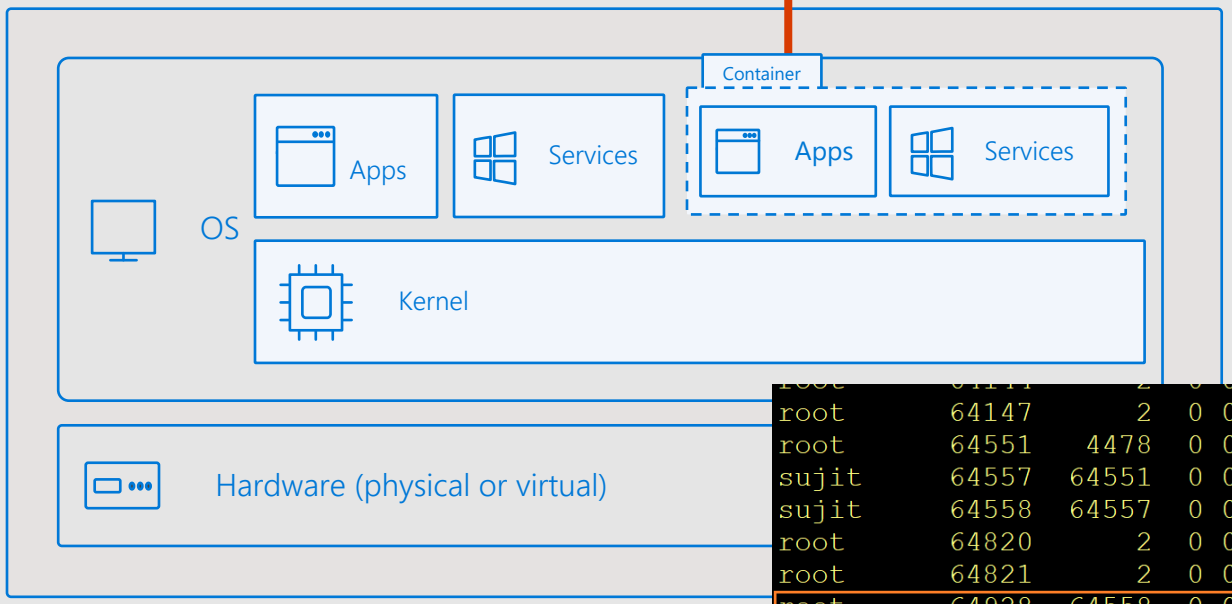
ps –ef at host bash prompt command showing **nginx** process

```
root      69890     2  0 04:46 ?        00:00:00 [kworker/1:14]
root      69891     2  0 04:46 ?        00:00:00 [kworker/1:15]
root      69892     2  0 04:46 ?        00:00:00 [kworker/1:16]
root      69893     2  0 04:46 ?        00:00:00 [kworker/1:17]
root      69894     2  0 04:46 ?        00:00:00 [kworker/1:18]
root      69895     2  0 04:46 ?        00:00:00 [kworker/1:19]
root      69896     2  0 04:46 ?        00:00:00 [kworker/1:20]
sujit     69951 64558  0 04:46 pts/1    00:00:00 top
root      70249     1  0 04:49 ?        00:00:00 nginx: master process nginx
nginx     70250 70249  0 04:49 ?        00:00:00 nginx: worker process
nginx     70251 70249  0 04:49 ?        00:00:00 nginx: worker process
root      70267     1  0 04:49 ?        00:00:00 /usr/libexec/fprintd
```

Same walkthrough—now with Containers

Containers

Example: Application




```
top - 04:22:17 up  4:38,  0 users,  load average: 0.12, 0.16, 0.15
Tasks:   2 total,   1 running,   1 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.2 us,  0.5 sy,  0.0 ni, 98.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  8155284 total,  5018192 free,   661224 used,  2475868 buff/cache
KiB Swap:        0 total,        0 free,        0 used.  7162536 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
     1 root      20   0   18124   2008   1516 S   0.0  0.0   0:00.13 bash
   253 root      20   0   41020   1796   1332 R   0.0  0.0   0:00.00 top
```

top command at container bash prompt

```
root      64147      2  0 04:17 ?        00:00:00 [kworker/0:16]
root      64551   4478  0 04:19 ?        00:00:00 sshd: sujit [priv]
sujit     64557  64551  0 04:19 ?        00:00:00 sshd: sujit@pts/1
sujit     64558  64557  0 04:19 pts/1    00:00:00 -bash
root      64820      2  0 04:20 ?        00:00:00 [kworker/1:19]
root      64821      2  0 04:20 ?        00:00:00 [kworker/1:20]
root      64928  64558  0 04:20 pts/1    00:00:00 sudo docker run -it --rm nginx bash
root      64930  64928  0 04:20 pts/1    00:00:00 /usr/bin/docker-current run -it --rm nginx bash
root      64955   5396  0 04:20 ?        00:00:00 /usr/bin/docker-containerd-shim-current 628cf55d7e5062
root      64970  64955  0 04:20 pts/2    00:00:00 bash
root      65489  64970  0 04:22 pts/2    00:00:00 top
root      66071      2  0 04:25 ?        00:00:00 [kworker/1:0]
root      66145      2  0 04:26 ?        00:00:00 [kworker/0:0]
root      66861      2  0 04:30 ?        00:00:00 [kworker/u256:2]
root      66956      2  0 04:30 ?        00:00:00 [kworker/1:1]
root      67244      2  0 04:32 ?        00:00:00 [kworker/0:1]
root      67370      1  0 04:33 ?        00:00:00 /usr/libexec/fprintd
root      67446  46385  0 04:33 pts/0    00:00:00 ps -fe
```

