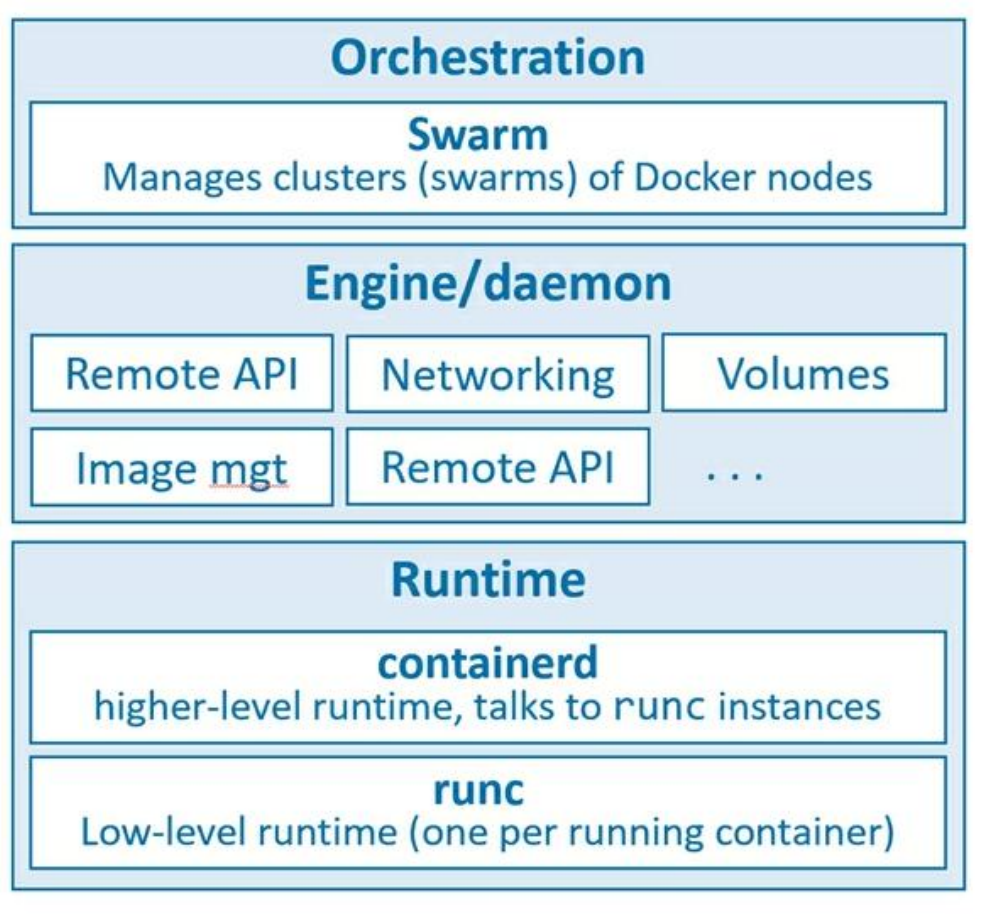
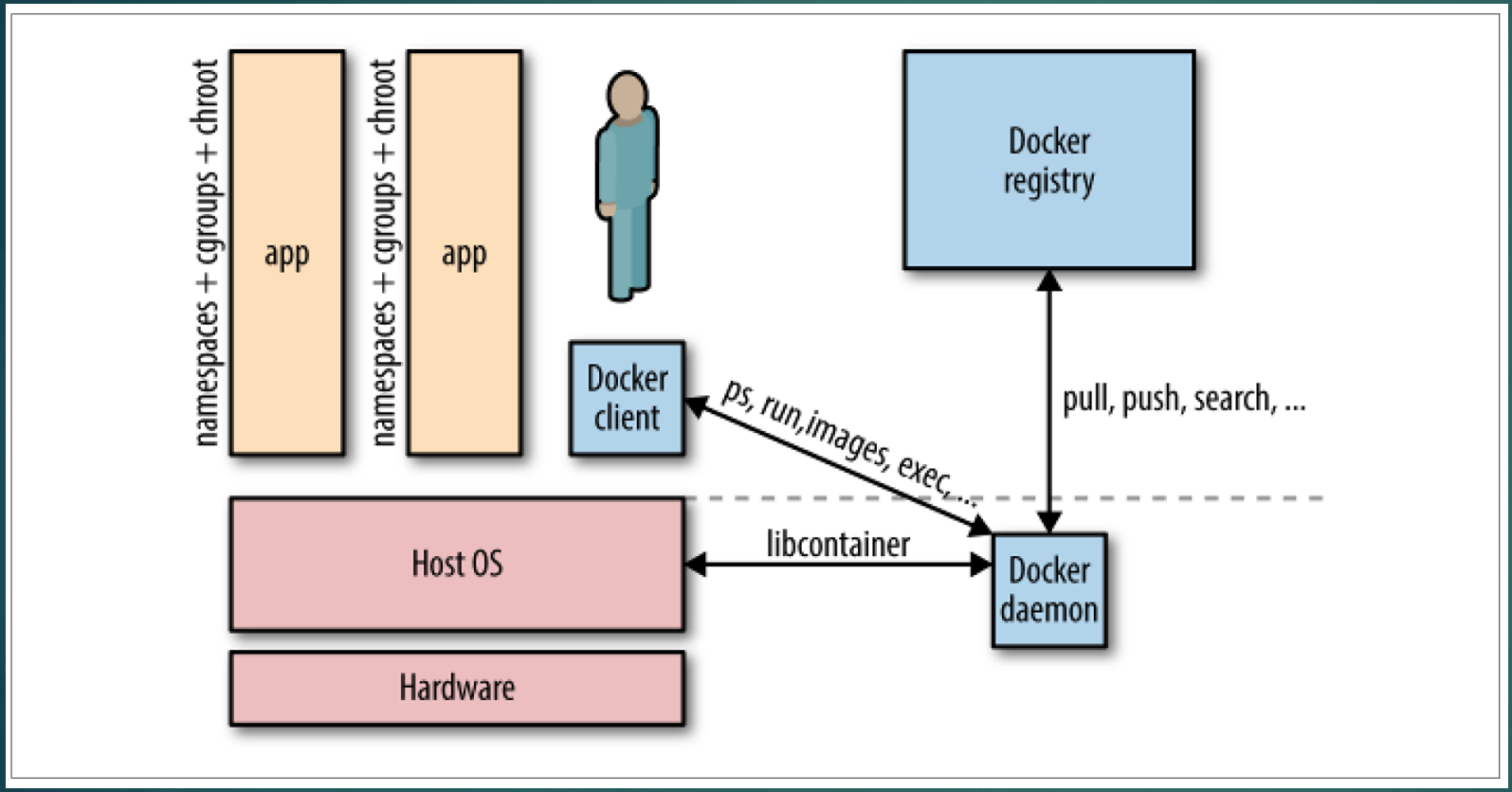
Docker

