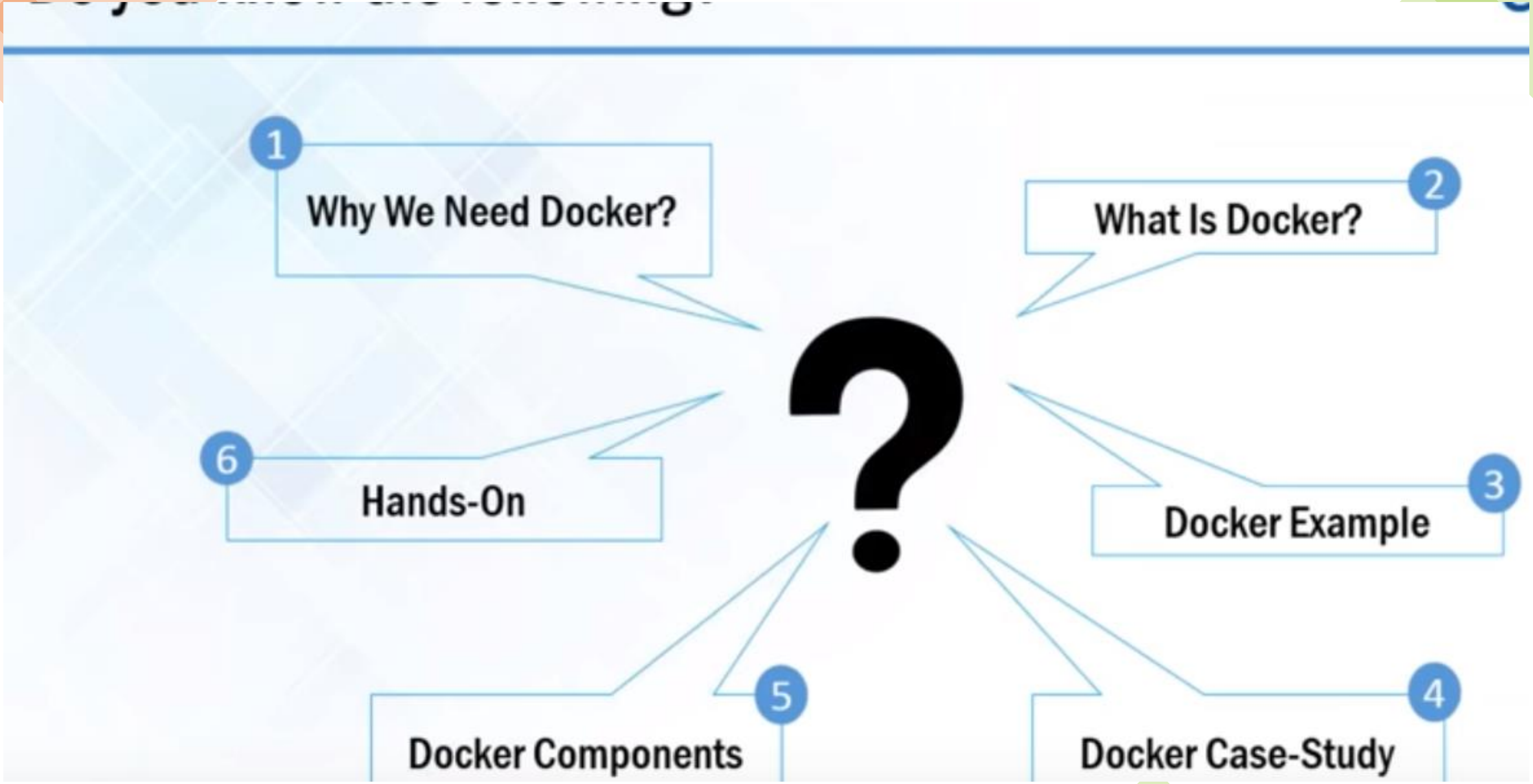
Agenda

- Docker Fundamentals and Training

01

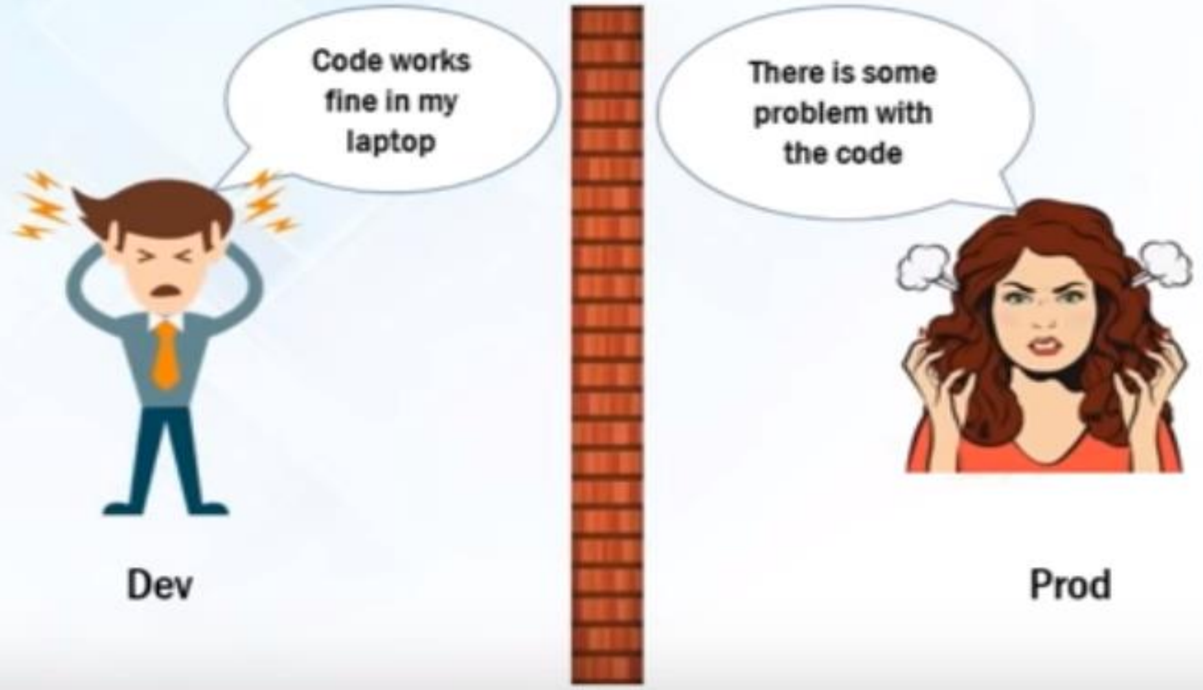
Introduction to Docker



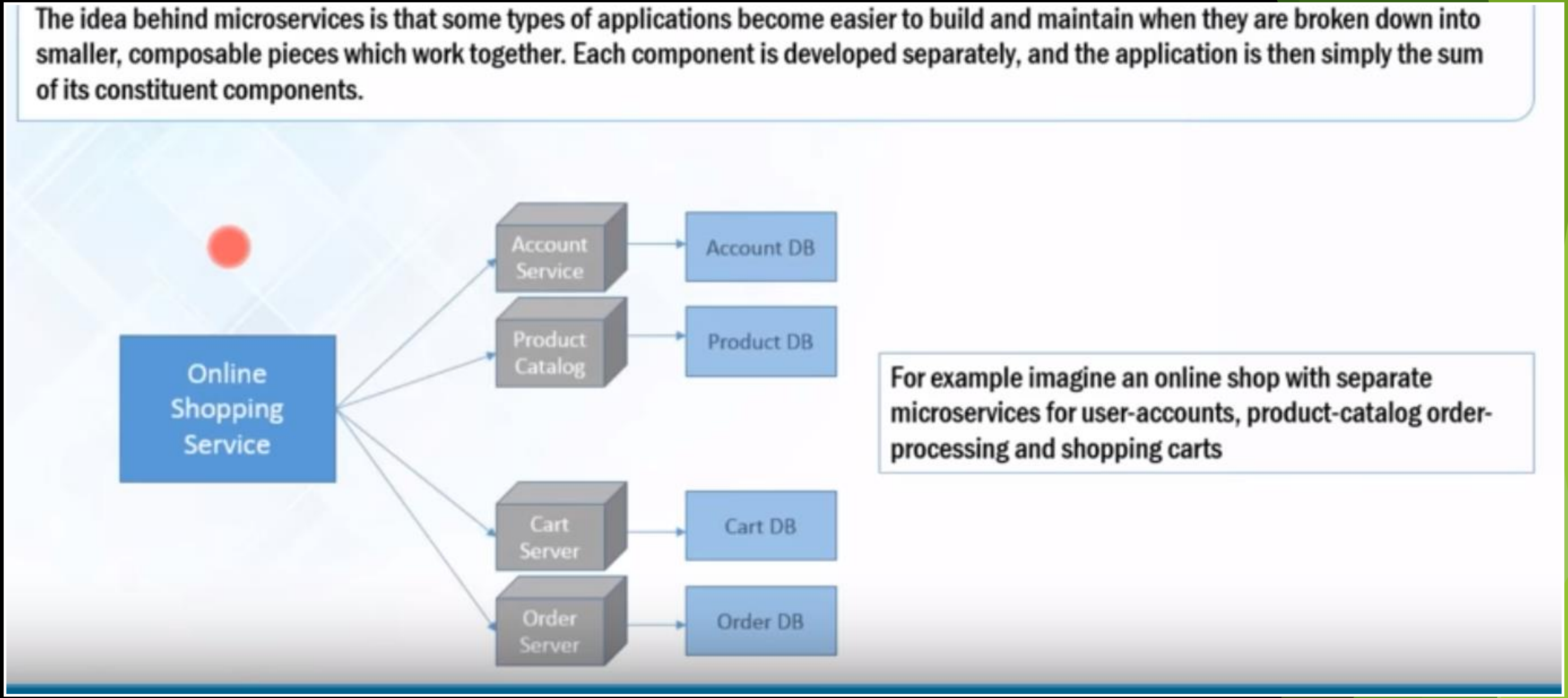
1. Why We Need Docker?
2. What Is Docker?
3. Docker Example
4. Docker Case-Study
5. Docker Components
6. Hands-On

Problems before Docker

An application works in developer's laptop but not in testing or production. This is due to difference in computing environment between Dev, Test and Prod.


Code works fine in my laptop
There is some problem with the code
Dev
Prod

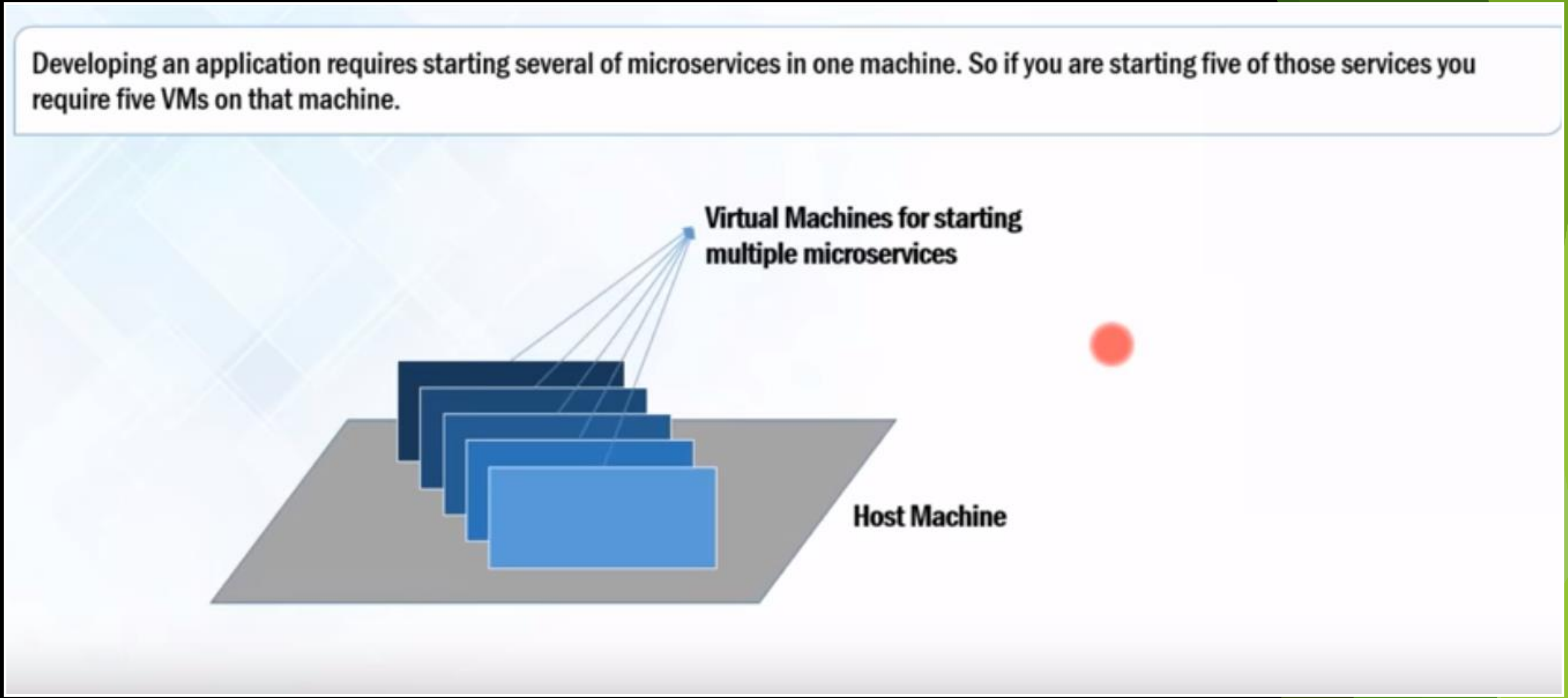
In Dev there can be a software that is upgraded and in Prod the old version of software might be present



The idea behind microservices is that some types of applications become easier to build and maintain when they are broken down into smaller, composable pieces which work together. Each component is developed separately, and the application is then simply the sum of its constituent components.

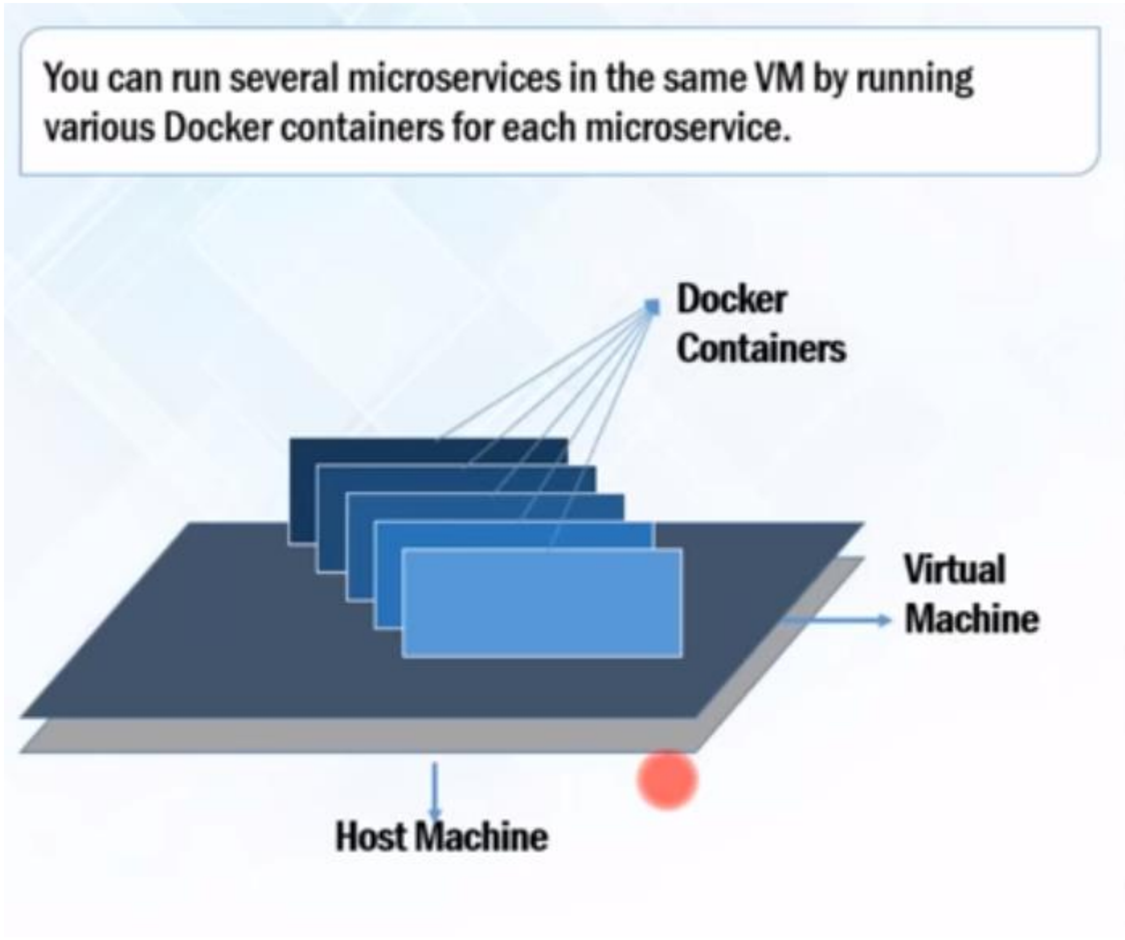
For example imagine an online shop with separate microservices for user-accounts, product-catalog order-processing and shopping carts

Problems before Docker

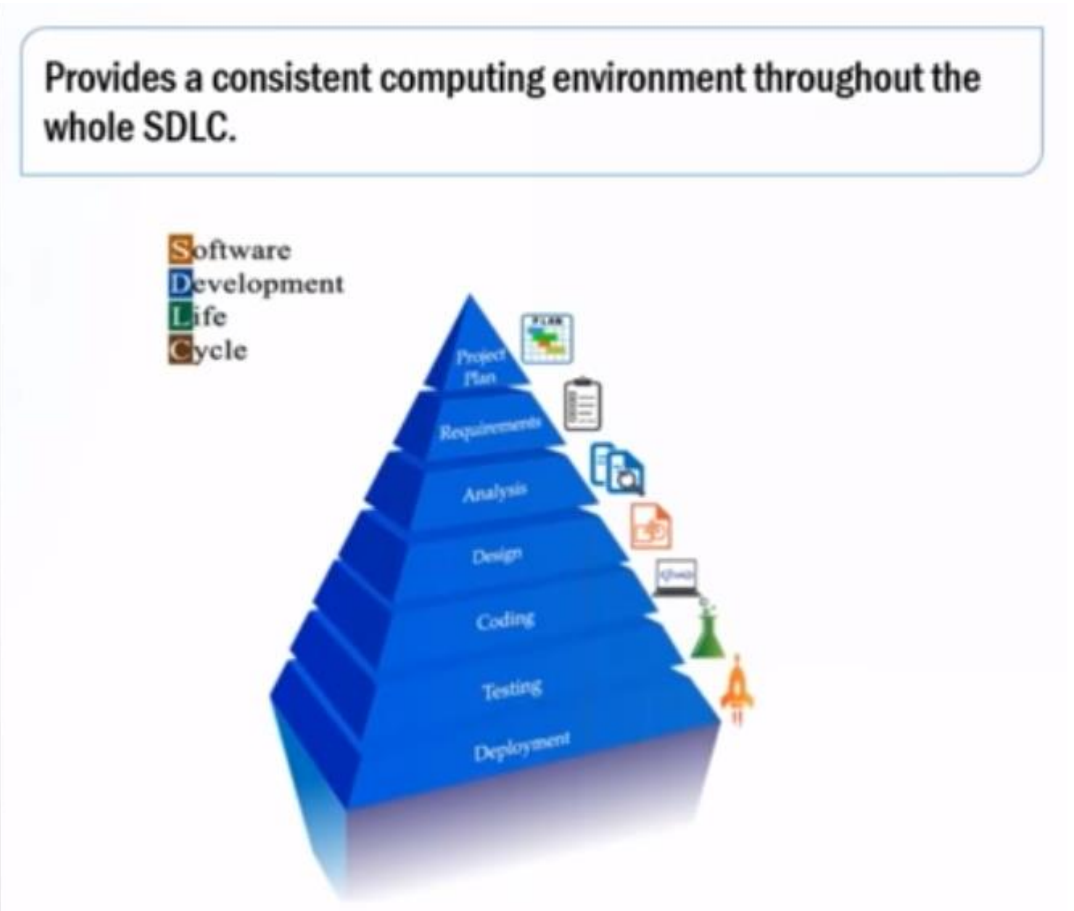

Developing an application requires starting several of microservices in one machine. So if you are starting five of those services you require five VMs on that machine.

Problems before Docker

How Docker Solves these problems

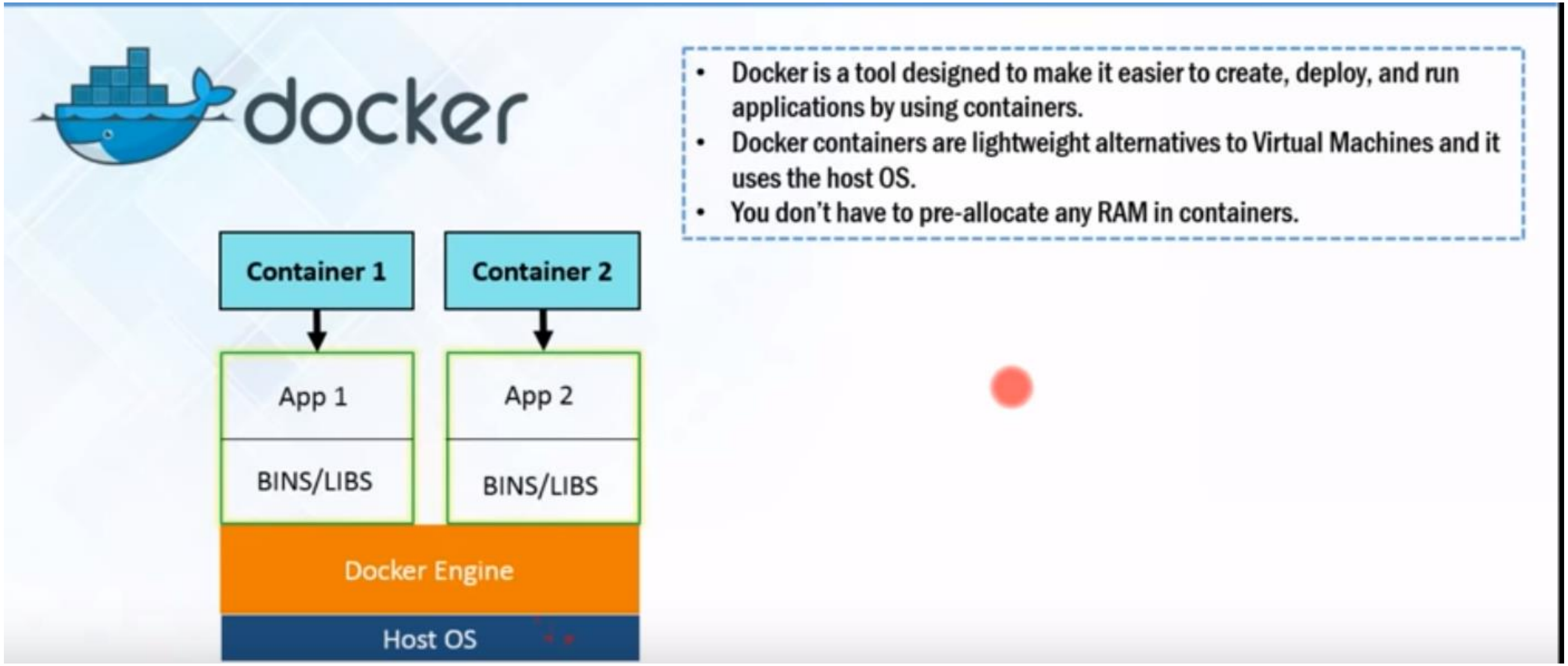


You can run several microservices in the same VM by running various Docker containers for each microservice.

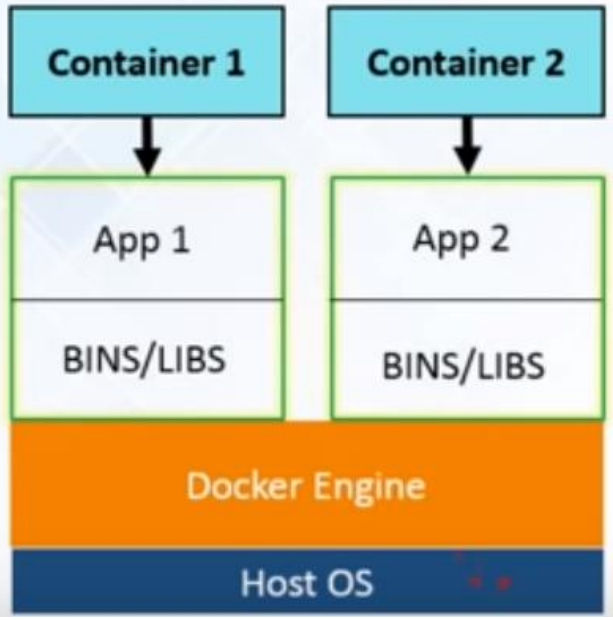


Provides a consistent computing environment throughout the whole SDLC.