Docker and container processes

Host system Linux processes

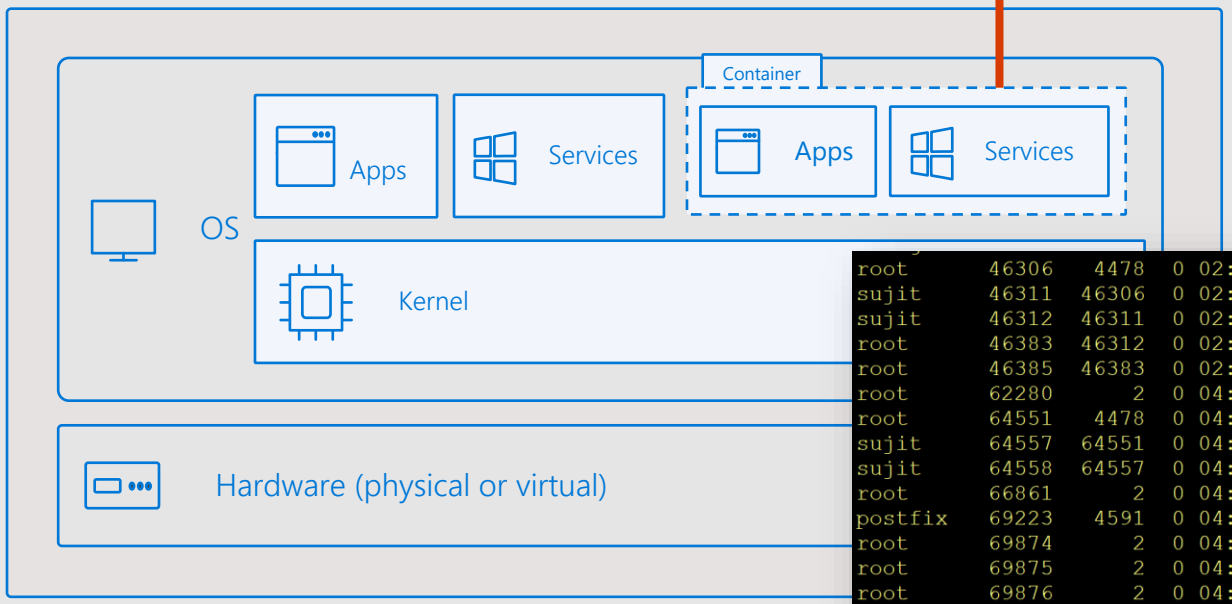
Containers

Example: Services

nginx container

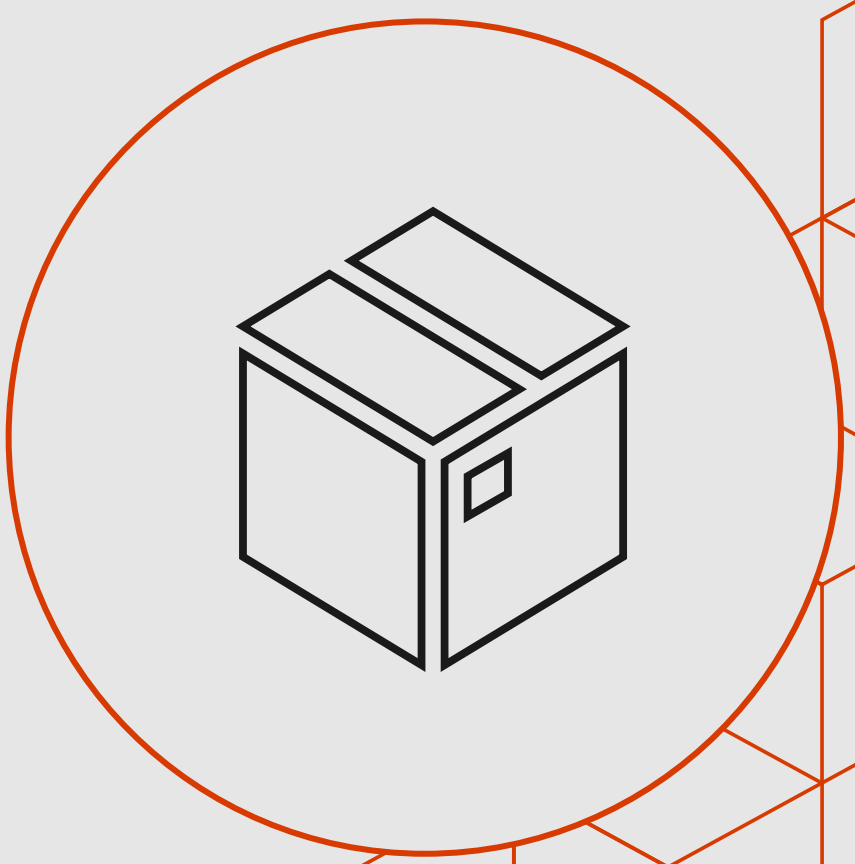
```
[sujit@labs ~]$ sudo docker run -d -p 8080:80 nginx
6cf9ca0ccaca3e3e70ea432c05fdc43b0fd1597083a23c3587d729e808844071
[sujit@labs ~]$
[sujit@labs ~]$ curl http://localhost:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
</p>
```