Docker Architecture



- The lowest level and is responsible for starting and stopping

Docker in Single Host



Basic Component

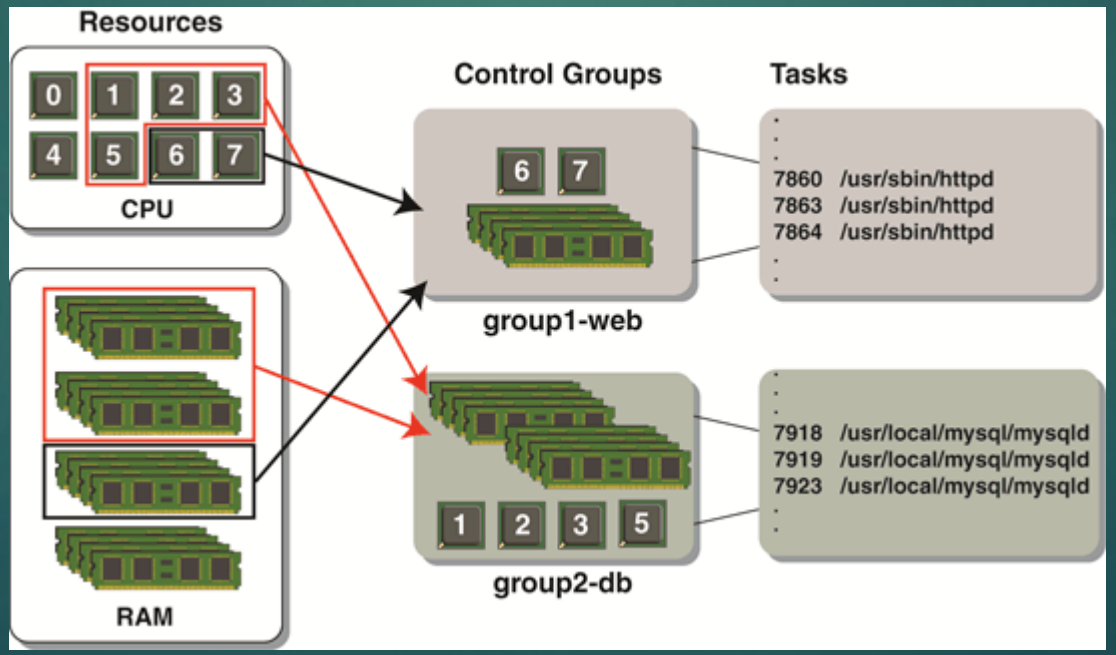
- Namespce
- Cgroups
- Chroot

Basic Component

- Namespce
- Cgroups
- Chroot

Control groups (cgroups)