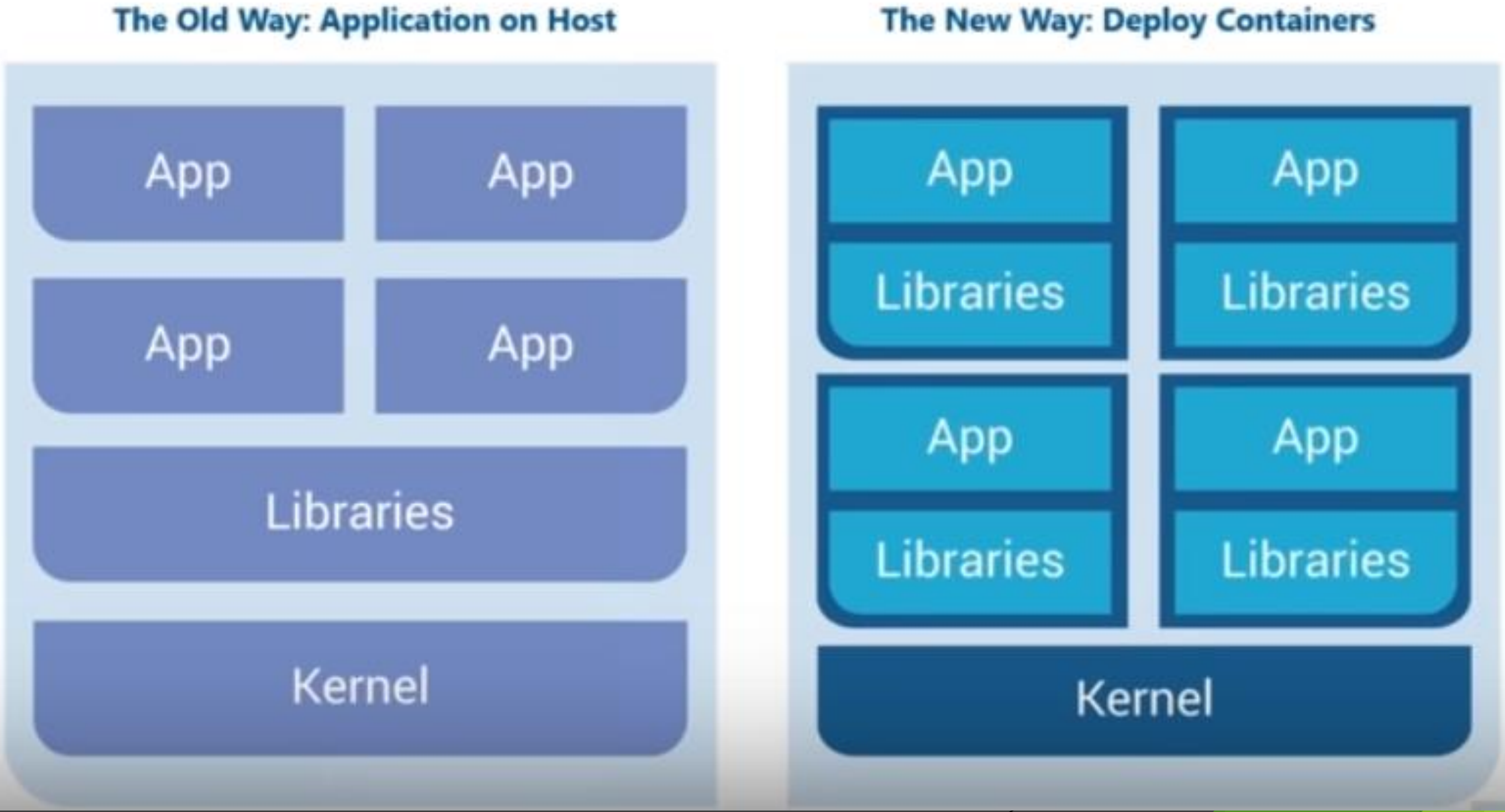
What is Docker ?



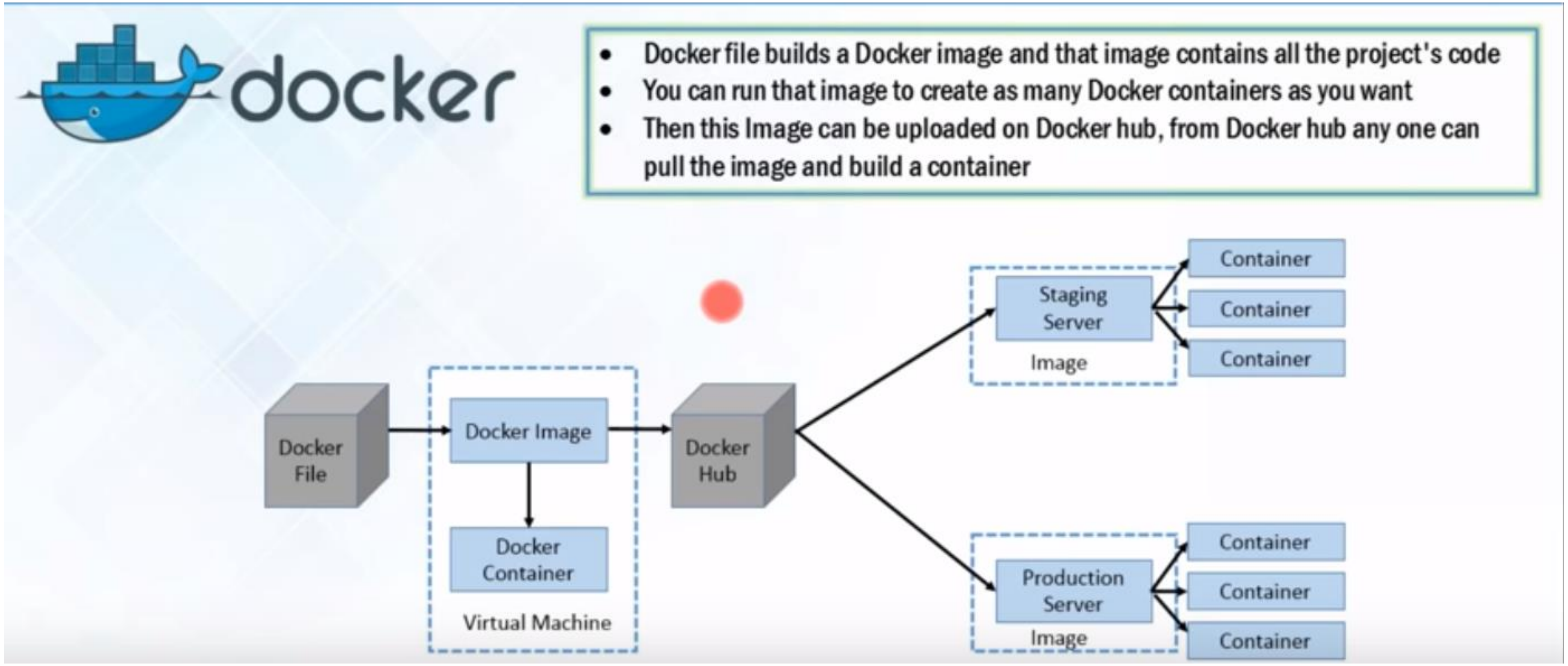
- Docker is a tool designed to make it easier to create, deploy, and run applications by using containers.
- Docker containers are lightweight alternatives to Virtual Machines and it uses the host OS.
- You don't have to pre-allocate any RAM in containers.



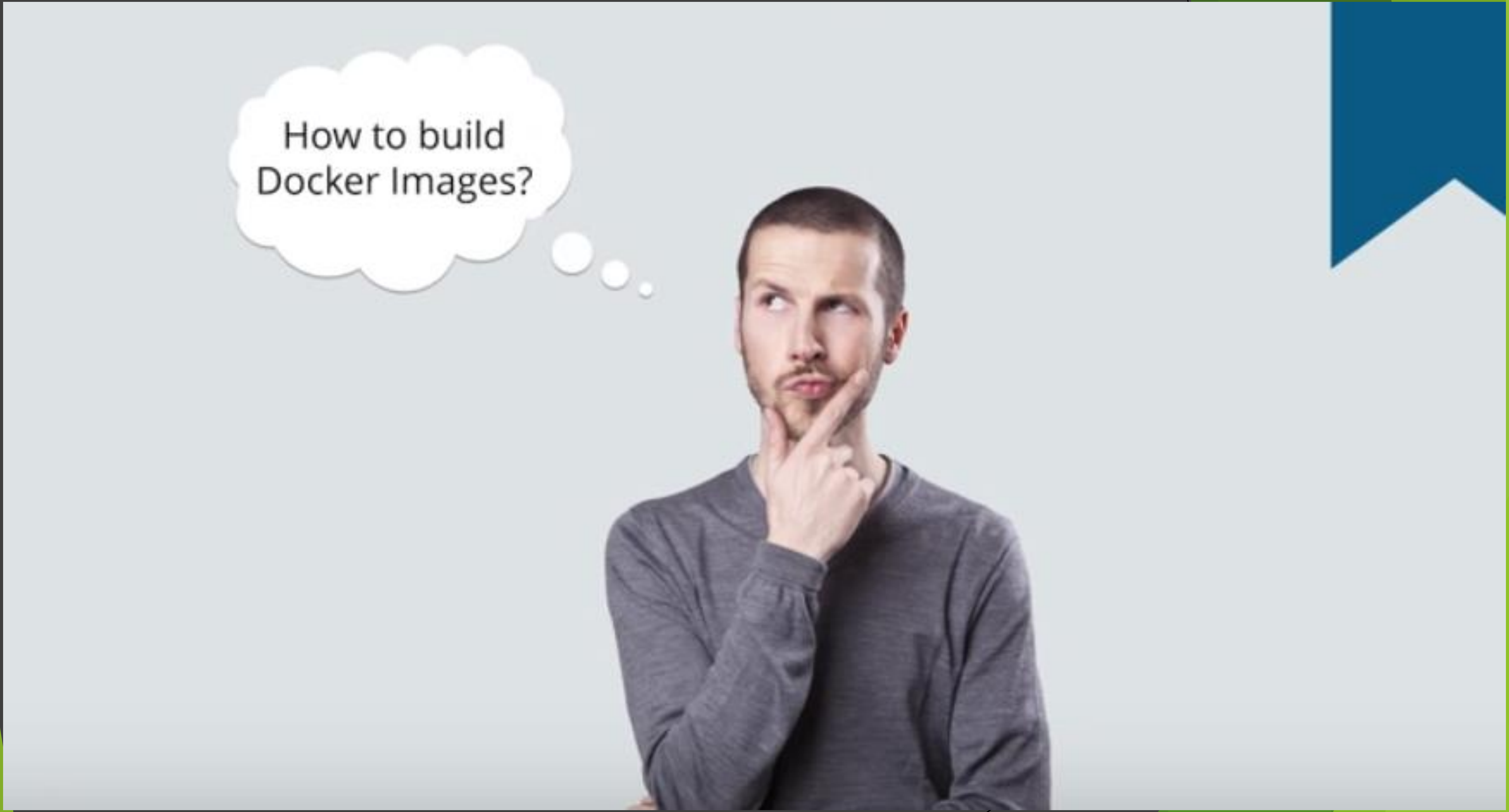
What is Docker?



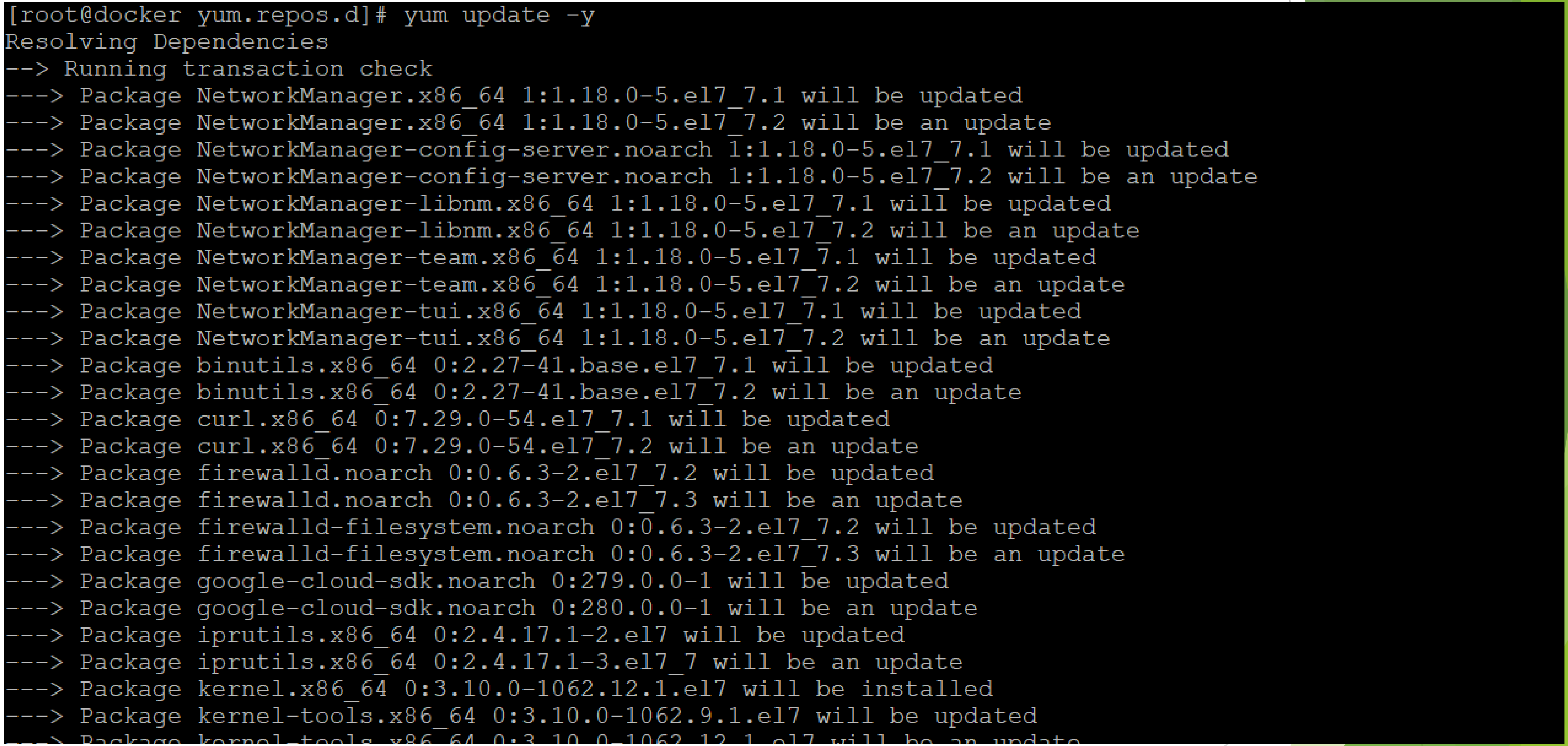
Docker In a Nutshell



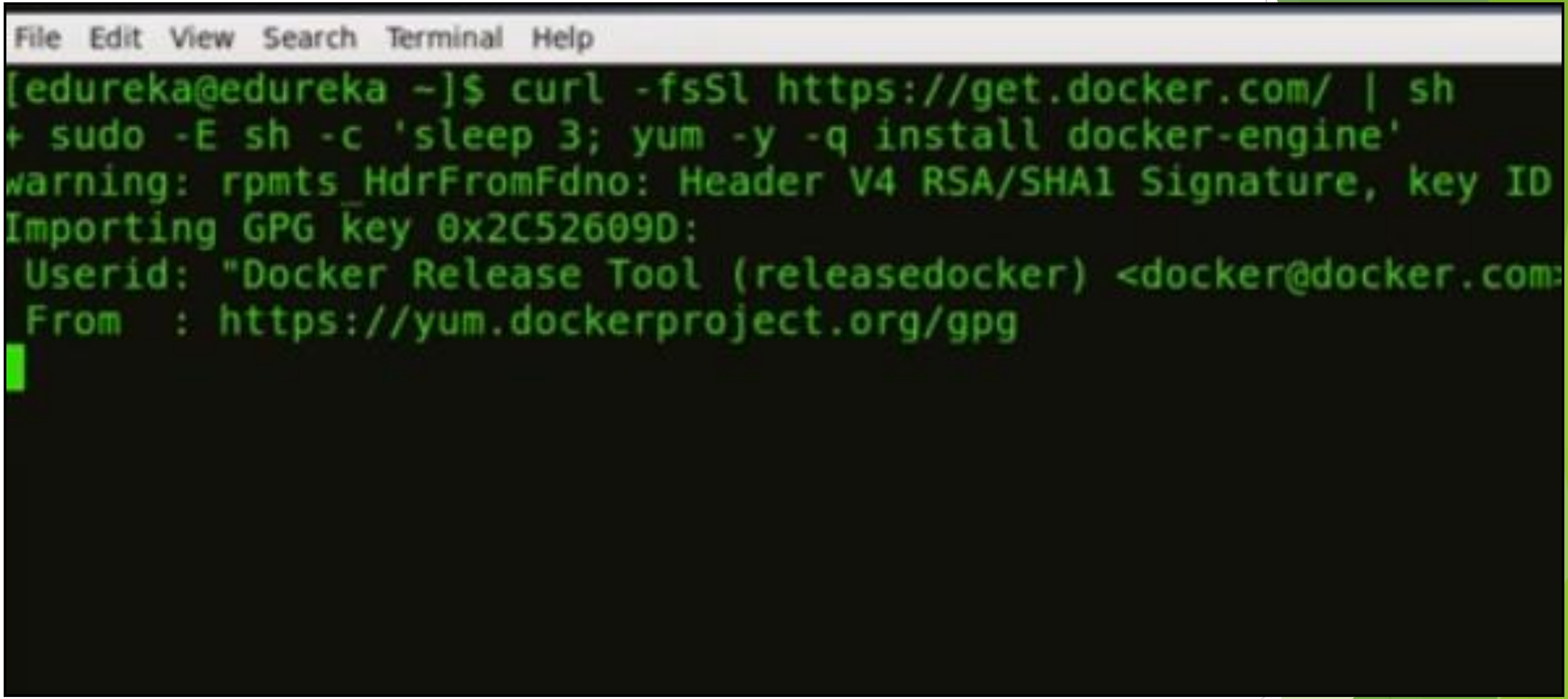
- Docker file builds a Docker image and that image contains all the project's code
- You can run that image to create as many Docker containers as you want
- Then this Image can be uploaded on Docker hub, from Docker hub any one can pull the image and build a container


How to build Docker Images?

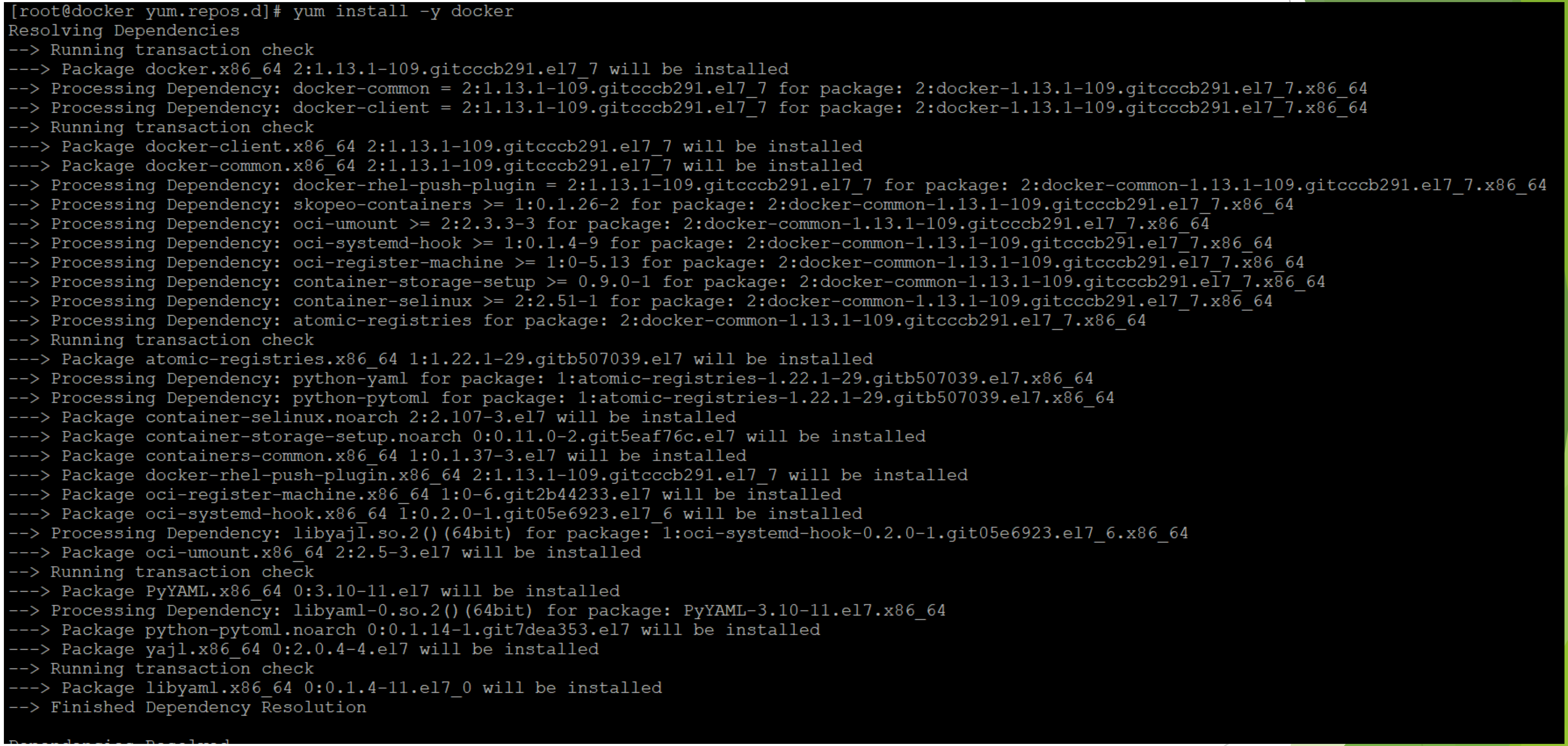
Docker Installation



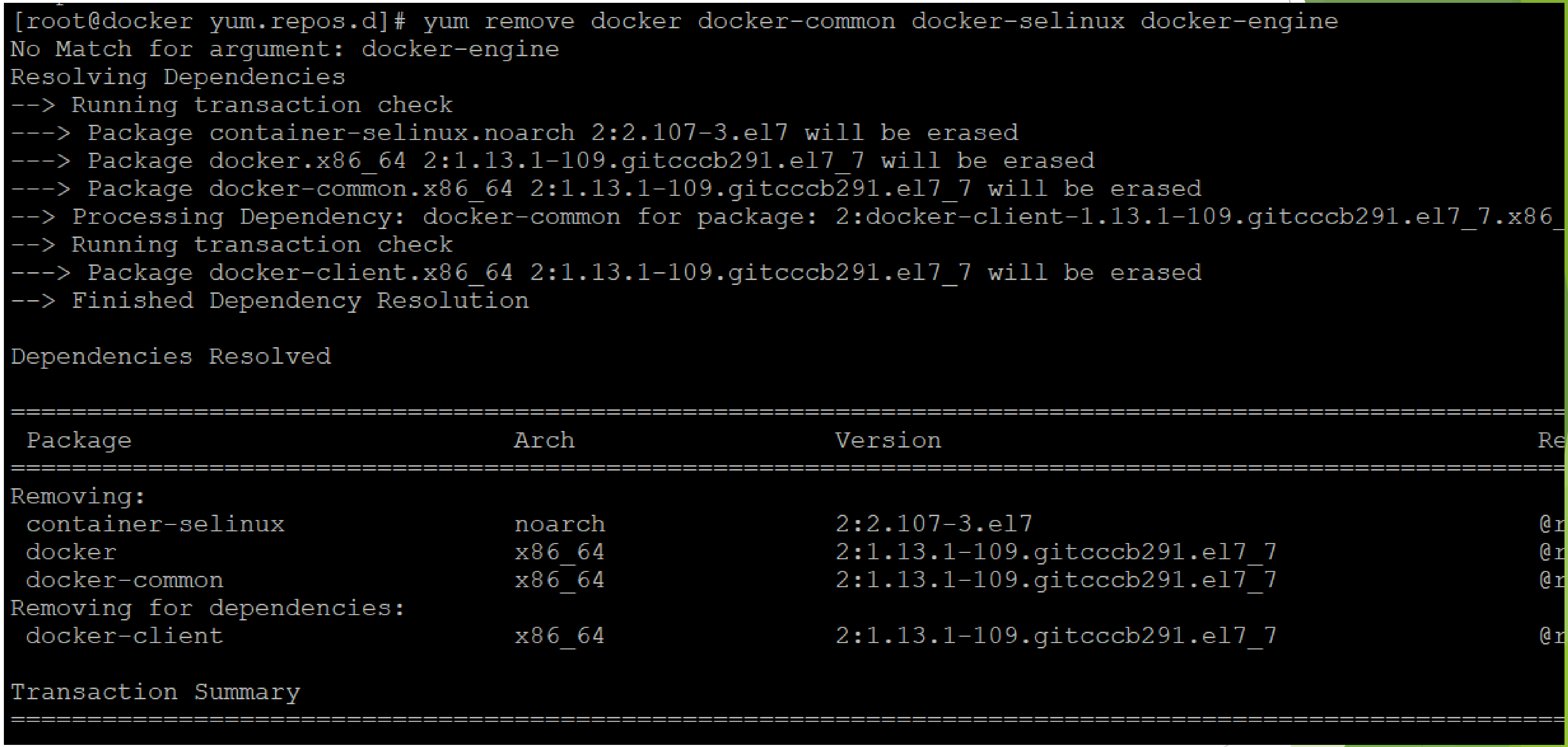
```
[root@docker yum.repos.d]# yum update -y
Resolving Dependencies
--> Running transaction check
---> Package NetworkManager.x86_64 1:1.18.0-5.el7_7.1 will be updated
---> Package NetworkManager.x86_64 1:1.18.0-5.el7_7.2 will be an update
---> Package NetworkManager-config-server.noarch 1:1.18.0-5.el7_7.1 will be updated
---> Package NetworkManager-config-server.noarch 1:1.18.0-5.el7_7.2 will be an update
---> Package NetworkManager-libnm.x86_64 1:1.18.0-5.el7_7.1 will be updated
---> Package NetworkManager-libnm.x86_64 1:1.18.0-5.el7_7.2 will be an update
---> Package NetworkManager-team.x86_64 1:1.18.0-5.el7_7.1 will be updated
---> Package NetworkManager-team.x86_64 1:1.18.0-5.el7_7.2 will be an update
---> Package NetworkManager-tui.x86_64 1:1.18.0-5.el7_7.1 will be updated
---> Package NetworkManager-tui.x86_64 1:1.18.0-5.el7_7.2 will be an update
---> Package binutils.x86_64 0:2.27-41.base.el7_7.1 will be updated
---> Package binutils.x86_64 0:2.27-41.base.el7_7.2 will be an update
---> Package curl.x86_64 0:7.29.0-54.el7_7.1 will be updated
---> Package curl.x86_64 0:7.29.0-54.el7_7.2 will be an update
---> Package firewalld.noarch 0:0.6.3-2.el7_7.2 will be updated
---> Package firewalld.noarch 0:0.6.3-2.el7_7.3 will be an update
---> Package firewalld-filesystem.noarch 0:0.6.3-2.el7_7.2 will be updated
---> Package firewalld-filesystem.noarch 0:0.6.3-2.el7_7.3 will be an update
---> Package google-cloud-sdk.noarch 0:279.0.0-1 will be updated
---> Package google-cloud-sdk.noarch 0:280.0.0-1 will be an update
---> Package iprutils.x86_64 0:2.4.17.1-2.el7 will be updated
---> Package iprutils.x86_64 0:2.4.17.1-3.el7_7 will be an update
---> Package kernel.x86_64 0:3.10.0-1062.12.1.el7 will be installed
---> Package kernel-tools.x86_64 0:3.10.0-1062.9.1.el7 will be updated
---> Package kernel-tools.x86_64 0:3.10.0-1062.12.1.el7 will be an update
```



