



Cognizant

KM Session

Presenter: Shivam Jha (631036)

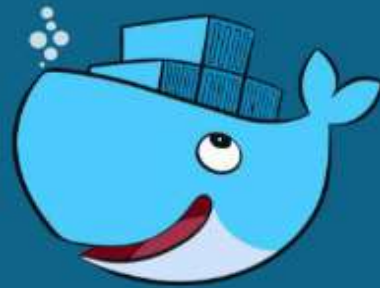
Date: 07/20/2018

© 2018 Cognizant

Course Outline

- Module 1 - Introduction to Docker Introducing Docker
- Module 2 - Image Creation and Sharing
- Module 3 - Docker Ecosystem

Module 1 – Introduction to Docker
















INTRODUCTION TO DOCKER

Objectives

- Shipping Transportation challenges
- Introducing Docker
- Docker Terminology
- Architecture of Docker

SHIPPING TRANSPORTATION CHALLENGES

Shipping Challenges

<div>Goods to be transported</div>      	?	?	?	?	?	?	?
	?	• Which truck/transport should we use to transport the goods?					
	?	• Are the workers skilled/trained enough to handle the goods?					
	?	The process of shipping the stuff is very tightly coupled to the goods being shipped,					
	?	In other words, the workers must be skilled/trained enough to handle the products.					
	?	For Example, Shipping of oil barrel					
	?	We are shipping Oil Barrels via various transports such as train, and ship.					
	?	During the shipment several loading and unloading of oil barrel requires various skilled workers.					
	?						
<div>Means of transport</div>       							

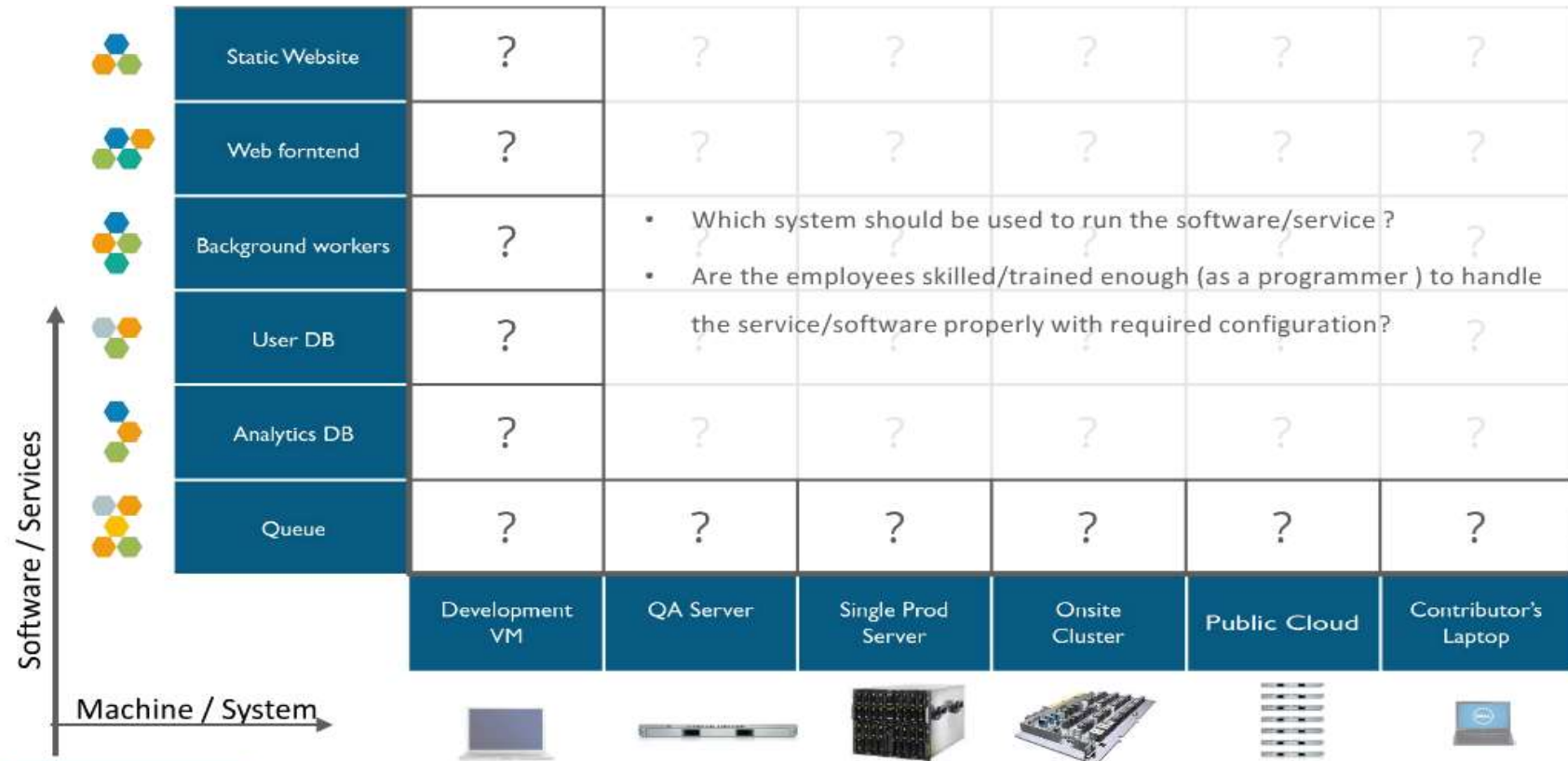
Shipping Challenges contd ...