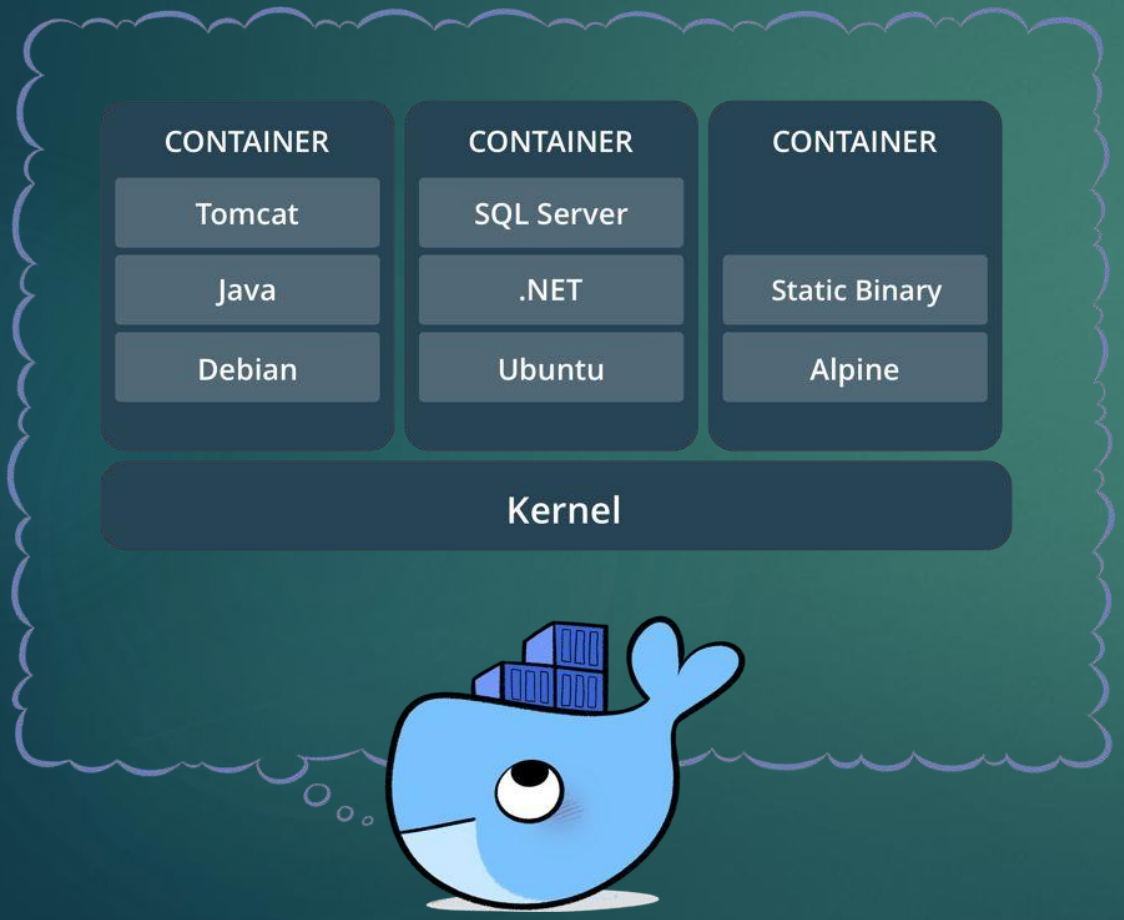
- Control groups provide a mechanism for aggregating/partitioning sets of tasks (processes), and all their future children, into hierarchical groups.



Chroot

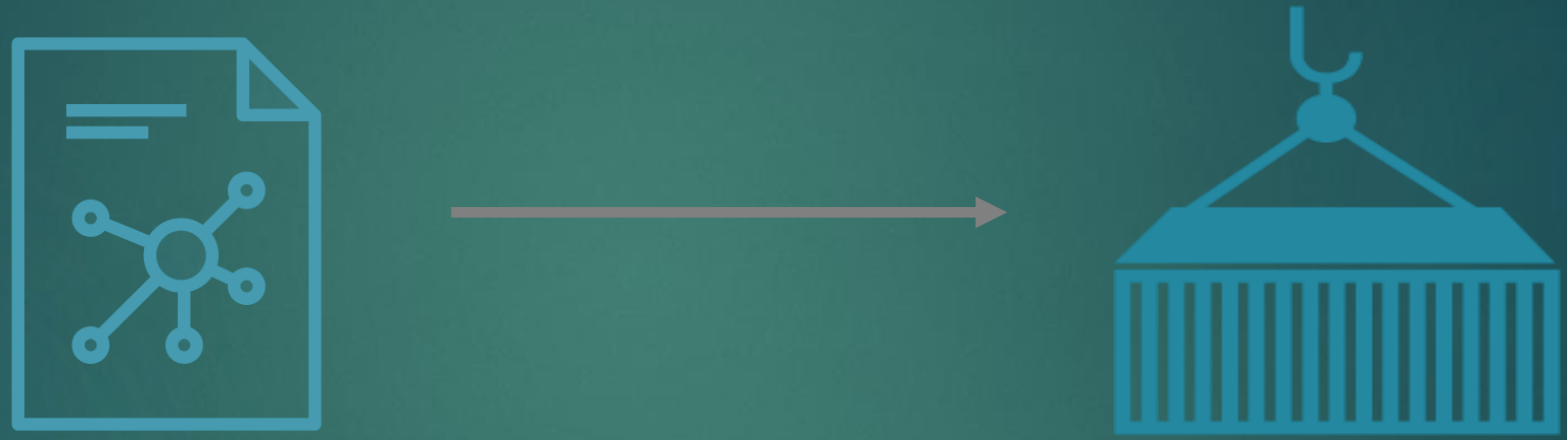
- Chroot – change root directory for a running process, along with children segregate and isolate processes, protecting global environment

What is a container?