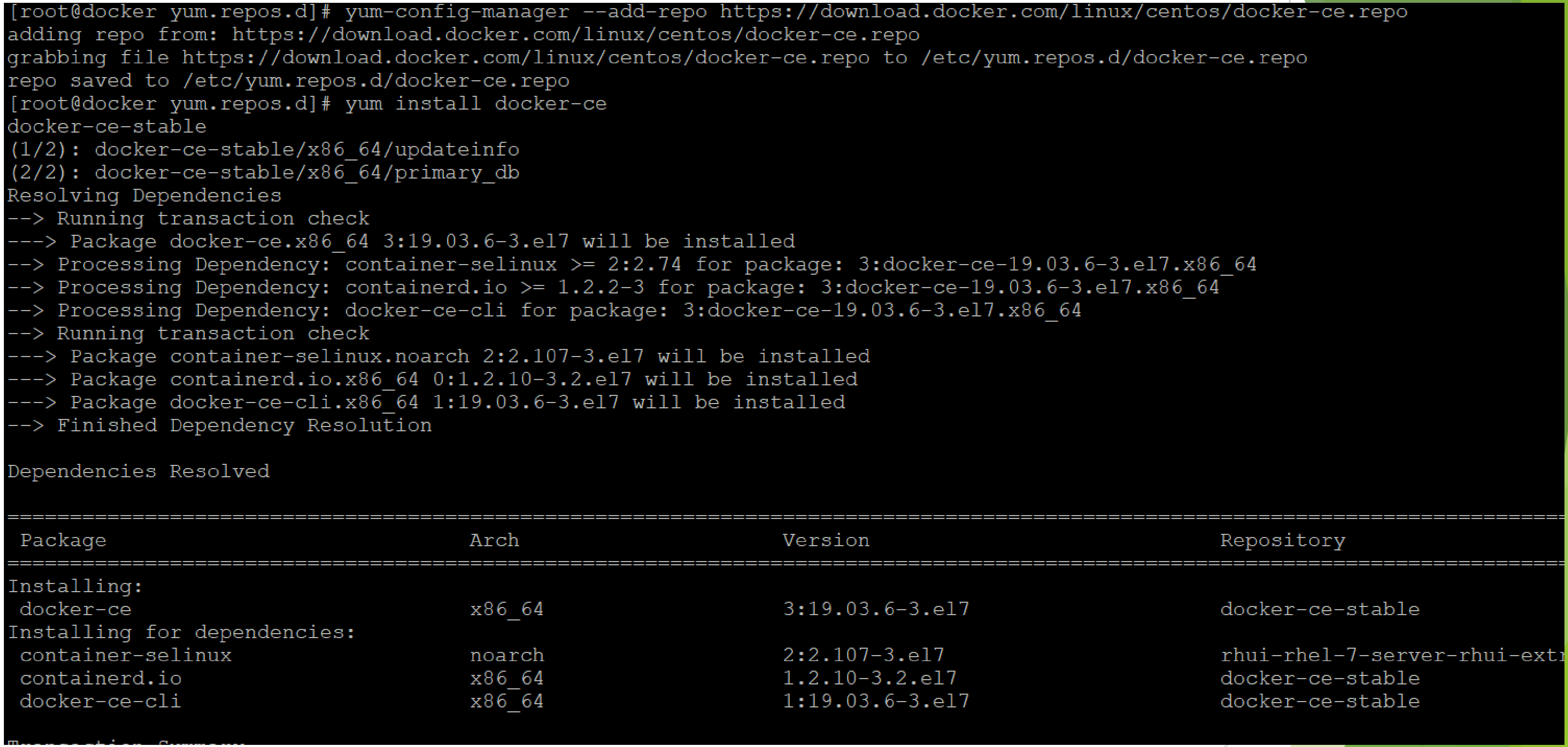
```
[edureka@edureka ~]$ curl -fsSl https://get.docker.com/ | sh
+ sudo -E sh -c 'sleep 3; yum -y -q install docker-engine'
warning: rpmts_HdrFromFdno: Header V4 RSA/SHA1 Signature, key ID
Importing GPG key 0x2C52609D:
 Userid: "Docker Release Tool (releasedocker) <docker@docker.com
 From   : https://yum.dockerproject.org/gpg
```



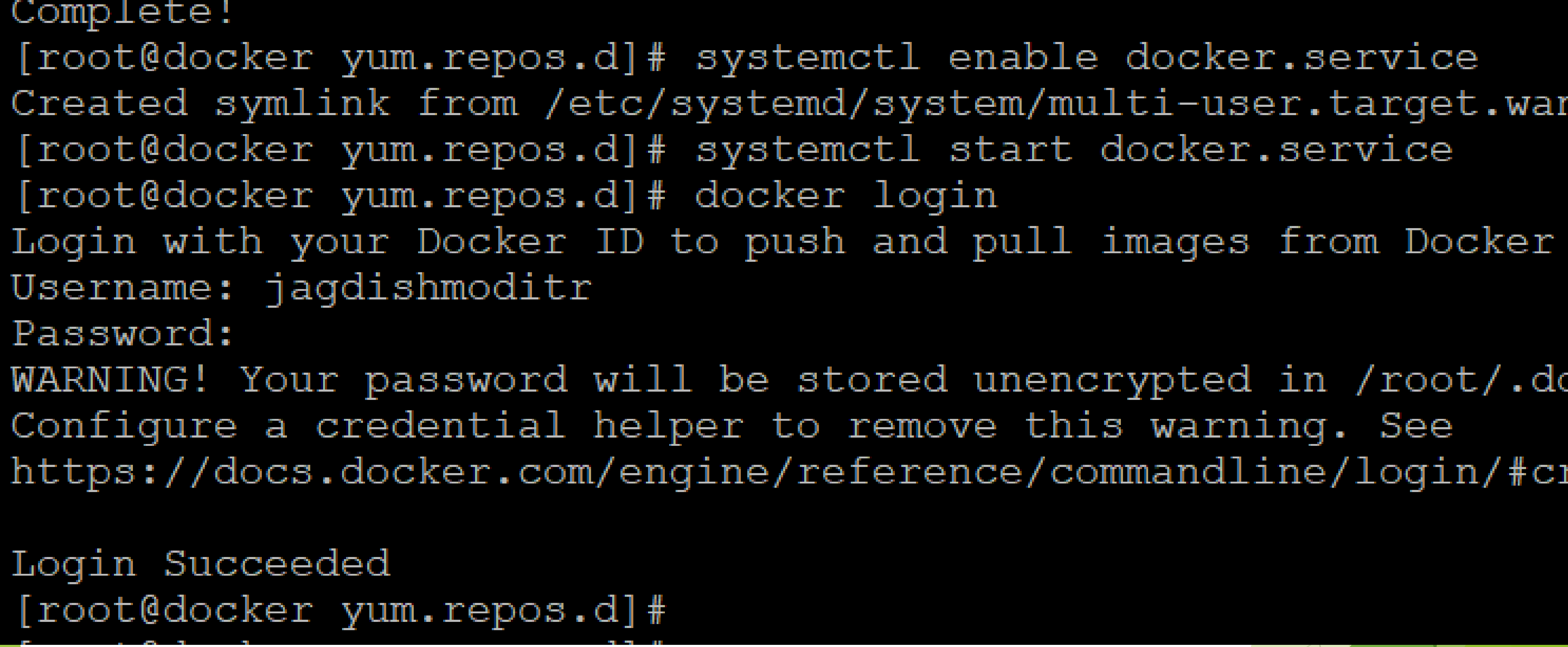
```
[root@docker yum.repos.d]# yum install -y docker
Resolving Dependencies
--> Running transaction check
---> Package docker.x86_64 2:1.13.1-109.gitcccb291.el7_7 will be installed
--> Processing Dependency: docker-common = 2:1.13.1-109.gitcccb291.el7_7 for package: 2:docker-1.13.1-109.gitcccb291.el7_7.x86_64
--> Processing Dependency: docker-client = 2:1.13.1-109.gitcccb291.el7_7 for package: 2:docker-1.13.1-109.gitcccb291.el7_7.x86_64
--> Running transaction check
---> Package docker-client.x86_64 2:1.13.1-109.gitcccb291.el7_7 will be installed
---> Package docker-common.x86_64 2:1.13.1-109.gitcccb291.el7_7 will be installed
--> Processing Dependency: docker-rhel-push-plugin = 2:1.13.1-109.gitcccb291.el7_7 for package: 2:docker-common-1.13.1-109.gitcccb291.el7_7.x86_64
--> Processing Dependency: skopeo-containers >= 1:0.1.26-2 for package: 2:docker-common-1.13.1-109.gitcccb291.el7_7.x86_64
--> Processing Dependency: oci-umount >= 2:2.3.3-3 for package: 2:docker-common-1.13.1-109.gitcccb291.el7_7.x86_64
--> Processing Dependency: oci-systemd-hook >= 1:0.1.4-9 for package: 2:docker-common-1.13.1-109.gitcccb291.el7_7.x86_64
--> Processing Dependency: oci-register-machine >= 1:0-5.13 for package: 2:docker-common-1.13.1-109.gitcccb291.el7_7.x86_64
--> Processing Dependency: container-storage-setup >= 0.9.0-1 for package: 2:docker-common-1.13.1-109.gitcccb291.el7_7.x86_64
--> Processing Dependency: container-selinux >= 2:2.51-1 for package: 2:docker-common-1.13.1-109.gitcccb291.el7_7.x86_64
--> Processing Dependency: atomic-registries for package: 2:docker-common-1.13.1-109.gitcccb291.el7_7.x86_64
--> Running transaction check
---> Package atomic-registries.x86_64 1:1.22.1-29.gitb507039.el7 will be installed
--> Processing Dependency: python-yaml for package: 1:atomic-registries-1.22.1-29.gitb507039.el7.x86_64
--> Processing Dependency: python-pytoml for package: 1:atomic-registries-1.22.1-29.gitb507039.el7.x86_64
---> Package container-selinux.noarch 2:2.107-3.el7 will be installed
---> Package container-storage-setup.noarch 0:0.11.0-2.git5eaf76c.el7 will be installed
---> Package containers-common.x86_64 1:0.1.37-3.el7 will be installed
---> Package docker-rhel-push-plugin.x86_64 2:1.13.1-109.gitcccb291.el7_7 will be installed
---> Package oci-register-machine.x86_64 1:0-6.git2b44233.el7 will be installed
---> Package oci-systemd-hook.x86_64 1:0.2.0-1.git05e6923.el7_6 will be installed
--> Processing Dependency: libyajl.so.2()(64bit) for package: 1:oci-systemd-hook-0.2.0-1.git05e6923.el7_6.x86_64
---> Package oci-umount.x86_64 2:2.5-3.el7 will be installed
--> Running transaction check
---> Package PyYAML.x86_64 0:3.10-11.el7 will be installed
--> Processing Dependency: libyaml-0.so.2()(64bit) for package: PyYAML-3.10-11.el7.x86_64
---> Package python-pytoml.noarch 0:0.1.14-1.git7dea353.el7 will be installed
---> Package yajl.x86_64 0:2.0.4-4.el7 will be installed
--> Running transaction check
---> Package libyaml.x86_64 0:0.1.4-11.el7_0 will be installed
--> Finished Dependency Resolution
```



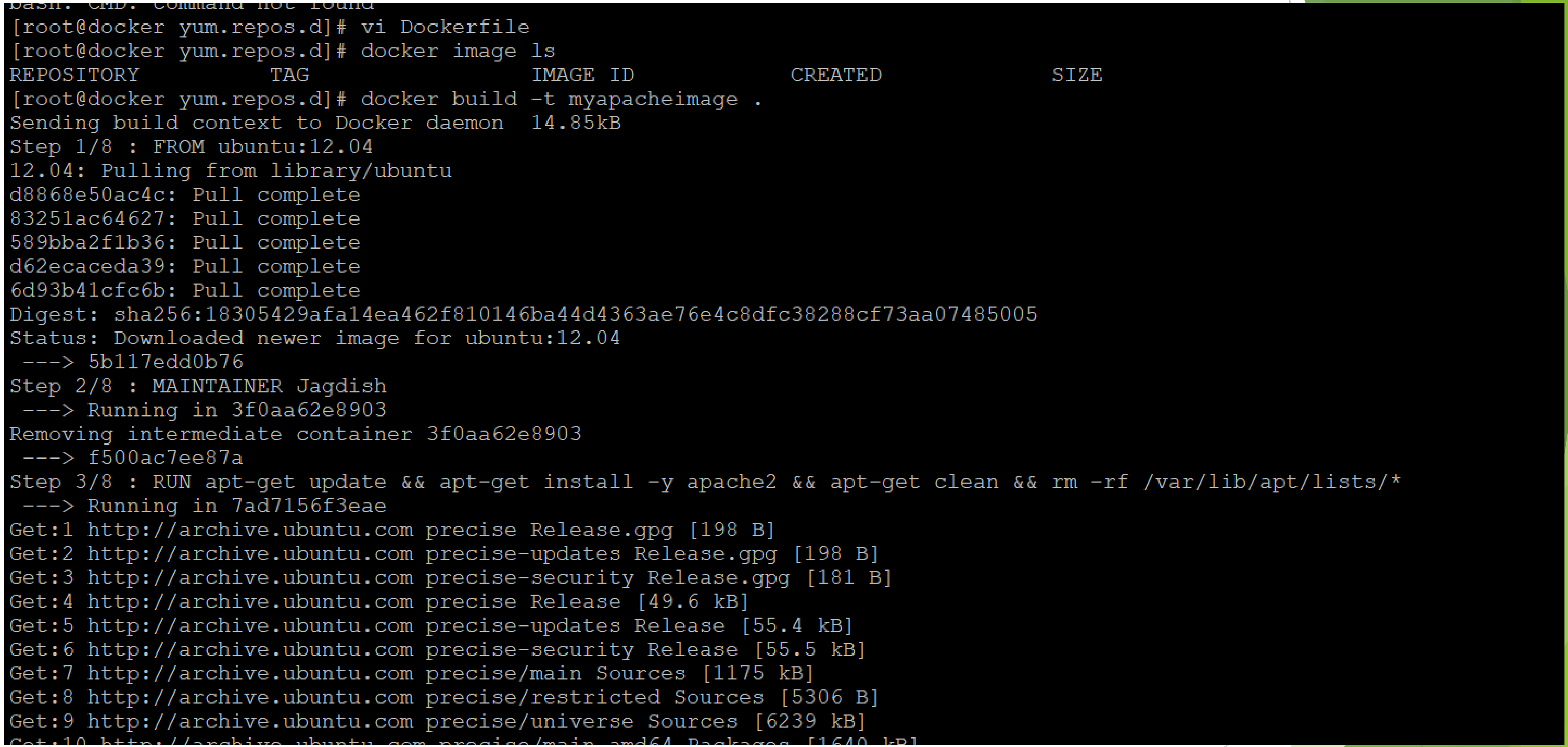
```
[root@docker yum.repos.d]# yum remove docker docker-common docker-selinux docker-engine
No Match for argument: docker-engine
Resolving Dependencies
--> Running transaction check
---> Package container-selinux.noarch 2:2.107-3.el7 will be erased
---> Package docker.x86_64 2:1.13.1-109.gitcccb291.el7_7 will be erased
---> Package docker-common.x86_64 2:1.13.1-109.gitcccb291.el7_7 will be erased
--> Processing Dependency: docker-common for package: 2:docker-client-1.13.1-109.gitcccb291.el7_7.x86_
--> Running transaction check
---> Package docker-client.x86_64 2:1.13.1-109.gitcccb291.el7_7 will be erased
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================================
 Package                         Arch                    Version                            Re
================================================================================================
Removing:
 container-selinux               noarch                  2:2.107-3.el7                      @r
 docker                          x86_64                  2:1.13.1-109.gitcccb291.el7_7      @r
 docker-common                   x86_64                  2:1.13.1-109.gitcccb291.el7_7      @r
Removing for dependencies:
 docker-client                   x86_64                  2:1.13.1-109.gitcccb291.el7_7      @r

Transaction Summary
================================================================================================
```



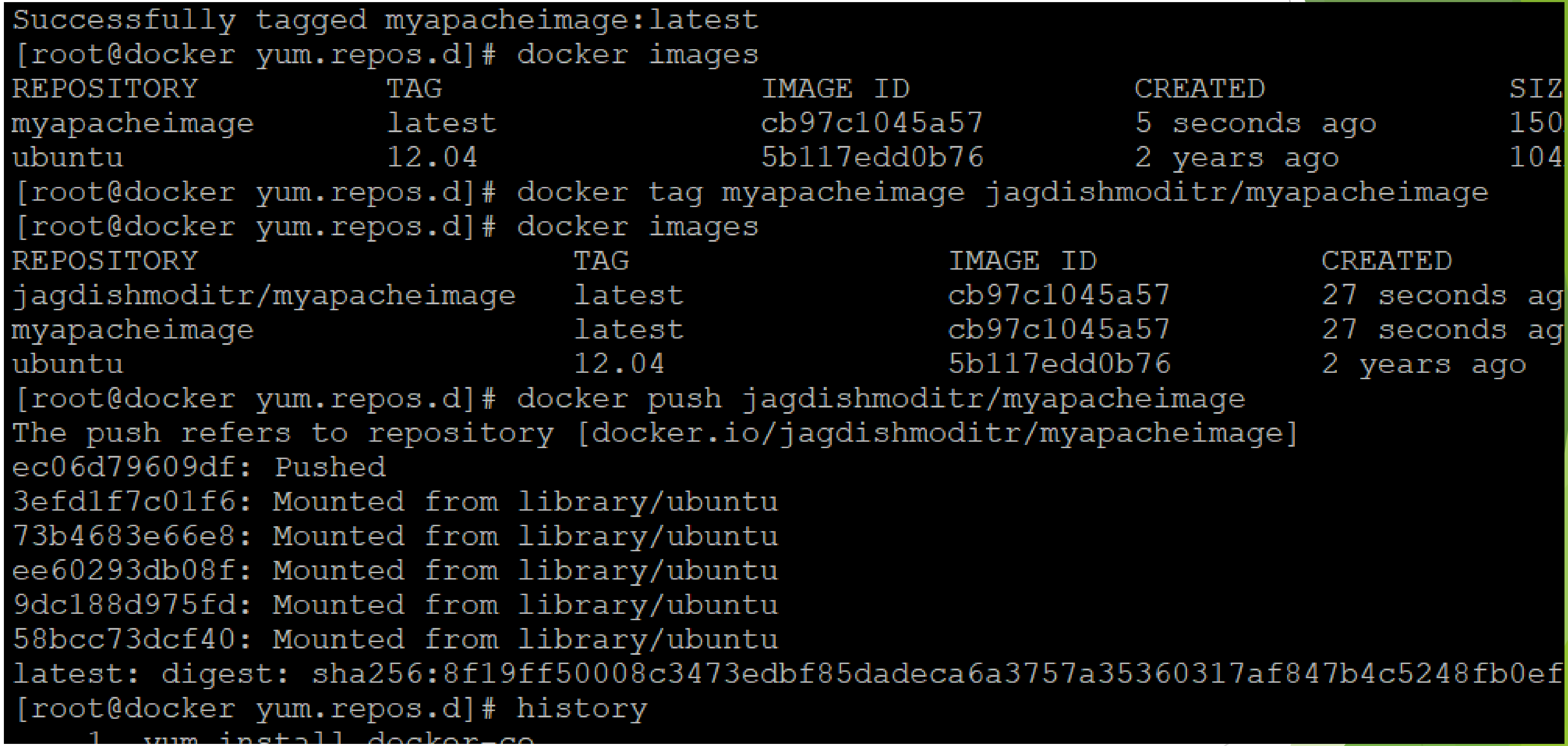
```
[root@docker yum.repos.d]# yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yum.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
[root@docker yum.repos.d]# yum install docker-ce
docker-ce-stable
(1/2): docker-ce-stable/x86_64/updateinfo
(2/2): docker-ce-stable/x86_64/primary_db
Resolving Dependencies
--> Running transaction check
---> Package docker-ce.x86_64 3:19.03.6-3.el7 will be installed
--> Processing Dependency: container-selinux >= 2:2.74 for package: 3:docker-ce-19.03.6-3.el7.x86_64
--> Processing Dependency: containerd.io >= 1.2.2-3 for package: 3:docker-ce-19.03.6-3.el7.x86_64
--> Processing Dependency: docker-ce-cli for package: 3:docker-ce-19.03.6-3.el7.x86_64
--> Running transaction check
---> Package container-selinux.noarch 2:2.107-3.el7 will be installed
---> Package containerd.io.x86_64 0:1.2.10-3.2.el7 will be installed
---> Package docker-ce-cli.x86_64 1:19.03.6-3.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

==============================================================================================================================
 Package                             Arch                     Version                            Repository
==============================================================================================================================
Installing:
 docker-ce                           x86_64                   3:19.03.6-3.el7                    docker-ce-stable
Installing for dependencies:
 container-selinux                   noarch                   2:2.107-3.el7                      rhui-rhel-7-server-rhui-extr
 containerd.io                       x86_64                   1.2.10-3.2.el7                     docker-ce-stable
 docker-ce-cli                       x86_64                   1:19.03.6-3.el7                    docker-ce-stable

Transaction Summary
```



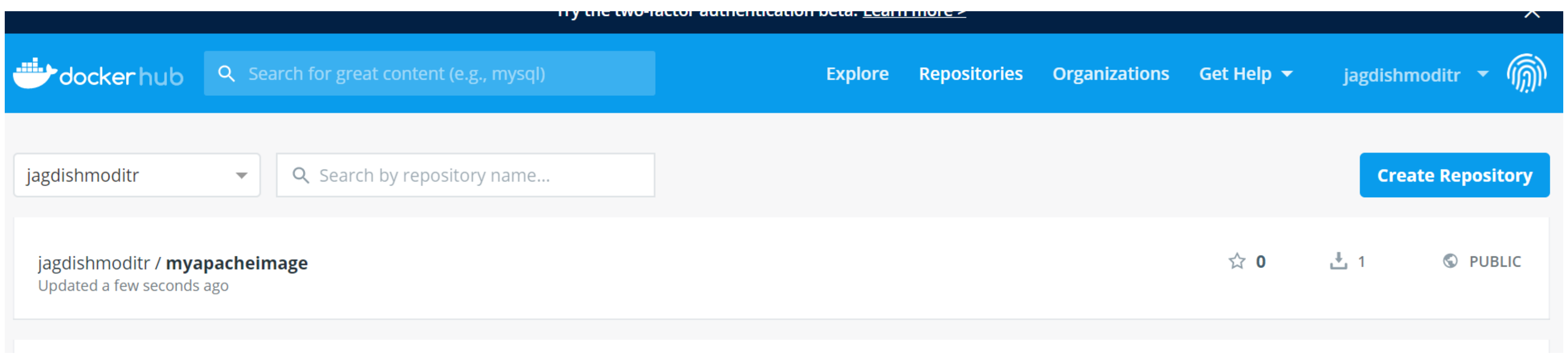
```
Complete!
[root@docker yum.repos.d]# systemctl enable docker.service
Created symlink from /etc/systemd/system/multi-user.target.war
[root@docker yum.repos.d]# systemctl start docker.service
[root@docker yum.repos.d]# docker login
Login with your Docker ID to push and pull images from Docker
Username: jagdishmoditr
Password:
WARNING! Your password will be stored unencrypted in /root/.do
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#cr

Login Succeeded
[root@docker yum.repos.d]#
```



