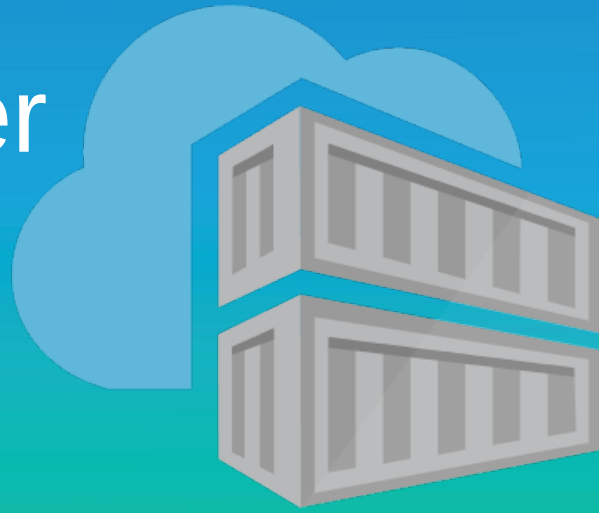


Introduction to Docker



docker

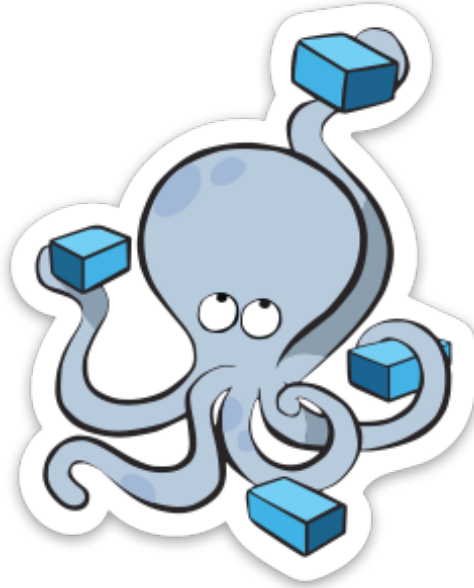
Agenda

Section 1:

What is Docker
What is Docker Not
Basic Docker Commands
Dockerfiles

Section 2:

Anatomy of a Docker image
Docker volumes



Section 3:

Networking

Section 4:

Docker compose / stacks
Demo

Section 1: What is Docker

Basic Docker Commands

Dockerfiles



What is docker

Docker is an open platform for developing, shipping, and running applications

Docker enables you to separate your applications from your infrastructure so you can deliver software quickly

With Docker, you can manage your infrastructure in the same ways you manage your applications

Docker provides the ability to package and run an application in a loosely isolated environment called a container

The isolation and security allows you to run many containers simultaneously on a given host.

Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host.

You can easily share containers while you work, and be sure that everyone you share with gets the same container that works in the same way.



Docker provides tooling and a platform to manage the lifecycle of your containers:

Develop your application and its supporting components using containers.

The container becomes the unit for distributing and testing your application.

When you're ready, deploy your application into your production environment, as a container or an orchestrated service.

This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.



What can I use Docker for?

Fast, consistent delivery of your applications

Containers are great for continuous integration and continuous delivery (CI/CD) workflows