- Standardized packaging for software and dependencies
- Isolate apps from each other
- Share the same OS kernel
- Works for all major Linux distributions
- Containers native to Windows Server 2016

The Role of Images and Containers



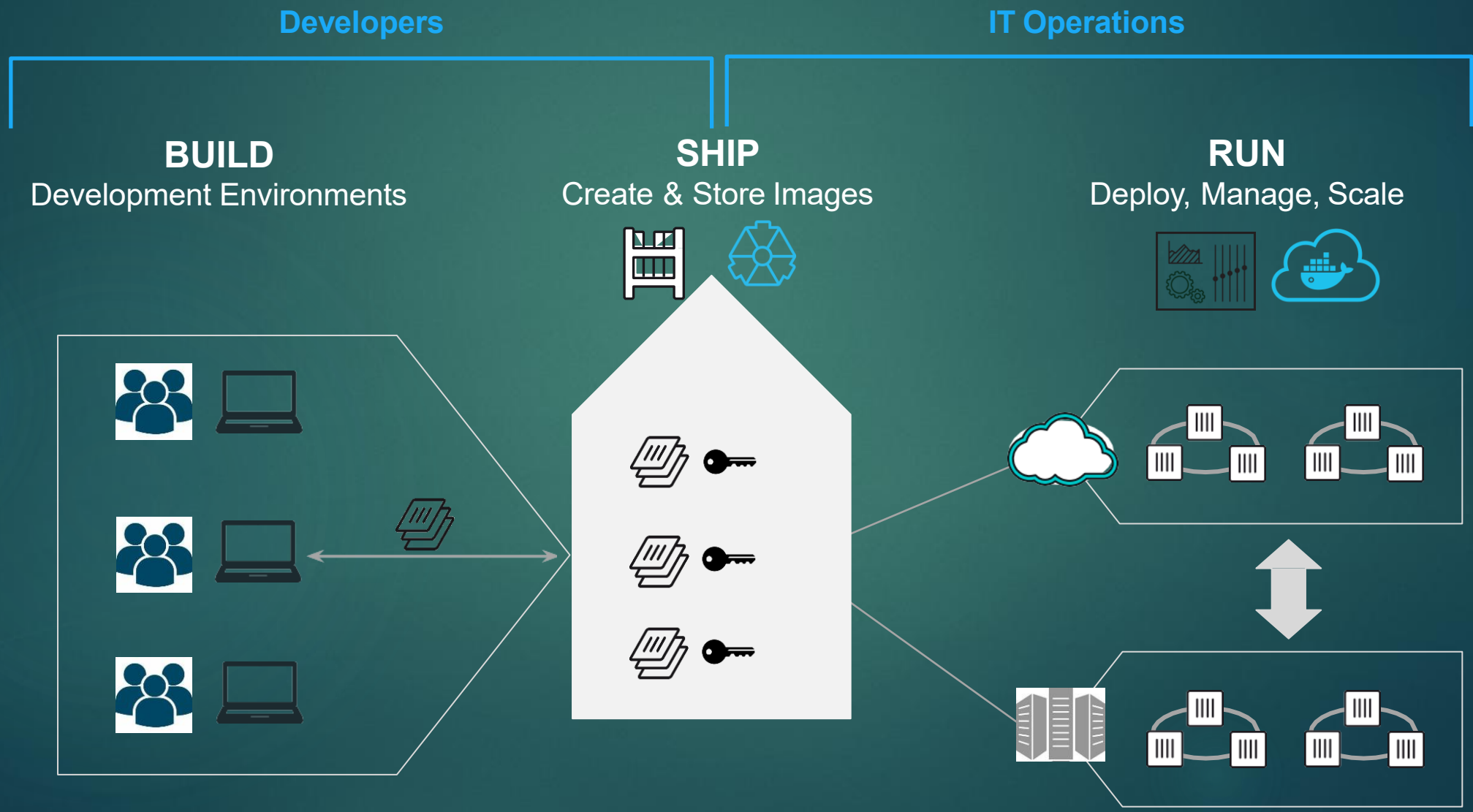
Docker Image

Example: Ubuntu with Node.js and Application Code

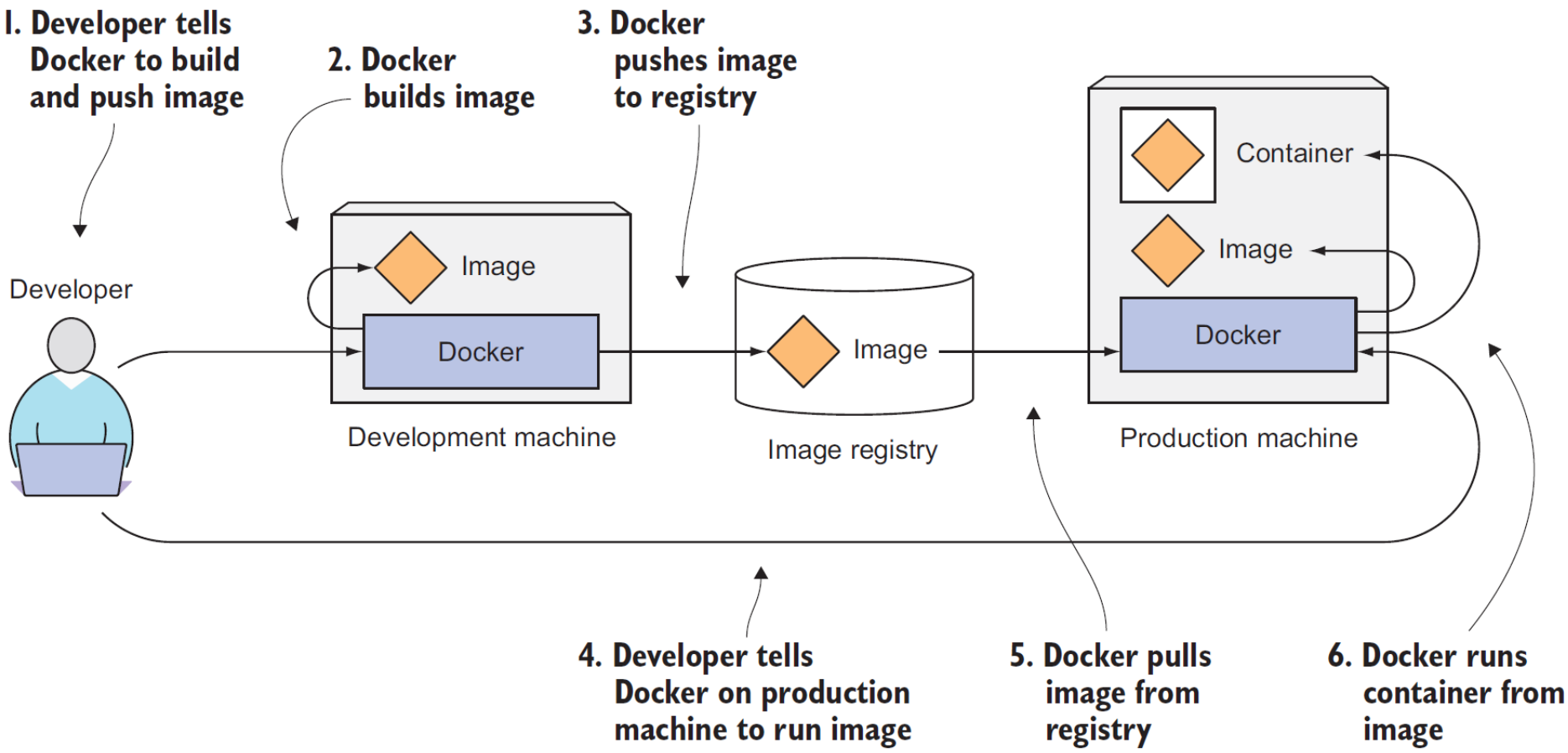
Docker Container

Created by using an image. Runs your application.

