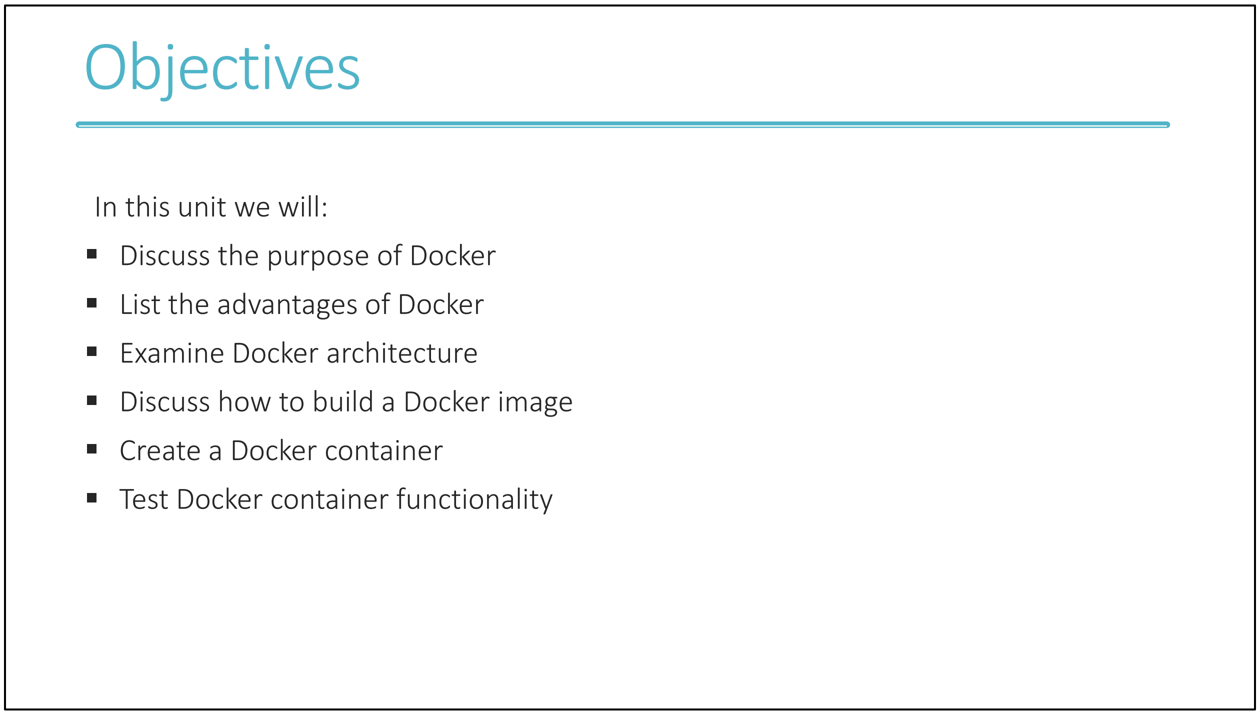
Docker Essentials

A simplified look at Docker

Steve Acquistapace, Technical Curriculum Developer

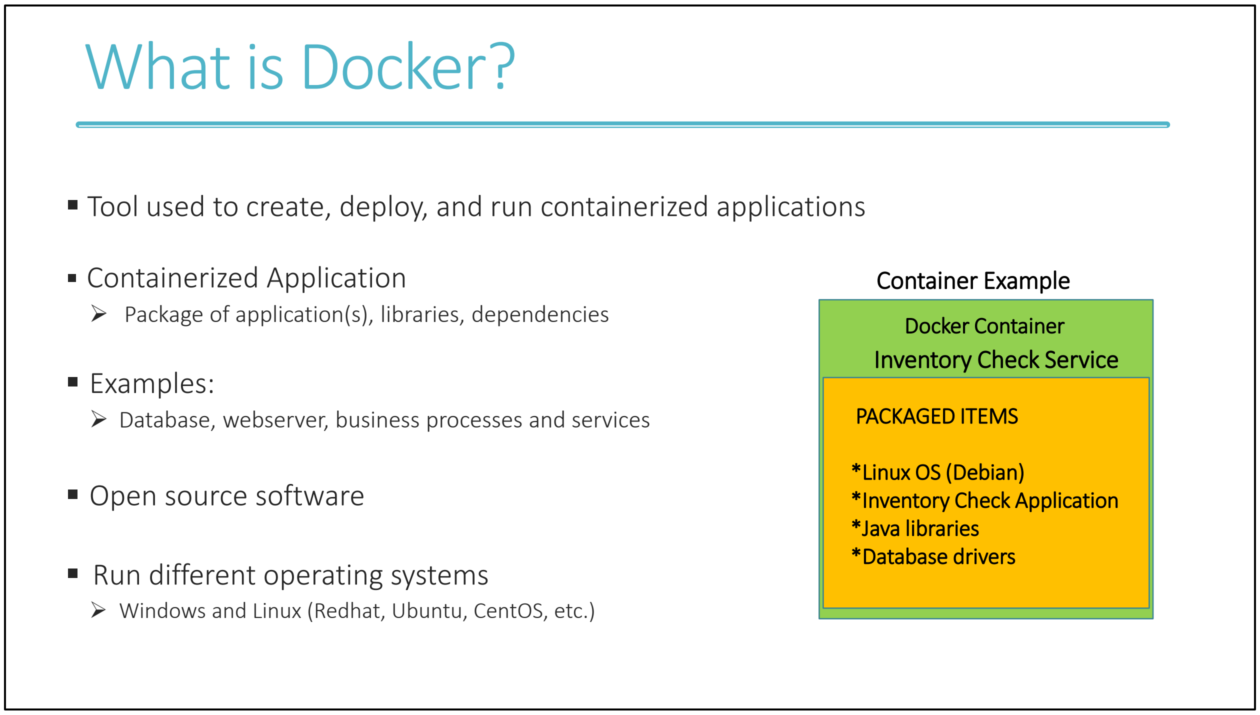


Objectives

In this unit we will:

- Discuss the purpose of Docker
- List the advantages of Docker
- Examine Docker architecture
- Discuss how to build a Docker image
- Create a Docker container
- Test Docker container functionality

The objectives listed above will be discussed followed by a Docker hands-on lab.