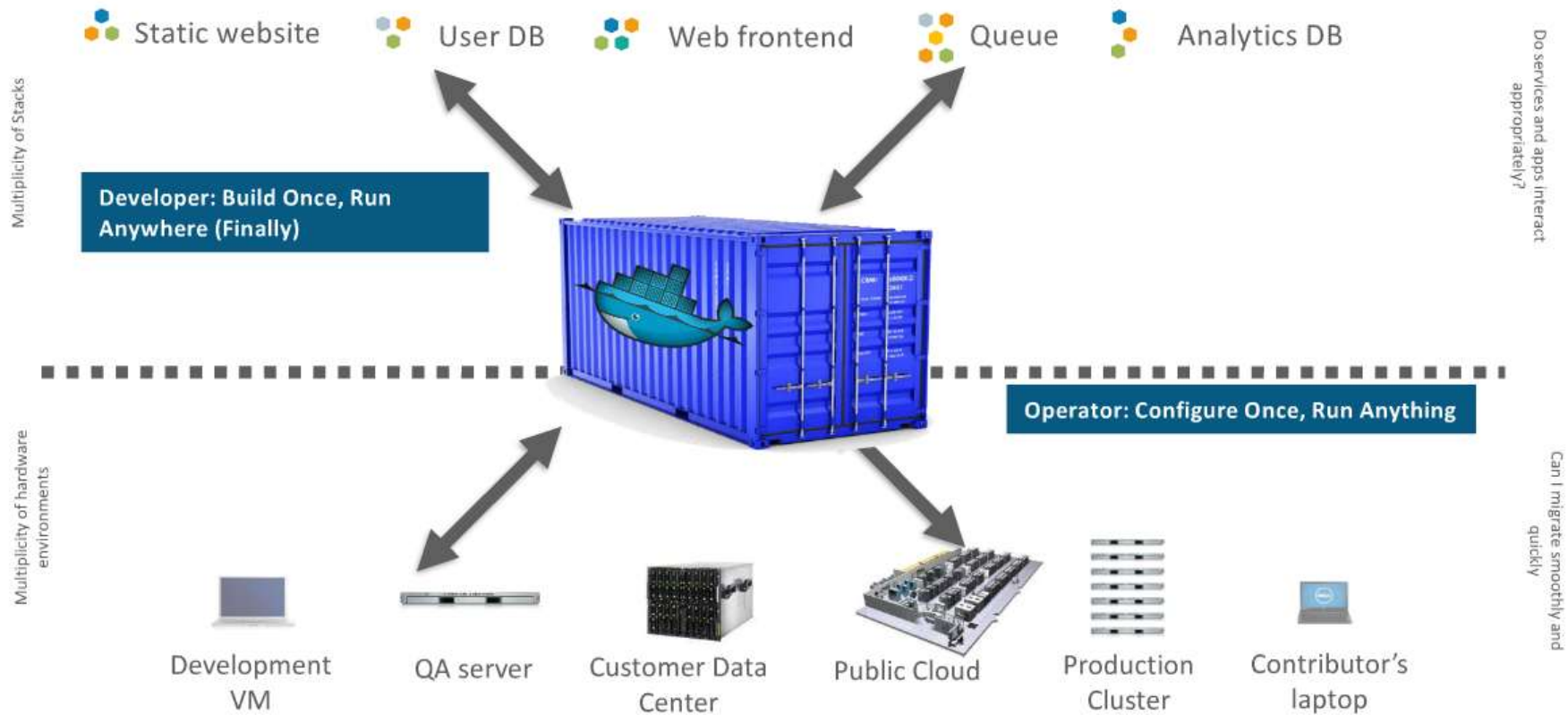
Shipping Solution



Challenges with System and Software Configurations

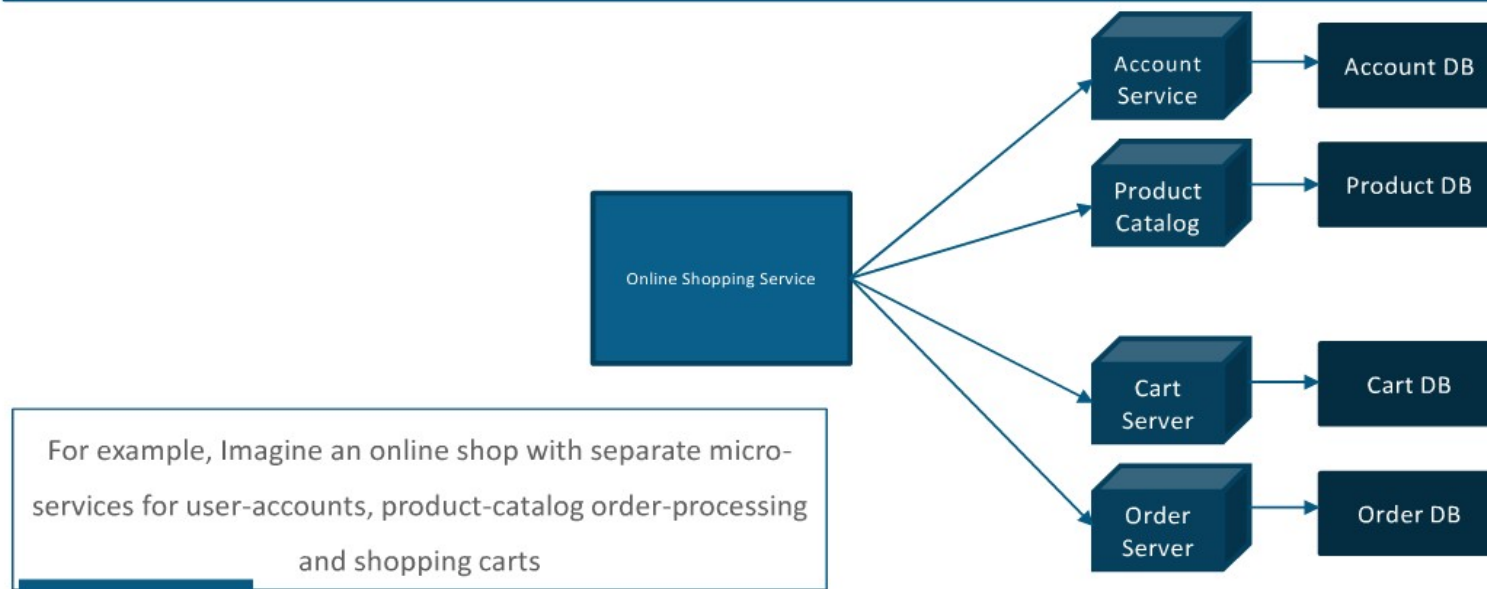


Docker – A shipping Container for node



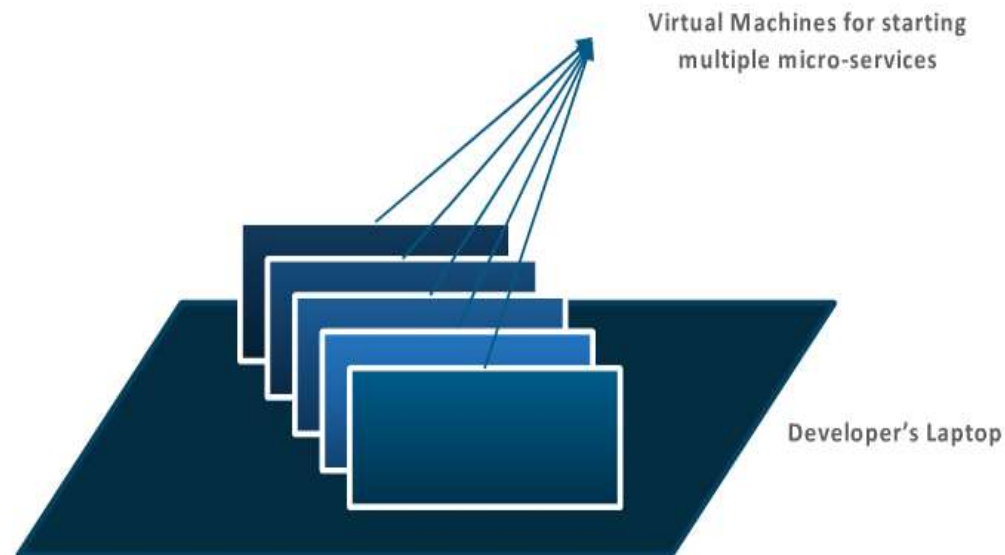
Use Case – Online Portal

Applications are easier to build and maintain when broken down to smaller, composable pieces which work together (micro-services). Each component is developed separately and the application is the sum of its constituent.



Online Portal – Handling micro-services

Developing an application requires starting several of micro-services in one machine. So if you are starting five of those services you require five VMs on that machine.



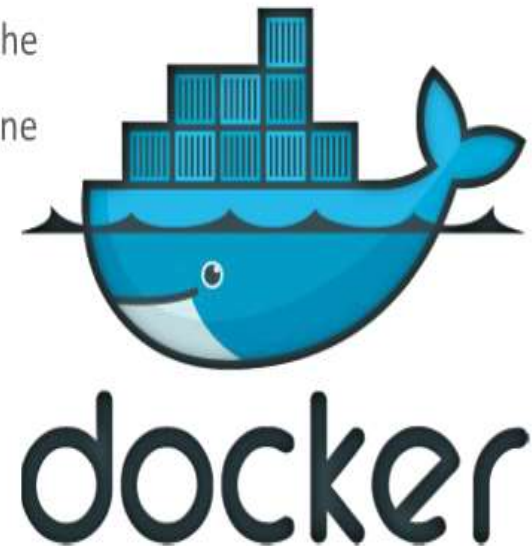


Let us see what is Docker!