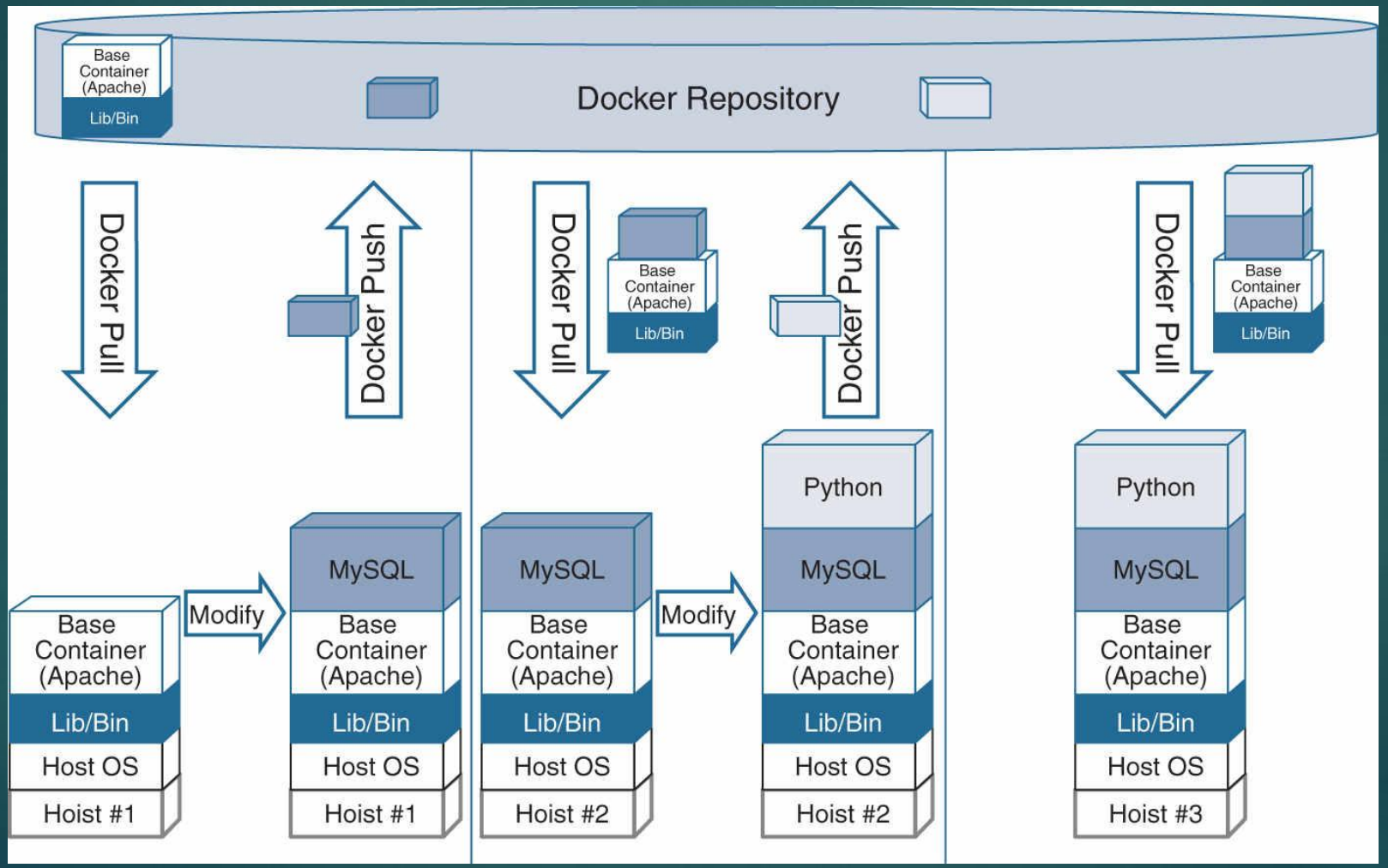
Using Docker: Build, Ship, Run Workflow



Docker Process



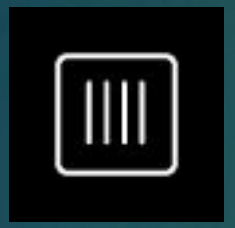
Docker Stack



Some Docker vocabulary

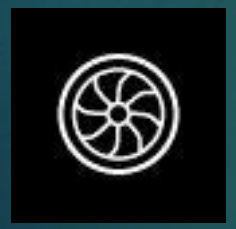
Docker Image

The basis of a Docker container. Represents a full application



Docker Container

The standard unit in which the application service resides and executes



Docker Engine

Creates, ships and runs Docker containers deployable on a physical or virtual, host locally, in a datacenter or cloud service provider

Registry Service (Docker Hub(Public) or Docker Trusted Registry(Private))

Cloud or server based storage and distribution service for your images

Images

- Image = an object that contains an OS file system, an application and all dependencies
- For developer image performs as a ***class***

More …

Images

- Images are read only templates used to create containers.