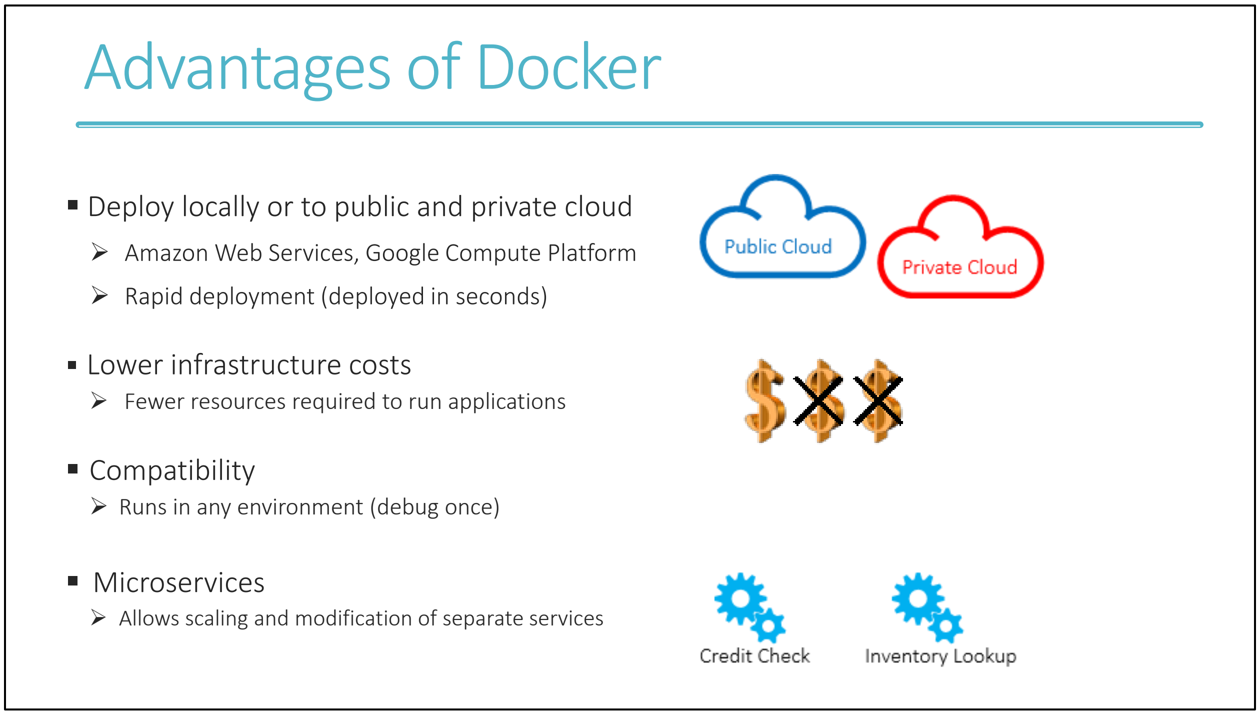


What is Docker?

- Tool used to create, deploy, and run containerized applications
- Containerized Application
  - ➢ Package of application(s), libraries, dependencies
- Examples:
  - ➢ Database, webserver, business processes and services
- Open source software
- Run different operating systems
  - ➢ Windows and Linux (Redhat, Ubuntu, CentOS, etc.)

Container Example
Docker Container
Inventory Check Service
PACKAGED ITEMS
*Linux OS (Debian)
*Inventory Check Application
*Java libraries
*Database drivers

Docker is a tool used to create, deploy, and run containerized applications. A containerized application is a package of one or more applications along with the libraries and dependencies required for functionality. Examples of containerized applications include a database, webserver and business processes and services that perform operations such as an item inventory lookup or a credit check service. Docker is open source software allowing for contributions from a global community of developers who can extend capabilities by adding additional features that are not currently available. Finally, a docker container can run different operating systems such as Windows and Linux (Redhat, Ubuntu, CentOS, Alpine, Debian, etc.).