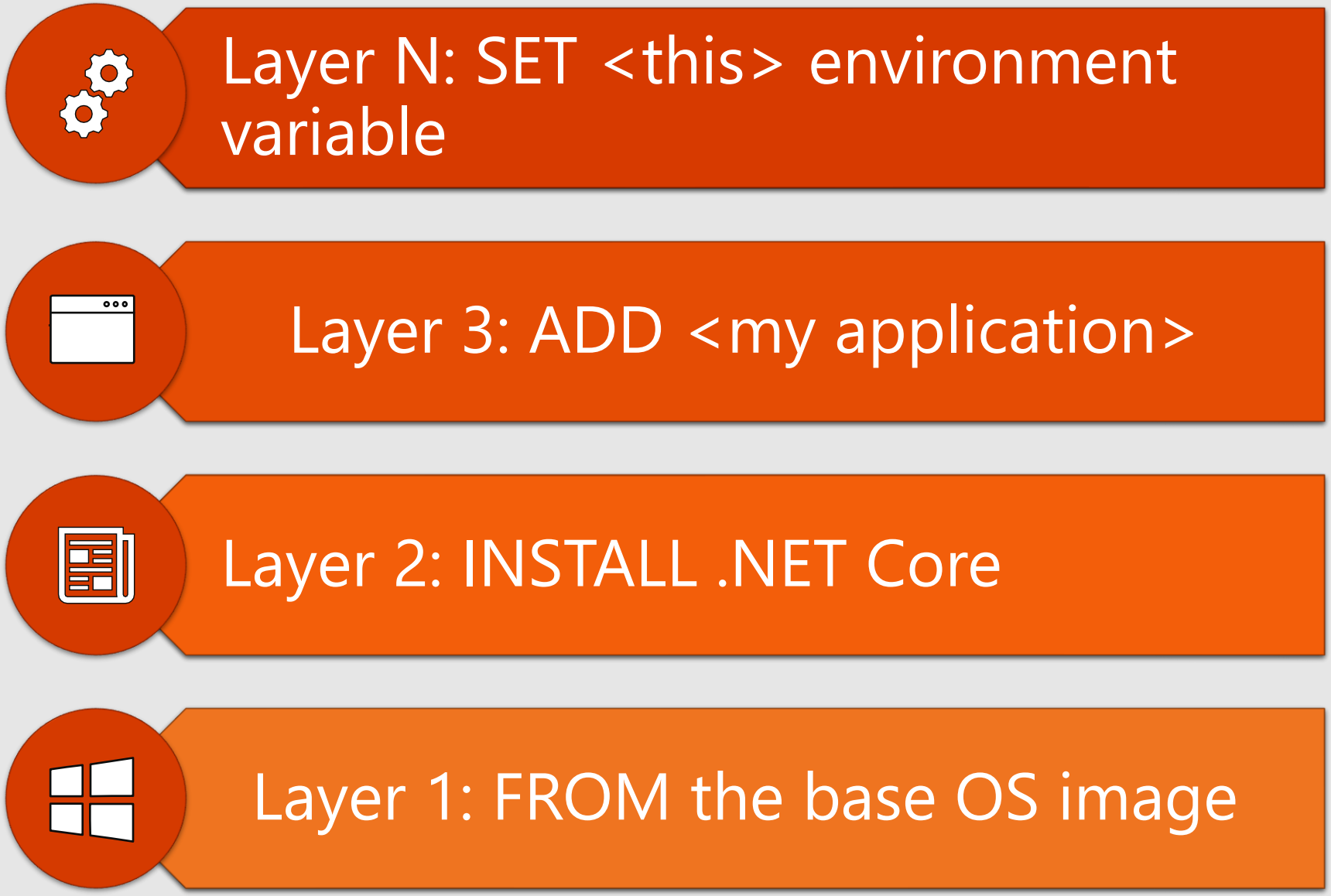



```
root      46306   4478  0 02:48 ?        00:00:00 sshd: sujit [priv]
sujit     46311  46306  0 02:48 ?        00:00:00 sshd: sujit@pts/0
sujit     46312  46311  0 02:48 pts/0    00:00:00 -bash
root      46383  46312  0 02:48 pts/0    00:00:00 sudo -i
root      46385  46383  0 02:48 pts/0    00:00:00 -bash
root      62280      2  0 04:06 ?        00:00:01 [kworker/u256:0]
root      64551   4478  0 04:19 ?        00:00:00 sshd: sujit [priv]
sujit     64557  64551  0 04:19 ?        00:00:00 sshd: sujit@pts/1
sujit     64558  64557  0 04:19 pts/1    00:00:00 -bash
root      66861      2  0 04:30 ?        00:00:00 [kworker/u256:2]
postfix   69223   4591  0 04:44 ?        00:00:00 pickup -l -t unix -u
root      69874      2  0 04:46 ?        00:00:00 [kworker/0:4]
root      69875      2  0 04:46 ?        00:00:00 [kworker/0:5]
root      69876      2  0 04:46 ?        00:00:00 [kworker/0:6]
root      69893      2  0 04:46 ?        00:00:00 [kworker/1:17]
root      69894      2  0 04:46 ?        00:00:00 [kworker/1:18]
root      69895      2  0 04:46 ?        00:00:00 [kworker/1:19]
root      70249      1  0 04:49 ?        00:00:00 nginx: master process nginx
nginx     70250  70249  0 04:49 ?        00:00:00 nginx: worker process
nginx     70251  70249  0 04:49 ?        00:00:00 nginx: worker process
root      70884      2  0 04:54 ?        00:00:00 [kworker/1:0]
root      70902   5391  0 04:54 ?        00:00:00 /usr/libexec/docker/docker-proxy-curren
root      70907   5396  0 04:54 ?        00:00:00 /usr/bin/docker-containerd-shim-current
root      70922  70907  0 04:54 ?        00:00:00 nginx: master process nginx -g daemon o
101       70946  70922  0 04:54 ?        00:00:00 nginx: worker process
root      71110      2  0 04:55 ?        00:00:00 [kworker/0:0]
root      71302      1  0 04:56 ?        00:00:00 /usr/libexec/fprintd
root      71343  46385  0 04:57 pts/0    00:00:00 ps -ef
[root@labs api]#
```