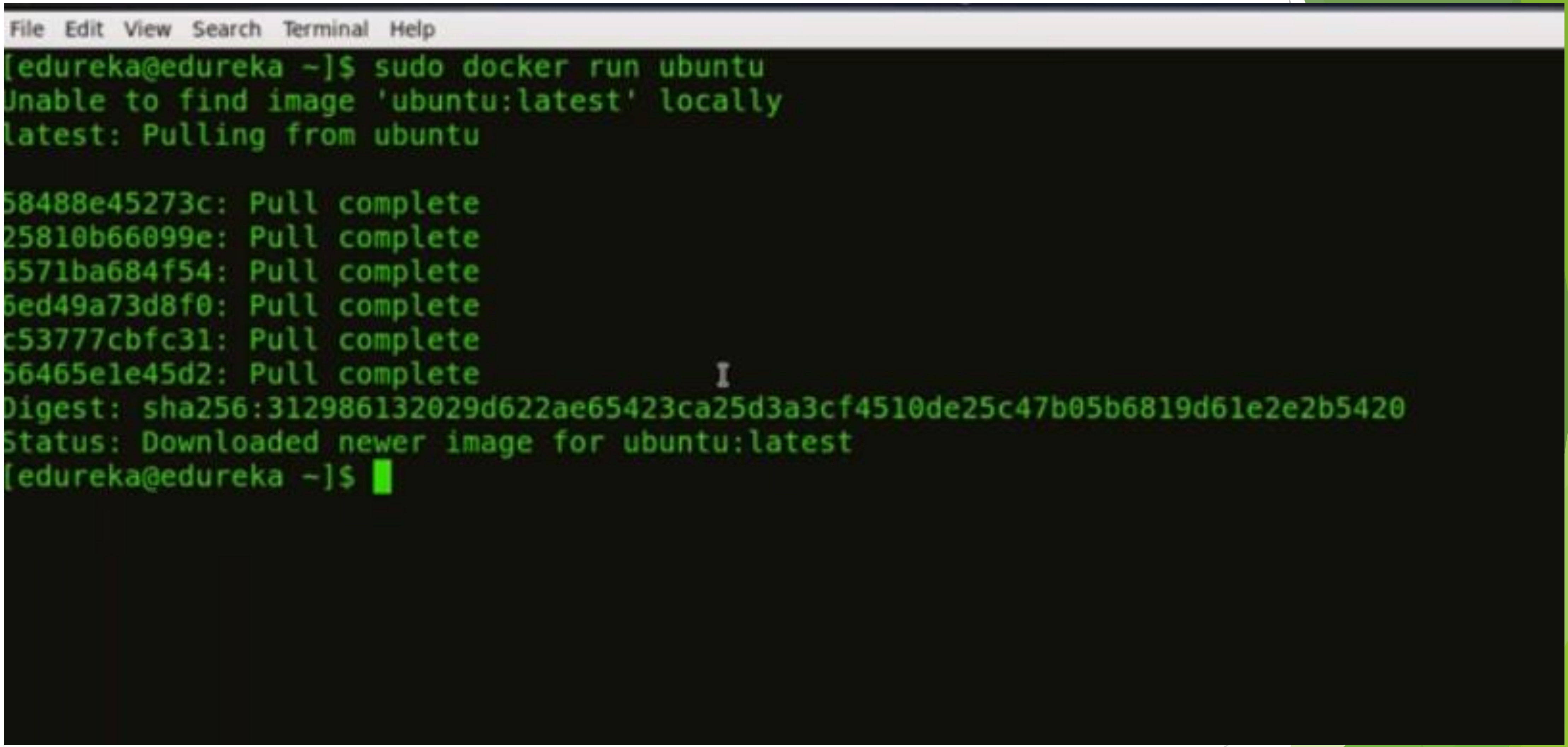
```
[root@docker yum.repos.d]# vi Dockerfile
[root@docker yum.repos.d]# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
[root@docker yum.repos.d]# docker build -t myapacheimage .
Sending build context to Docker daemon  14.85kB
Step 1/8 : FROM ubuntu:12.04
12.04: Pulling from library/ubuntu
d8868e50ac4c: Pull complete
83251ac64627: Pull complete
589bba2f1b36: Pull complete
d62ecaceda39: Pull complete
6d93b41cfc6b: Pull complete
Digest: sha256:18305429afa14ea462f810146ba44d4363ae76e4c8dfc38288cf73aa07485005
Status: Downloaded newer image for ubuntu:12.04
 ---> 5b117edd0b76
Step 2/8 : MAINTAINER Jagdish
 ---> Running in 3f0aa62e8903
Removing intermediate container 3f0aa62e8903
 ---> f500ac7ee87a
Step 3/8 : RUN apt-get update && apt-get install -y apache2 && apt-get clean && rm -rf /var/lib/apt/lists/*
 ---> Running in 7ad7156f3eae
Get:1 http://archive.ubuntu.com precise Release.gpg [198 B]
Get:2 http://archive.ubuntu.com precise-updates Release.gpg [198 B]
Get:3 http://archive.ubuntu.com precise-security Release.gpg [181 B]
Get:4 http://archive.ubuntu.com precise Release [49.6 kB]
Get:5 http://archive.ubuntu.com precise-updates Release [55.4 kB]
Get:6 http://archive.ubuntu.com precise-security Release [55.5 kB]
Get:7 http://archive.ubuntu.com precise/main Sources [1175 kB]
Get:8 http://archive.ubuntu.com precise/restricted Sources [5306 B]
Get:9 http://archive.ubuntu.com precise/universe Sources [6239 kB]
```



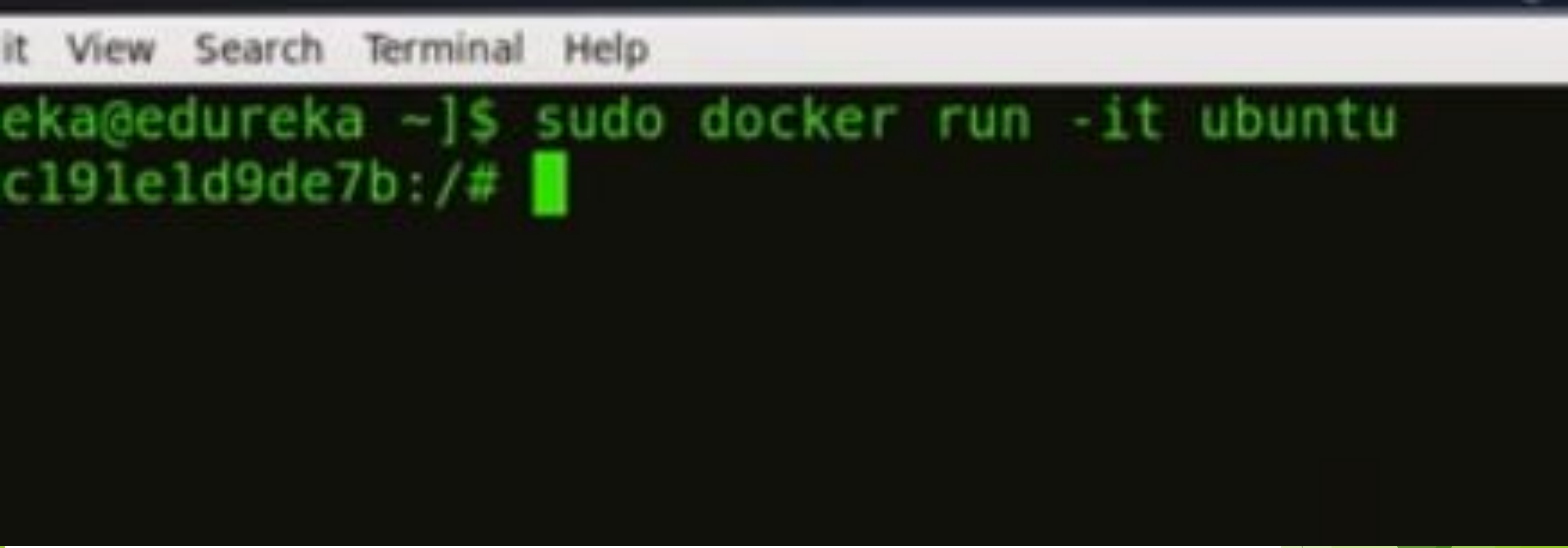
```
Successfully tagged myapacheimage:latest
[root@docker yum.repos.d]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZ
myapacheimage       latest              cb97c1045a57        5 seconds ago       150
ubuntu              12.04               5b117edd0b76        2 years ago         104
[root@docker yum.repos.d]# docker tag myapacheimage jagdishmoditr/myapacheimage
[root@docker yum.repos.d]# docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED
jagdishmoditr/myapacheimage   latest              cb97c1045a57        27 seconds ag
myapacheimage                 latest              cb97c1045a57        27 seconds ag
ubuntu                        12.04               5b117edd0b76        2 years ago
[root@docker yum.repos.d]# docker push jagdishmoditr/myapacheimage
The push refers to repository [docker.io/jagdishmoditr/myapacheimage]
ec06d79609df: Pushed
3efd1f7c01f6: Mounted from library/ubuntu
73b4683e66e8: Mounted from library/ubuntu
ee60293db08f: Mounted from library/ubuntu
9dc188d975fd: Mounted from library/ubuntu
58bcc73dcf40: Mounted from library/ubuntu
latest: digest: sha256:8f19ff50008c3473edbf85dadeca6a3757a35360317af847b4c5248fb0ef
[root@docker yum.repos.d]# history
```


Try the two-factor authentication beta. Learn more >
docker hub
Search for great content (e.g., mysql)
Explore
Repositories
Organizations
Get Help
jagdishmoditr
jagdishmoditr
Search by repository name...
Create Repository

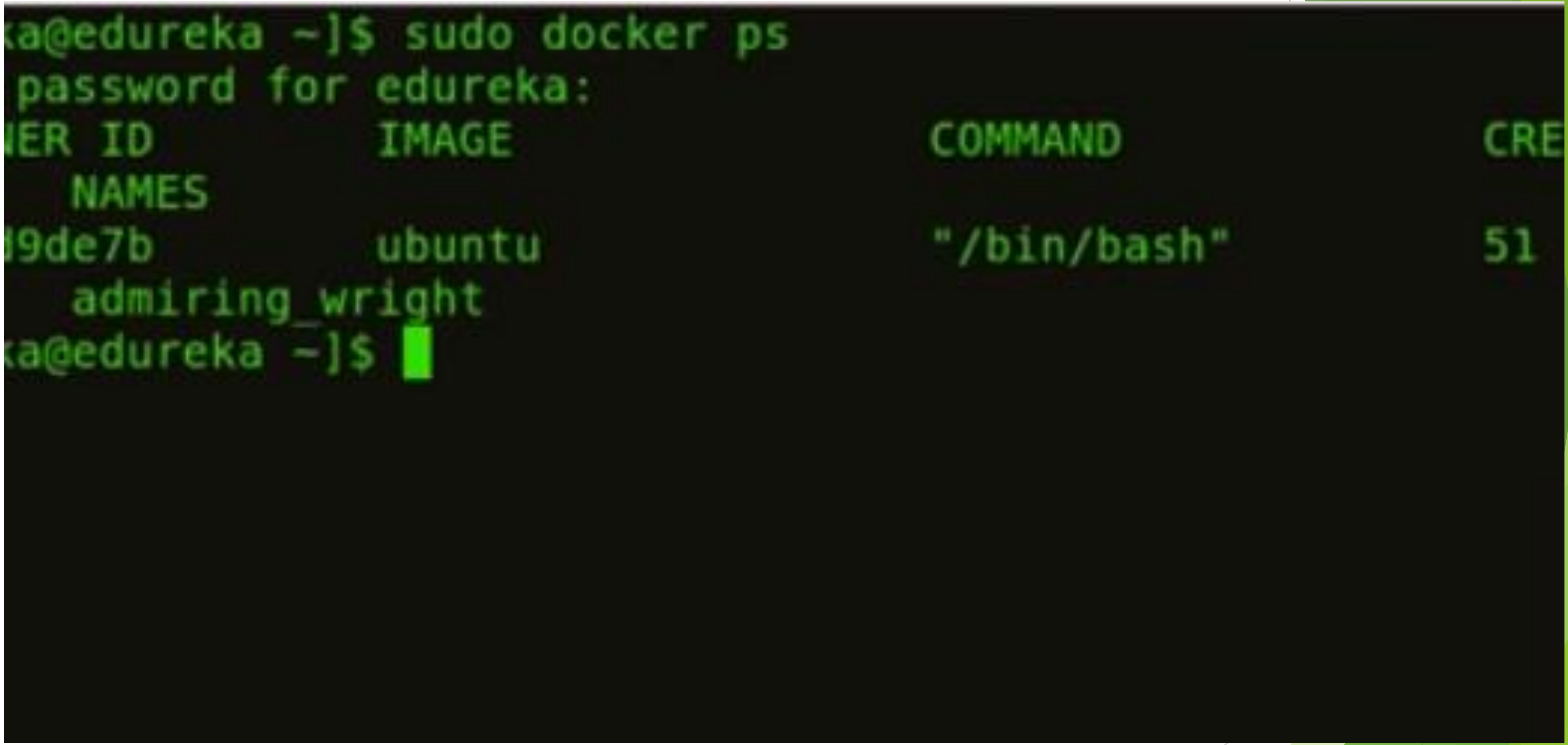
jagdishmoditr / **myapacheimage**
Updated a few seconds ago
0 1 PUBLIC



```
[edureka@edureka ~]$ sudo docker run ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from ubuntu

58488e45273c: Pull complete
25810b66099e: Pull complete
6571ba684f54: Pull complete
6ed49a73d8f0: Pull complete
c53777cbfc31: Pull complete
56465e1e45d2: Pull complete
Digest: sha256:312986132029d622ae65423ca25d3a3cf4510de25c47b05b6819d61e2e2b5420
Status: Downloaded newer image for ubuntu:latest
[edureka@edureka ~]$
```



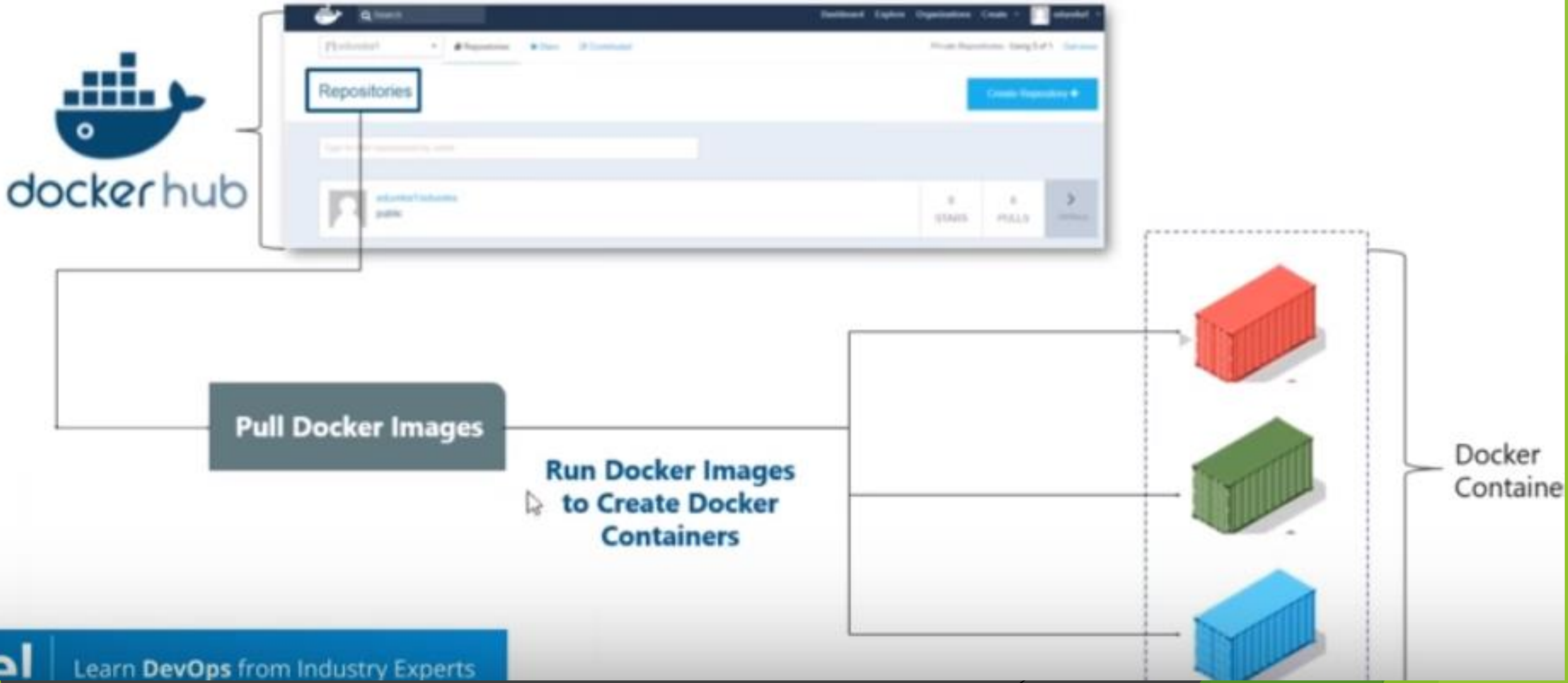
```
eka@edureka ~]$ sudo docker run -it ubuntu
c191e1d9de7b:/# 
```



```
ka@edureka ~]$ sudo docker ps
 password for edureka:
NER ID          IMAGE                   COMMAND           CRE
   NAMES
l9de7b          ubuntu                  "/bin/bash"       51
   admiring_wright
ka@edureka ~]$ █
```

What if I want to create my own image?

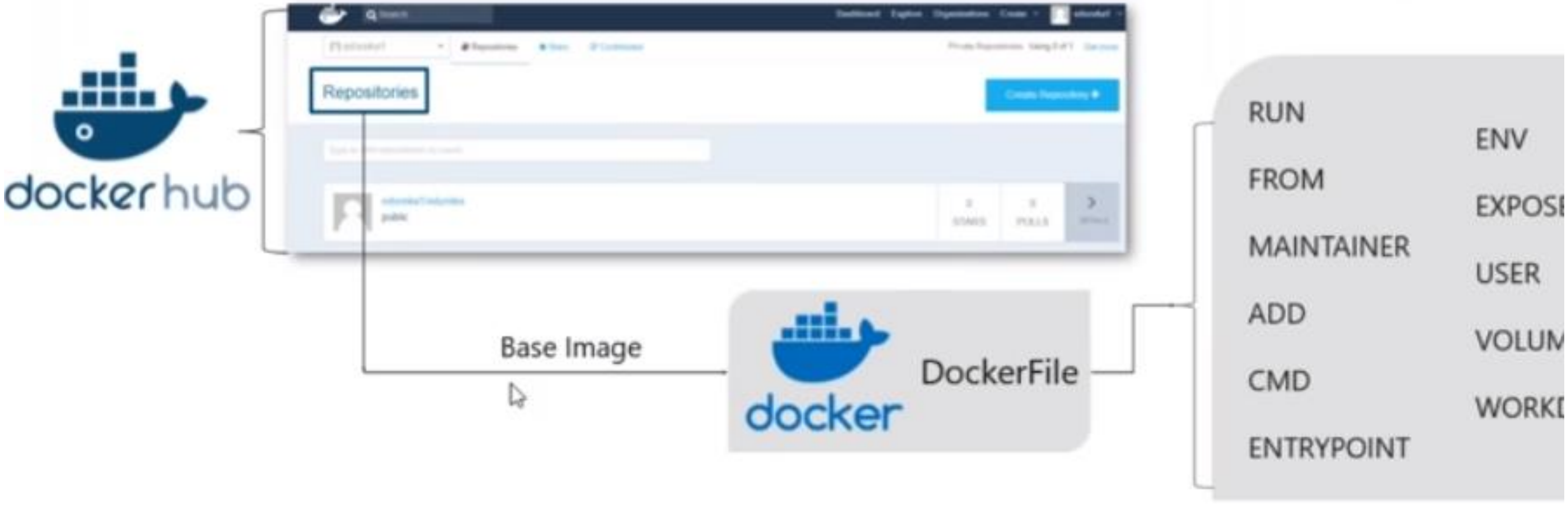
How To Build Docker Images? – Using Predefined Images



el | Learn **DevOps** from Industry Experts

How To Build Docker Images? – Using DockerFile

Dockerfile is a script, composed of various commands (instructions) and arguments listed successively to automatically perform actions on a base image in order to create (or form) a new one



How To Write A DockerFile?

DockerFile Syntax

Dockerfile syntax consists of two kind of main line blocks: **comments** and **commands** + **arguments**

Syntax

```
# Line blocks used for commenting
command argument argument1...
```

Example

```
# Print "Welcome To Edureka!"
RUN echo "Welcome To Edureka!"
```

Let's have a look at Different Set of Commands which DockerFile can Contain Before working on a DockerFile Example

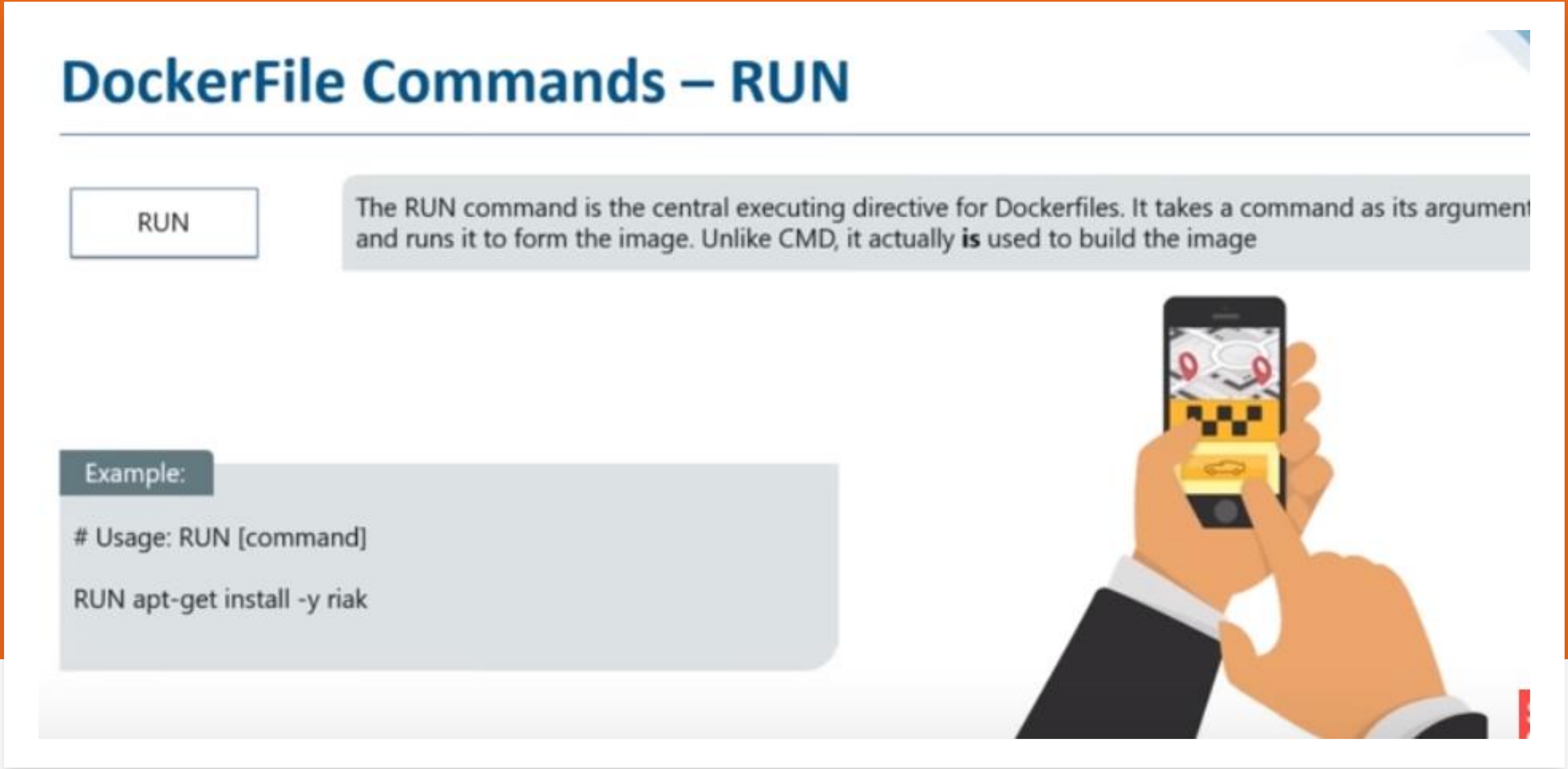
DockerFile Commands – FROM

FROM

FROM directive is probably the most crucial amongst all others for Dockerfiles. It defines the base image to use to start the build process