Responsive deployment and scaling

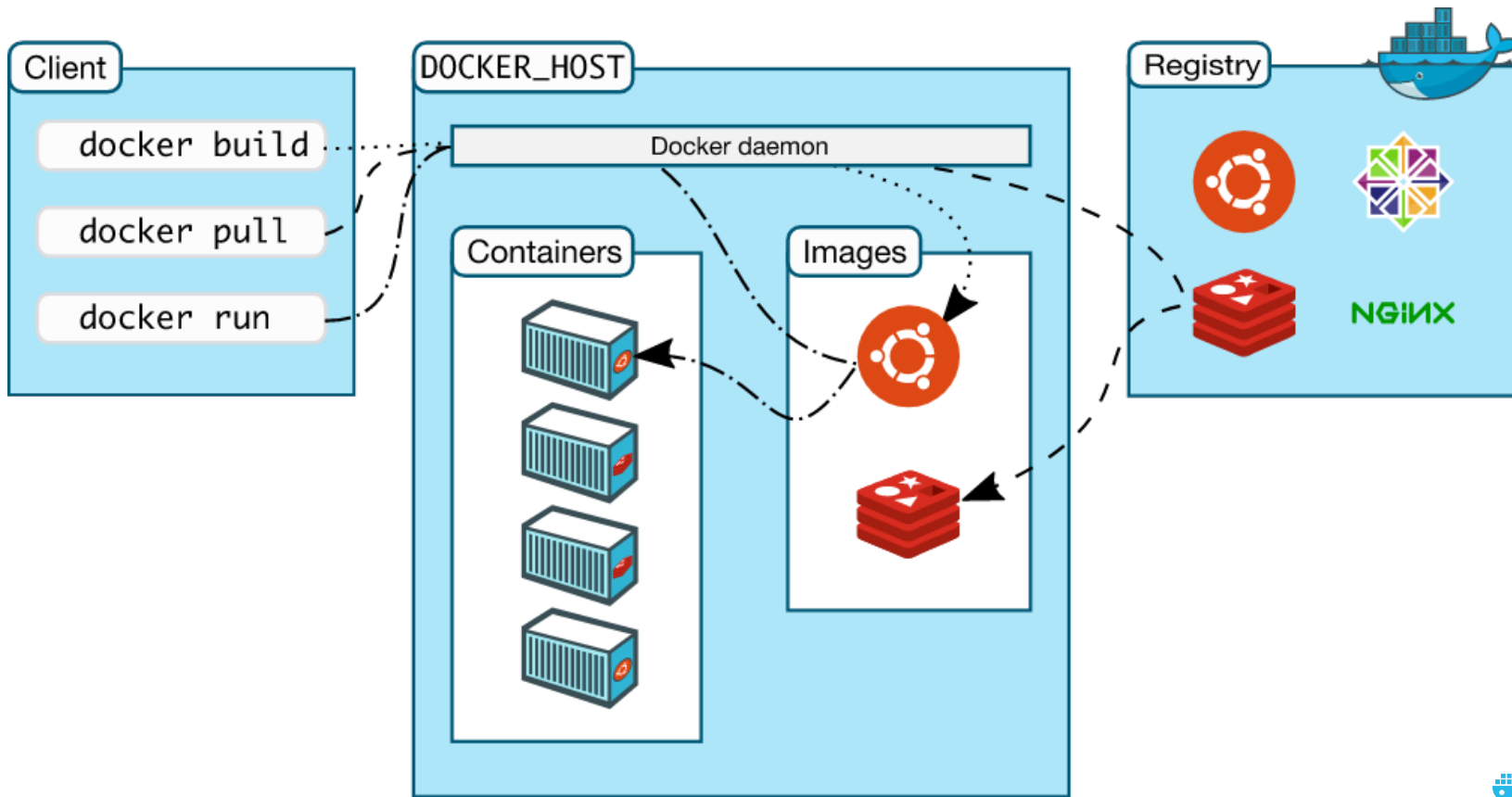
Docker's portability and lightweight nature also make it easy to dynamically manage workloads, scaling up or tearing down applications and services as business needs dictate, in near real time.

Running more workloads on the same hardware

Docker is lightweight and fast. It provides a viable, cost-effective alternative to hypervisor-based virtual machines, so you can use more of your compute capacity to achieve your business goals



Docker architecture



The Docker daemon

The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

The Docker client

The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as `docker run`, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

Docker Desktop

Docker Desktop is an easy-to-install application for your Mac or Windows environment that enables you to build and share containerized applications and microservices. Docker Desktop includes the Docker daemon (dockerd), the Docker client (docker), Docker Compose, Docker Content Trust, Kubernetes, and Credential Helper.

Docker Registry

A Docker registry stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.

When you use the `docker pull` or `docker run` commands, the required images are pulled from your configured registry. When you use the `docker push` command, your image is pushed to your configured registry.

