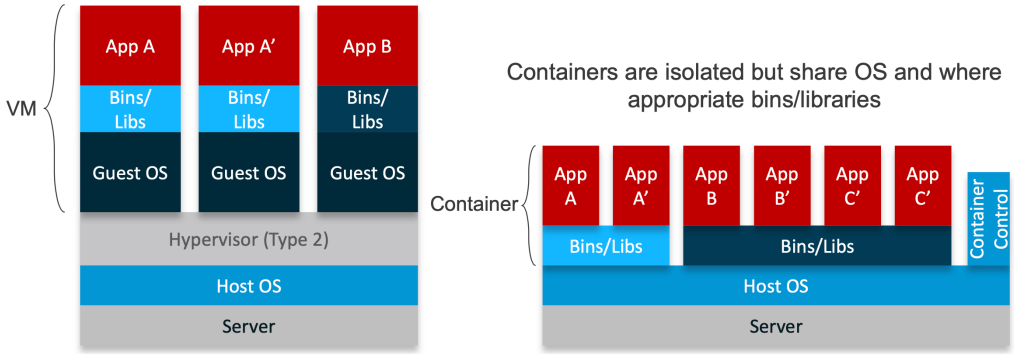
UNDERSTANDING DOCKER

Virtual Machines vs Containers

- Virtual Machines
  - Can run any OS
  - Access to dedicated hardware
  - Your support staff knows how to manage
- Container
  - Share a single operating system kernel
  - Require much less resources
  - Faster to launch



Containers are almost like Virtual Machines

SmartPro

- Containers have their own network interface (and IP address)
  - Can be bridged, routed ... Just like with Xen, KVM etc
- Container have their own filesystem
  - For example, a Debian host can run Fedora container (and vice-versa)
- Security: Containers are isolated from each other
  - Two container can't harm (or even see) each other
- Resource Control: Containers are isolated and can have dedicated resources:
  - Soft & hard quotas for RAM, CPU, I/O...

Though...

Apps in Containers share the kernel of the host (i.e Linux guests only)

Container are light-weight, fast to start, allow for > 10x density compared to VMs

Containers Are...

- A way to package up our applications and dependencies.
- A way to guarantee execution consistency and portability.
- A way to keep your applications isolated.
- A way to use your compute resources without the overhead of VM's.

Containers Are not...

Microservices

- We hear containers and microservice used a lot together.
- Microservices benefit from a lightweight packaging, distribution and deployment solution.
- However, you can put package anything into a container, including a badly written legacy app in some cases, using containers doesn't magically make bad code better.

VM's

- Containers are purely user-space, if you need kernel extensions/modules or a custom kernel,
- containers probably aren't what you're looking for.

Magic

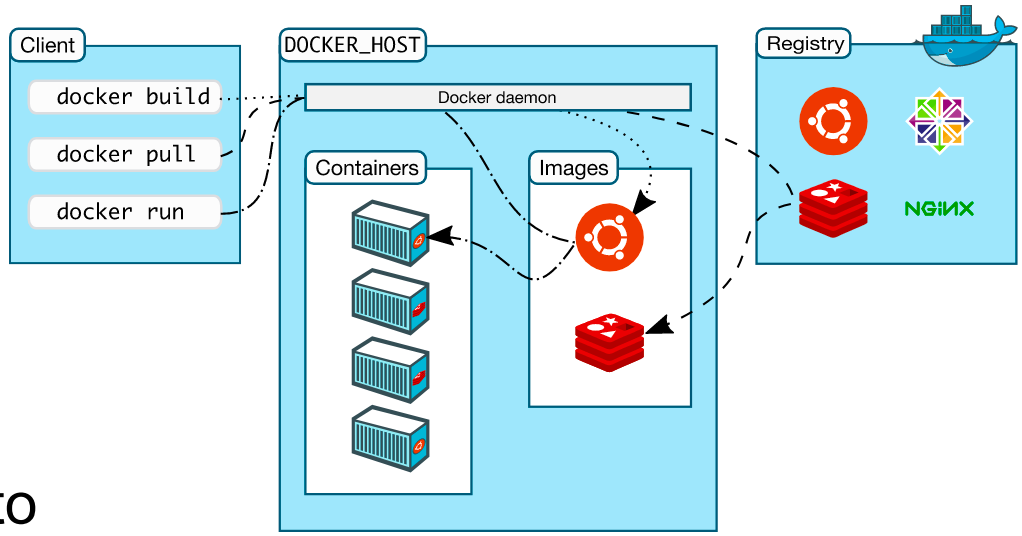
- They bring their own nuances and require deployment consideration just like any other toolchain.