Example:

```
# Usage: FROM [image name]

FROM ubuntu
```



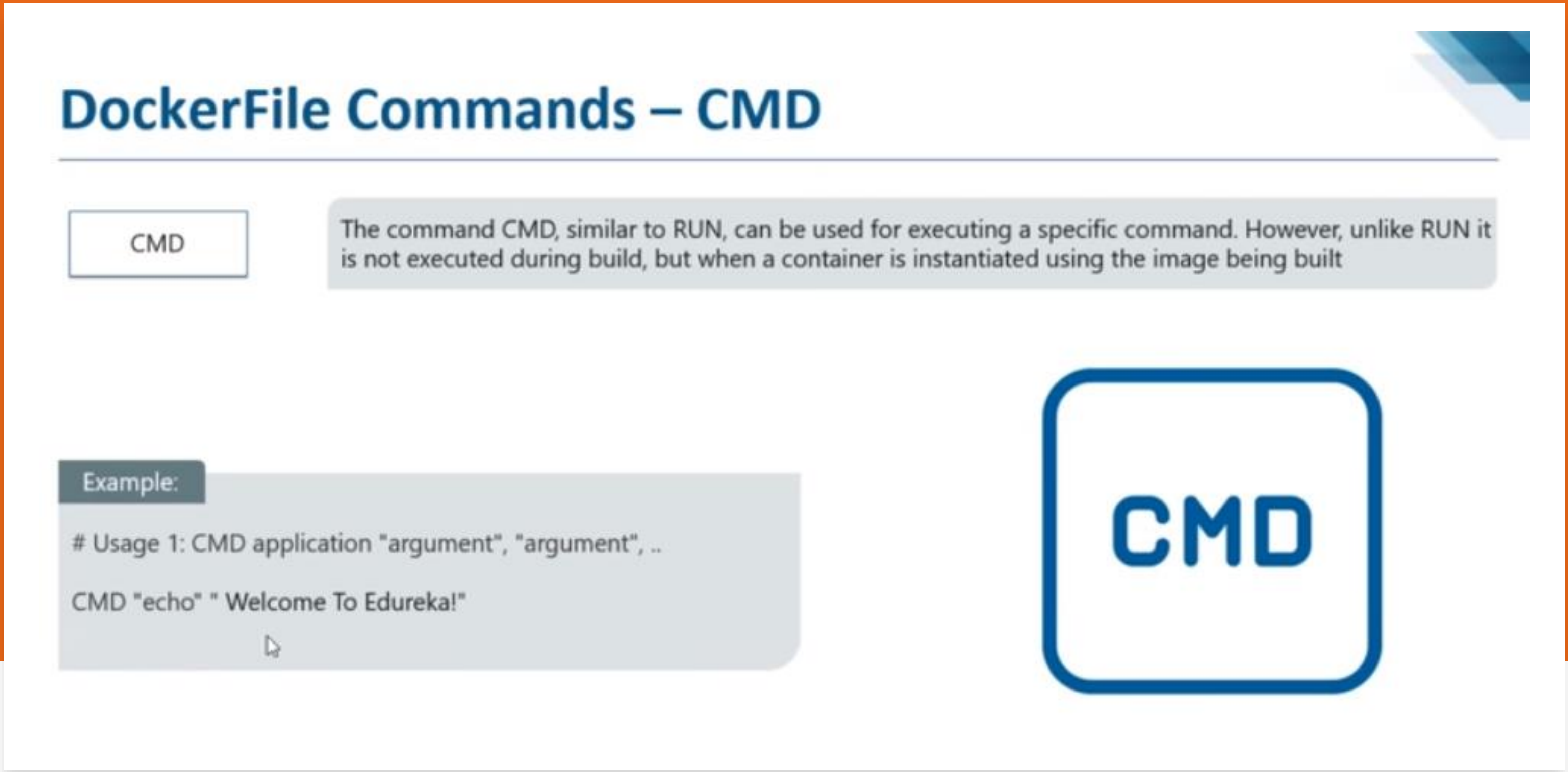
DockerFile Commands – RUN

RUN

The RUN command is the central executing directive for Dockerfiles. It takes a command as its argument and runs it to form the image. Unlike CMD, it actually **is** used to build the image

Example:

```
# Usage: RUN [command]

RUN apt-get install -y riak
```



DockerFile Commands – CMD

CMD

The command CMD, similar to RUN, can be used for executing a specific command. However, unlike RUN it is not executed during build, but when a container is instantiated using the image being built

Example:

```
# Usage 1: CMD application "argument", "argument", ..

CMD "echo" " Welcome To Edureka!"
```

CMD

DockerFile Commands – ENTRYPOINT

ENTRYPOINT

ENTRYPOINT argument sets the concrete default application that is used every time a container is created using the image

Example:

```
# Usage: ENTRYPOINT application "argument", "argument", ..

# Remember: arguments are optional. They can be provided by CMD
# or during the creation of a container.

ENTRYPOINT echo
# Usage example with CMD:

# Arguments set with CMD can be overridden during *run* CMD "Hello docker!"
CMD "Welcome to edureka!"
```

DockerFile Commands – ADD

ADD

The ADD command gets two arguments: a source and a destination. It basically copies the files from the source on the host into the container's own filesystem at the set destination

Example:

```
# Usage: ADD [source directory or URL] [destination directory]

ADD /my_app_folder /my_app_folder
```

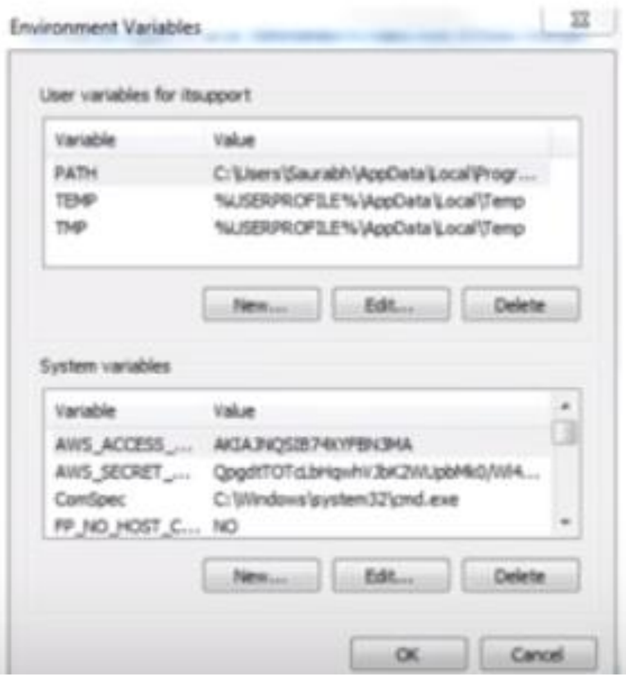
DockerFile Commands – ENV

ENV

The ENV command is used to set the environment variables (one or more). These variables consist of "key value" pairs which can be accessed within the container by scripts and applications alike

Example:

```
# Usage: ENV key value

ENV SERVER_WORKS 4
```



Variable	Value
PATH	C:\Users\Saurabh\AppData\Local\Progr...
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

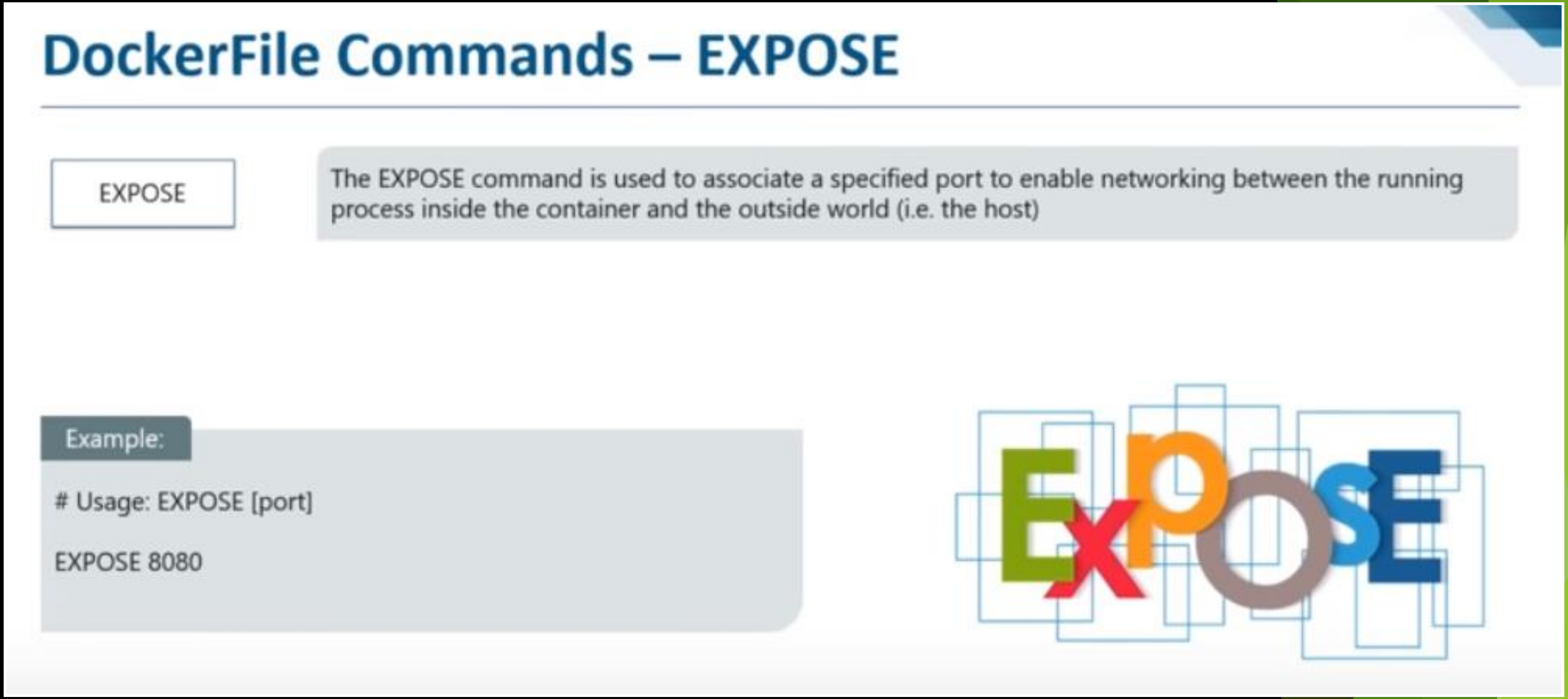
Variable	Value
AWS_ACCESS_...	AKIAJNQSIB74KYFBN3MA
AWS_SECRET_...	QpgdtTOTcLbHqwhV3bK2WUpbMk0/W4...
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_C...	NO

WORKDIR

The WORKDIR directive is used to set where the command defined with CMD is to be executed

Example:

```
# Usage:

WORKDIR /path WORKDIR ~/
```



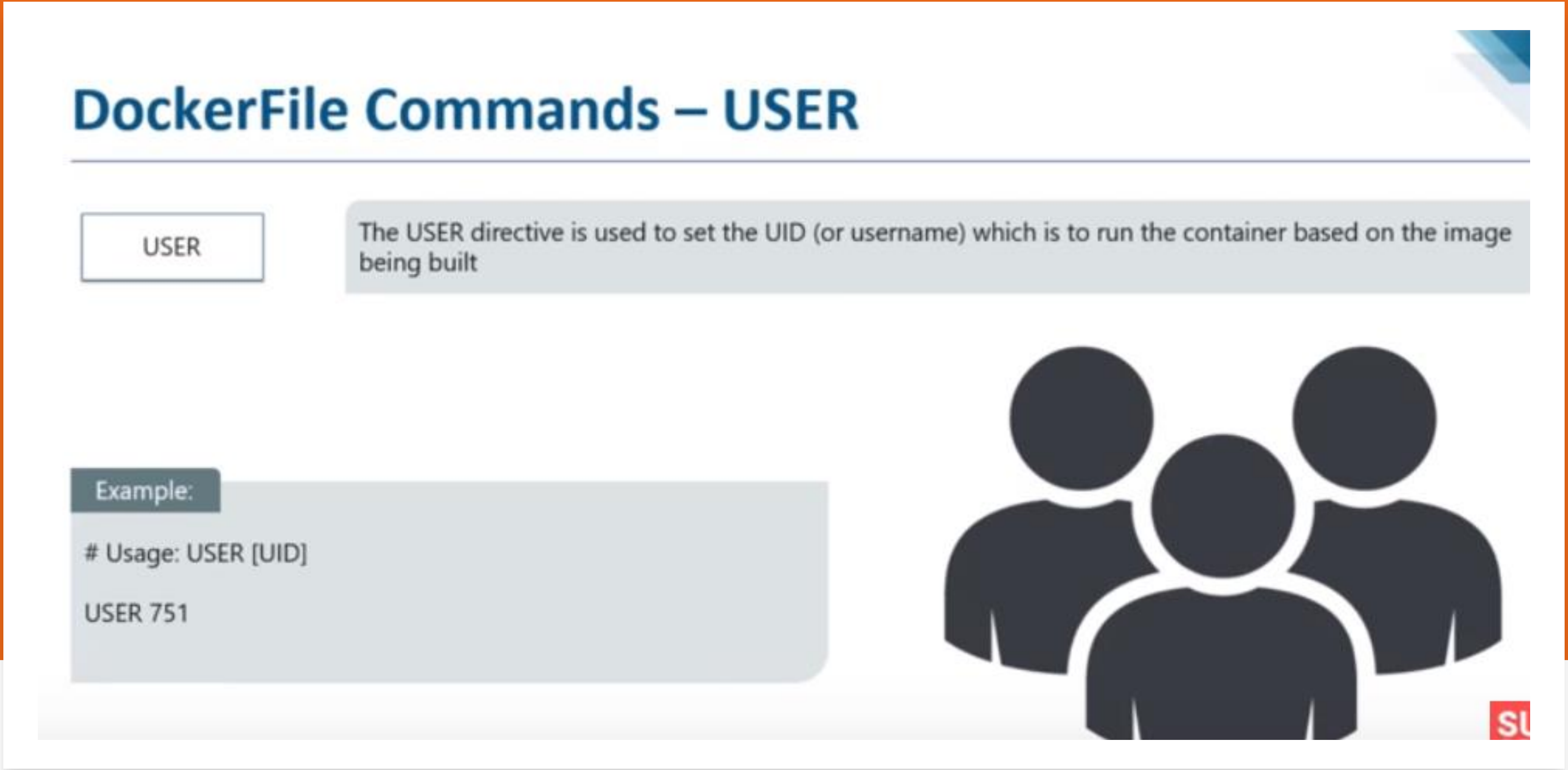
DockerFile Commands – EXPOSE

EXPOSE

The EXPOSE command is used to associate a specified port to enable networking between the running process inside the container and the outside world (i.e. the host)

Example:

```
# Usage: EXPOSE [port]

EXPOSE 8080
```



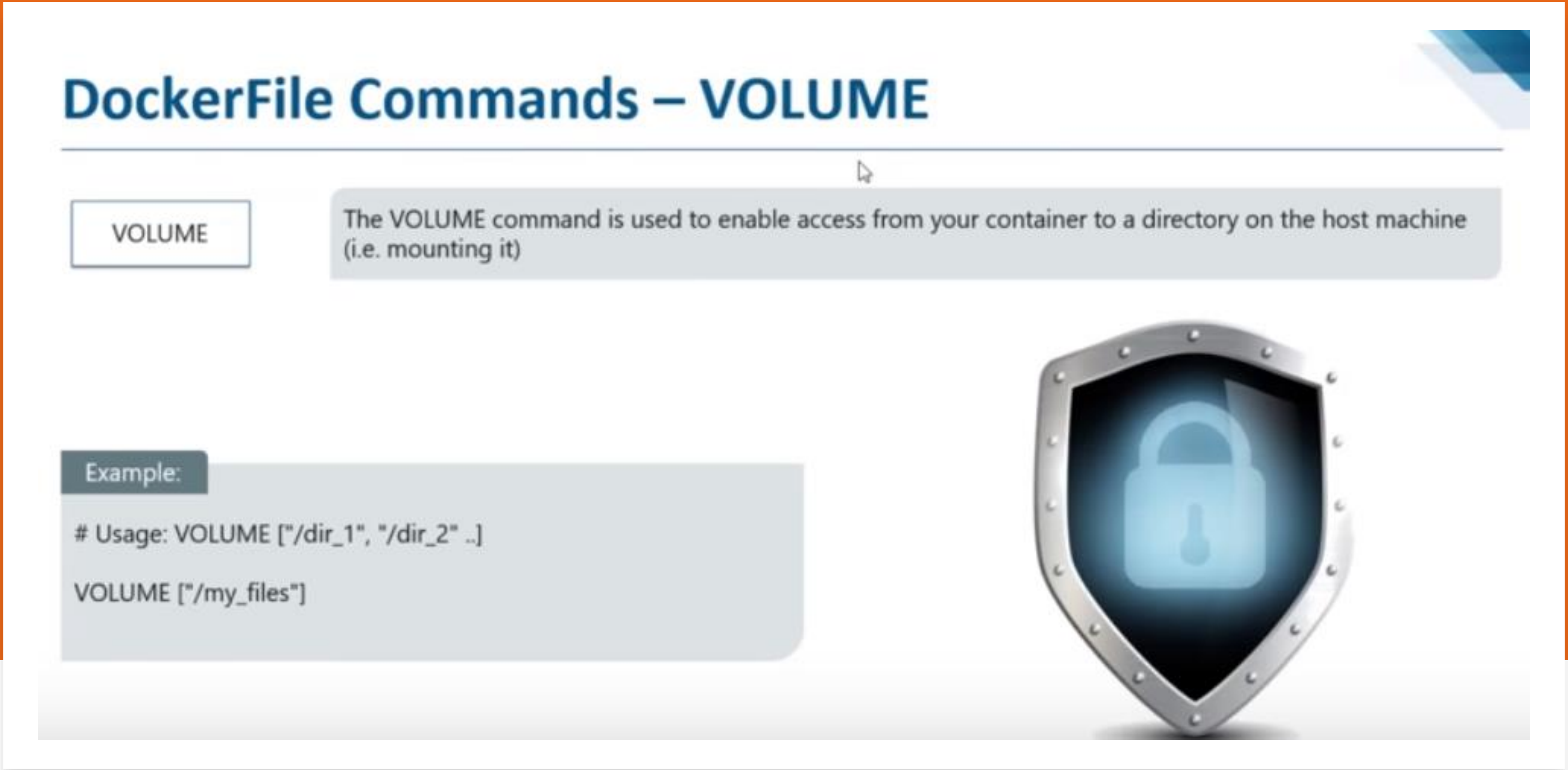
DockerFile Commands – USER

USER

The USER directive is used to set the UID (or username) which is to run the container based on the image being built

Example:

```
# Usage: USER [UID]

USER 751
```



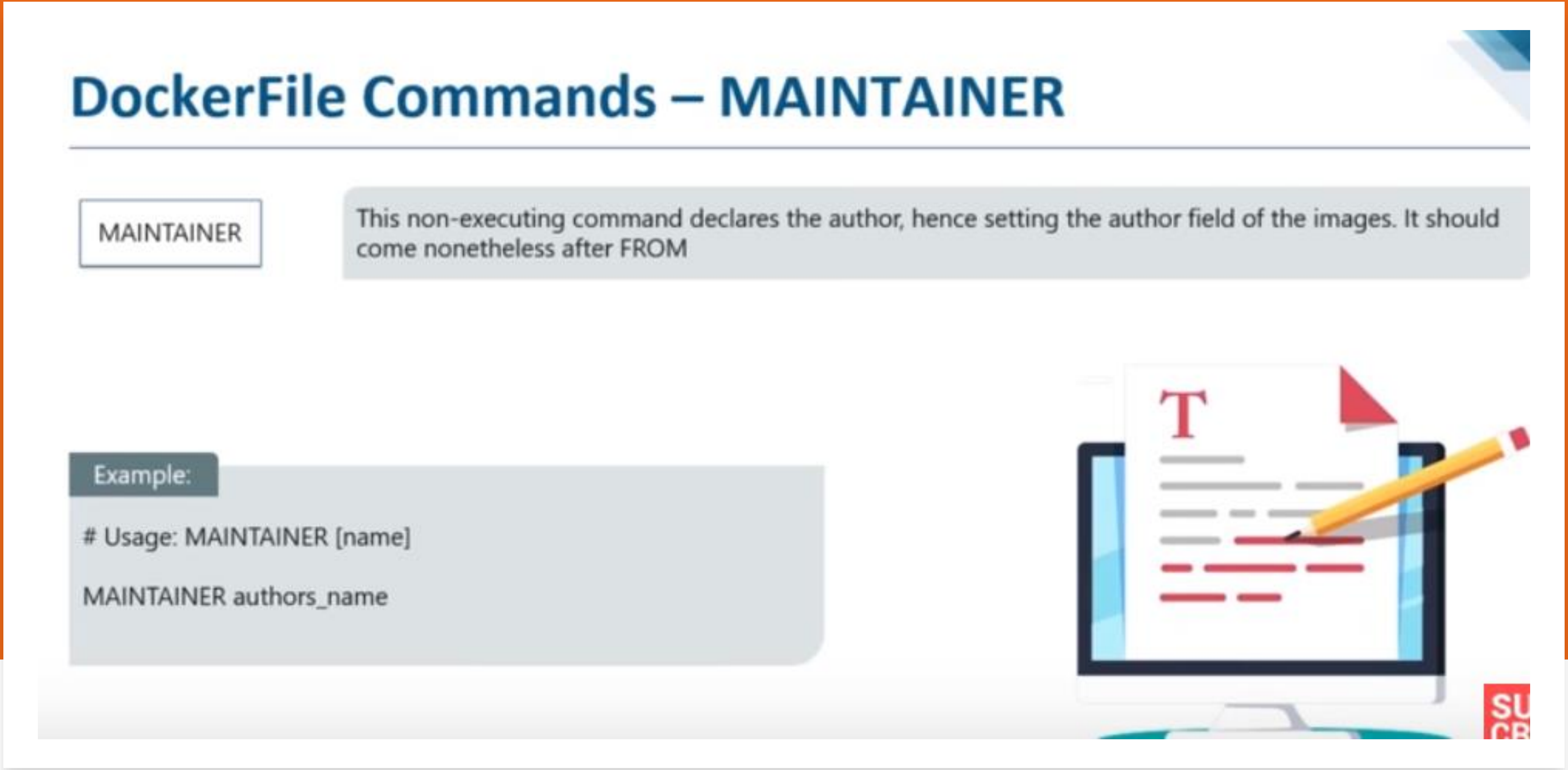
DockerFile Commands – VOLUME

VOLUME

The VOLUME command is used to enable access from your container to a directory on the host machine (i.e. mounting it)

Example:

```
# Usage: VOLUME ["/dir_1", "/dir_2" ..]

VOLUME ["/my_files"]
```



DockerFile Commands – MAINTAINER

MAINTAINER

This non-executing command declares the author, hence setting the author field of the images. It should come nonetheless after FROM

Example:

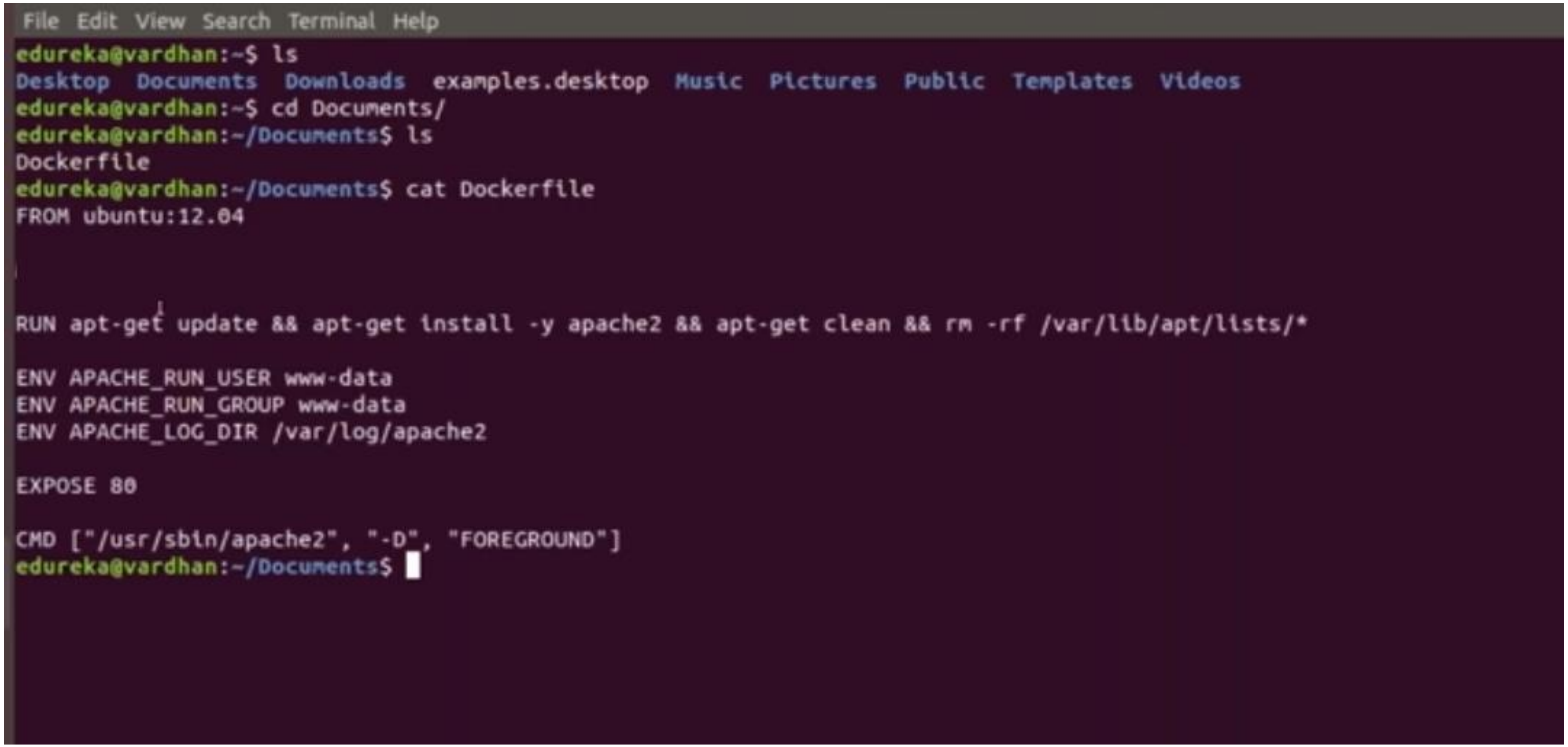
```
# Usage: MAINTAINER [name]

MAINTAINER authors_name
```

Creating an Image to Install Apache Web Server

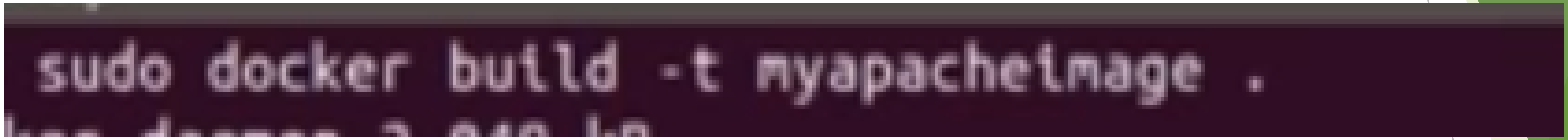
DockerFile For Installing Apache

In addition to its HTTP server capabilities, NGINX can also function as a proxy server for email (IMAP, POP3, and SMTP) and a reverse proxy and load balancer for HTTP, TCP, and UDP servers.