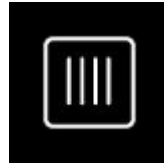


Some Docker vocabulary



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

Docker objects

When you use Docker, you are creating and using images, containers, networks, volumes, plugins, and other objects. This section is a brief overview of some of those objects.

Images

An image is a read-only template with instructions for creating a Docker container. Often, an image is based on another image, with some additional customization. For example, you may build an image which is based on the ubuntu image, but installs the Apache web server and your application, as well as the configuration details needed to make your application run.

Containers

A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.



The Role of Images and Containers



Docker
Image

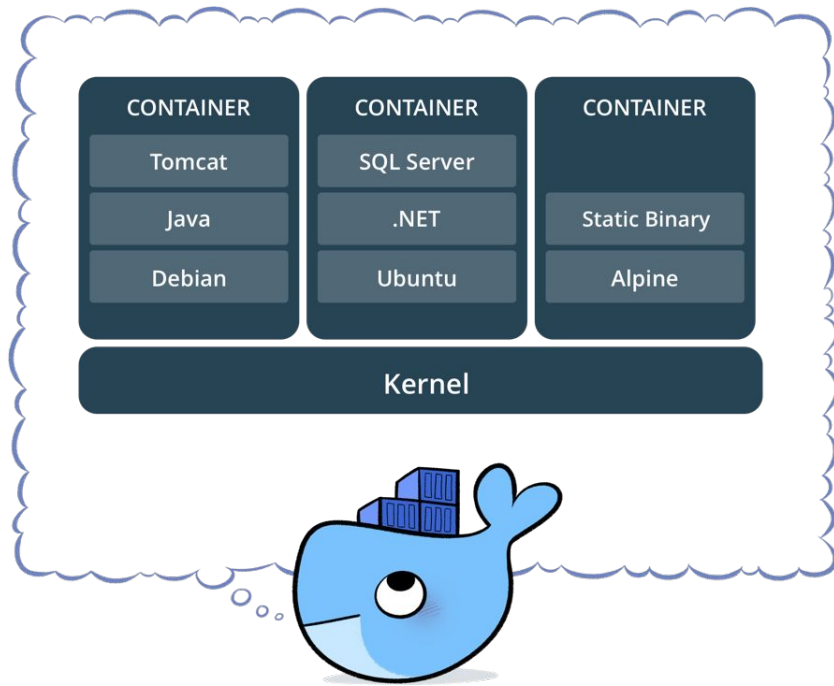
Example: Ubuntu with
Node.js and
Application Code



Docker
Container

Created by using an image.
Runs your
application.

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

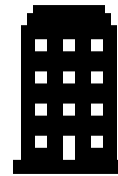
Containers native to Windows Server 2016

Docker containers are NOT VMs

Easy connection to make
Fundamentally different architectures
Fundamentally different benefits



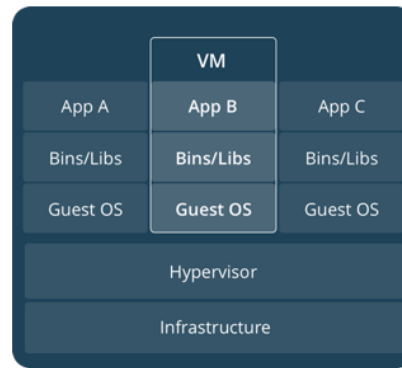
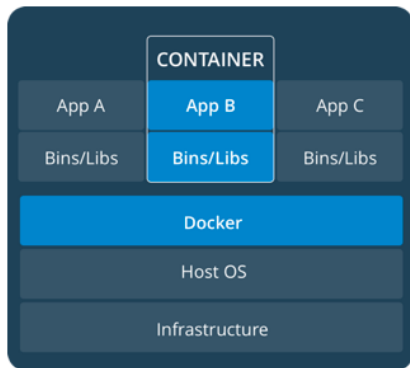
VM