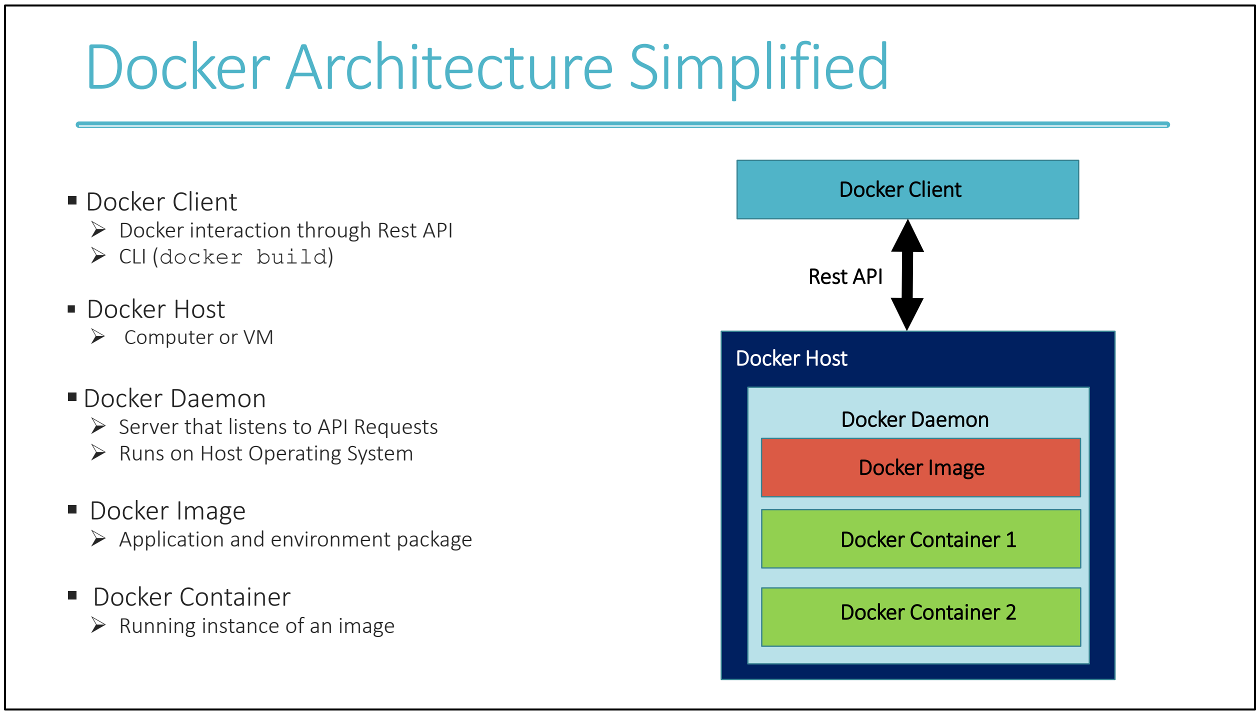


Advantages of Docker

- Deploy locally or to public and private cloud
  - Amazon Web Services, Google Compute Platform
  - Rapid deployment (deployed in seconds)
- Lower infrastructure costs
  - Fewer resources required to run applications
- Compatibility
  - Runs in any environment (debug once)
- Microservices
  - Allows scaling and modification of separate services

Public Cloud
Private Cloud
Credit Check
Inventory Lookup

Docker can be deployed locally or to a public cloud such as Amazon Web Services, or a private cloud, proprietary environment, typically running on-premises, and dedicated to a single organization. Docker containers are rapidly deployed in a matter of seconds because there is no need to boot an entire operating system during the deployment process. Lower infrastructure costs are a key advantage because Docker requires fewer resources to run applications. With Docker container deployment you can run more apps with fewer physical servers. A docker container can run in any environment, public or private cloud, and on different operating systems. You only need to test and debug once before deployment. Docker also takes advantage of microservices that can be easily scaled to meet the needs of your organization and separate services can be easily modified without effecting other microservices in your solution.



Docker Architecture Simplified

- Docker Client
  - Docker interaction through Rest API
  - CLI (`docker build`)
- Docker Host
  - Computer or VM
- Docker Daemon
  - Server that listens to API Requests
  - Runs on Host Operating System
- Docker Image
  - Application and environment package
- Docker Container
  - Running instance of an image

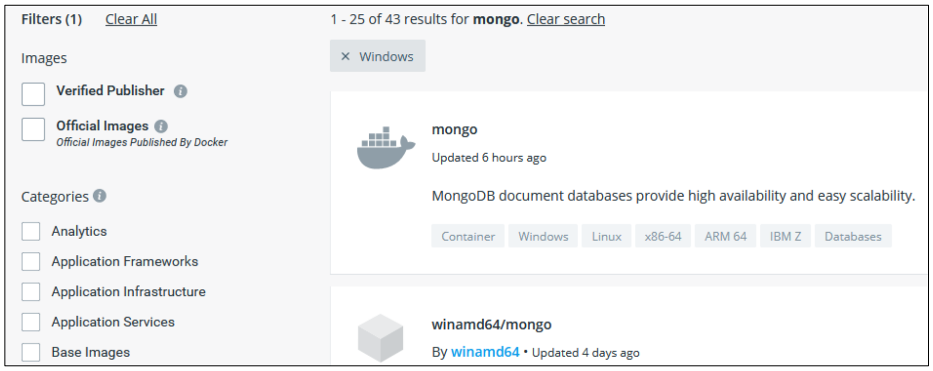
Docker Client
Rest API
Docker Host
Docker Daemon
Docker Image
Docker Container 1
Docker Container 2

Docker architecture key components include the following:

- Docker Client: Command line interface that allows Docker interaction through the REST API.
- Docker Host: Computer or virtual machine that runs the Docker daemon.
- Docker Daemon: Server that listens to API requests and performs container operations.
- Docker Image: Application and environment package that is used to build a container.
- Docker Container: A running instance of an image. A deployed running container runs your application.

Docker Hub: hub.docker.com

- Docker Service
  - Find and share container images
  - Private repositories of container images (Teams, Organizations)
  - Search for images on different categories: Analytics, Databases, Linux, Windows



Filters (1) Clear All

1 - 25 of 43 results for **mongo**. Clear search

× Windows

Images

- **Verified Publisher**
- **Official Images** *Official Images Published By Docker*