What is Docker?

- Created in 2010 by Solomon Hykes and Sebastien Pahl
- Docker is a container technology similar to Linux Containers (LXC) that...
  - Provides isolation for application processes from the host processes using Linux **namespaces**
  - Provides resource caps for the application using Linux **cgroups**
  - Provides industry preferred **packaging** model using docker
  - images, docker index, and docker registry concepts
  - Provides the basis for application **lifecycle management** automation due to good integration with devops automation tools such as Puppet/Chef

Docker architecture

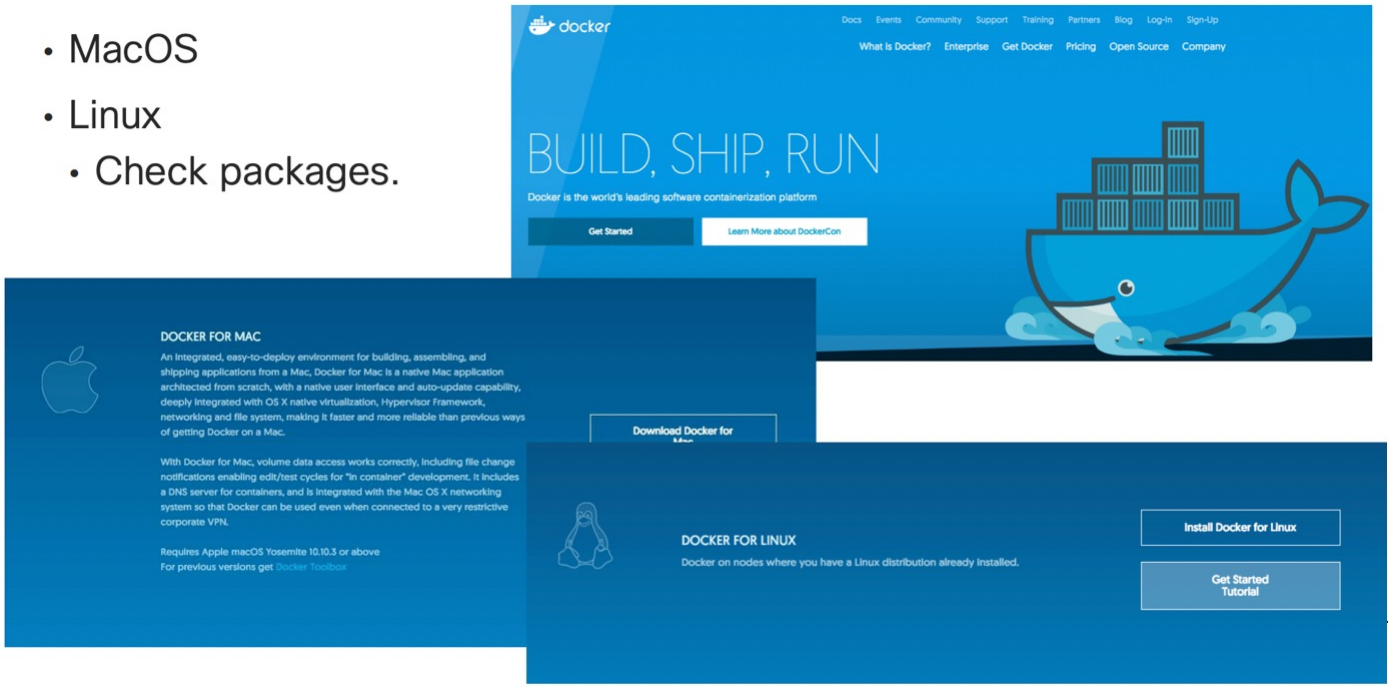
- Client
  - CLI to Docker engine
  - Local or remote Docker engine
  - Uses RESTful API
- Docker Host
  - Runs Docker engine
  - Hosts containers
  - Stores images locally
- Docker Registry
  - Software distribution to hosts and engine
  - Docker Hub is public registry



USING DOCKER

Getting Docker...