What is Docker ?

- Docker is a tool designed to create, deploy, and run applications with ease by using containers.
- It allows a developer to package up an application with all of the requirements such as libraries and other dependencies and ship it all as one package.
- It ensure that your application works seamlessly in any environment; be it Development, Test or Production.

“BUILD, SHIP & RUN ANY SOFTWARE ANY WHERE”

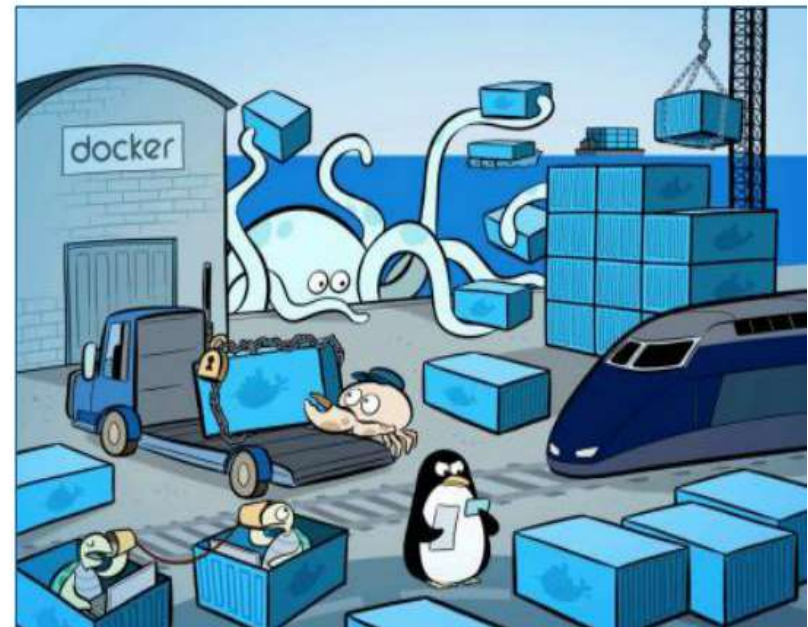


Who uses Docker ?

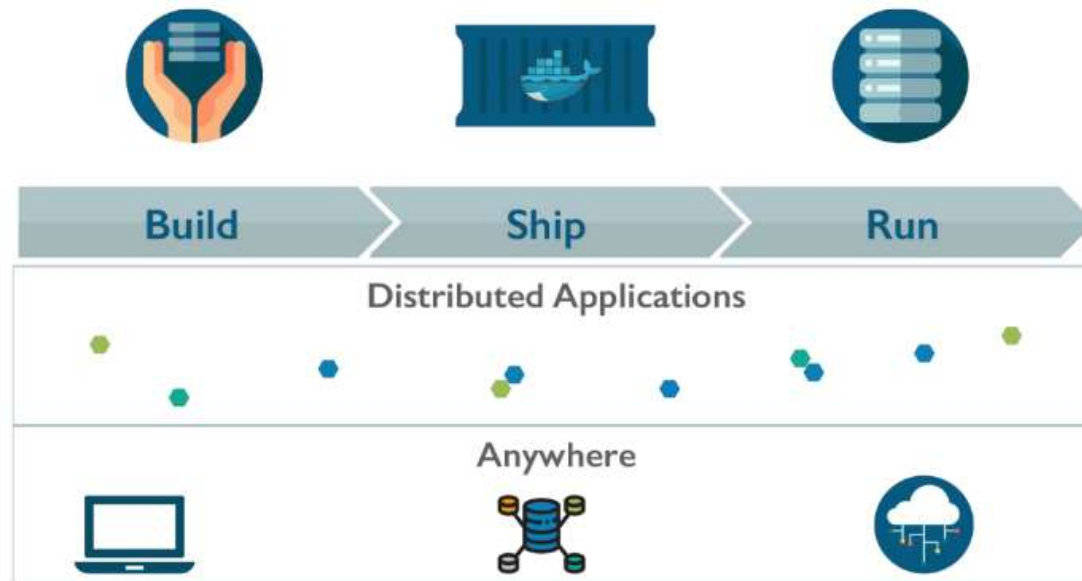
Developer: Docker helps the developer to focus only on building great software by automating the repetitive tasks of setting up and configuring development environment.

Sysadmin: Docker helps the sysadmin to streamline the software delivery, such as develop and deploy bug fixes and new features without any roadblock.

Enterprise: Docker is at the heart of the modern app platform, bridging developer and IT, Linux and Windows. It works in the cloud just as well as on premise; and supports both traditional and microservices architectures.



Docker containers are popular !



Develop an app using Docker containers with any language and any toolchain.

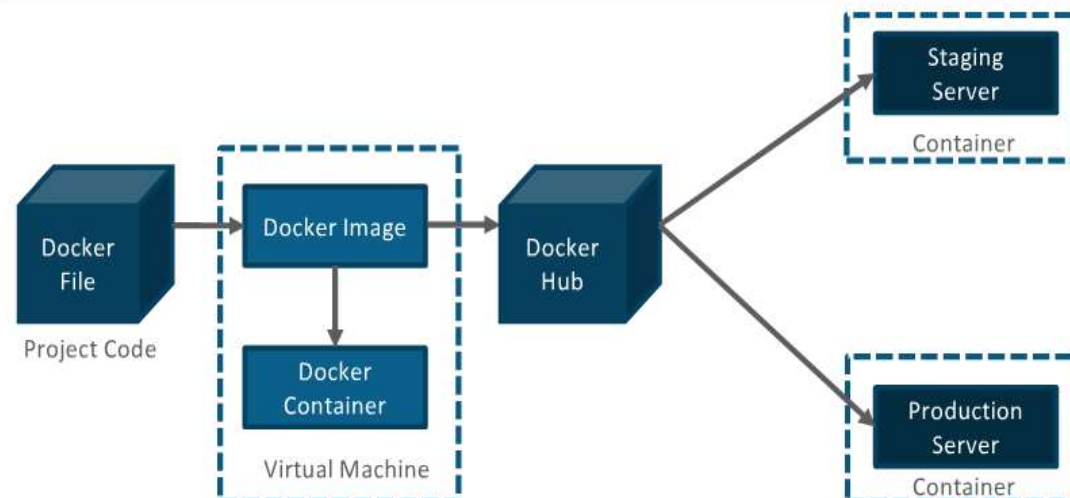
Ship the "Dockerized" app and dependencies anywhere - to QA, teammates, or the cloud - without breaking anything.

Scale to 1000s of nodes, move between data centers and clouds, update with zero downtime and more.