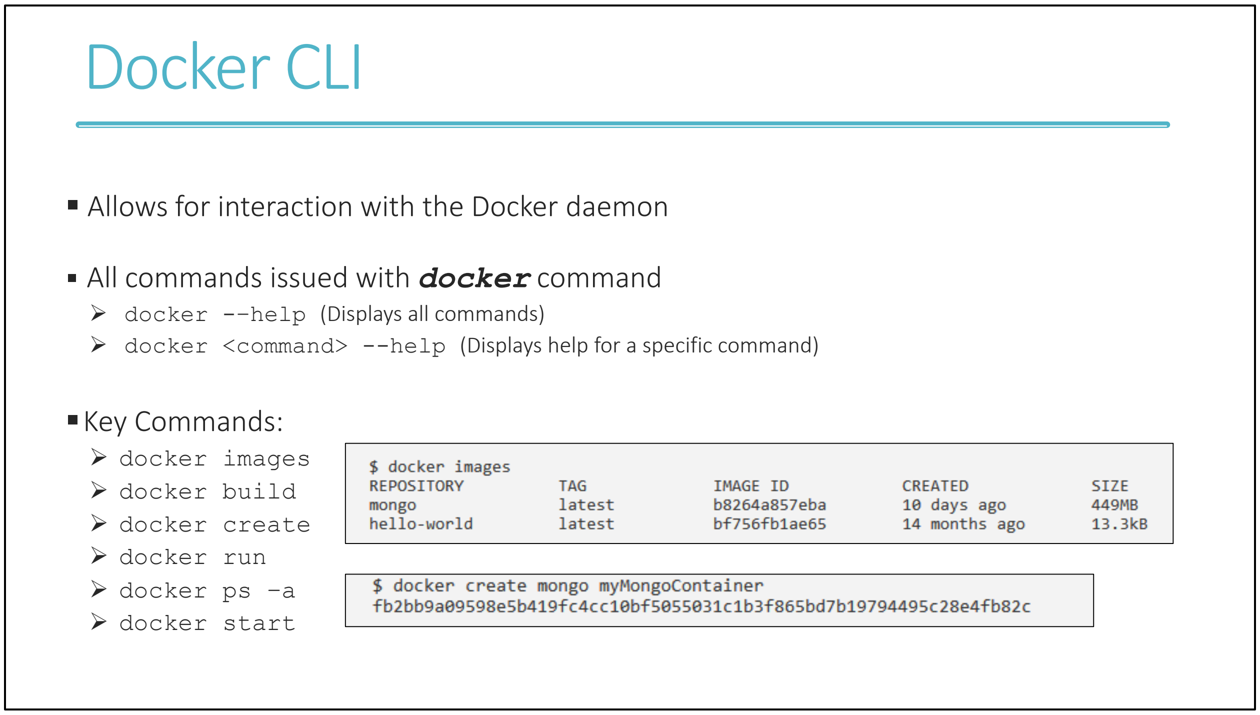
Categories

- Analytics
- Application Frameworks
- Application Infrastructure
- Application Services
- Base Images

mongo
Updated 6 hours ago
MongoDB document databases provide high availability and easy scalability.
Container Windows Linux x86-64 ARM 64 IBM Z Databases

winamd64/mongo
By winamd64 • Updated 4 days ago

Docker Hub is a docker service that allows for the sharing of docker container images. It also contains private repositories of container images that are shared by teams and organizations. Docker Hub allows users to search for images on different categories such as analytics, databases, Linux and Windows.



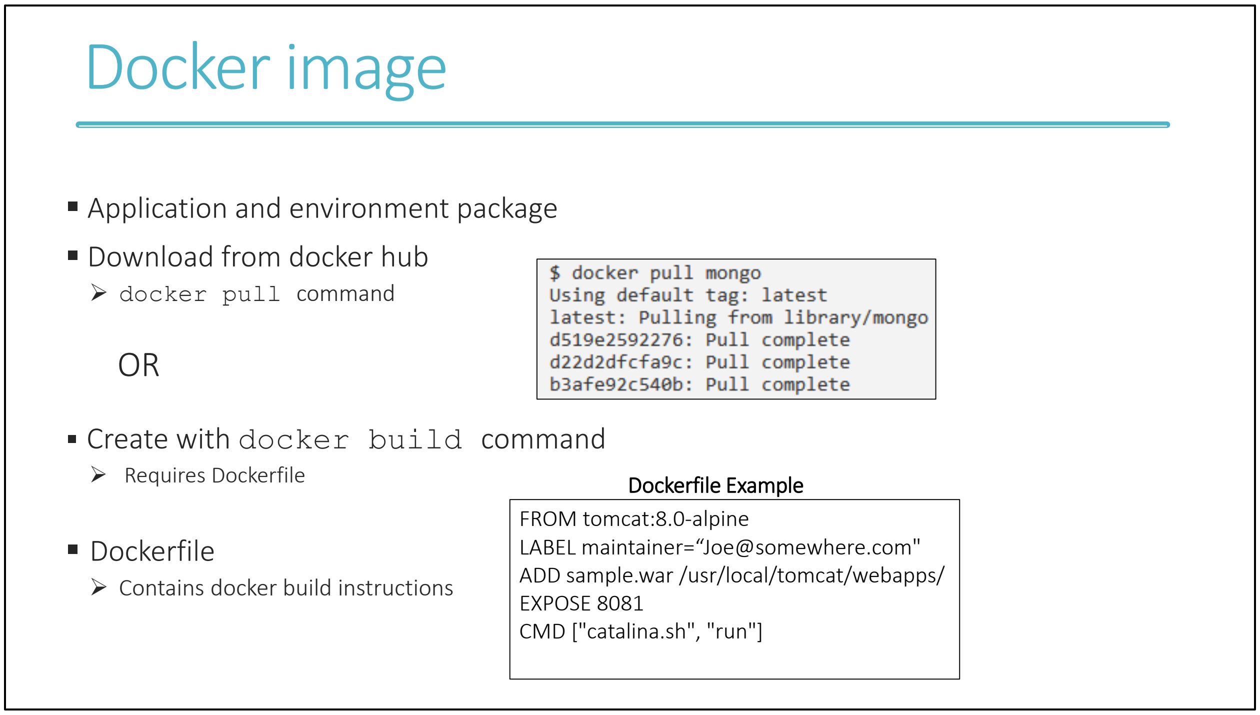
Docker CLI

- Allows for interaction with the Docker daemon
- All commands issued with ***docker*** command
  - `docker --help` (Displays all commands)
  - `docker <command> --help` (Displays help for a specific command)
- Key Commands:
  - `docker images`
  - `docker build`
  - `docker create`
  - `docker run`
  - `docker ps -a`
  - `docker start`

```
$ docker images
REPOSITORY          TAG             IMAGE ID          CREATED            SIZE
mongo               latest          b8264a857eba      10 days ago        449MB
hello-world         latest          bf756fb1ae65      14 months ago      13.3kB
```

```
$ docker create mongo myMongoContainer
fb2bb9a09598e5b419fc4cc10bf5055031c1b3f865bd7b19794495c28e4fb82c
```

The Docker command line interface allows for interaction with the Docker daemon. Command details are available using the **docker --help** command. The CLI allows you to perform key operations such as build or download docker images and create and run containers. Some key commands include **docker images** which allows you to view all images in your repository and **docker create** which allows you to create a docker container.