- Images are created with the docker build command, either by us or by other docker users.
- Images are composed of layers of other images.
- Images are stored in a Docker registry.

Containers

Containers

- If an image is a class, then a container is an instance of a class - a runtime object.
- Containers are lightweight and portable encapsulations of an environment in which to run applications.
- Containers are created from images. Inside a container, it has all the binaries and dependencies needed to run the application.

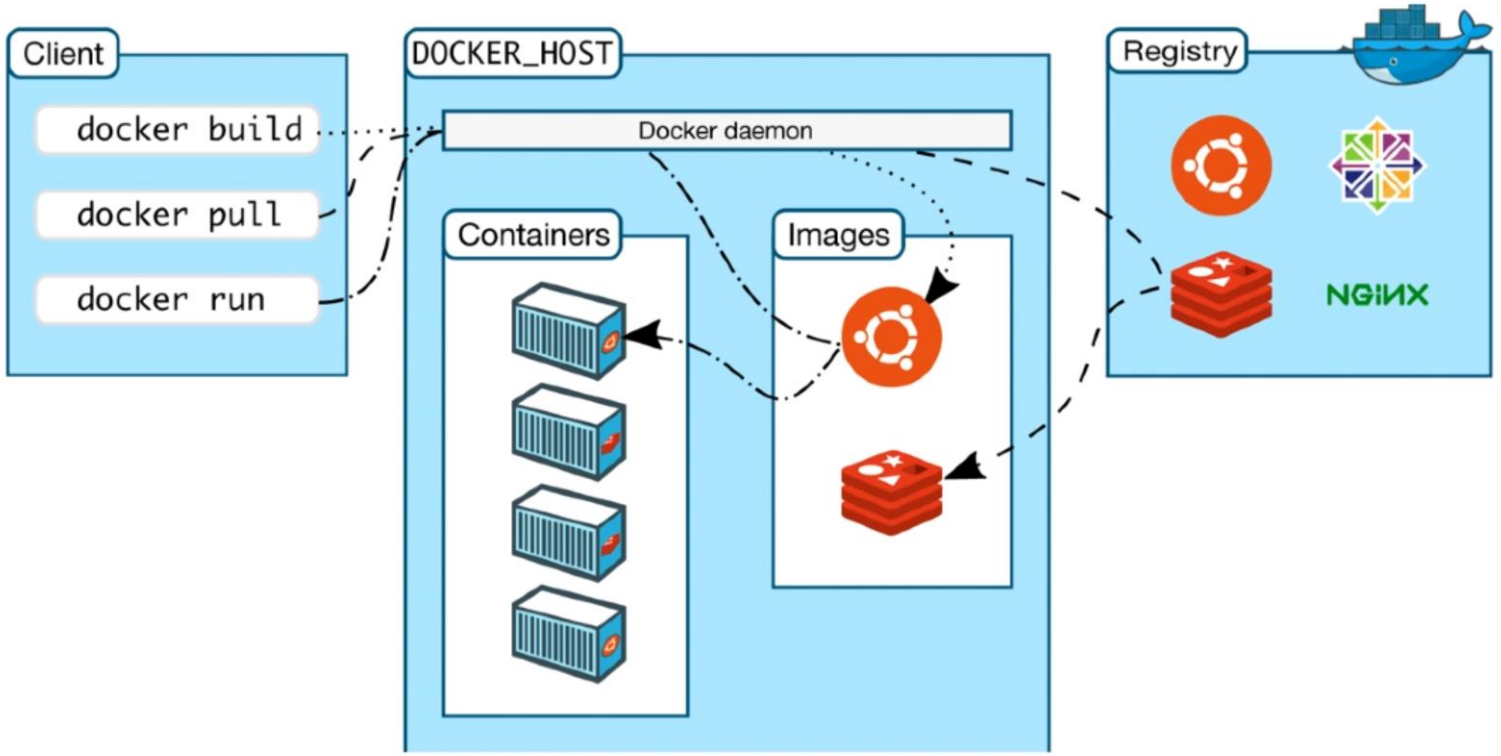
Registries and Repositories

- A registry is where we store our images.
- You can host your own registry, or you can use Docker's public registry which is called DockerHub.
- Inside a registry, images are stored in repositories.
- Docker repository is a collection of different docker images with the same name, that have different tags, each tag usually represents a different version of the image.