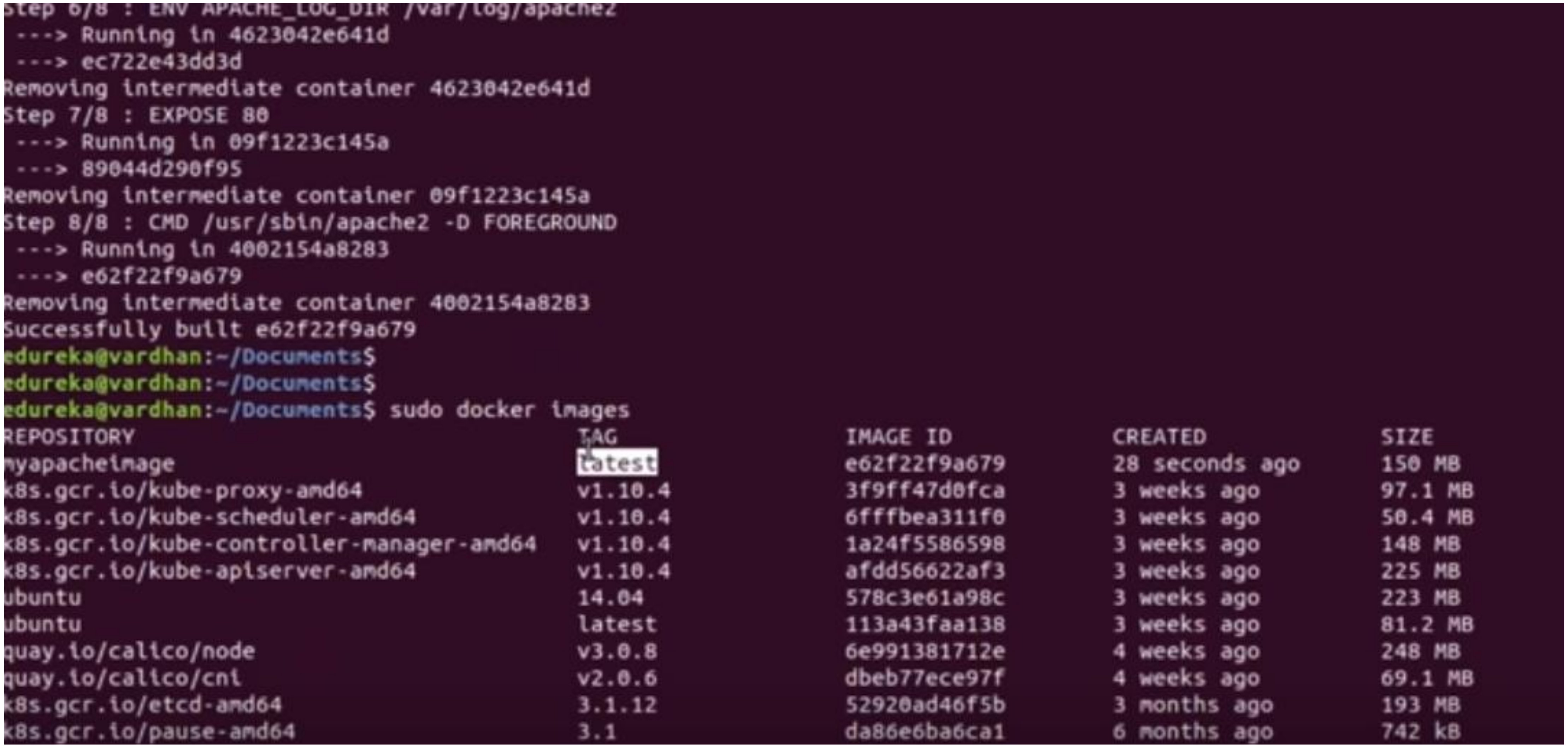
```
FROM ubuntu:12.04


RUN apt-get update && apt-get install -y apache2 && apt-get clean && rm -rf /var/lib/apt/lists/*

ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2
```



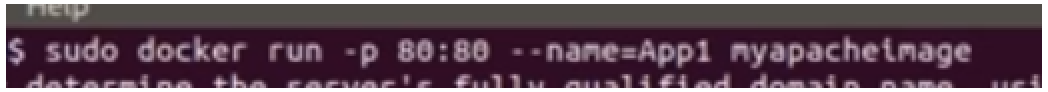
```
edureka@vardhan:~$ ls
Desktop  Documents  Downloads  examples.desktop  Music  Pictures  Public  Templates  Videos
edureka@vardhan:~$ cd Documents/
edureka@vardhan:~/Documents$ ls
Dockerfile
edureka@vardhan:~/Documents$ cat Dockerfile
FROM ubuntu:12.04



RUN apt-get update && apt-get install -y apache2 && apt-get clean && rm -rf /var/lib/apt/lists/*

ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2

EXPOSE 80

CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]
edureka@vardhan:~/Documents$
```



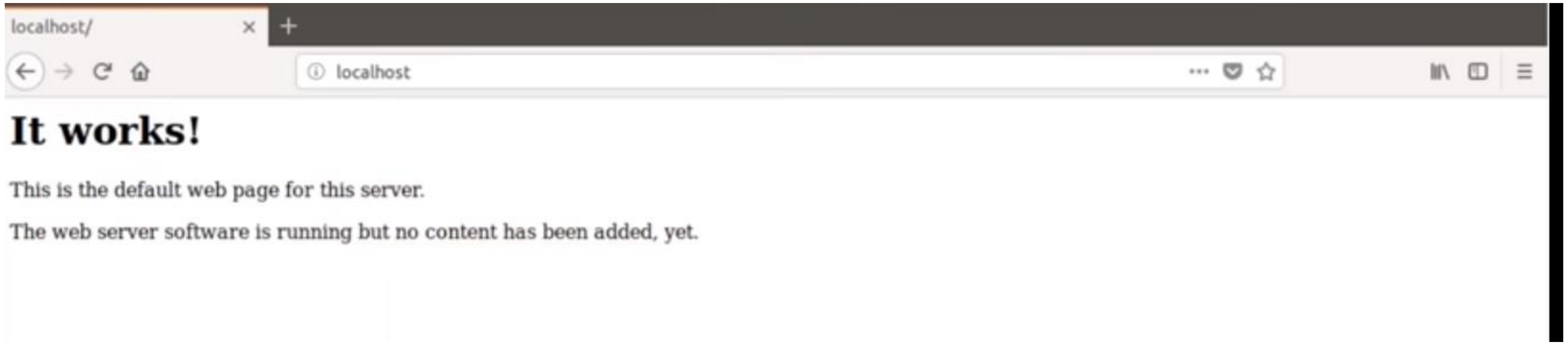
```
 sudo docker build -t myapacheimage .
```



```
Step 6/8 : ENV APACHE_LOG_DIR /var/log/apache2
 ---> Running in 4623042e641d
 ---> ec722e43dd3d
Removing intermediate container 4623042e641d
Step 7/8 : EXPOSE 80
 ---> Running in 09f1223c145a
 ---> 89044d290f95
Removing intermediate container 09f1223c145a
Step 8/8 : CMD /usr/sbin/apache2 -D FOREGROUND
 ---> Running in 4002154a8283
 ---> e62f22f9a679
Removing intermediate container 4002154a8283
Successfully built e62f22f9a679
edureka@vardhan:~/Documents$
edureka@vardhan:~/Documents$
edureka@vardhan:~/Documents$ sudo docker images
REPOSITORY                                TAG           IMAGE ID        CREATED            SIZE
myapacheimage                             latest        e62f22f9a679    28 seconds ago     150 MB
k8s.gcr.io/kube-proxy-amd64               v1.10.4       3f9ff47d0fca    3 weeks ago        97.1 MB
k8s.gcr.io/kube-scheduler-amd64           v1.10.4       6fffbea311f0    3 weeks ago        50.4 MB
k8s.gcr.io/kube-controller-manager-amd64  v1.10.4       1a24f5586598    3 weeks ago        148 MB
k8s.gcr.io/kube-apiserver-amd64           v1.10.4       afdd56622af3    3 weeks ago        225 MB
ubuntu                                    14.04         578c3e61a98c    3 weeks ago        223 MB
ubuntu                                    latest        113a43faa138    3 weeks ago        81.2 MB
quay.io/calico/node                       v3.0.8        6e991381712e    4 weeks ago        248 MB
quay.io/calico/cni                        v2.0.6        dbeb77ece97f    4 weeks ago        69.1 MB
k8s.gcr.io/etcd-amd64                     3.1.12        52920ad46f5b    3 months ago       193 MB
k8s.gcr.io/pause-amd64                    3.1           da86e6ba6ca1    6 months ago       742 kB
```



```
Help
$ sudo docker run -p 80:80 --name=App1 myapacheimage
 determine the server's fully qualified domain name, usi
```

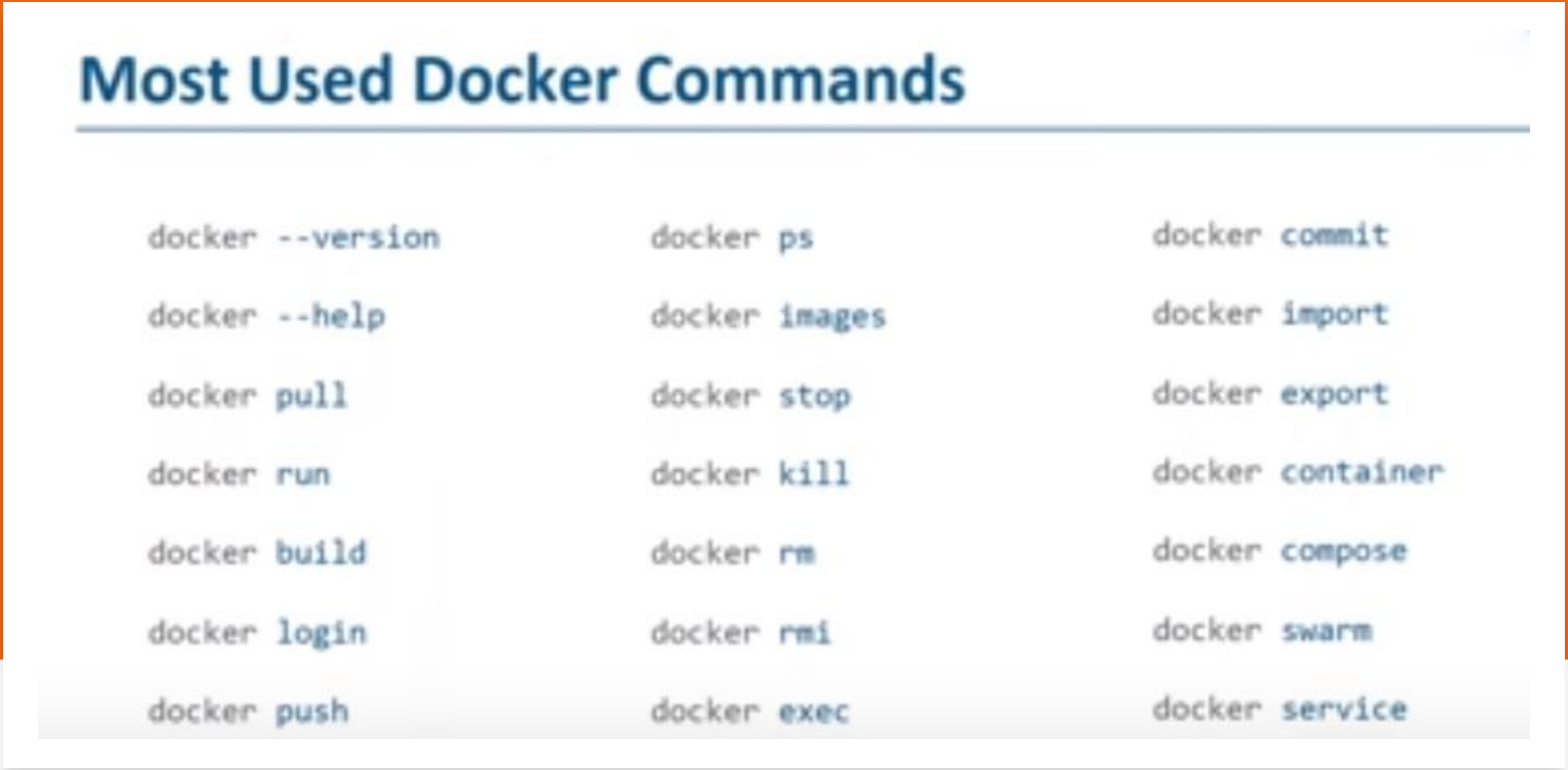


It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.

Docker Commands



Most Used Docker Commands

```
docker --version      docker ps         docker commit
docker --help         docker images     docker import
docker pull           docker stop       docker export
docker run            docker kill       docker container
docker build          docker rm         docker compose
docker login          docker rmi        docker swarm
docker push           docker exec       docker service
```

Basic Docker Commands

docker pull

```
$ docker pull ubuntu
```

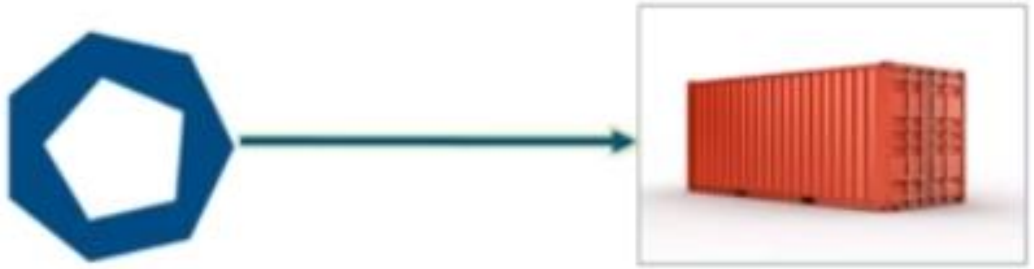
This command pulls a new Docker image from the Docker Hub

Basic Docker Commands

docker run

```
$ docker run ubuntu
```

This command executes a Docker image on your local repo & creates a running Container out of it

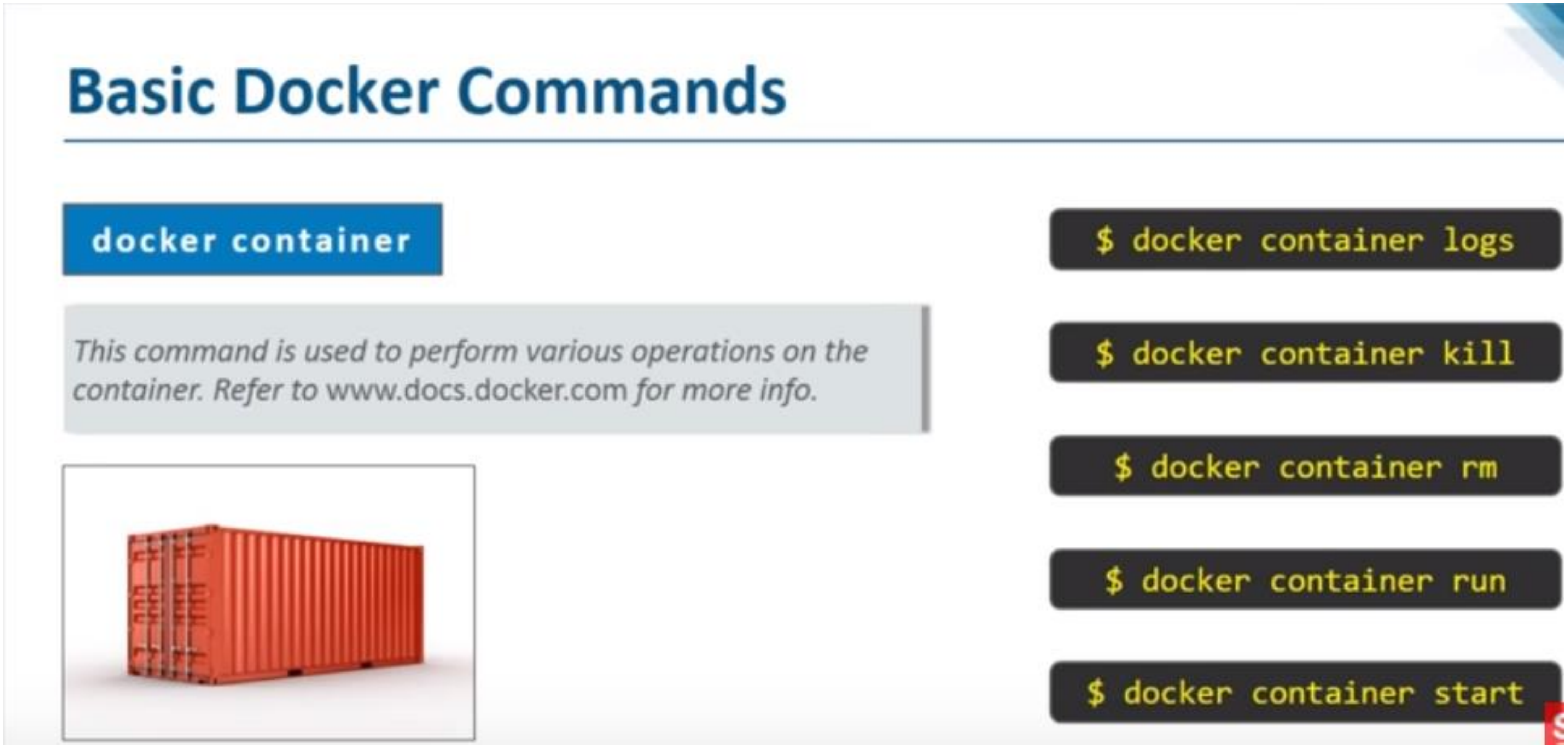


Basic Docker Commands

docker build

```
$ docker build -t MyUbuntuImage .
```

This command is used to compile the Dockerfile, for building custom Docker images based on the



Basic Docker Commands

docker container

This command is used to perform various operations on the container. Refer to www.docs.docker.com *for more info.*

```
$ docker container logs
```

```
$ docker container kill
```

```
$ docker container rm
```

```
$ docker container run
```

```
$ docker container start
```

Basic Docker Commands

docker login

```
$ docker login
```

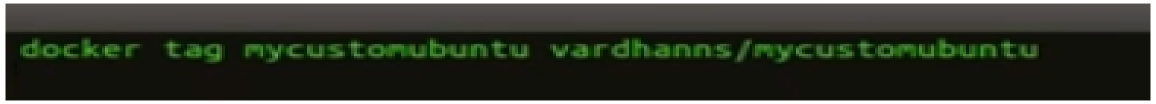
This command is used to Login to Docker Hub repo from the CLI

Basic Docker Commands

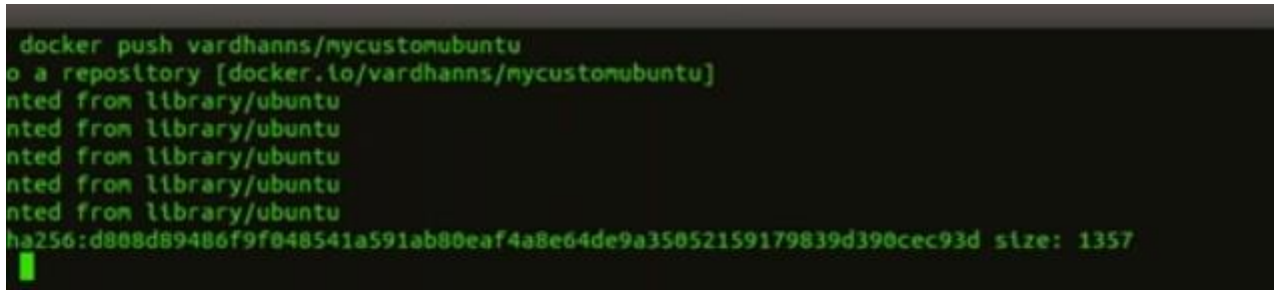
docker push

```
$ docker push vardhanns/MyUbuntuImage
```

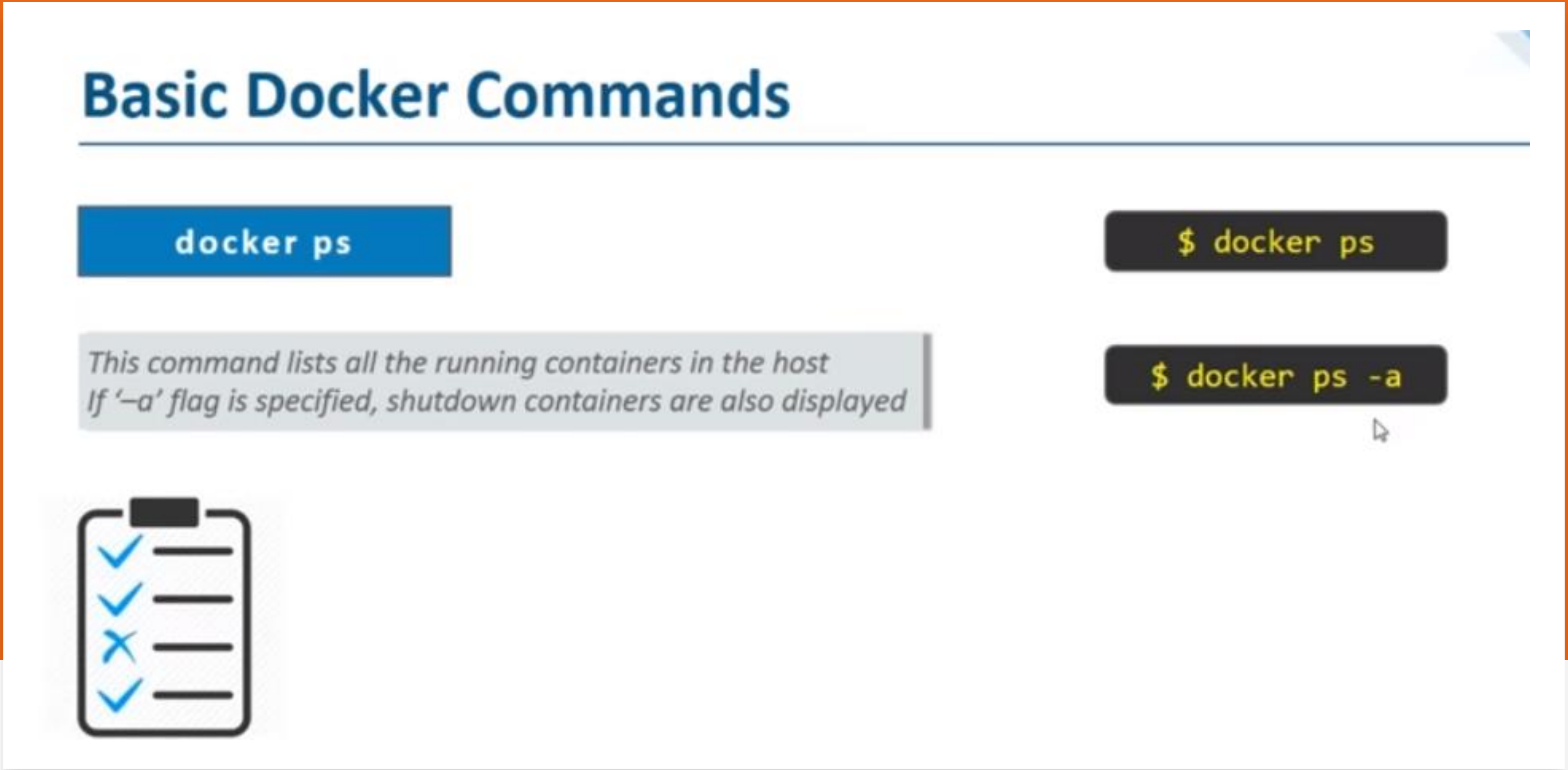
This command pushes a Docker image on your local repo to the Docker Hub



```
docker tag mycustomubuntu vardhanns/mycustomubuntu
```



```
 docker push vardhanns/mycustomubuntu
o a repository [docker.io/vardhanns/mycustomubuntu]
nted from library/ubuntu
nted from library/ubuntu
nted from library/ubuntu
nted from library/ubuntu
nted from library/ubuntu
ha256:d808d89486f9f048541a591ab80eaf4a8e64de9a35052159179839d390cec93d size: 1357
```



Basic Docker Commands

docker ps

This command lists all the running containers in the host
If '–a' flag is specified, shutdown containers are also displayed

```
$ docker ps
```

```
$ docker ps -a
```

Basic Docker Commands

docker stop

```
$ docker stop fe6e370a1c9c
```

This command shuts down the container whose Container ID is specified in arguments. Container is shut down gracefully by waiting for other dependencies to shut