What’s Inside a Container?



What's Inside a Container?



Dockerfiles

Recipes for building containers

Dockerfile

```
FROM microsoft/dotnet:2.1-aspnetcore-runtime
COPY ./published /app
WORKDIR /app

EXPOSE 5000/tcp
ENV ASPNETCORE_URLS http://*:5000

ENTRYPOINT ["dotnet", "test.dll"]
```

Containers and Images

Can be confusing

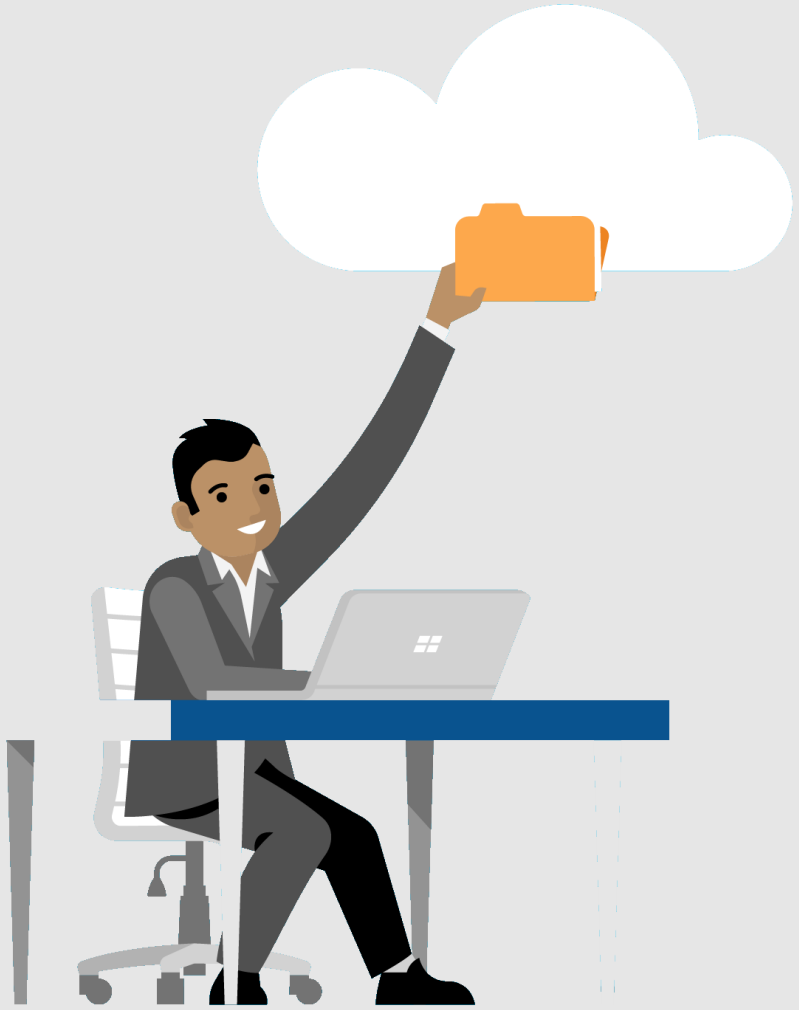
Container = Running instance of the workload
Image = Template for the container (like a VHD)