Docker image

- Application and environment package
- Download from docker hub
  - `docker pull` command

OR

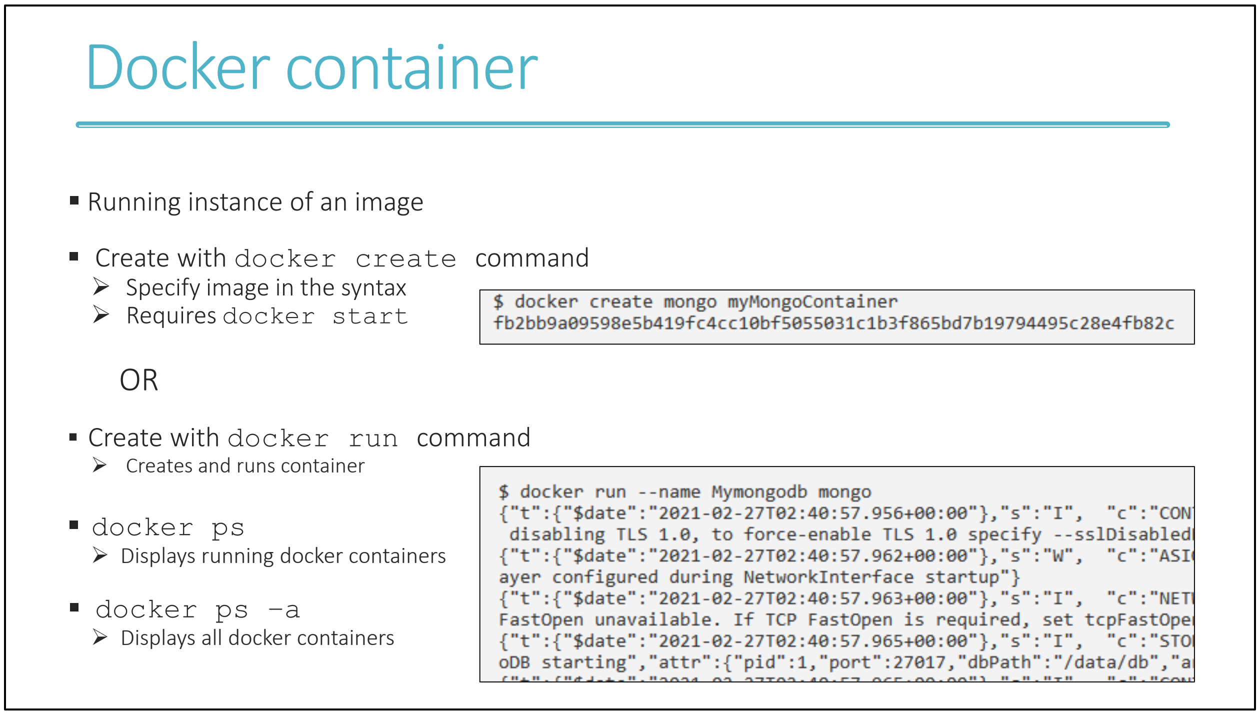
- Create with `docker build` command
  - Requires Dockerfile
- Dockerfile
  - Contains docker build instructions

```
$ docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
d519e2592276: Pull complete
d22d2dfcfa9c: Pull complete
b3afe92c540b: Pull complete
```

Dockerfile Example

```
FROM tomcat:8.0-alpine
LABEL maintainer=“Joe@somewhere.com"
ADD sample.war /usr/local/tomcat/webapps/
EXPOSE 8081
CMD ["catalina.sh", "run"]
```

The Docker image contains the application and environment package including libraries and other dependencies that are necessary for running the application. Docker Hub allows for the download of shared images using the **docker pull** command. Alternatively, you can create a docker image using the **docker build** command. The **docker build** command requires the use of a dockerfile. The dockerfile is a text document that contains build instructions such as the base operating system that will be contained in the image. Build process instructions may also include commands to add applications, libraries, and other dependencies.



Docker container

- Running instance of an image
- Create with `docker create` command
  - ➢ Specify image in the syntax
  - ➢ Requires `docker start`