Docker in a Nutshell



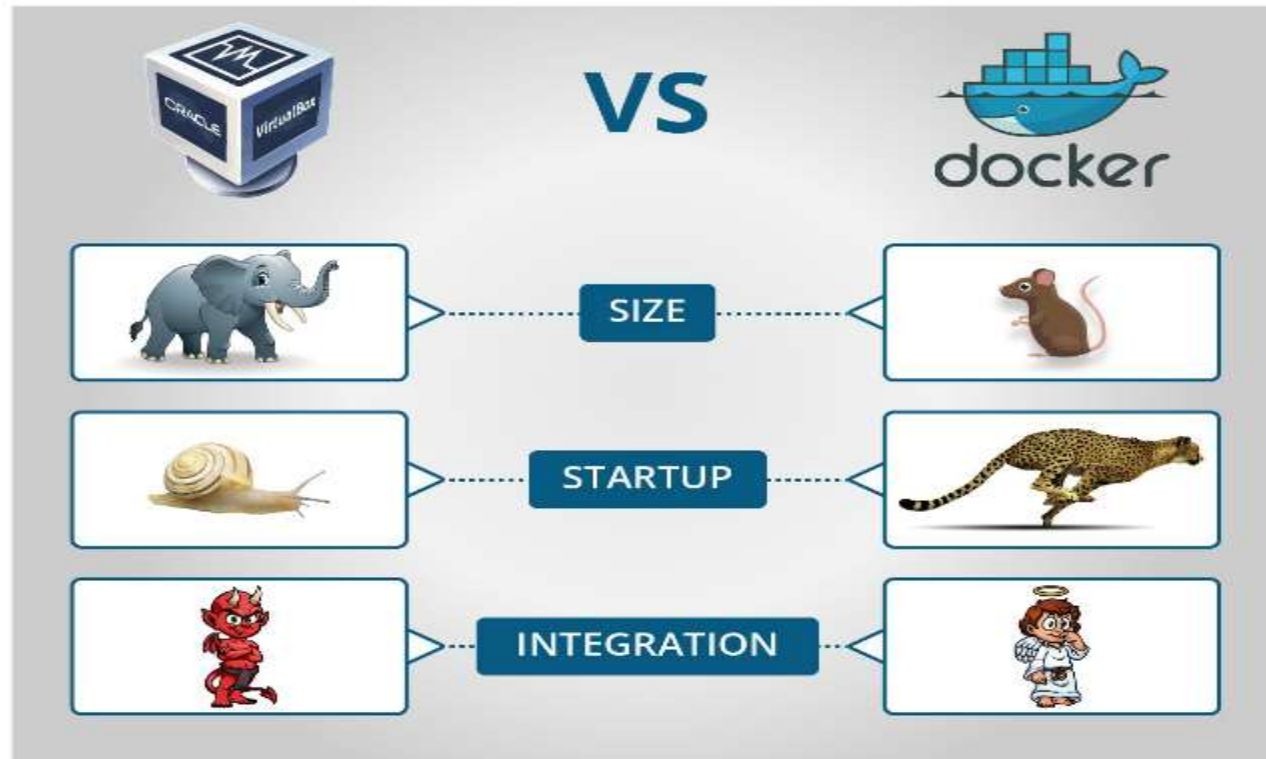
- Docker file builds a Docker image and that image contains all the project's code
- You can run that image to create as many docker containers as you want
- The created Images can be uploaded on Docker hub from where the image can be pulled and built in a container





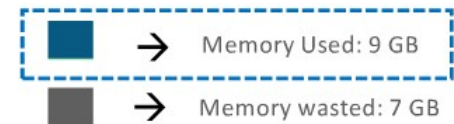
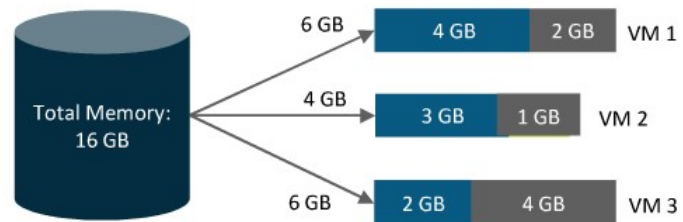
BENEFITS OF DOCKER OVER VM

VM vs. Docker



Resource / Memory Management

In case of Virtual Machines



7 GB of memory is blocked and cannot be allotted to a new VM

VM vs Docker

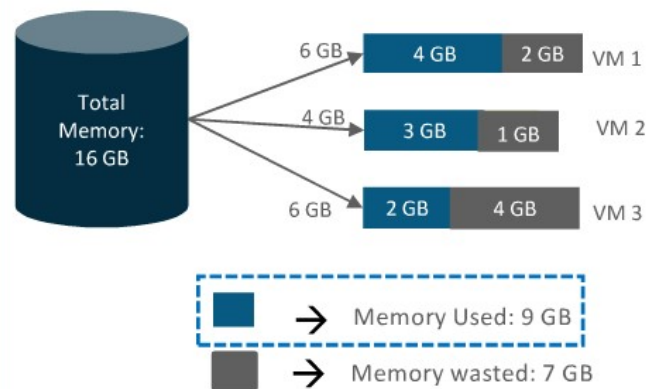
1 Size

2 Startup

3 Integration

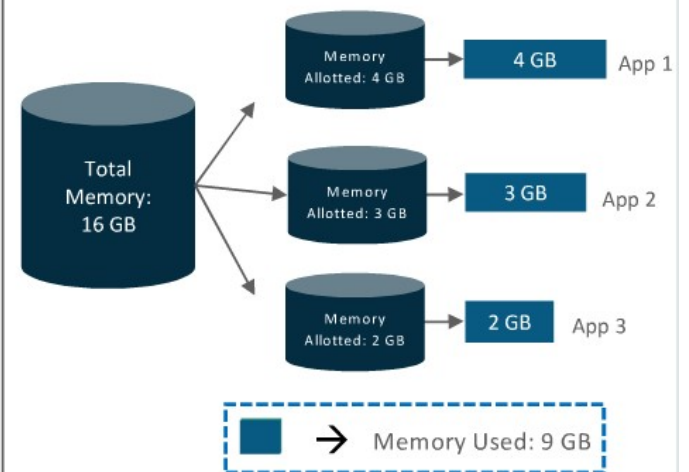
Resource / Memory Management

In case of Virtual Machines



7 GB of memory is blocked and cannot be allotted to a new VM

In case of Docker



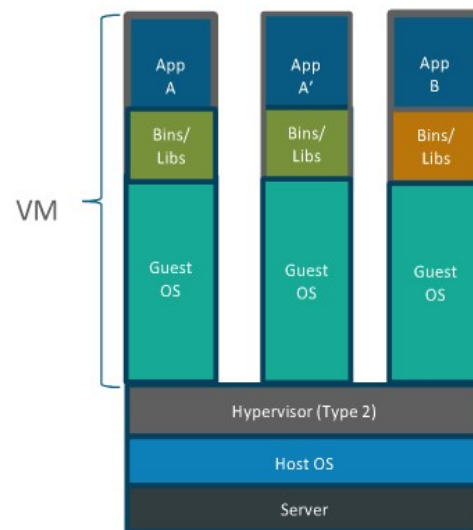
Only 9 GB memory utilized;
7 GB can be allotted to a new Container