- MacOS
- Linux
  - Check packages.



Getting Docker...

```
$ docker -v
```

```
$ docker -v
Docker version 20.10.14, build a224086
$
```

https://labs.play-with-docker.com/

DOCKER BASIC COMMANDS

Docker images, Containers, and Registries

- Images
  - Read-only templates used to create Docker containers
- Containers
  - Like a directory
  - Consists of image files that hold the components the app needs to run
- Registries
  - Stateless, scalable, server-side applications that stores & lets you distribute Docker images
  - You can use public and private registries

Working with Containers

When working with containers, the key commands are as follows:

- **build:** Create a container from an image.
- **start:** Start an existing container.
- **run:** Create a new container and start it.
- **ps:** List running containers.
- **inspect:** Get detailed information regarding the container.
- **logs:** Print run logs from the container's execution.
- **stop:** Gracefully stop running the container.
- **kill:** Stop the main process in the container abruptly.
- **rm:** Delete a stopped container.

Searching for public images

```
$ docker search <keyword>
```

```
$ sudo docker search hello-world
NAME                                   DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
hello-world                            Hello World! (an example of minimal Dockeriz…   1722      [OK]
kitematic/hello-world-nginx            A light-weight nginx container that demonstr…   151
tutum/hello-world                      Image to test docker deployments. Has Apache…   88                   [OK]
dockercloud/hello-world                Hello World!                                    19                   [OK]
crccheck/hello-world                   Hello World web server in under 2.5 MB          15                   [OK]
vad1mo/hello-world-rest                A simple REST Service that echoes back all t…   5                    [OK]
ansibleplaybookbundle/hello-world-db-apb An APB which deploys a sample Hello World! a…  2                    [OK]
ppc64le/hello-world                    Hello World! (an example of minimal Dockeriz…   2
rancher/hello-world                                                                    1
souravpatnaik/hello-world-go           hello-world in Golang                           1
ansibleplaybookbundle/hello-world-apb  An APB which deploys a sample Hello World! a…  1                    [OK]
thomaspoignant/hello-world-rest-json   This project is a REST hello-world API to bu…   1
strimzi/hello-world-consumer                                                           0
koudaiii/hello-world                                                                   0
businessgeeks00/hello-world-nodejs                                                     0
garystafford/hello-world               Simple hello-world Spring Boot service for t…   0                    [OK]
freddiedevops/hello-world-spring-boot                                                  0
strimzi/hello-world-streams                                                            0
tsepotesting123/hello-world                                                            0
okteto/hello-world                                                                     0
armswdev/c-hello-world                 Simple hello-world C program on Alpine Linux…   0
dandando/hello-world-dotnet                                                            0
kevindockercompany/hello-world                                                         0
rsperling/hello-world3                                                                 0
strimzi/hello-world-producer                                                           0
$
```

Running public images

SmartPro

```
$ docker run <image>
```

```
$ sudo docker run hello-world

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.
    (amd64)
 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 run options

SmartPro

$ docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

- **i:** interactive mode
- t: terminal mode
- d: run in background
- -- name: set container name

https://docs.docker.com/engine/reference/commandline/run/

```
$ sudo docker run redis
1:C 02 May 2022 00:16:23.506 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 02 May 2022 00:16:23.506 # Redis version=7.0.0, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 02 May 2022 00:16:23.507 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
1:M 02 May 2022 00:16:23.508 * monotonic clock: POSIX clock_gettime
1:M 02 May 2022 00:16:23.509 * Running mode=standalone, port=6379.
1:M 02 May 2022 00:16:23.510 # Server initialized
1:M 02 May 2022 00:16:23.510 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
1:M 02 May 2022 00:16:23.511 * The AOF directory appendonlydir doesn't exist
1:M 02 May 2022 00:16:23.511 * Ready to accept connections
```