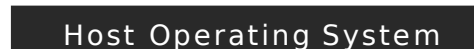
VM with
Container

Container And VM

Container Vs VM



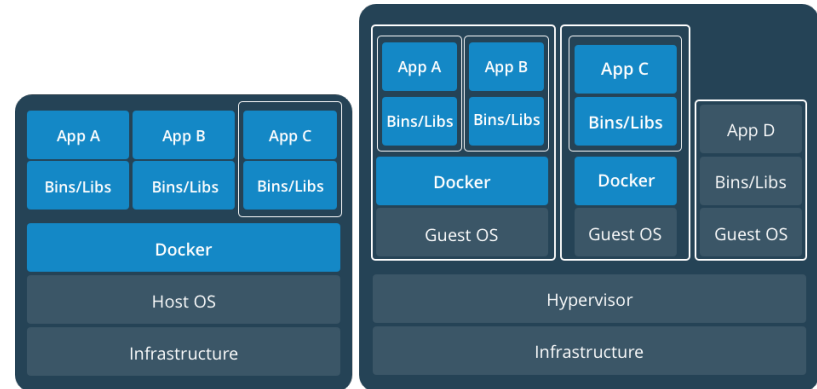
Docker Containers Versus Virtual Machines



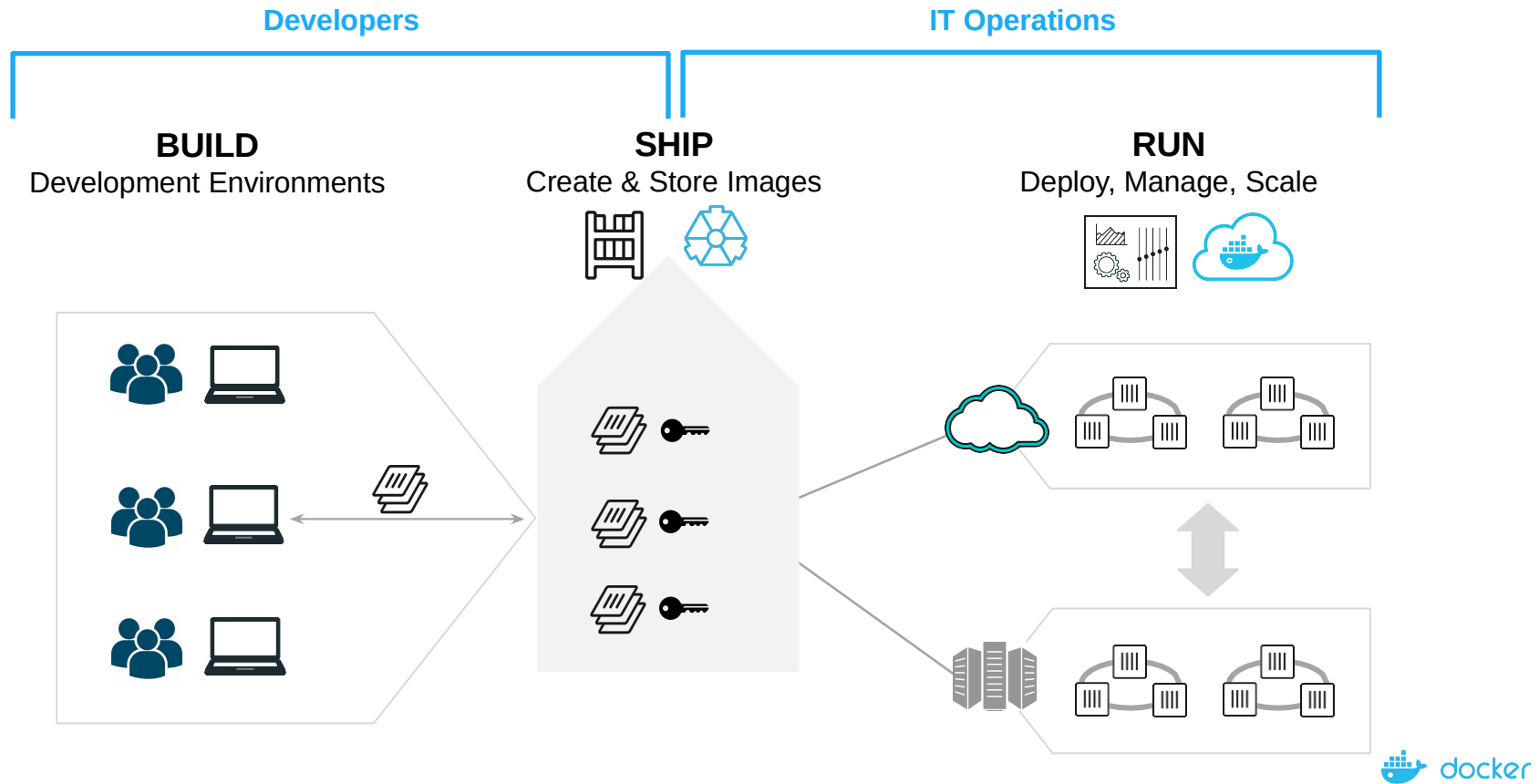
Virtual
Machines



Docker
Containers



Using Docker: Build, Ship, Run Workflow



The underlying technology

Docker is written in the [Go programming language](#) and takes advantage of several features of the Linux kernel to deliver its functionality.

Docker uses a technology called namespaces to provide the isolated workspace called the container.

When you run a container, Docker creates a set of namespaces for that container.

These namespaces provide a layer of isolation.

Each aspect of a container runs in a separate namespace and its access is limited to that namespace.



Installing Docker

- Installing docker on Windows:
- <https://docs.docker.com/docker-for-windows/install/>

Installing docker on ubuntu

- <https://tecadmin.net/install-docker-on-ubuntu/>
- <https://docs.docker.com/install/linux/docker-ce/ubuntu/>

Installing docker on mac

- <https://docs.docker.com/docker-for-mac/install/>

Container

- <https://www.docker.com/resources/what-container>

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