Demo 1

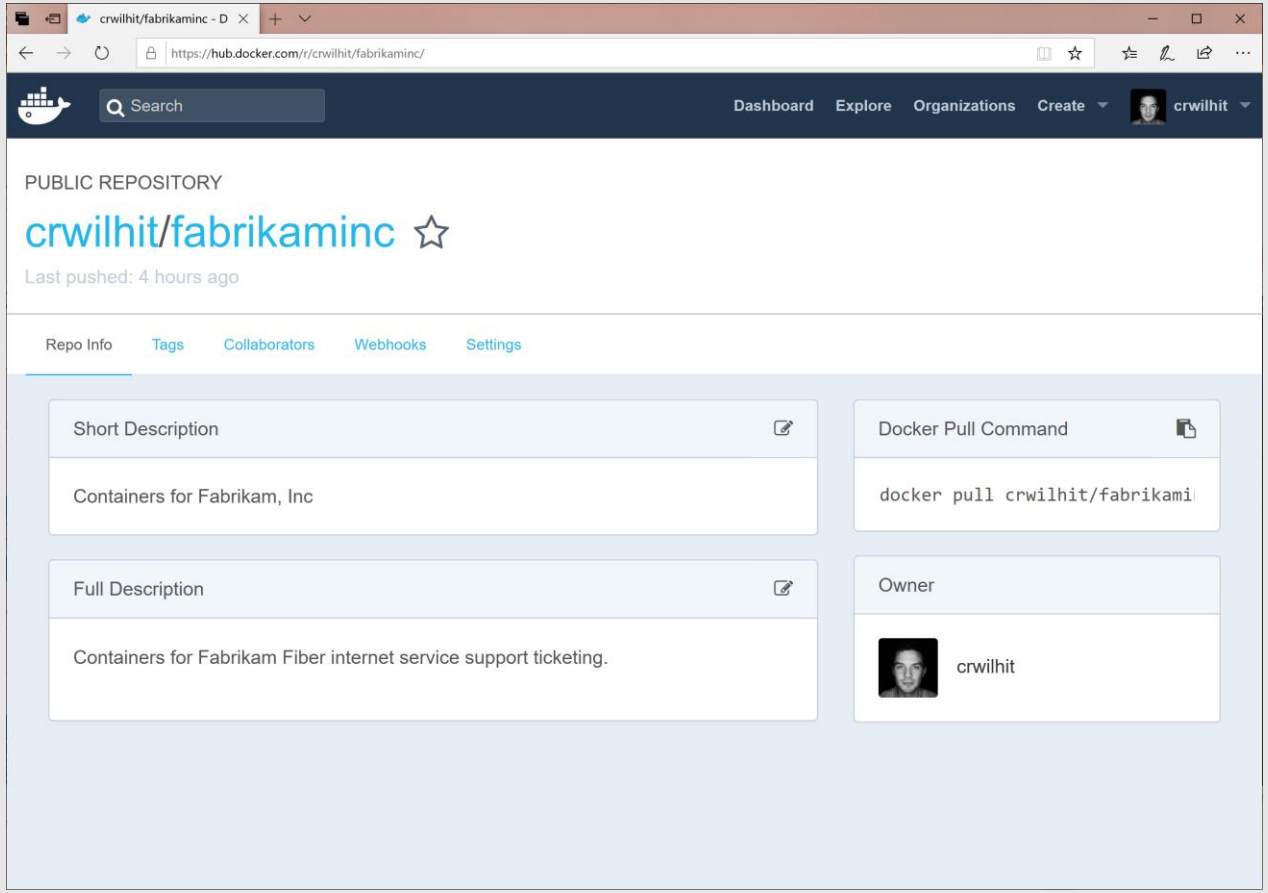
1. Wordpress in seconds
2. ASP.NET Core App in a Docker Container