Basic Docker Commands

docker rm

```
$ docker rm fe6e370a1c9c
```

This command removes the container whose Container ID is specified in arguments

Basic Docker Commands

docker exec

```
$ docker exec -it fe6e370a1c9c bash
```

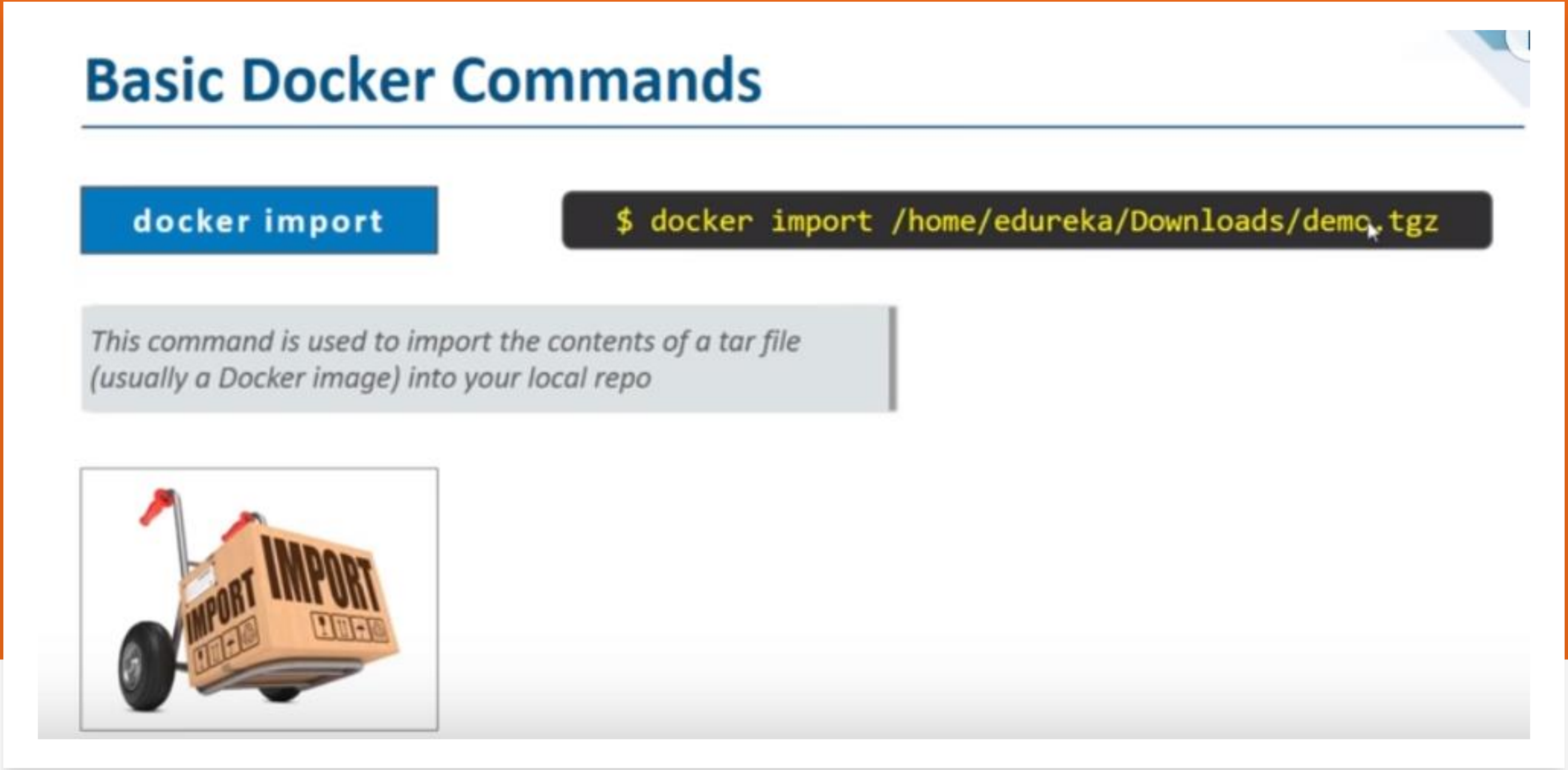
This command is used to access an already running container and perform operations inside the container

Basic Docker Commands

docker export

```
$ docker export --output="latest.tar" mycontainer
```

This command is used to export a Docker image into a tar file in your local system

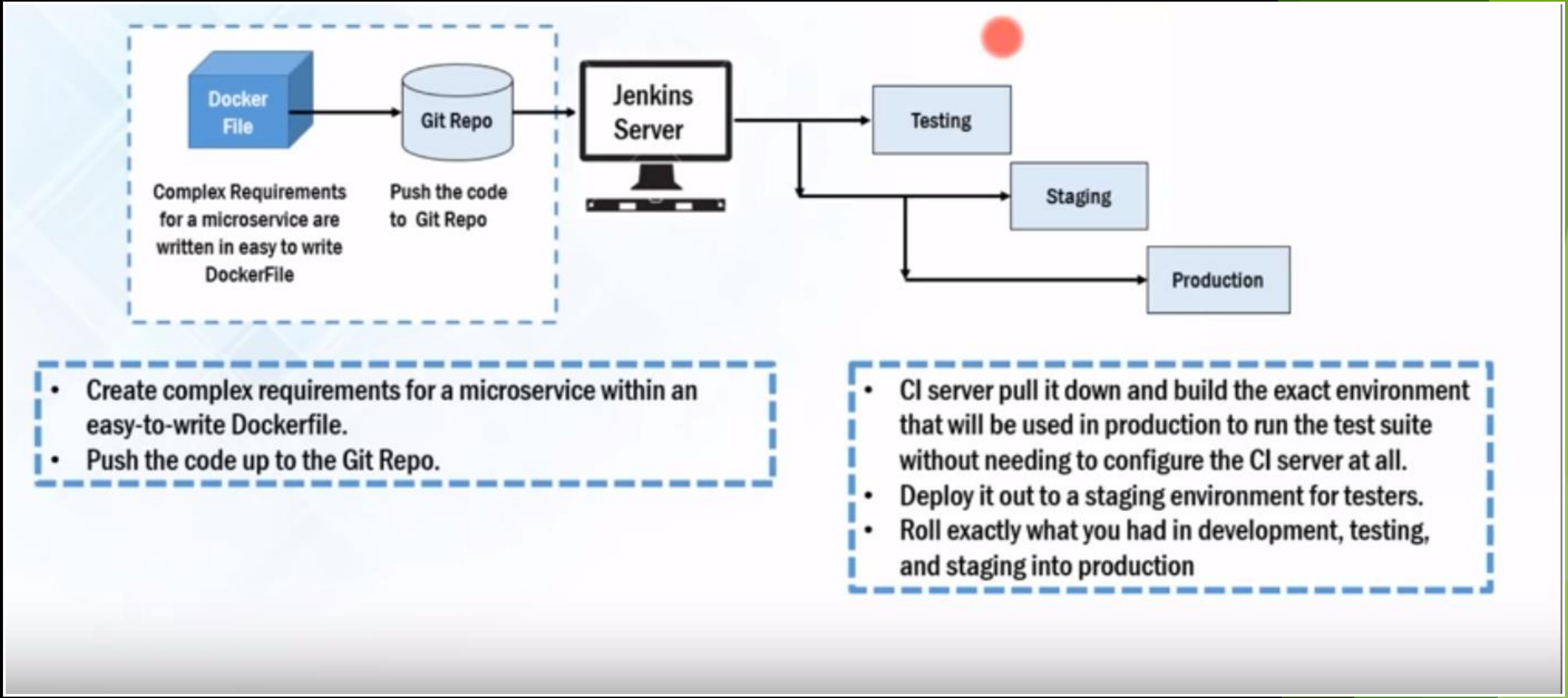


Basic Docker Commands

docker import

```
$ docker import /home/edureka/Downloads/demo.tgz
```

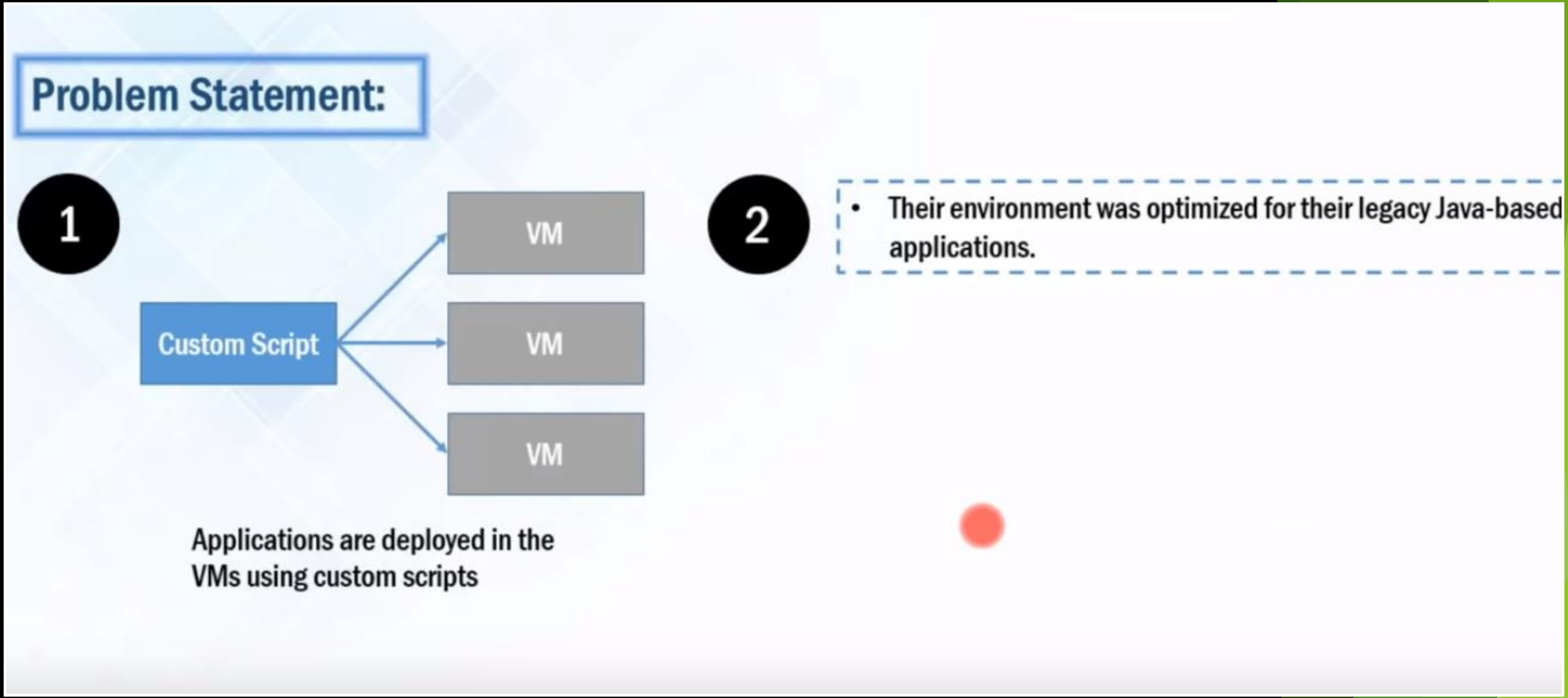
This command is used to import the contents of a tar file (usually a Docker image) into your local repo


Docker File
Git Repo
Jenkins Server
Testing
Staging
Production
Complex Requirements for a microservice are written in easy to write DockerFile
Push the code to Git Repo

- Create complex requirements for a microservice within an easy-to-write Dockerfile.
- Push the code up to the Git Repo.

- CI server pull it down and build the exact environment that will be used in production to run the test suite without needing to configure the CI server at all.
- Deploy it out to a staging environment for testers.
- Roll exactly what you had in development, testing, and staging into production

Docker Example



Problem Statement:

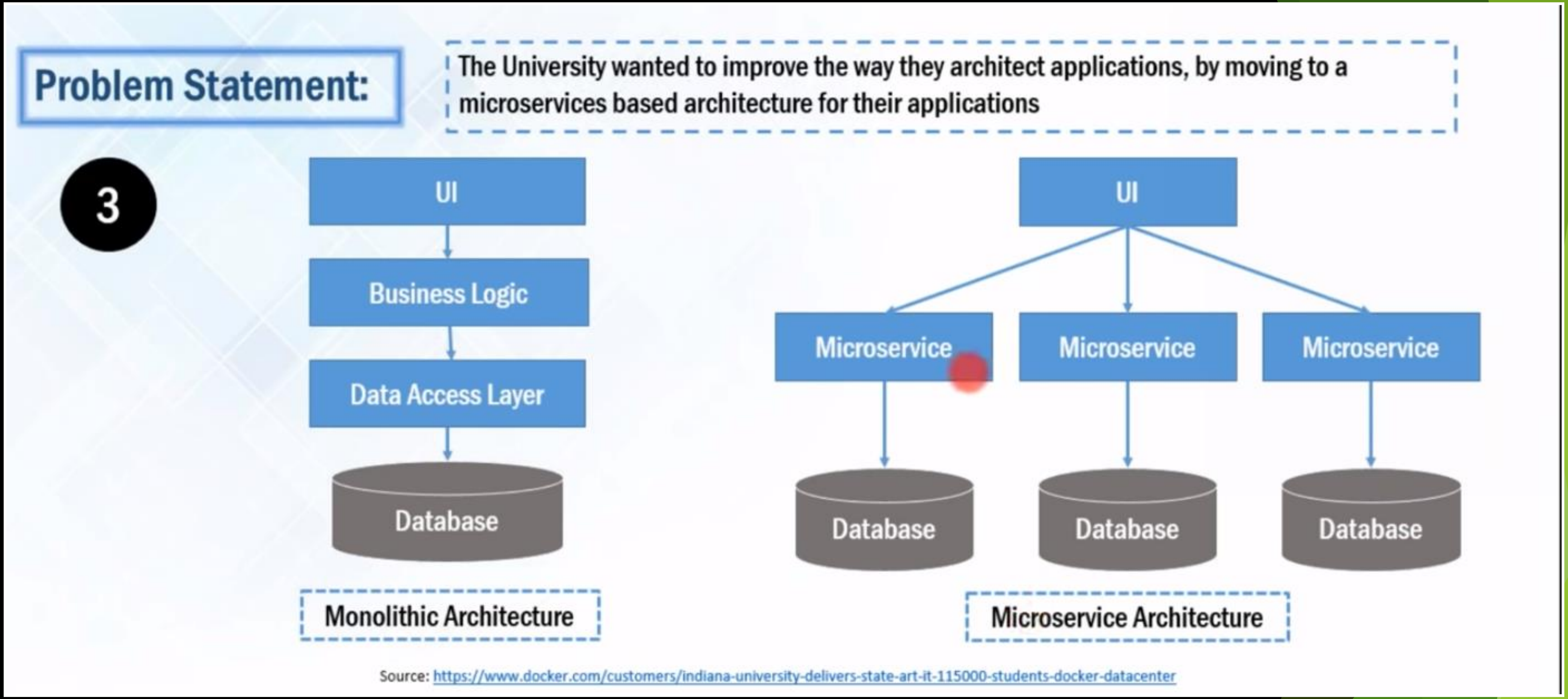
1

Applications are deployed in the VMs using custom scripts

2

- Their environment was optimized for their legacy Java-based applications.

Docker Case-Study Indiana University



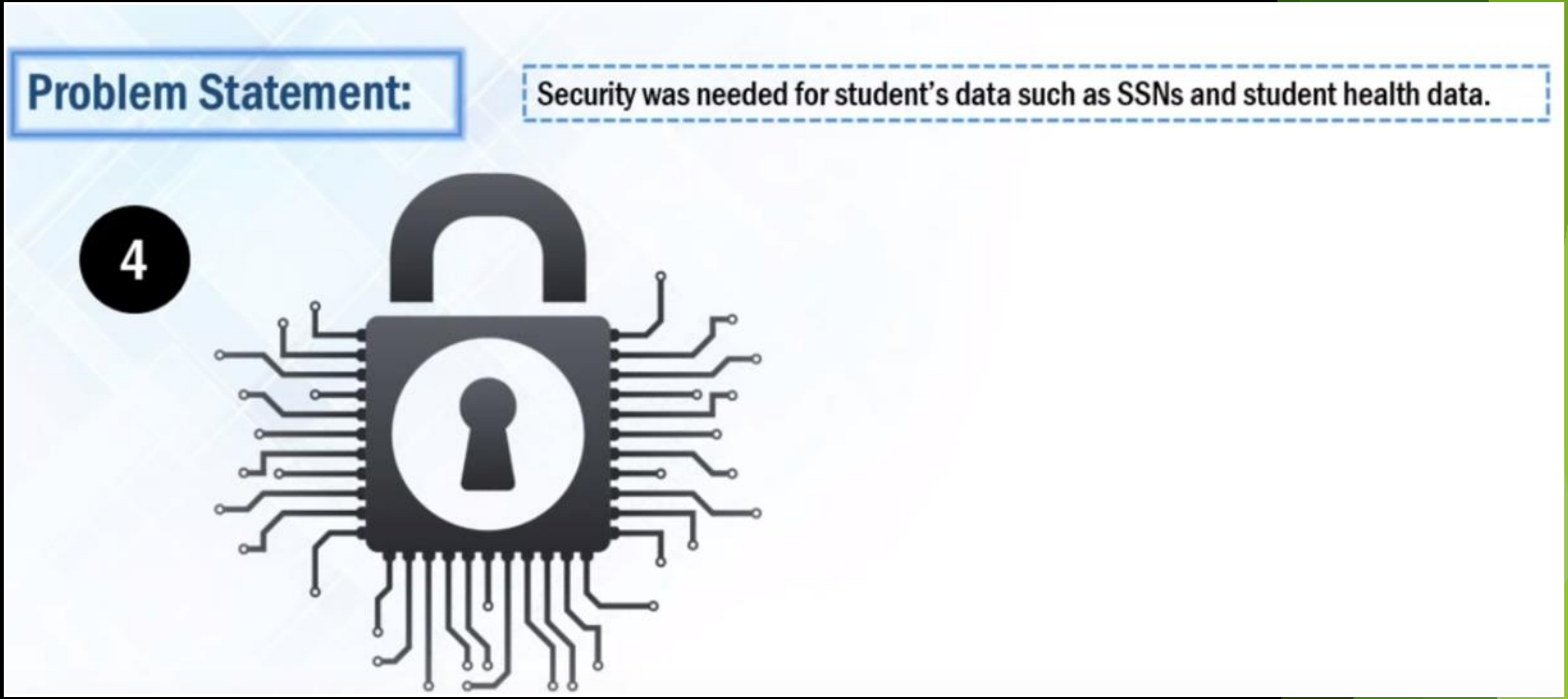
Problem Statement:

The University wanted to improve the way they architect applications, by moving to a microservices based architecture for their applications

3

Source: https://www.docker.com/customers/indiana-university-delivers-state-art-it-115000-students-docker-datacenter

Docker Case-Study Indiana University

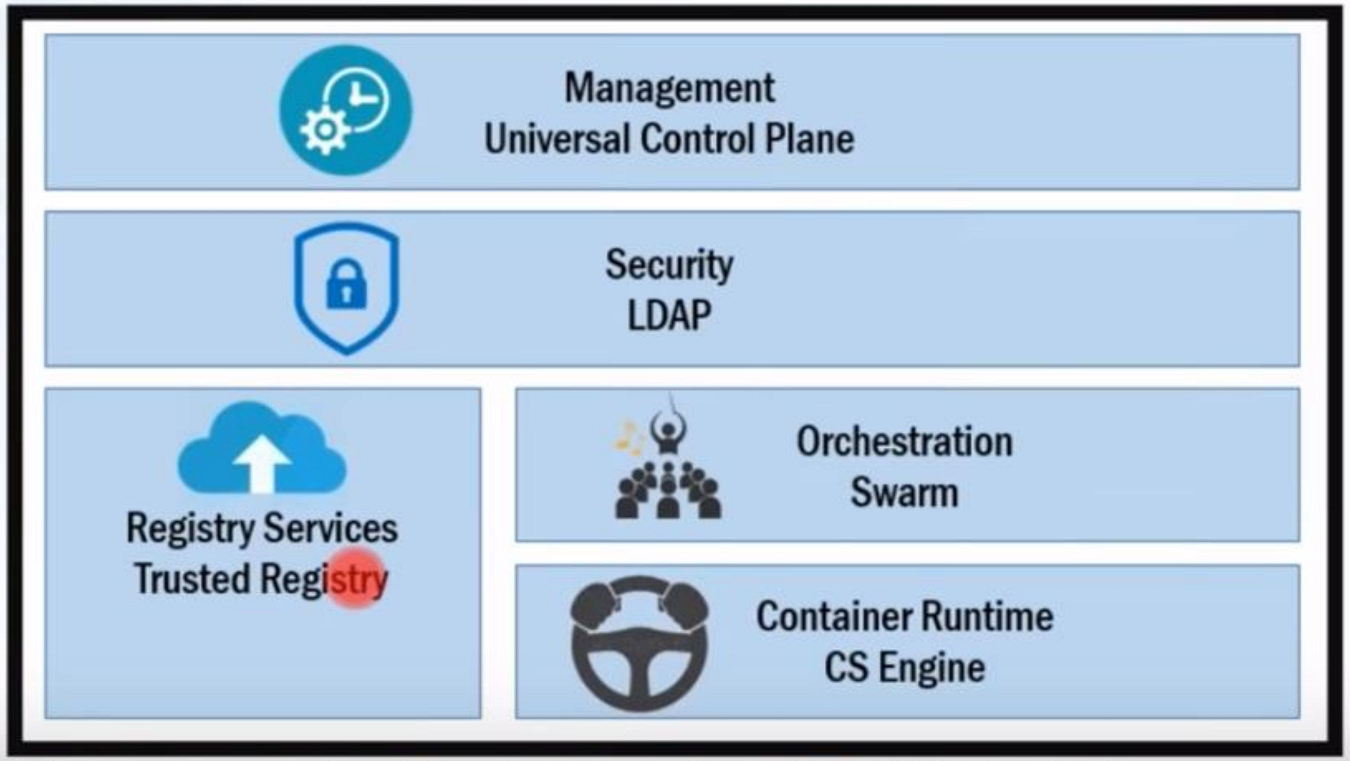

Problem Statement:

Security was needed for student's data such as SSNs and student health data.

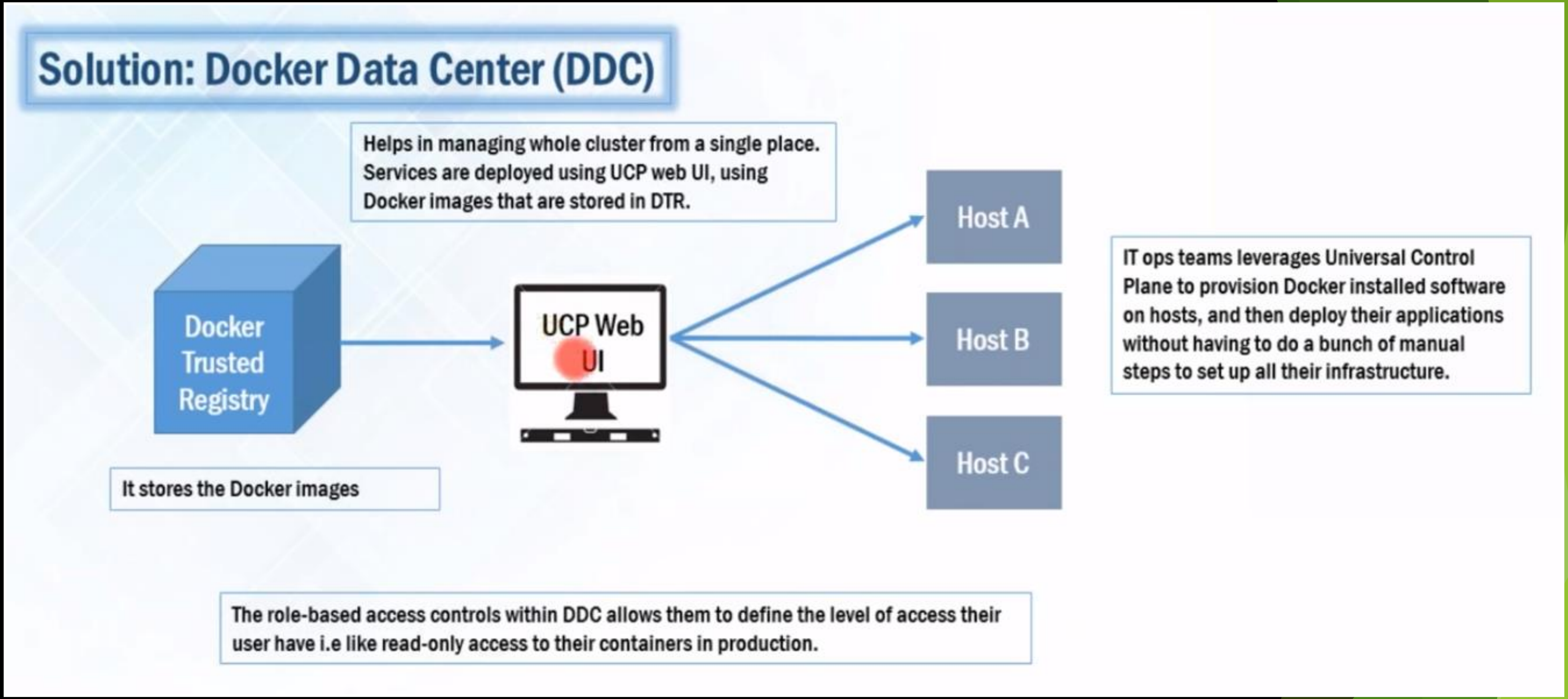
4

Docker Case-Study Indiana University

Solution: Docker Data Center (DDC)



Source: https://www.docker.com/customers/indiana-university-delivers-state-art-it-115000-students-docker-datacenter



Solution: Docker Data Center (DDC)

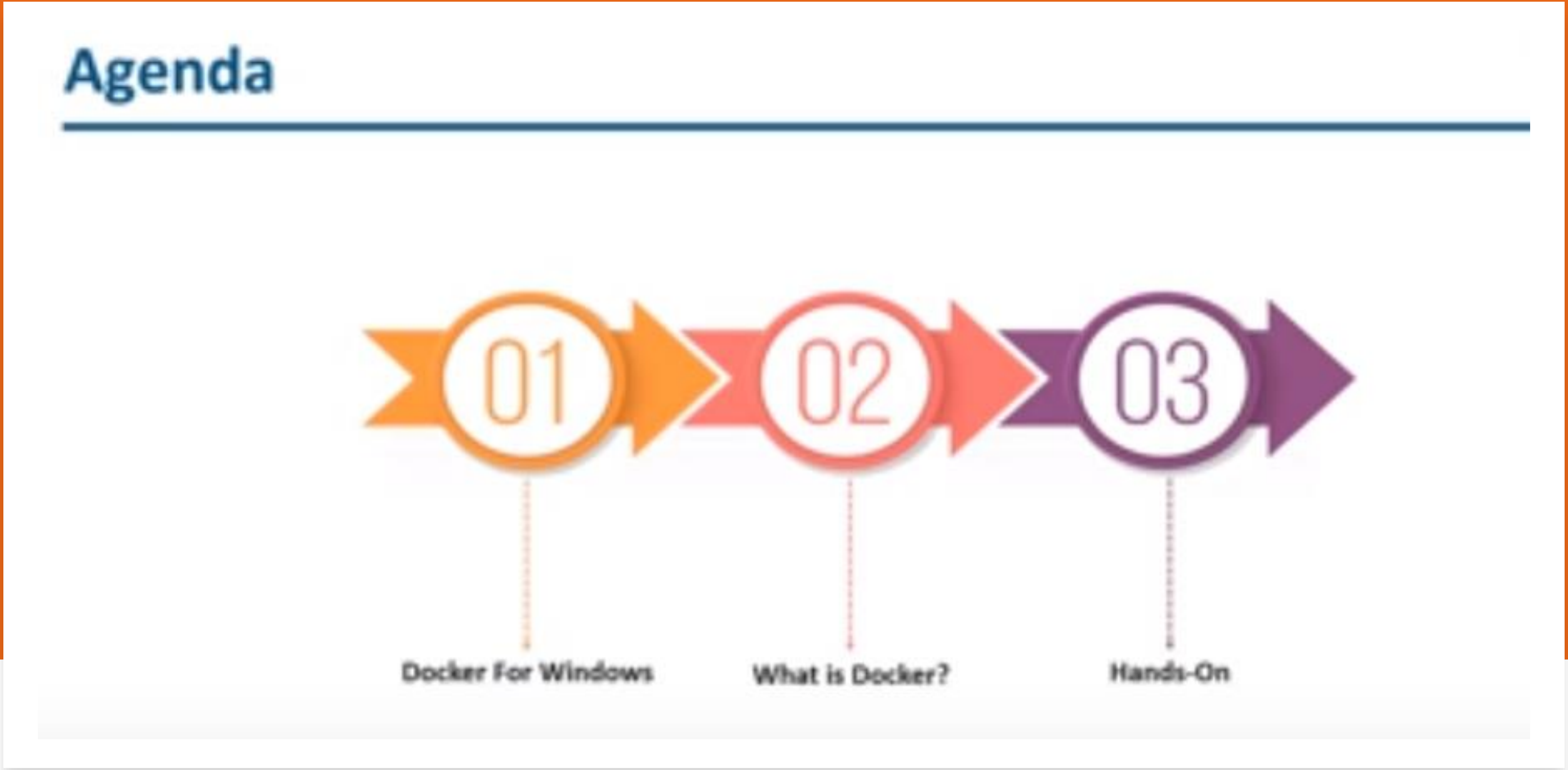
Helps in managing whole cluster from a single place. Services are deployed using UCP web UI, using Docker images that are stored in DTR.