```
$ sudo docker run -d redis
f1c4b64f6780dfb64cad836f34c7e0e7f576af074965d5b7ef42720cb0d38b97
$
```

```
$ sudo docker run ubuntu
$
```

```
$ sudo docker run -it ubuntu
root@ef40bd533cbe:/#
root@ef40bd533cbe:/#
root@ef40bd533cbe:/#
```

docker ps

- **Docker ps:** show running containers
- **Docker ps --all , -a:** show all containers

```
$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS      NAMES
f1c4b64f6780   redis     "docker-entrypoint.s…"   5 minutes ago   Up 5 minutes   6379/tcp   boring_shirley
$
```

docker exe, docker attach

- Performance a command in a running container

```
$ docker exec <docker id>  <command>
```

- Attach to a running container

```
$ Docker attach <docker id>
```

Docker stop

$ Docker stop <id> | <name>
$ Docker rm <id> | <name> <id> | <name> <id> | <name>....
$ docker container prune
(stop all containers)

```
$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS      NAMES
1372cffd0184   ubuntu    "sleep 1000"             10 minutes ago   Up 10 minutes              amazing_meninsk
y
f1c4b64f6780   redis     "docker-entrypoint.s…"   17 minutes ago   Up 17 minutes   6379/tcp   boring_shirley
$
$ sudo docker stop 1372cffd0184
1372cffd0184
$ sudo docker rm 1372cffd0184
1372cffd0184
$
```

Docker inspect

$ docker inspect <id> | <name>

```
$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS      NAMES
1372cffd0184   ubuntu    "sleep 1000"             9 minutes ago    Up 9 minutes               amazing_meninsk
y
f1c4b64f6780   redis     "docker-entrypoint.s…"   16 minutes ago   Up 16 minutes   6379/tcp   boring_shirley
$ sudo docker inspect 1372cffd0184
[
    {
        "Id": "1372cffd018404955be6d3237fb53802b57e3044b900553653f7b86f97c731b7",
        "Created": "2022-05-02T00:24:26.935018192Z",
        "Path": "sleep",
        "Args": [
            "1000"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 44372,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2022-05-02T00:24:27.322449377Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
        "Image": "sha256:d2e4e1f511320dfb2d0baff2468fcf0526998b73fe10c8890b4684bb7ef8290f",
        "ResolvConfPath": "/var/lib/docker/containers/1372cffd018404955be6d3237fb53802b57e3044b900553653f7b86
f97c731b7/resolv.conf",
        "HostnamePath": "/var/lib/docker/containers/1372cffd018404955be6d3237fb53802b57e3044b900553653f7b86f9
7c731b7/hostname",
        "HostsPath": "/var/lib/docker/containers/1372cffd018404955be6d3237fb53802b57e3044b900553653f7b86f97c7
31b7/hosts",
        "LogPath": "/var/lib/docker/containers/1372cffd018404955be6d3237fb53802b57e3044b900553653f7b86f97c731
b7/1372cffd018404955be6d3237fb53802b57e3044b900553653f7b86f97c731b7-json.log",
        "Name": "/amazing_meninsky",
        "RestartCount": 0,
        "Driver": "overlay2",
        "Platform": "linux",
```

Docker images

```
$ Docker images
$ Docker rmi
$ docker image prune -a
```

```
$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
ubuntu        latest    d2e4e1f51132   2 days ago    77.8MB
redis         latest    a10f849e1540   4 days ago    117MB
wordpress     latest    b44d413c437a   10 days ago   606MB
$
```

Let’s pull an image from Docker Hub...