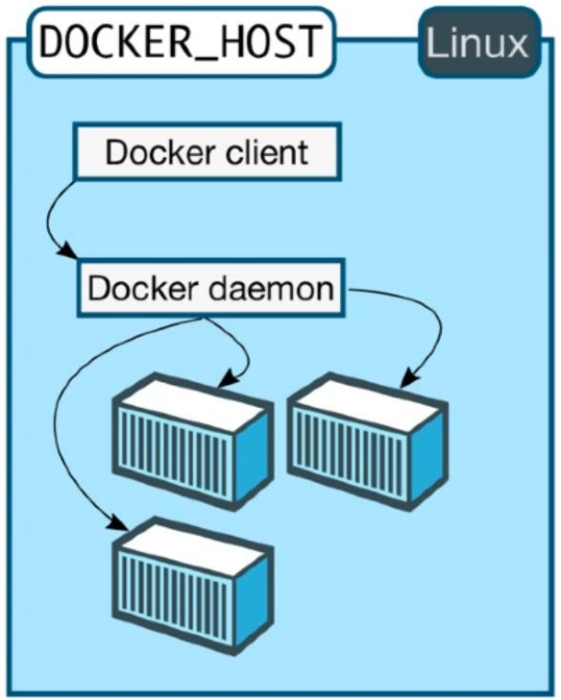
Basic Docker Commands

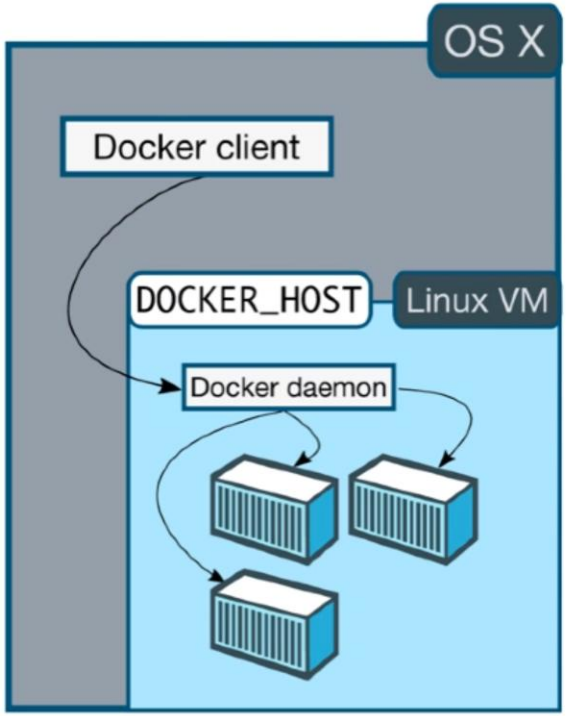
```
$ docker image pull node:latest

$ docker image ls
$ docker container run –d –p 5000:5000 –-name node node:latest

$ docker container ps

$ docker container stop node(or <container id>)

$ docker container rm node (or <container id>)

$ docker image rmi (or <image id>)

$ docker build –t node:2.0 .

$ docker image push node:2.0

$ docker --help
```

Docker Interaction



Running in Linux and other





In windows 10 or OS X, Docker needs the lightweight Linux

Install on Windows

- Install “Docker Desktop”
- Requirement:
  - 64-bit version of Windows 10 Pro/Enterprise/Education (does not work with Home edition)
  - Hardware virtualization support must be enabled in your system’s BIOS
  - The Hyper-V and Containers features must be enabled in Windows

Powershell command

```
$ docker version
Client: Docker Engine - Community
 Version:           19.03.8
 API version:       1.40
 Go version:        go1.12.17
 Git commit:        afacb8b
 Built:             Wed Mar 11 01:23:10 2020
 OS/Arch:           windows/amd64
 Experimental:      true
Server: Docker Engine - Community
 Engine:
  Version:          19.03.8
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.17
  Git commit:       afacb8b
  Built:            Wed Mar 11 01:29:16 2020
  OS/Arch:          linux/amd64
  Experimental:     true
<Snip>
```

- OS/Arch : linux/amd64
  - because default installation assumes Linux containers

Switch to Window containers

- Right-click the docker whale icon
  - Selecting to Window containers
  - Locate \Program Files\Docker\ Docker directory

```
C:\Program Files\Docker\Docker> .\dockercli -SwitchDaemon
```

```
C:\> docker version
Client:
 <Snip>

Server:
 Engine:
  Version:      19.03.8
  API version:  1.40 (minimum version 1.24)
  Go version:   go1.12.17
  Git commit:   afacb8b
  Built:        Wed Mar 11 01:37:20 2020
  OS/Arch:      windows/amd64
  Experimental: true
```

Installing Docker on Linux

1. Update the apt package index.

```
$ sudo apt-get update
Get:1 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu focal InRelease [265 kB]
...
```

2. Install Docker from the official repo.

```
$ sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree
...
```

Docker is now installed and you can test by running some commands.

```
$ sudo docker --version
Docker version 19.03.8, build afacb8b7f0

$ sudo docker info
Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 ...
```

Big Picture

- **The Ops perspective**
  - download an image,
  - start a new container, log in to the new container,
  - run a command inside of it, and then destroy it.
- **The Dev perspective**
  - focus more on the app
  - clone some app-code from GitHub,
  - inspect a Dockerfile, containerize the app, run it as a container.