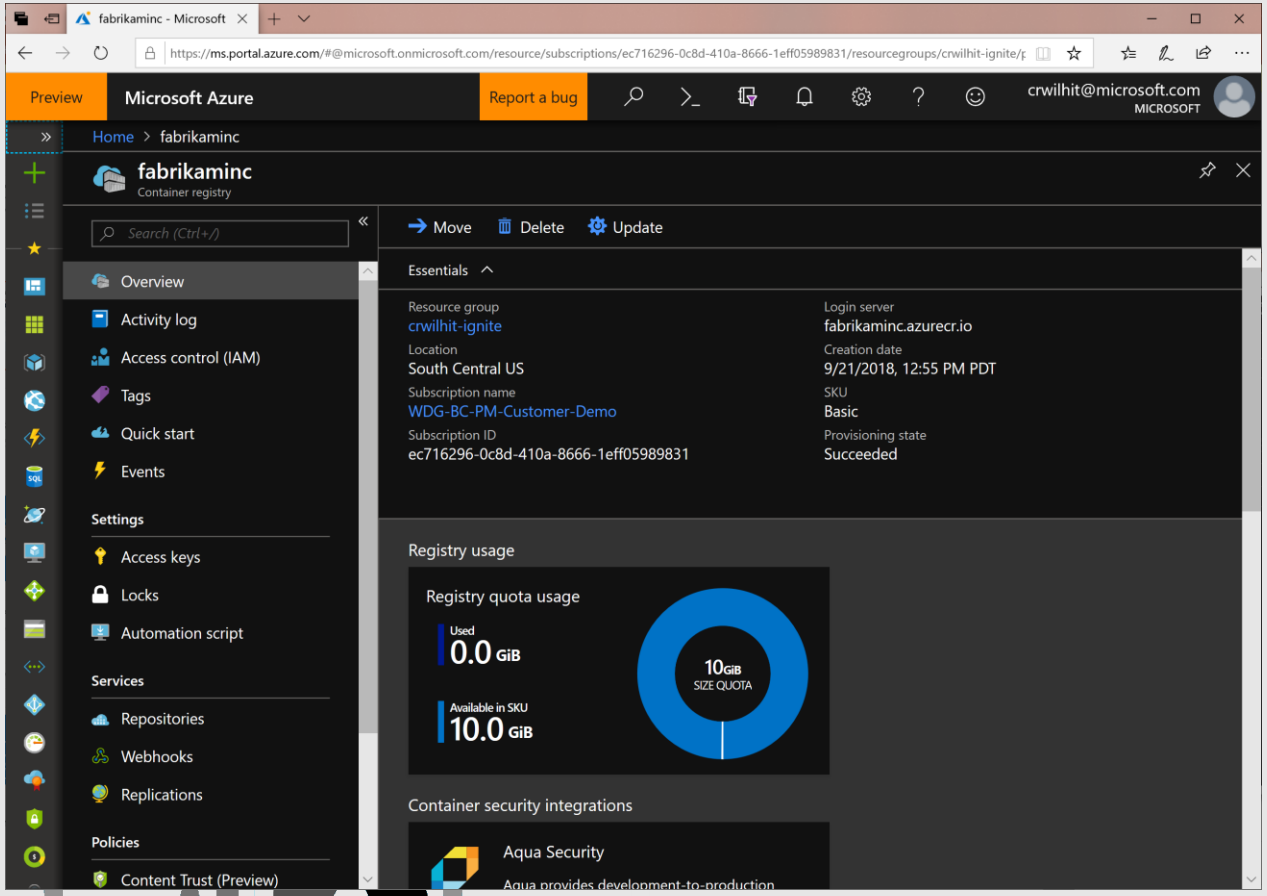
Storing a Container with Container Registries