VM vs Docker

- 1 Size
- 2 Startup
- 3 Integration

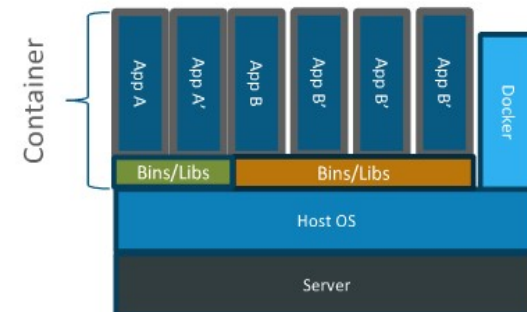
Building & Deployment

In case of Virtual Machines



New Builds → **Multiple OS** → Separate Libraries
→ Heavy → **More Time**

In case of Docker

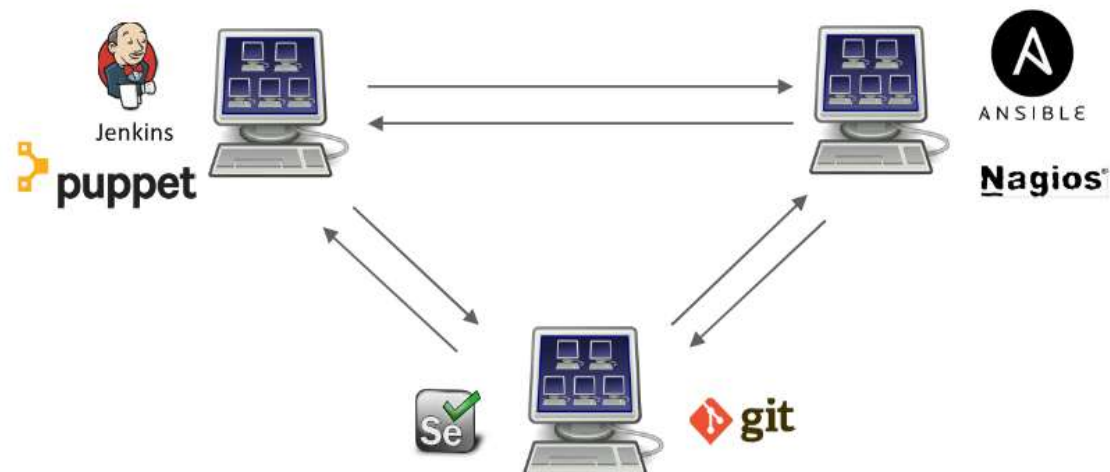


New Builds → **Same OS** → Separate Libraries →
Lightweight → **Less Time**

Integration in VMs

Integration in virtual machines is possible, but:

- **Costly** due to infrastructure requirements
- Not **easily scalable**



Install Docker

There are loads of ways and places to install Docker. There's Windows, there's Mac, Linux also in the cloud, on premises, on your laptop, and more...

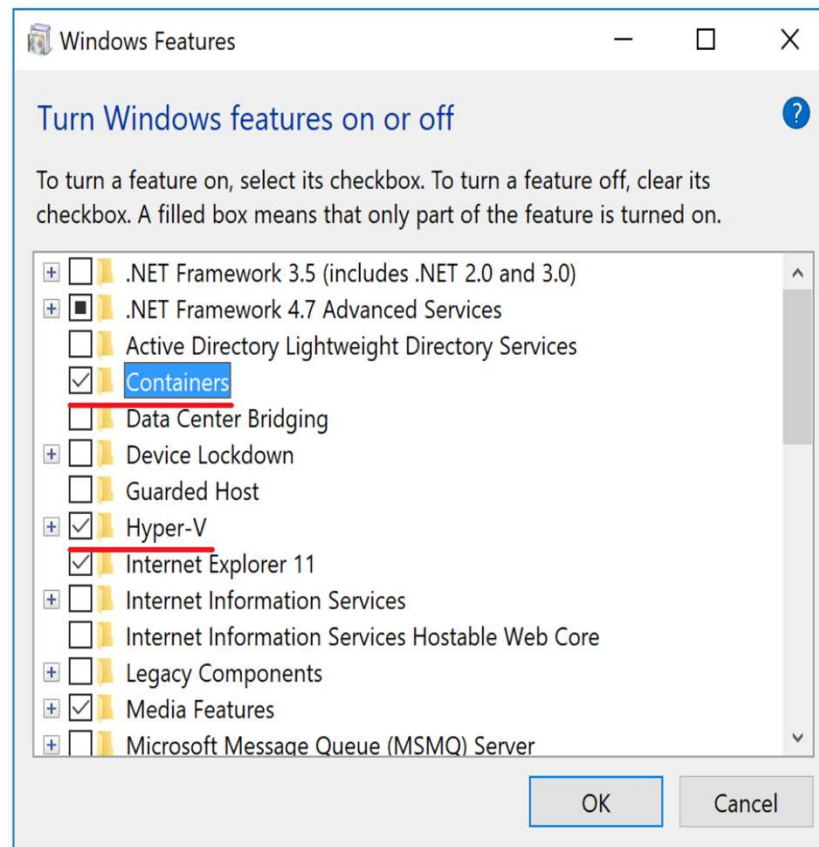
Docker for Windows (DfW)

Pre-requisites for Docker for Windows requires:

- Windows 10 Pro | Enterprise | Education
- Must be 64-bit Windows 10
- The Hyper-V and Containers features must be enabled in Windows
- Hardware virtualization support must be enabled in your system's BIOS

The first thing to do in Windows 10, is make sure the Hyper-V and Containers features are installed and enabled.

- Right-click the Windows Start button and choose Apps and Features.
- Click the Programs and Features link (a small link on the right).
- Click Turn Windows features on or off.
- Check the Hyper-V and Containers checkboxes and click OK.



Install Docker on Windows

Once you've installed the Hyper-V and Containers features, and restarted your machine, it's time to install Docker for Windows.

- Head over to <https://www.docker.com/get-docker> and click the GET DOCKER COMMUNITY EDITION link.
- Click the Download from Docker Store link beneath the DOCKER CE FOR WINDOWS section. This will take you to the Docker Store and you may need to login with your Docker ID.
- Click one of the Get Docker download links.

Docker for Windows has a stable and edge channel. The edge channel contains newer features but may not be as stable.

An installer package called Docker for Windows Installer.exe will be downloaded to your default downloads directory.

- Locate and launch the installer package downloaded in the previous step.
- Open a command prompt or PowerShell terminal and try the following commands:
C:\> docker version

Verify the Installation

Docker for Windows includes the Docker Engine (client and daemon), Docker Compose, Docker Machine, and the command line. Use the following commands to verify that each was successfully installed:

```
C:\> docker --version
```

```
Docker version 18.01.0-ce, build 03596f5
```

```
C:\> docker-compose --version
```

```
docker-compose version 1.18.0, build 8dd22a96
```

```
C:\> docker-machine --version
```

```
docker-machine.exe version 0.13.0, build 9ba6da9
```

Installing Docker – Ubuntu Linux

The first thing you need to decide is which edition to install. There are currently two editions:

- Community Edition (CE)
- Enterprise Edition (EE)

Docker CE is free and is the version we'll be demonstrating. Docker EE is the same as CE, but comes with commercial support and access to other Docker products such as Docker Trusted Registry and Universal Control Plane.

Note - You should ensure that your system is up-to-date with the latest packages and security patches before continuing .

Installing Docker – Ubuntu Linux

1. Open a new shell on your Linux machine.
2. Use wget to retrieve and run the Docker install script from <https://get.docker.com> and pipe it through your shell.

```
$ wget -qO- https://get.docker.com/ | sh
```

If you would like to use Docker as a non-root user, you should now consider adding your user to the "docker" group with something like:

```
$ sudo usermod -aG docker shivam  
$ cat /etc/group | grep docker  
docker:x:999:shivam
```

```

root@shivam5:~# wget -qO- https://get.docker.com/ | sh
# Executing docker install script, commit: 36b78b2
+ sh -c apt-get update -qq >/dev/null
+ sh -c apt-get install -y -qq apt-transport-https ca-certificates curl >/dev/null
+ sh -c curl -fsSL "https://download.docker.com/linux/ubuntu/gpg" | apt-key add -qq - >/dev/null
+ sh -c echo "deb [arch=amd64] https://download.docker.com/linux/ubuntu trusty edge" > /etc/apt/sources.list.d/docker.list
+ [ ubuntu = debian ]
+ sh -c apt-get update -qq >/dev/null
+ sh -c apt-get install -y -qq --no-install-recommends docker-ce >/dev/null
+ sh -c docker version
Client:
Version:      18.06.0-ce
API version:  1.38
Go version:   go1.10.3
Git commit:   0ffa825
Built:        Wed Jul 18 19:10:22 2018
OS/Arch:      linux/amd64
Experimental: false

Server:
Engine:
Version:      18.06.0-ce
API version:  1.38 (minimum version 1.12)
Go version:   go1.10.3
Git commit:   0ffa825
Built:        Wed Jul 18 19:08:26 2018
OS/Arch:      linux/amd64
Experimental: false

If you would like to use Docker as a non-root user, you should now consider
adding your user to the "docker" group with something like:

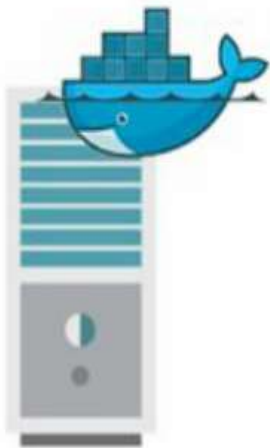
    sudo usermod -aG docker your-user

root@shivam5:~# sudo usermod -aG docker user
root@shivam5:~# cat /etc/group | grep docker
docker:x:999:user
root@shivam5:~#