```
$ sudo docker pull wordpress
Using default tag: latest
latest: Pulling from library/wordpress
1fe172e4850f: Already exists
012a3732d045: Pull complete
43092314d50d: Pull complete
4f615e42d863: Pull complete
cd39010a4efc: Pull complete
d983c9ce24de: Pull complete
ecbdd59ae430: Pull complete
9d02b88c8618: Pull complete
50a246031d43: Pull complete
a6c0267e6c34: Pull complete
787ca6348cef: Pull complete
da8ad43595e2: Pull complete
e191f9e80e29: Pull complete
fed8d3fd90f9: Pull complete
9ffdaa9000ed: Pull complete
5774aeca6412: Pull complete
6978431bb9e2: Pull complete
fb4d3fb05351: Pull complete
23d3af42839e: Pull complete
a5b33728e4a6: Pull complete
766e2b674cd0: Pull complete
Digest: sha256:abc1a527c810542eea7cd0be5c5e8a1d087f16c363a46178ea615e8083700077
Status: Downloaded newer image for wordpress:latest
docker.io/library/wordpress:latest
```

```
$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
ubuntu        latest    d2e4e1f51132   2 days ago    77.8MB
redis         latest    a10f849e1540   4 days ago    117MB
wordpress     latest    b44d413c437a   10 days ago   606MB
$
```

BUILDING AN IMAGE

Building an image from a code repository

$ git clone

Source control, download the code repository holding the Dockerfile & dependent files.

$ cd <folder>

changing into the directory

$ docker build .

Looks for a Dockerfile in the local directory and uses it to build a Docker image.

$ docker images

Show the local docker images (both downloaded from public and built locally).

$ docker run –ti

Run our locally built image.

CREATING YOUR OWN IMAGES

Steps to build your own images

Step 1: Start a container from existing image

```
$ docker run -it ubuntu
```

Step 2: Install the packages and performance the necessary commands

```
root@ca50f04282ae:/# apt-get update
root@ca50f04282ae:/# apt-get install -y python3 python3-pip
root@ca50f04282ae:/#pip3 install flask
...
root@61d6c2bd66f6:/# history
```

Step 3: Create and edit docker file

```
$ touch Dockerfile
$ nano Dockerfile
```

```
FROM ubuntu
RUN apt-get update
RUN apt-get install -y python3 python3-pip
RUN pip3 install flask
COPY app.py /app.py
RUN ["chmod", "+x", "/app.py"]
CMD ["python3", "/app.py"]
```

Step 4: Create an image

```
$ docker build . -t simple-web-app
```

The Dockerfile

$ docker build .

Uses Dockerfile to create a docker image.

- **FROM:** Selects the base image used to start the build process or can be set to **scratch** to build a totally new image.
- **MAINTAINER:** Lets you select a name and email address for the image creator.
- **RUN:** Creates image layers and executes commands within a container.
- **CMD:** Executes a single command within a container. Only one can exist in a Dockerfile.
- **WORKDIR:** Sets the path where the command defined with **CMD** is to be executed.
- **ENTRYPOINT:** Executes a default application every time a container is created with the image.
- **COPY:** Copies the files from the local host into the container's file system.
- **ADD:** Copies the files from the local host or remotely via a URL into the container's file system.
- **ENV:** Sets environment variables within the container.
- **EXPOSE:** Associates a specific port for networking binding.
- **USER:** Sets the UID (or username) of the user that is to run the container.
- **VOLUME:** Sets up a sharable directory that can be mapped to a local host directory.
- **LABEL:** Provides a label to identify the created Docker image.

NAMING, DISTRIBUTING.

Tag an image == give an image a name & version

hub.docker.com
Common docker repository offering free public repo's
Others are available
Requires signup

$ docker tag <image id> registry:version
Name is the repository URL you're planning to push the image to.
Version is arbitrary and under your
No URL defaults to Docker Hub.

$docker tag 8a0d280fc794 trxuk/testrepo:0.1
$docker images

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
trxuk/testrepo      0.1                 8a0d280fc794        23 minutes ago      203 MB
<none>              <none>              3ab28aaf0423        31 minutes ago      203 MB
```

Push images to a registry

SmartPro

$ docker login

Authenticates your local docker CLI with the docker registry. You'll need to signup for the docker registry at hub.docker.com (free) to get credentials.

$ docker push trxuk/testrepo

Name My docker hub account ID is trxuk. This will try to upload new images i've tagged locally as trxuk/testrepo to the docker registry for public consumption.

Other users could then
$docker run trxuk/testrepo
to run the latest version of my container image.