List all image

```
$ docker image ls
REPOSITORY   TAG      IMAGE ID      CREATED      SIZE
```

- Pulling image
  - Linux, pull the ubuntu:latest image.
  - Windows, pull the mcr.microsoft.com/powershell:lts-nanoserver-1903 image.

```
$ docker image pull ubuntu:latest
latest: Pulling from library/ubuntu
50aff78429b1: Pull complete
f6d82e297bce: Pull complete
275abb2c8a6f: Pull complete
9f15a39356d6: Pull complete
fc0342a94c89: Pull complete
Digest: sha256:fbaf303...c0ea5d1212
Status: Downloaded newer image for ubuntu:latest
```

```
$ docker images
REPOSITORY   TAG      IMAGE ID      CREATED      SIZE
ubuntu       latest   1d622ef86b13  16 hours ago  73.9MB
```

Container

For Linux:

```
$ docker container run -it ubuntu:latest /bin/bash
root@6dc20d508db0:/#
```

For Windows:

```
> docker container run -it mcr.microsoft.com/powershell:lts-nanoserver-1903 pwsh.exe

PowerShell 7.0.0
Copyright (C) Microsoft Corporation. All rights reserved.
PS C:\>
```

Run ps command

Linux example:

```
root@6dc20d508db0:/# ps -elf
F S UID   PID  PPID  NI ADDR SZ WCHAN  STIME TTY      TIME CMD
4 S root    1     0   0 -  4560 -      13:38 pts/0    00:00:00 /bin/bash
0 R root    9     1   0 -  8606 -      13:38 pts/0    00:00:00 ps -elf
```

Press Ctrl-PQ exit the container without terminating it

Windows example:

```
PS C:\> ps

NPM(K)    PM(M)      WS(M)     CPU(s)      Id  SI ProcessName
------    -----      -----     ------      --  -- -----------
     5     0.90       3.78       0.00    1068   1 CExecSvc
     6     0.97       4.12       0.03    1184   1 conhost
     6     0.87       2.16       0.00     972   1 csrss
     0     0.06       0.01       0.00       0   0 Idle
    18     4.38      12.32       0.00     272   1 lsass
    54    34.82      65.09       1.27    1212   1 pwsh
     9     1.61       4.99       0.00    1020   1 services
     4     0.49       1.18       0.00     948   0 smss
    14     1.98       6.61       0.00     628   1 svchost
    12     2.95      10.02       0.00     752   1 svchost
     8     1.83       6.02       0.00     788   1 svchost
     7     1.42       4.70       0.00    1040   1 svchost
    16     6.12      11.41       0.00    1120   1 svchost
    24     3.73      10.38       0.00    1168   1 svchost
    15     9.60      18.96       0.00    1376   1 svchost
```

Attaching to running containers

docker container exce <options> <container name or container-id>

- Linux

```
$ docker container exec -it vigilant_borg bash
root@6dc20d508db0:/#
```

- Verify stil running

```
$ docker container ls
CONTAINER ID  IMAGE       COMMAND     CREATED  STATUS    NAMES
6dc20d508db0  ubuntu:latest  "/bin/bash"  9 mins  Up 9 min  vigilant_borg
```

- Windows

```
> docker container exec -it pensive_hamilton pwsh.exe

PowerShell 7.0.0
Copyright (C) Microsoft Corporation. All rights reserved.
PS C:\>
```

Press Ctrl-PQ exit the container

Stop - remove

```
$ docker container stop vigilant_borg
vigilant_borg

$ docker container rm vigilant_borg
vigilant_borg
```

- Verify all gone

```
$ docker container ls -a
CONTAINER ID   IMAGE   COMMAND   CREATED   STATUS   PORTS   NAMES
```

The Dev Perspective

- Simple Web apps
- Linux
  - https://github.com/nigelpoulton/psweb.git
- Windows
  - https://github.com/nigelpoulton/win-web.git

Linux (windows substisute repo)

```
$ git clone https://github.com/nigelpoulton/psweb.git
Cloning into 'psweb'...
remote: Counting objects: 15, done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 15 (delta 2), reused 15 (delta 2), pack-reused 0
Unpacking objects: 100% (15/15), done.
Checking connectivity... done.
```

```
$ cd psweb
$ ls -l
total 40
-rw-r--r--@ 1 ubuntu ubuntu  338 24 Apr 19:29 Dockerfile
-rw-r--r--@ 1 ubuntu ubuntu  396 24 Apr 19:32 README.md
-rw-r--r--@ 1 ubuntu ubuntu  341 24 Apr 19:29 app.js
-rw-r--r--  1 ubuntu ubuntu  216 24 Apr 19:29 circle.yml
-rw-r--r--@ 1 ubuntu ubuntu  377 24 Apr 19:36 package.json
drwxr-xr-x  4 ubuntu ubuntu  128 24 Apr 19:29 test
drwxr-xr-x  3 ubuntu ubuntu   96 24 Apr 19:29 views
```

The Linux example is a simple nodejs web app. The Windows example is an IIS server running some static HTML.