```

Docker Daemon

- The Docker daemon runs on a host machine.
- The user uses the Docker client to interact with the daemon.



Why Use Docker Daemon?

- Responsible for creating, running, and monitoring containers
- Building and storing images

Docker Client

- The Docker client is the primary UI to Docker.
- It accepts commands and configuration flags and communicates with a Docker daemon via HTTP.
- One client can even communicate with multiple unrelated daemons.



Why Use Docker Client?

- Since all communication has to be done over HTTP, it is easy to connect to remote Docker
- The API used for communication with daemon allows developers to write programs that interface directly with the daemon, without using Docker

Docker Registry

- Docker Registry is a storage component for Docker Images
- We can store the Images in either Public/Private repositories
- Docker Hub is Docker's very own cloud repository



Why Use Docker Registries?

- Control where your images are being stored
- Integrate image storage with your in-house development workflow

Private or Public Registry

When choosing between these, some points to consider are:

- Performance, depending mainly on roll-out frequency and cluster size
- Security issues such as access control and digitally signing Docker image

With a private registry,

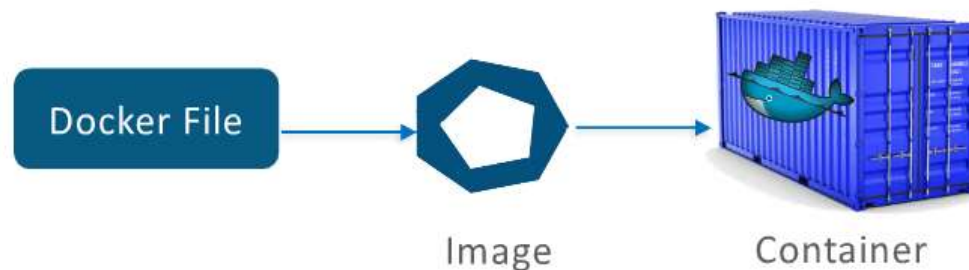
- You are in full control.
- No external dependencies in the CD pipeline, so your build is faster (or appear to be faster)
- Have to manage storage yourself – and this can increase drastically as the adoption of DevOps and number of build increase.

Before doing any decisions,

- Evaluate how many builds/images are being pushed and the average increase in size
- Also factor in growth rates for your applications/builds
- These will give you metrics' for determining network bandwidth and storage requirements

What is an image ?

- An image is a text file with a set of pre-written commands, usually called as a Docker file
- Docker Images are made up of multiple layers which are read-only filesystem
- A layer is created for each instruction in a Docker file and placed on top of the previous layer
- When an image is turned into a container the Docker engine takes the image and adds a read-write filesystem on top (as well as initializing various settings such as the IP address, name, ID, and resource limits)



Docker Images & Containers



Docker Images

- Read only template used to create containers
- Built by Docker users
- Stored in DockerHub or your local registry

Docker Images & Containers



Docker Images

- Read only template used to create containers
- Built by Docker users
- Stored in DockerHub or your local registry

run

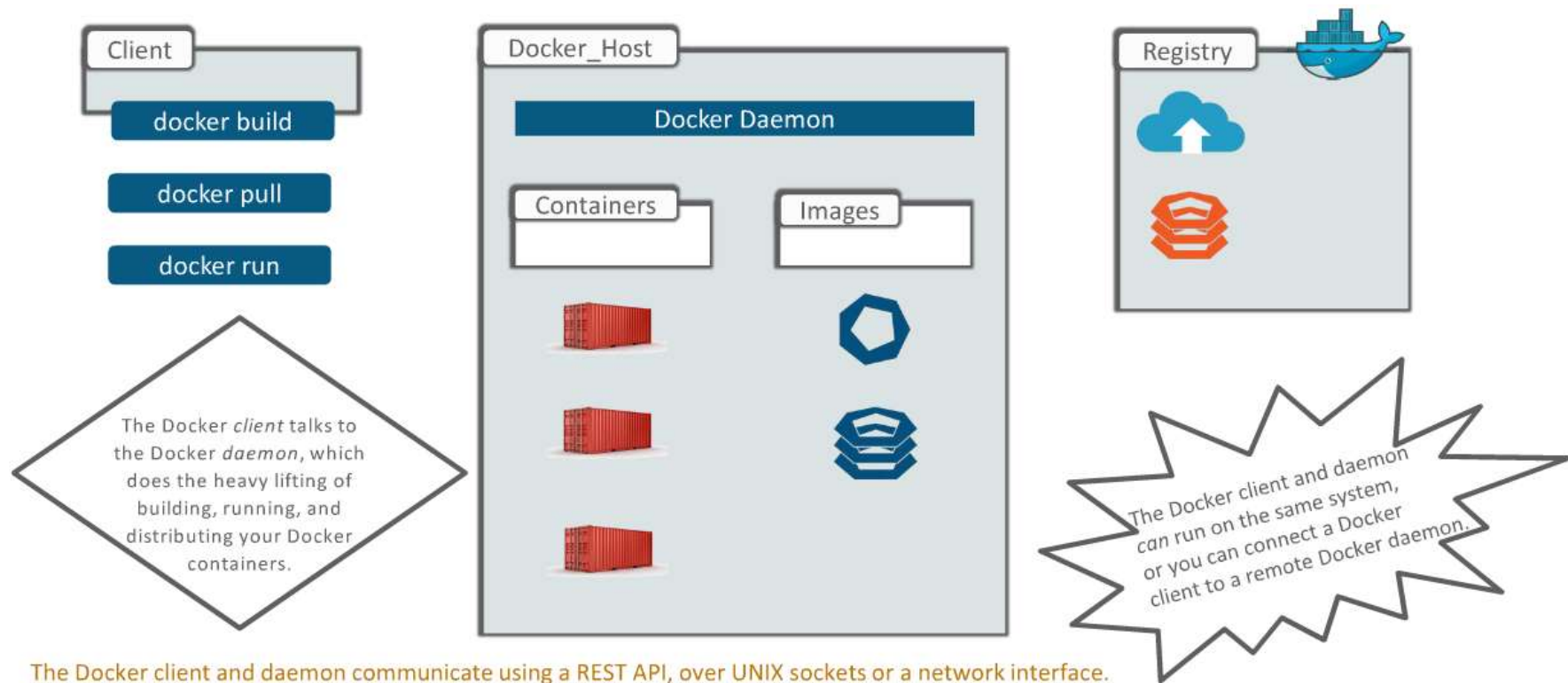


Docker Containers

- Isolated application platform
- Contains everything needed to run the application
- Built from one or more images