Container

Docker is a shipping container system for code



```
root@logan: ~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
28cdd9cc7f28        [REDACTED]         "catalina.sh run"   10 days ago         Up 10 days (unhealthy)  0.0.0.0:8086->8080/tcp
```

```
docker exec -it -u root 28cdd9cc7f28 bash
```

```
bash-4.4# ls
LICENSE  NOTICE  RELEASE-NOTES  RUNNING.txt  bin      conf      include  lib      logs      native-jni-lib  temp      webapps  work
```

A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings

```
C:\gitrama\spring-boot-splunk>docker ps
```

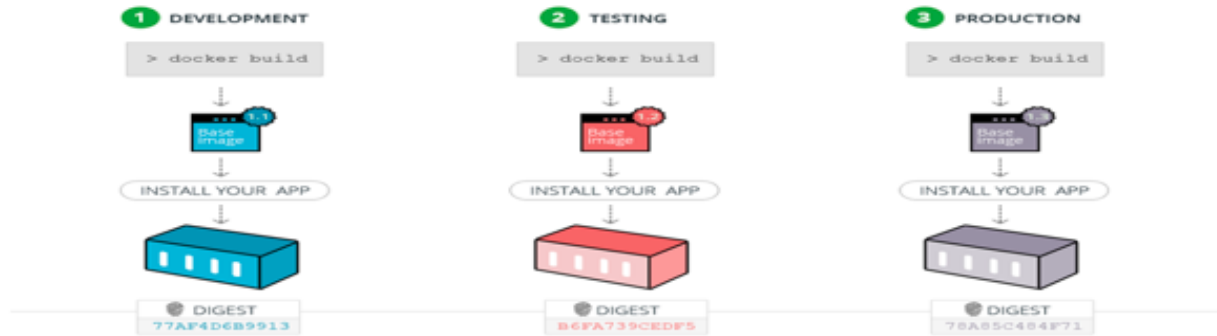
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
4fa1c8f6124a	splunk/universalforwarder:6.5.3-monitor spring-boot-splunk_splunkforwarder_1	"/sbin/entrypoint.sh..."	7 weeks ago	Up 3 minutes	1514/tcp, 8088-8089/tcp
42d49705bf62	splunk/splunk spring-boot-splunk_splunk_1	"/sbin/entrypoint.sh..."	7 weeks ago	Up 28 seconds (health: starting)	4001/tcp, 8065/tcp, 8088-8089/tcp, 8191/tcp, 9887/tcp, 9997/tcp, 0.0.0.0:8000->8000/tcp
e7b18e96cb7d	falcon007/spring-boot-splunk:0.0.1-SNAPSHOT spring-boot-splunk_demo-application_1	"java -Djava.securit..."	7 weeks ago	Up 28 seconds	0.0.0.0:8080->8080/tcp

```
C:\gitrama\spring-boot-splunk>
```