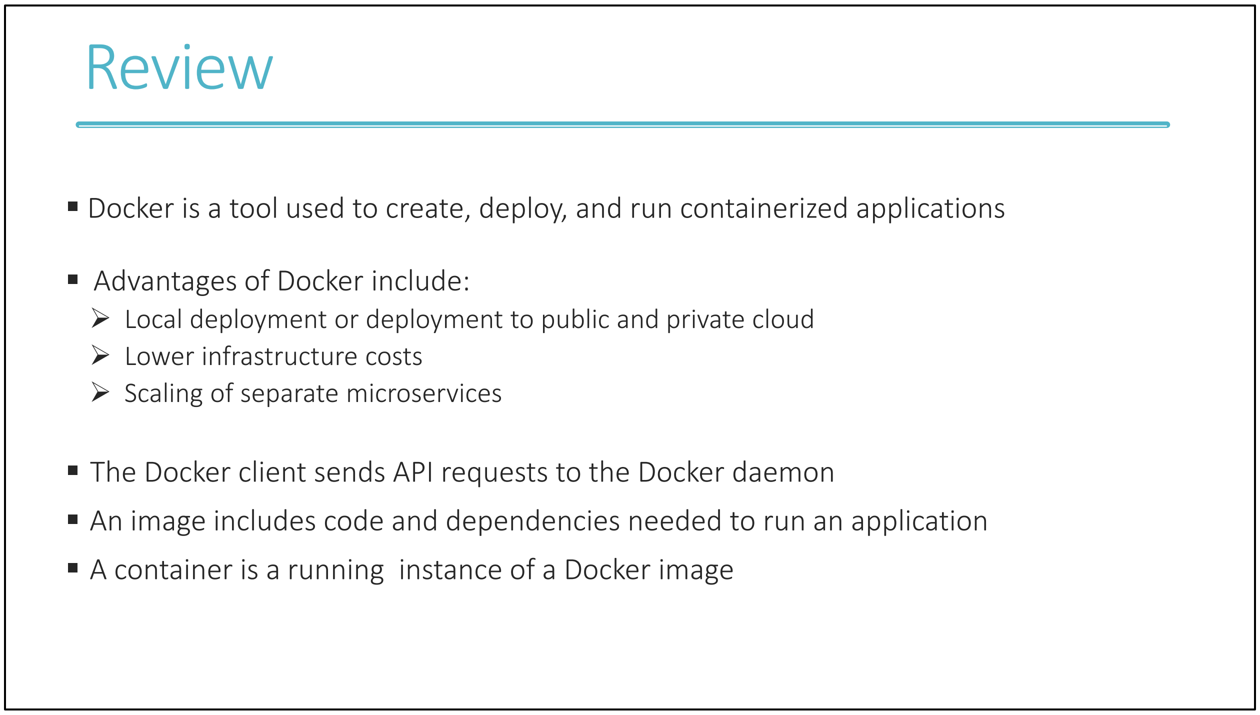
```
$ docker create mongo myMongoContainer
fb2bb9a09598e5b419fc4cc10bf5055031c1b3f865bd7b19794495c28e4fb82c
```

OR

- Create with `docker run` command
  - ➢ Creates and runs container
- `docker ps`
  - ➢ Displays running docker containers
- `docker ps -a`
  - ➢ Displays all docker containers

```
$ docker run --name Mymongodb mongo
{"t":{"$date":"2021-02-27T02:40:57.956+00:00"},"s":"I",  "c":"CON
 disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabled
{"t":{"$date":"2021-02-27T02:40:57.962+00:00"},"s":"W",  "c":"ASI
ayer configured during NetworkInterface startup"}
{"t":{"$date":"2021-02-27T02:40:57.963+00:00"},"s":"I",  "c":"NET
FastOpen unavailable. If TCP FastOpen is required, set tcpFastOpe
{"t":{"$date":"2021-02-27T02:40:57.965+00:00"},"s":"I",  "c":"STO
oDB starting","attr":{"pid":1,"port":27017,"dbPath":"/data/db","a
```

The Docker container is a running instance of an image. The container is created with the CLI by issuing the **docker create** command; however, you can also create and run a container with a single command using **docker run** and the appropriate syntax. Running containers can be viewed with the **docker ps** command, Additionally, all containers running or stopped can be viewed using the **docker ps -a** command.



Review

- Docker is a tool used to create, deploy, and run containerized applications
- Advantages of Docker include:
  - ➢ Local deployment or deployment to public and private cloud
  - ➢ Lower infrastructure costs
  - ➢ Scaling of separate microservices
- The Docker client sends API requests to the Docker daemon
- An image includes code and dependencies needed to run an application
- A container is a running instance of a Docker image

Review Question 1

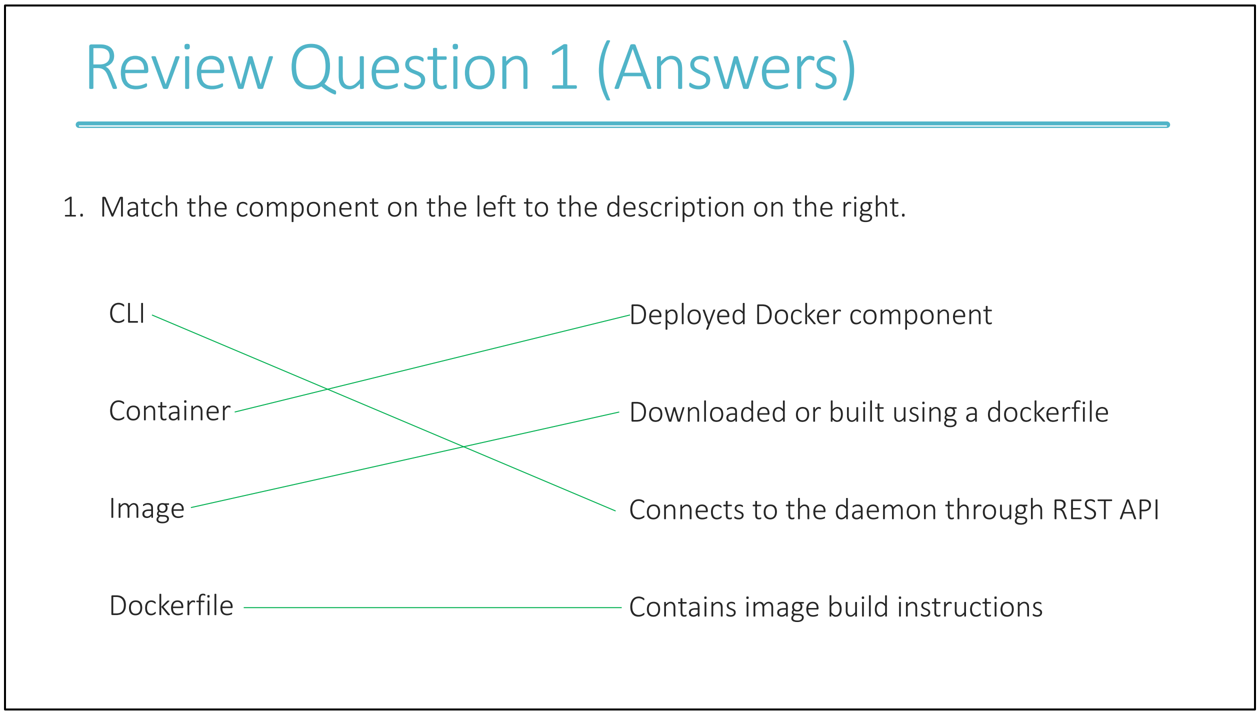
1. Match the component on the left to the description on the right.