- Registries track and manage container images
- Can be public or private



Storing a Container with Container Registries

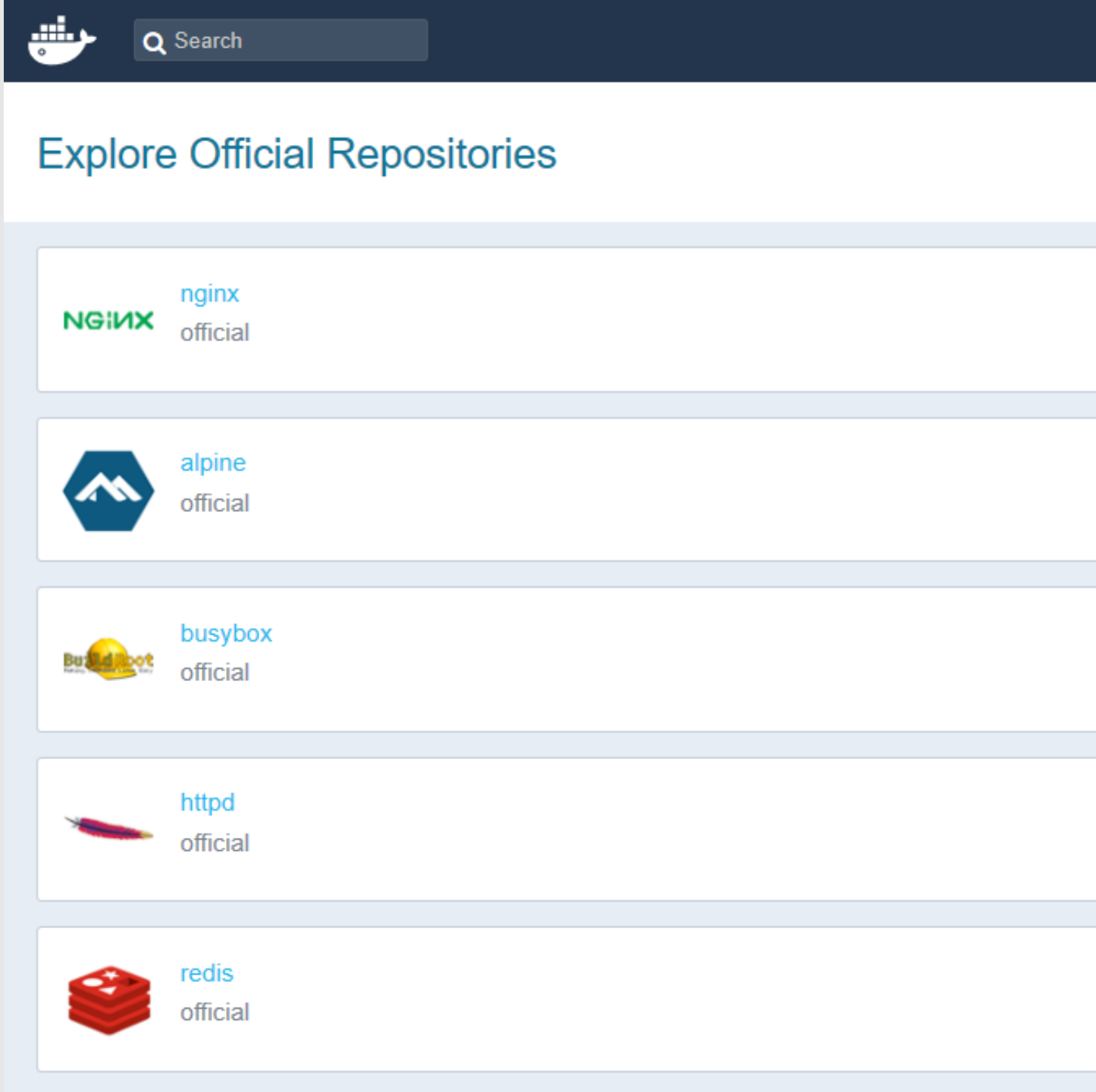


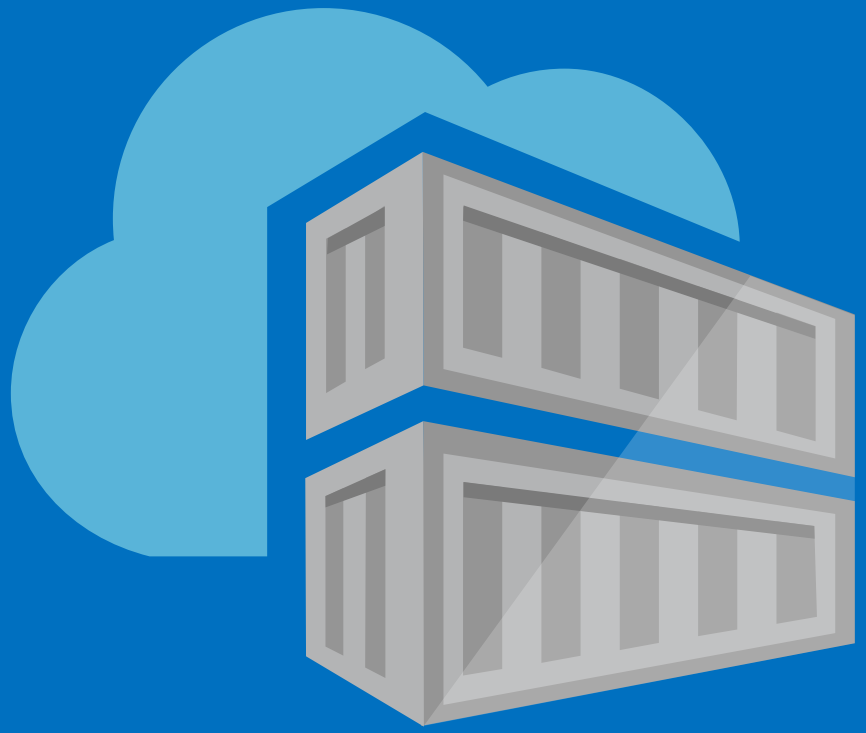


Docker Hub

Largest public container registry.

Explore and find container images.