Docker Image

- A **Docker image** is a file, comprised of multiple layers, used to execute code in a **Docker container**. An **image** is essentially built from the instructions for a complete and executable version of an application, which relies on the host OS kernel.

Image



```
C:\LearningWorkspace\datacache>docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
tanu319/datacache   latest             dcfc64cf92b8       39 hours ago       119MB
openjdk              8-jdk-alpine       54ae553cb104       2 weeks ago        103MB
mysql                latest             6a834f03bd02       3 weeks ago        484MB
vromero/activemq-artemis 1.5.5             35abaeaa2a5a       2 months ago       680MB
tomcat               8.5-jre8-alpine    cb132aeae2f7       2 months ago       107MB
mongo                3.4.7             b39de1d79a53       13 months ago      359MB
oscarfonts/h2        1.4.196           27ca89bbe21f       15 months ago      605MB
d4w/nsenter          latest            9e4f13a0901e       2 years ago         83.8kB
```

- An **image** is an executable package that includes everything needed to run an application- the code, a runtime, libraries, environment variables, and configuration files.

Docker images

```
C:\gitrama\spring-boot-splunk>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
falcon007/spring-boot-splunk	0.0.1-SNAPSHOT	9330f7c33190	7 weeks ago	127MB
openjdk	8-jdk-alpine	ece449e0acb8	7 weeks ago	105MB
splunk/universalforwarder	latest	67d362ab67e6	2 months ago	219MB
falcon007/spring-boot-splunk	latest	4d01685ea5d9	3 months ago	127MB
landoop/fast-data-dev	latest	cea723c43ac0	4 months ago	1.05GB
splunk/splunk	latest	de17de3d9fbc	4 months ago	539MB
barrycommins/spring-boot-sleuth-splunk-demo	latest	2305e3763a11	4 months ago	127MB
lh-ea-handler	latest	7ca9545e907b	4 months ago	190MB
redis	latest	0f55cf3661e9	4 months ago	95MB
openjdk	8-alpine	792ff45a2a17	4 months ago	105MB
consul	latest	c5811022a71c	4 months ago	107MB
splunk/splunk	<none>	bb2d3b5e7b01	6 months ago	535MB
anapsix/alpine-java	8_server-jre_unlimited	4ca48d28c780	6 months ago	127MB
docker4w/nsenter-dockerd	latest	2f1c802f322f	8 months ago	187kB
wnameless/oracle-xe-11g	latest	698cc7361de4	13 months ago	2.13GB
cassandra	3.10	3cf8fb744275	2 years ago	386MB
splunk/universalforwarder	6.5.3-monitor	a659c5928029	2 years ago	241MB
webcenter/activemq	5.14.3	ab2a33f6de2b	2 years ago	422MB

```
C:\gitrama\spring-boot-splunk>
```

Docker Commands Lab



Basic Docker Commands

1) Pull docker images

```
$ docker image pull hello-world:latest
```

2) List docker image

```
$ docker image ls
```

3) Start container or Run Container

```
$ docker run --name hello hello-world:latest
```

4) List all running Container