CLI	Deployed Docker component
Container	Downloaded or built using a dockerfile
Image	Connects to the daemon through REST API
Dockerfile	Contains image build instructions

Review Question 2

2. Select all items that are part of Docker architecture.

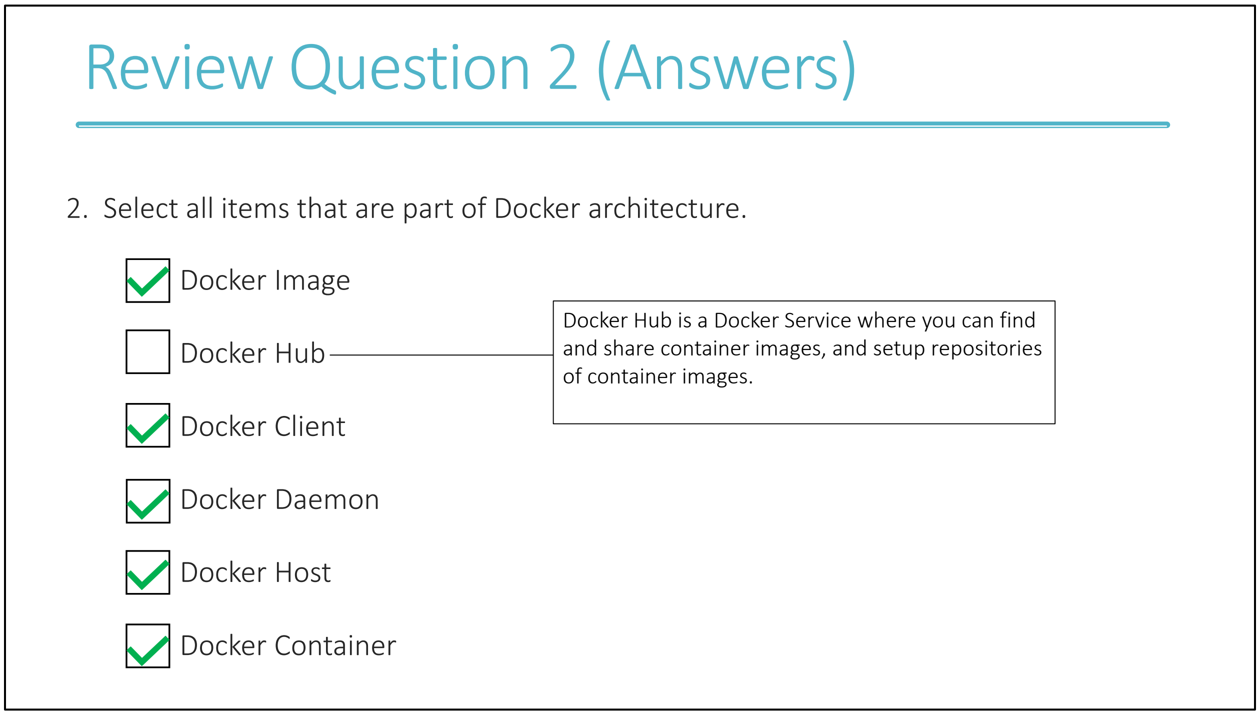
- ☐ Docker Image
- ☐ Docker Hub
- ☐ Docker Client
- ☐ Docker Daemon
- ☐ Docker Host
- ☐ Docker Container



Review Question 1 (Answers)

1. Match the component on the left to the description on the right.

CLI	Deployed Docker component
Container	Downloaded or built using a dockerfile
Image	Connects to the daemon through REST API
Dockerfile	Contains image build instructions



Review Question 2 (Answers)

2. Select all items that are part of Docker architecture.

- [✓] Docker Image
- [ ] Docker Hub
- [✓] Docker Client
- [✓] Docker Daemon
- [✓] Docker Host
- [✓] Docker Container

Docker Hub is a Docker Service where you can find and share container images, and setup repositories of container images.

Working with Docker

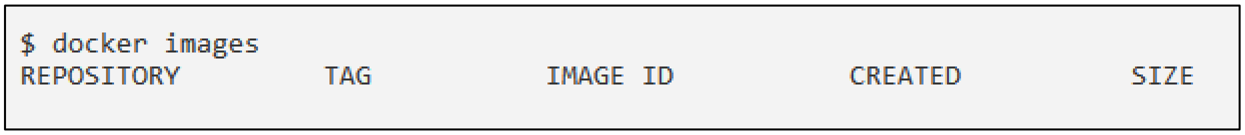
In this lab exercise you will download a MongoDB Docker image, create and run a MongoDB Docker container, and test container functionality.