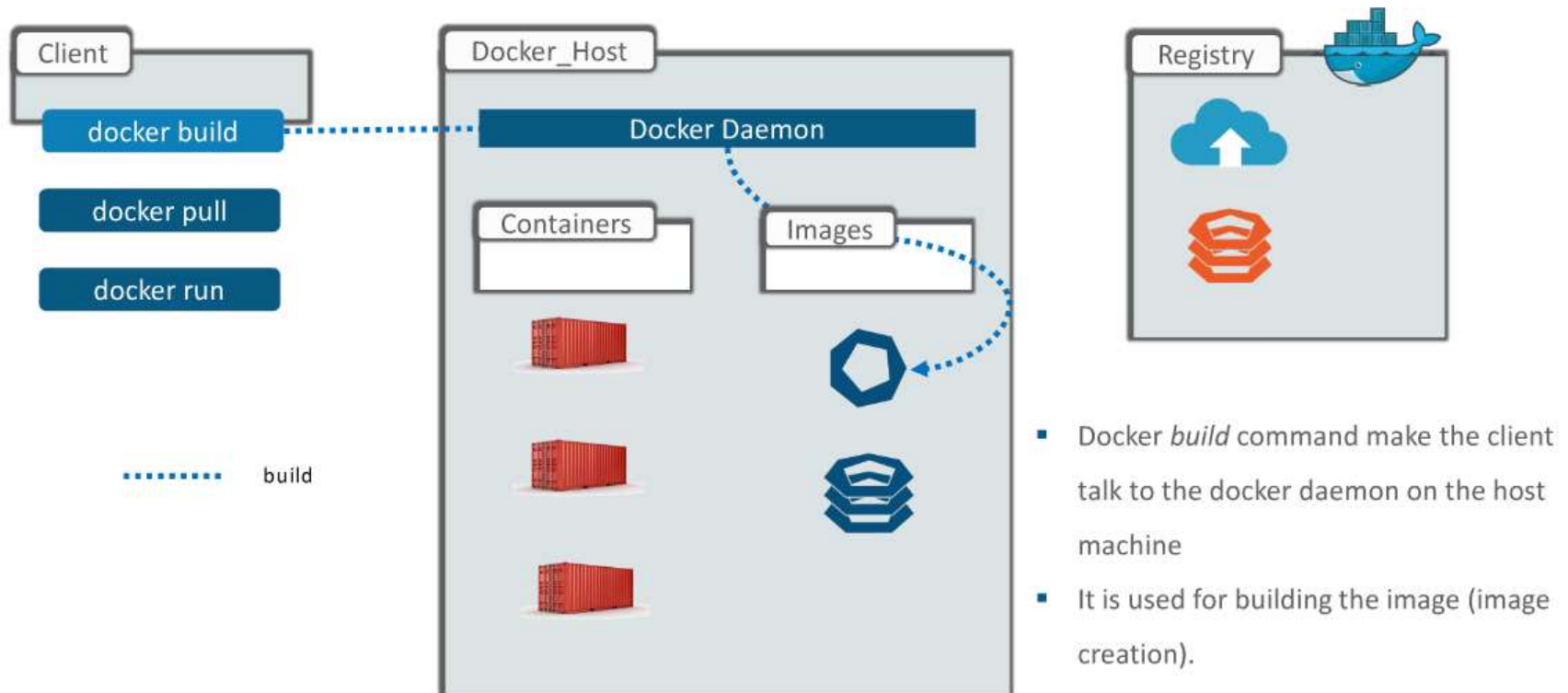
DOCKER ARCHITECTURE

Docker Architecture

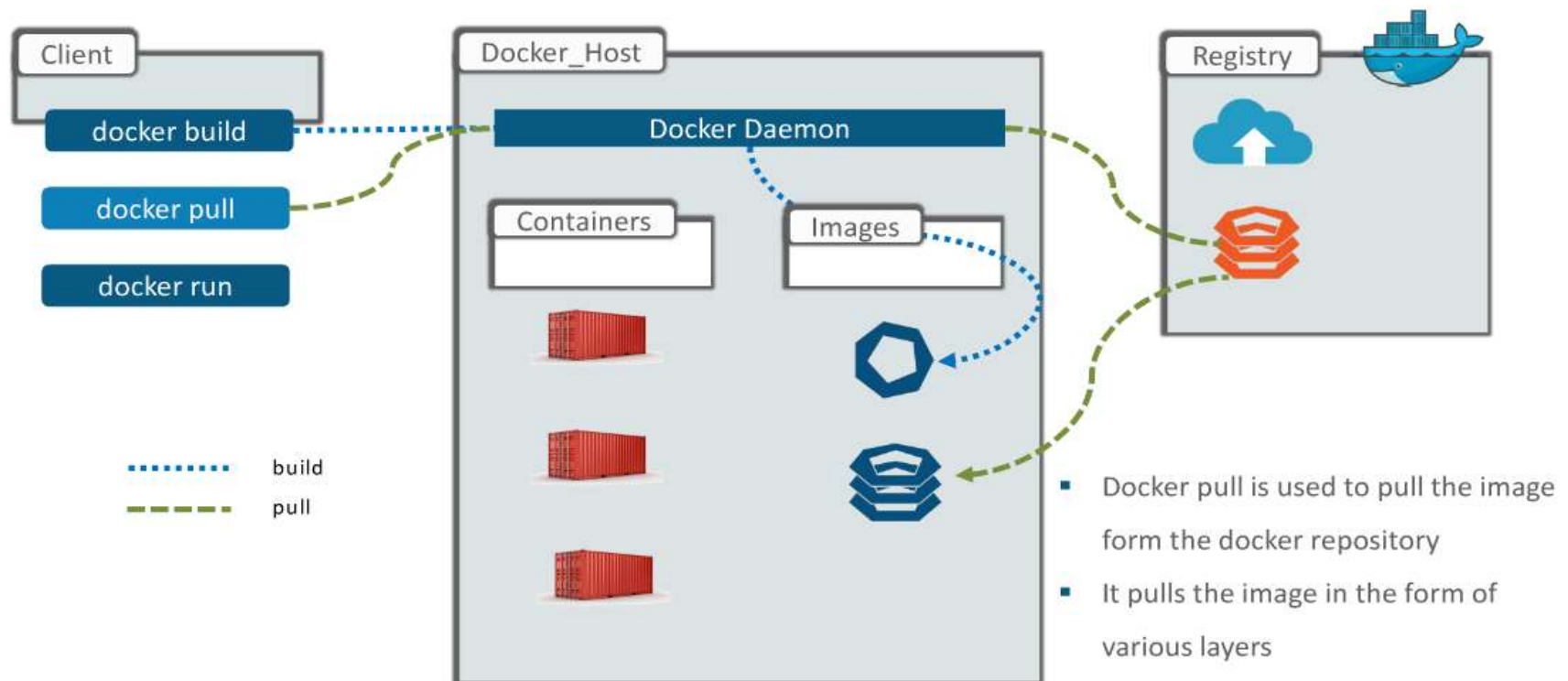


The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface.