```
$ docker ps
```

5) Stop container

```
$ docker stop hello(or <container id>)
```



Docker Hub Signup

- <https://hub.docker.com/>



Docker Identification

In order to get you started, let us get you a Docker ID.
Already have an account? [Sign In](#)



Docker ID is required.



- ☐ I agree to Docker's [Terms of Service](#).
- ☐ I agree to Docker's [Privacy Policy](#) and [Data Processing Terms](#).
- ☐ (Optional) I would like to receive email updates from Docker, including its various services and products.



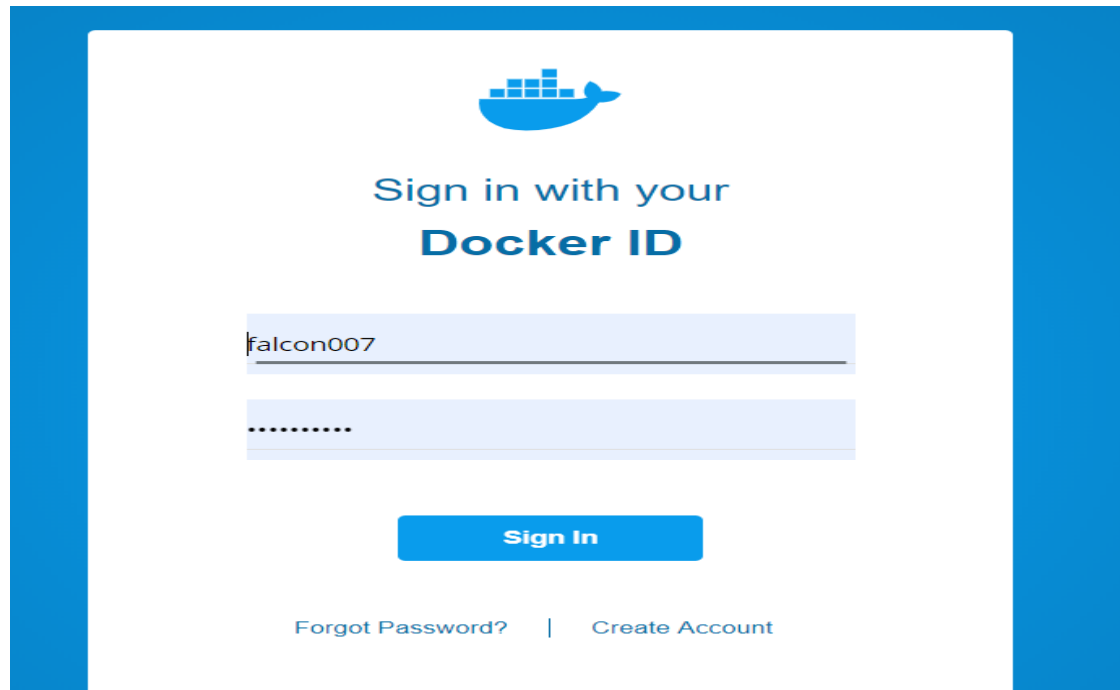
I'm not a robot




[Privacy](#) - [Terms](#)

Continue

Docker Hub Sign In



The image shows a Docker Hub sign-in form. At the top is the Docker logo, a blue whale with a stack of containers on its back. Below the logo, the text "Sign in with your" is in a light blue font, and "Docker ID" is in a bold blue font. There are two input fields: the first contains the text "falcon007" and the second contains a series of dots, indicating a password. Below the input fields is a blue button with the text "Sign In" in white. At the bottom, there are two links: "Forgot Password?" and "Create Account", separated by a vertical line.



Sign in with your
Docker ID

falcon007

.....

Sign In

[Forgot Password?](#) | [Create Account](#)

Login Screen

Welcome to Docker Hub

Download and Take a Tutorial

Get started by downloading Docker Desktop, and learn how you can build, tag and share a sample image on Hub.

[Get started with Docker Desktop](#)

6) Remove docker container

```
$ docker rm hello (or <container id>)
```

7) Remove docker image

```
$ docker rmi hello-world(or <image id>)
```

8) Build docker image

```
$ docker build -t hello-world:2.0 .
```

9) Push docker image to repository

```
$ docker image push hello:2.0
```

10) List docker images

```
$ docker images
```

11) docker -help





docker