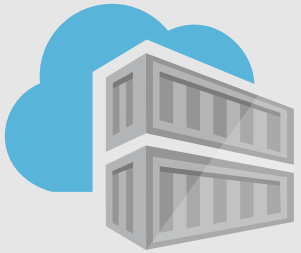


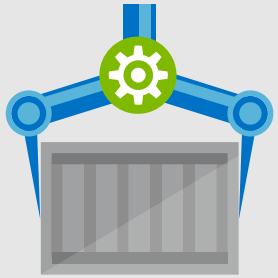


Azure Container Registry

Azure Container Registry

Use familiar, open-source Docker CLI tools





Expand registry functionality with triggers and webhooks

Manage a single registry across multiple regions

ACR Tasks.

Streamlined container image builds in Azure.

Fast build success validation.

Automatically push built images to your container registry

Windows or Linux containers

How to use container registry

- Steps:
- Create container image
  - docker build
- Tag container image with registry reference
  - docker tag
- Log in container registry
  - docker login
- Push container image to registry
  - docker push
  - To use your new image use docker pull

Demo 2

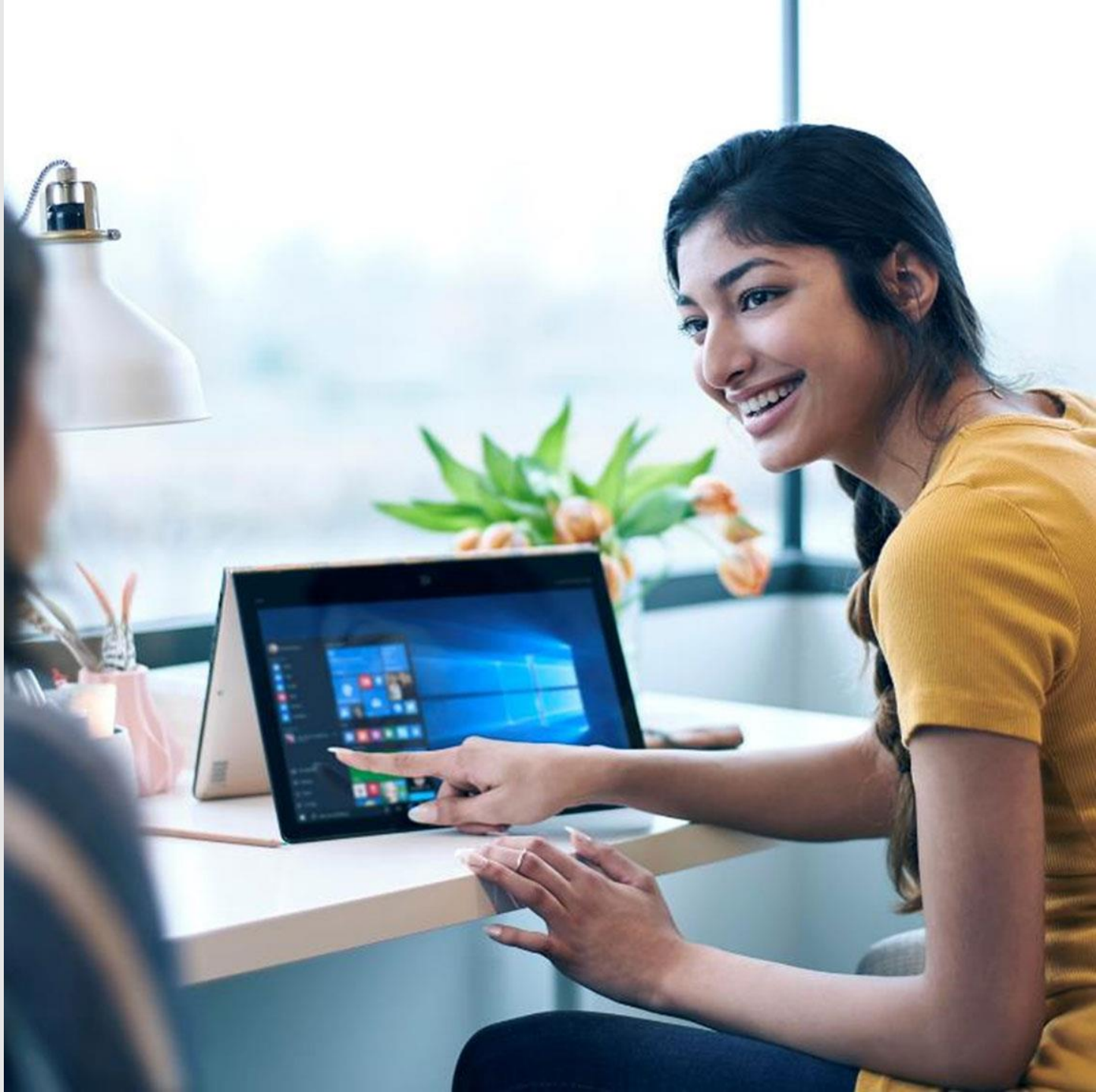
Using Docker Hub

Azure Container Registry

Knowledge Check

1. What is a container image?
2. How to create an image?
3. What is Docker Hub?
4. What is Azure Container Registry?

Lab 3 - Create your first Docker image



Microsoft

Thank you! Questions?