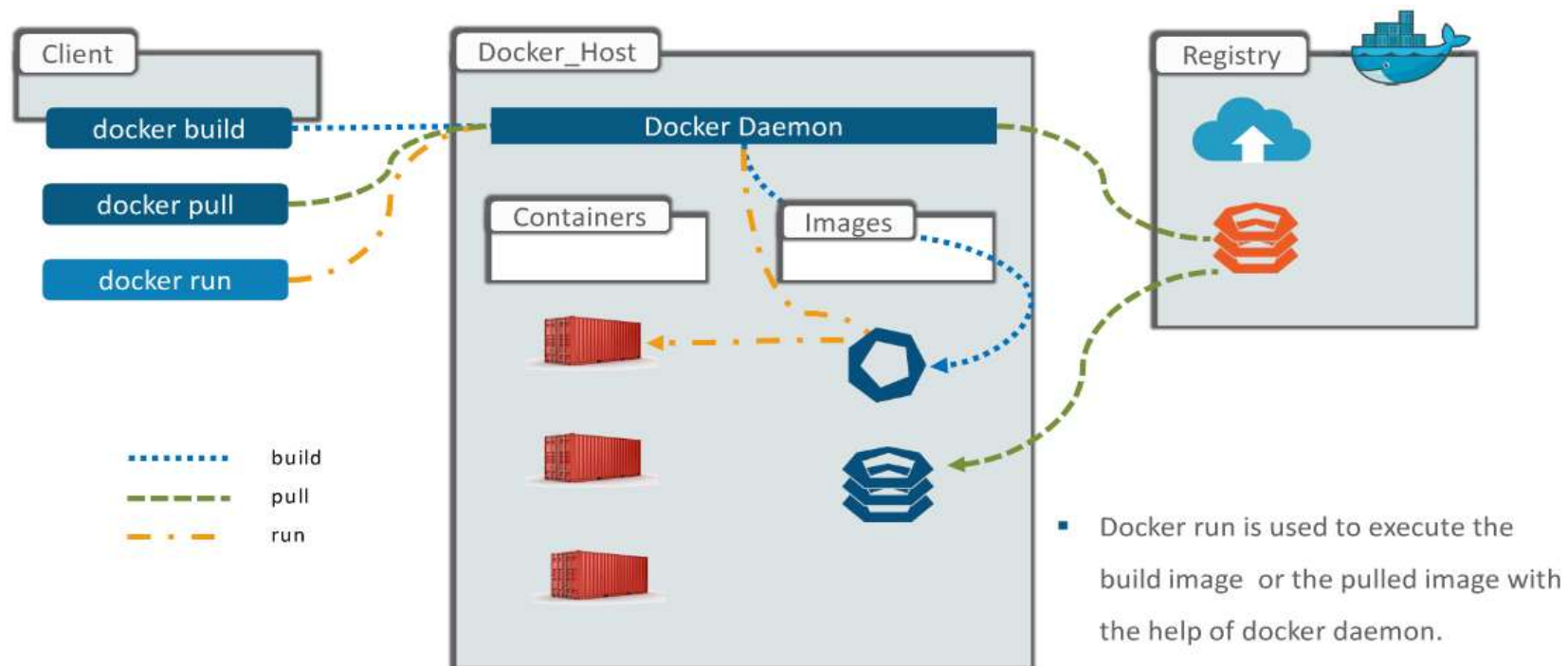
Docker Architecture contd...



Docker Architecture contd...



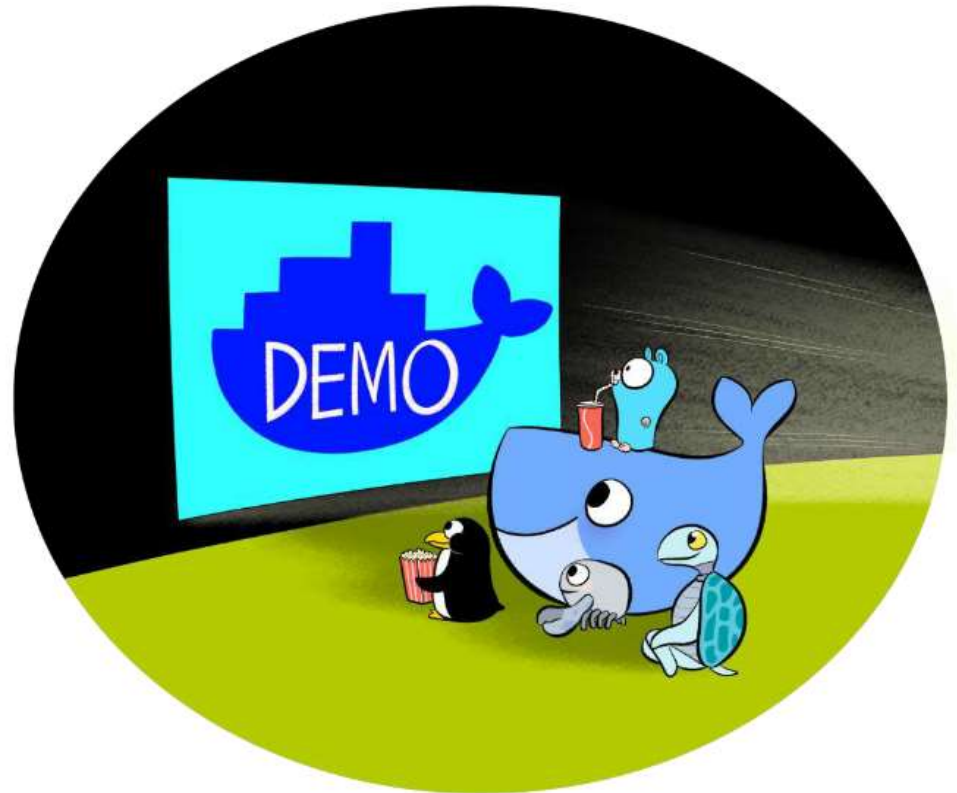
Docker Architecture contd...



DOCKER HANDS - ON

Docker Hands - On

- Version Check
- Searching hello-world
- Pull hello-world
- Execute hello-world



Version Check

```
[user@shivam3 ~]$ docker version
Client:
 Version:           18.06.0-ce
 API version:       1.38
 Go version:        go1.10.3
 Git commit:        0ffa825
 Built:             Wed Jul 18 19:08:18 2018
 OS/Arch:           linux/amd64
 Experimental:      false

Server:
 Engine:
  Version:          18.06.0-ce
  API version:      1.38 (minimum version 1.12)
  Go version:       go1.10.3
  Git commit:       0ffa825
  Built:            Wed Jul 18 19:10:42 2018
  OS/Arch:          linux/amd64
  Experimental:     false
[user@shivam3 ~]$
```

Docker Basic Commands

- `$ docker help` - Displays all the useful commands for Docker and other general help commands
- `$ docker images` - Displays a list of existing images in Docker system. It also displays the following details:
 - **REPOSITORY:** Name of the repository
 - **TAG:** Every image has an attached tag.
 - **IMAGE ID:** Each image is assigned a unique ID
 - **CREATED:** The date when the image was created
 - **SIZE:** The size of the image

Docker Basic Commands

```
[user@shivam3 ~]$ docker help

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
  --config string      Location of client config files (default "/home/user/.docker")
  -D, --debug          Enable debug mode
  -H, --host list      Daemon socket(s) to connect to
  -l, --log-level string Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
  --tls               Use TLS; implied by --tlsverify
  --tlscacert string  Trust certs signed only by this CA (default "/home/user/.docker/ca.pem")
  --tlscert string    Path to TLS certificate file (default "/home/user/.docker/cert.pem")
  --tlskey string     Path to TLS key file (default "/home/user/.docker/key.pem")
  --tlsverify         Use TLS and verify the remote
  -v, --version       Print version information and quit

Management Commands:
  config      Manage Docker configs
  container   Manage containers
  image       Manage images
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes
```

Docker Basic Commands

```
[user@shivam3 ~]$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
------------	-----	----------	---------	------

```
[user@shivam3 ~]$
```




KEEP CHALLENGING™