IT ops teams leverages Universal Control Plane to provision Docker installed software on hosts, and then deploy their applications without having to do a bunch of manual steps to set up all their infrastructure.

It stores the Docker images

The role-based access controls within DDC allows them to define the level of access their user have i.e like read-only access to their containers in production.

Docker for Windows
Windows
63%



Agenda

01 Docker For Windows

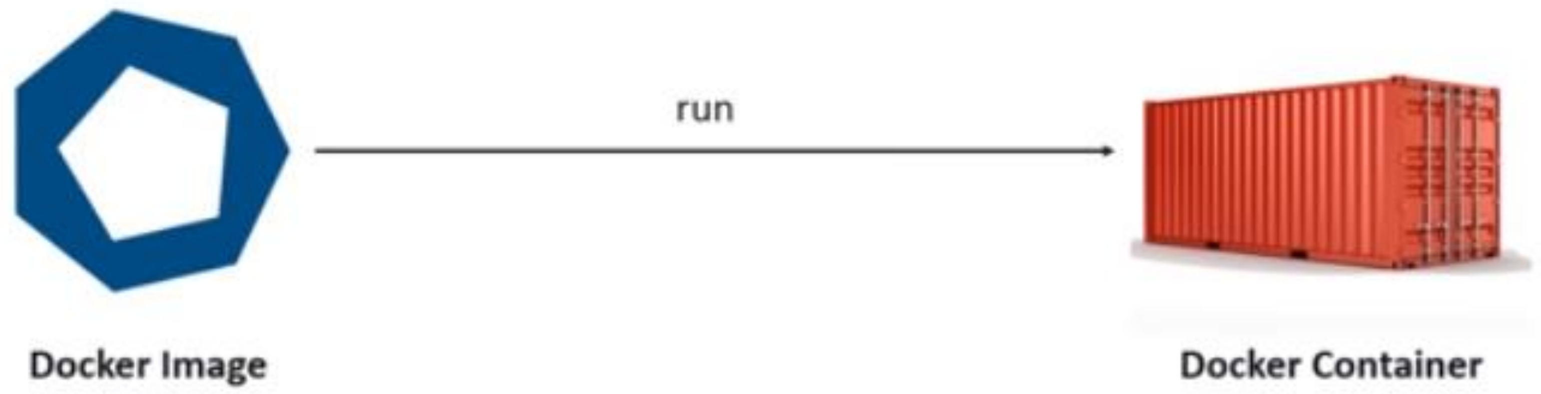
02 What is Docker?

03 Hands-On

Pre-requisites

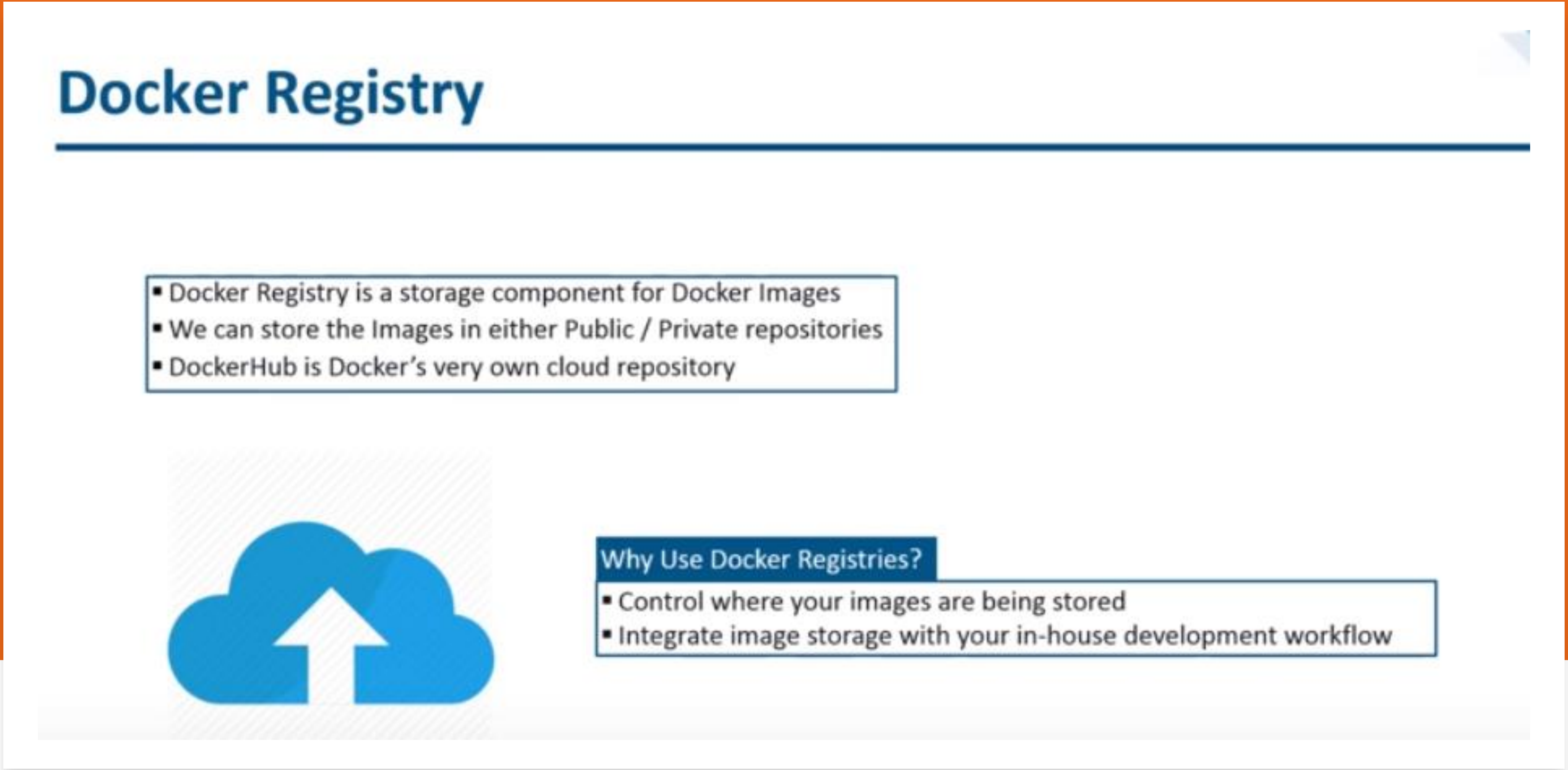
- ✓ 64bit Windows 10(Pro, Edu, Stu)
- ✓ Enable Hyper-V
- ✓ Enable virtualization

Docker Image & Container



- Read Only Template Used To Create Containers
- Built By Docker Users
- Stored In Docker Hub Or Your Local Registry

- Isolated Application Platform
- Contains Everything Needed To Run The Applicatio
- Built From One Or More Images



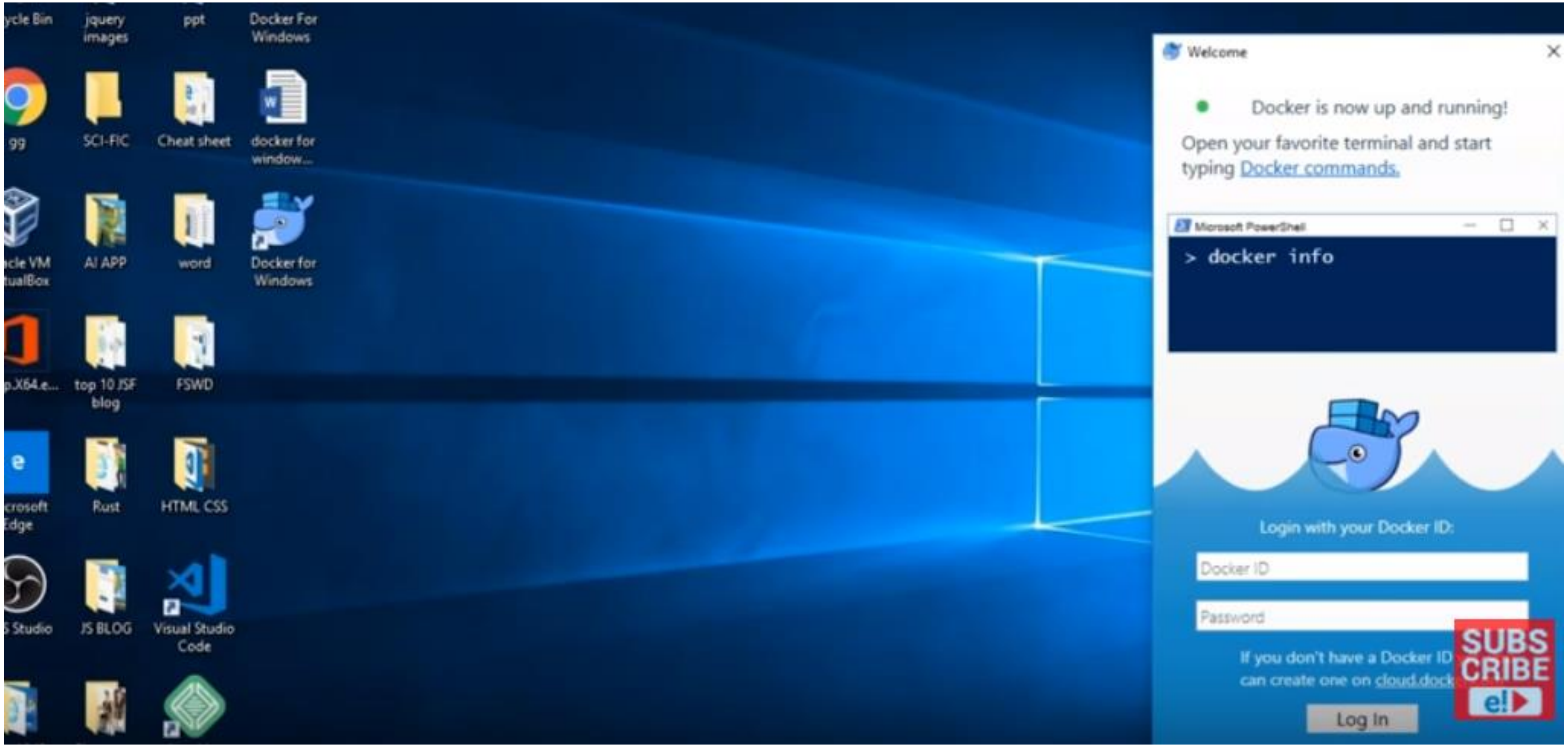
Docker Registry

- Docker Registry is a storage component for Docker Images
- We can store the Images in either Public / Private repositories
- DockerHub 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

Docker For Windows Demo

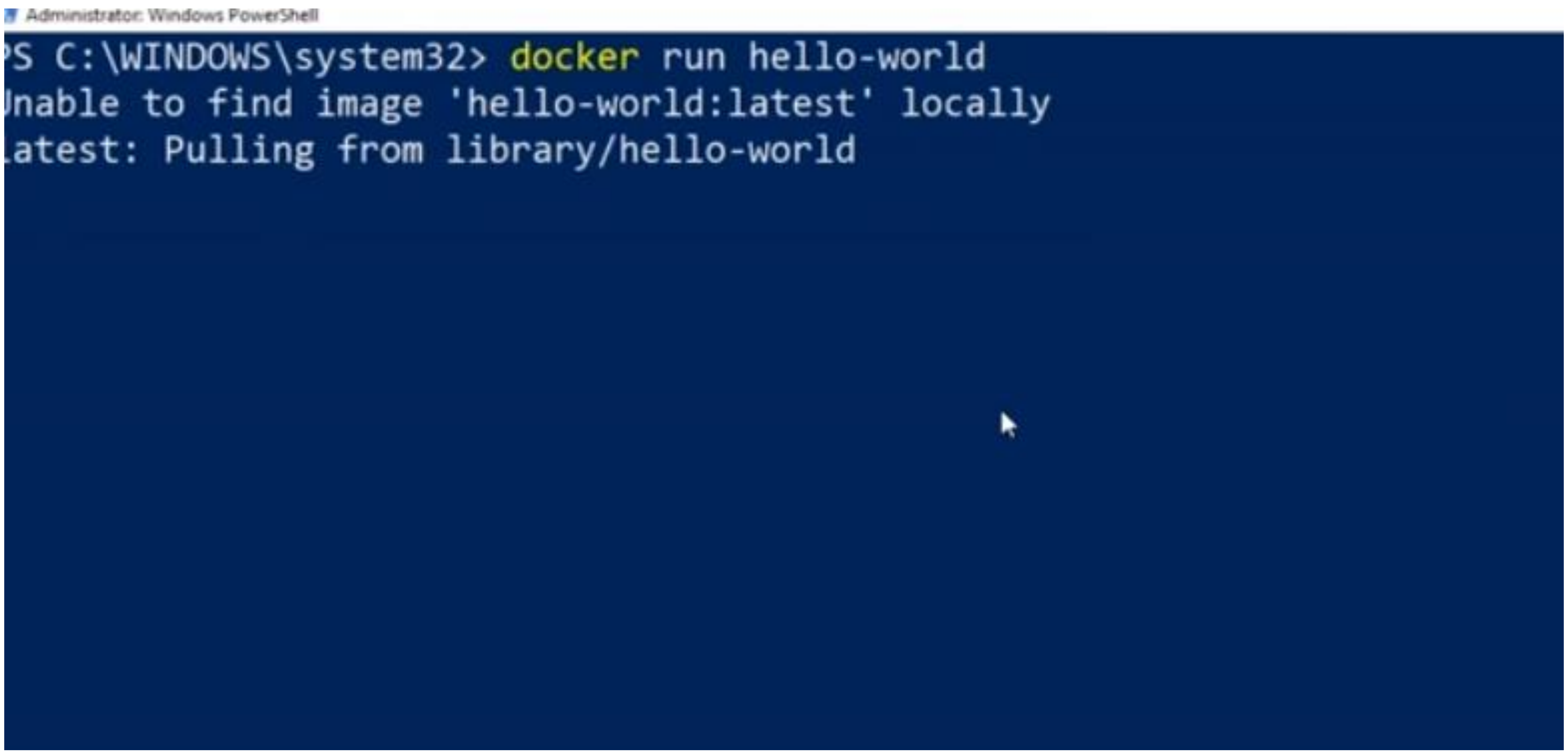


Docker is now up and running!

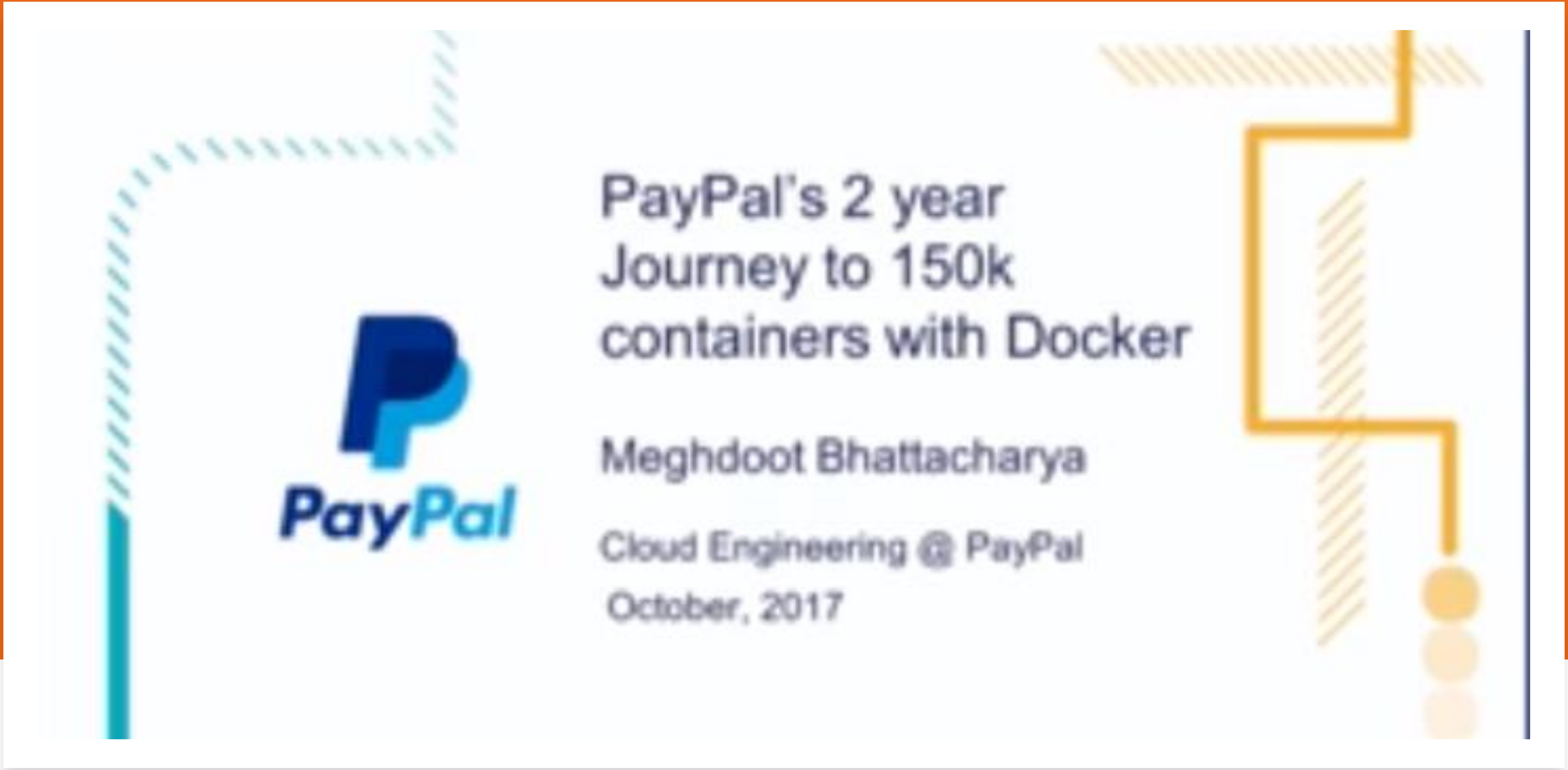
Open your favorite terminal and start typing Docker commands.

```
> docker info
```

If you don't have a Docker ID can create one on cloud.dock


Administrator: Windows PowerShell

```
'S C:\WINDOWS\system32> docker run hello-world
Jnable to find image 'hello-world:latest' locally
.atest: Pulling from library/hello-world
```

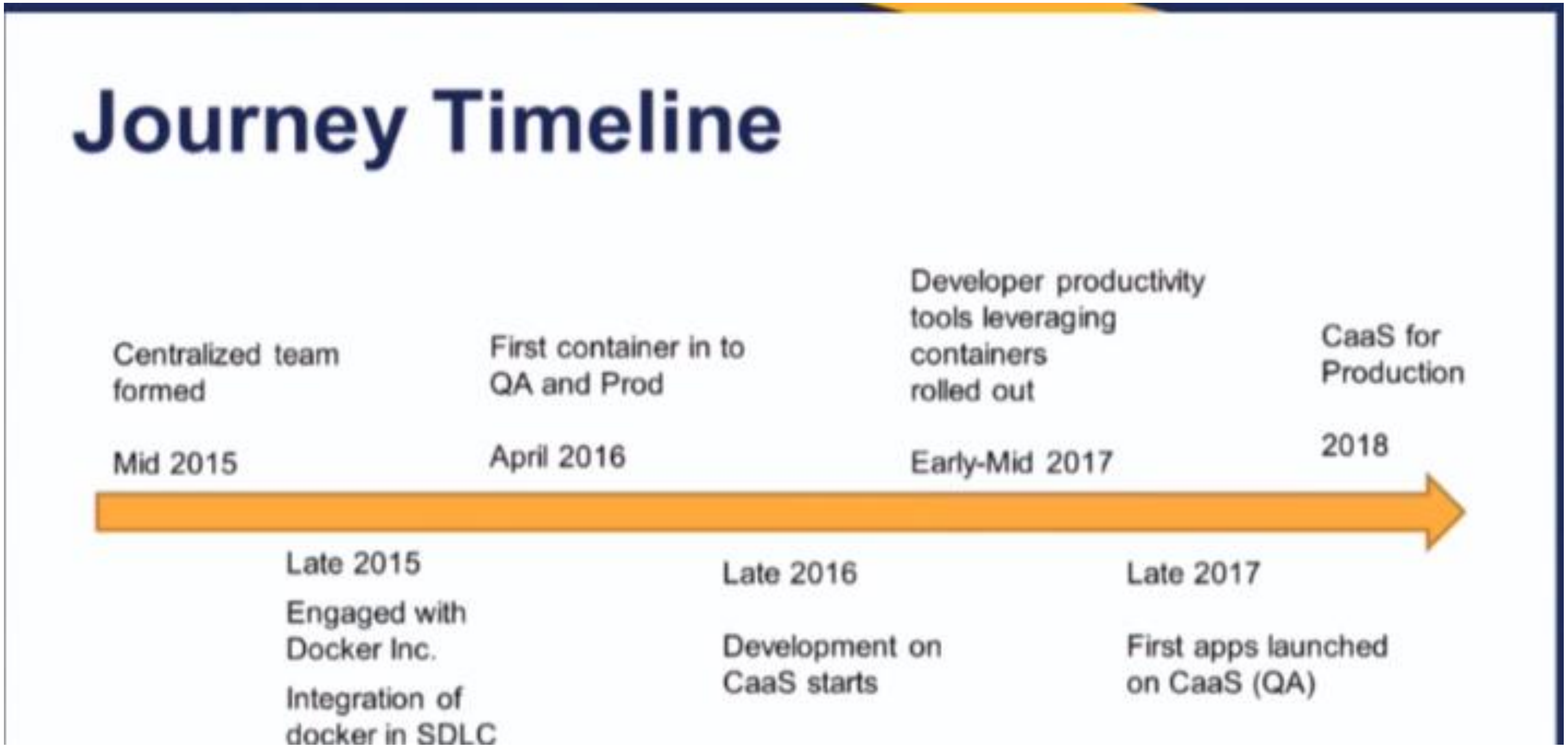


PayPal's 2 year Journey to 150k containers with Docker

Meghdoot Bhattacharya

Cloud Engineering @ PayPal

October, 2017



Journey Timeline

Centralized team formed
Mid 2015
Late 2015
Engaged with Docker Inc.
Integration of docker in SDLC
First container in to QA and Prod
April 2016
Late 2016
Development on CaaS starts
Developer productivity tools leveraging containers rolled out
Early-Mid 2017
Late 2017
First apps launched on CaaS (QA)
CaaS for Production
2018