Prerequisites:

- Docker Toolbox-19.03.1 installed
- Internet Connection

Lab Steps:

1. Launch a Terminal and enter the command to view docker images.
   - Select *Start* > ***Docker Quickstart Terminal***.
   - Enter the command **`docker images`**

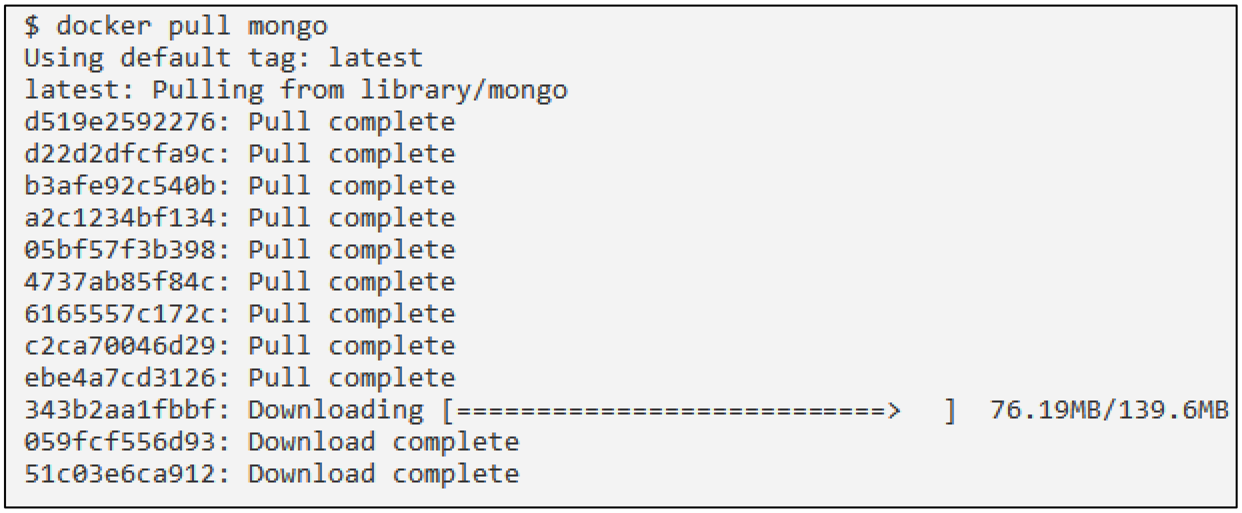


```
$ docker images
REPOSITORY          TAG            IMAGE ID            CREATED            SIZE
```

   - Observe that there are no images in your Docker environment.

2. Download the MongoDB image from Docker Hub.
   - Enter the following command:

     `docker pull mongo`

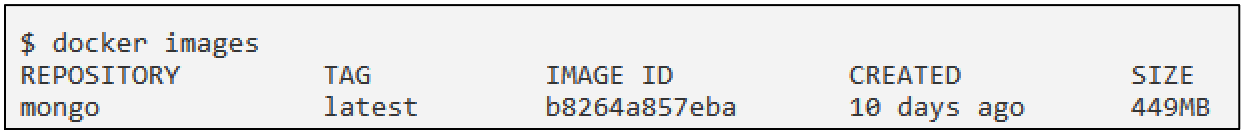


```
$ docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
d519e2592276: Pull complete
d22d2dfcfa9c: Pull complete
b3afe92c540b: Pull complete
a2c1234bf134: Pull complete
05bf57f3b398: Pull complete
4737ab85f84c: Pull complete
6165557c172c: Pull complete
c2ca70046d29: Pull complete
ebe4a7cd3126: Pull complete
343b2aa1fbbf: Downloading [===========================>   ]  76.19MB/139.6MB
059fcf556d93: Download complete
51c03e6ca912: Download complete
```

   - Observe the image download process. This may take a few minutes. Continue to the next step when the download is complete.

3. Verify that the image now resides in your Docker environment.
   - Enter the following command:

     `docker images`

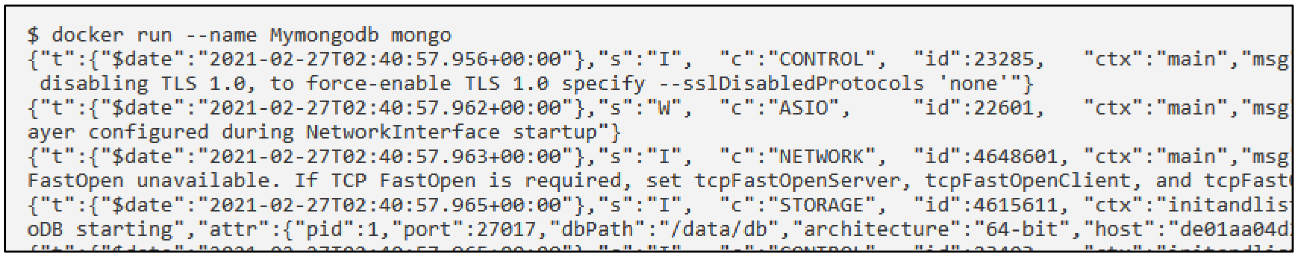


```
$ docker images
REPOSITORY          TAG            IMAGE ID          CREATED          SIZE
mongo               latest         b8264a857eba      10 days ago      449MB
```

   - Observe the image repository name of mongo.

4. Create and run a Mongodb container instance.
   - Enter the following command:

     `docker run --name Mymongodb mongo`



```
$ docker run --name Mymongodb mongo
{"t":{"$date":"2021-02-27T02:40:57.956+00:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg
 disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2021-02-27T02:40:57.962+00:00"},"s":"W",  "c":"ASIO",     "id":22601,   "ctx":"main","msg
ayer configured during NetworkInterface startup"}
{"t":{"$date":"2021-02-27T02:40:57.963+00:00"},"s":"I",  "c":"NETWORK",  "id":4648601, "ctx":"main","msg
FastOpen unavailable. If TCP FastOpen is required, set tcpFastOpenServer, tcpFastOpenClient, and tcpFast
{"t":{"$date":"2021-02-27T02:40:57.965+00:00"},"s":"I",  "c":"STORAGE",  "id":4615611, "ctx":"initandlis
oDB starting","attr":{"pid":1,"port":27017,"dbPath":"/data/db","architecture":"64-bit","host":"de01aa04d
```

Note: The syntax above creates and runs a container instance with the name of Mymongodb using the mongo Docker image.

Alternatively, you can create, verify, and start a container with the commands shown below:

The following command creates the container instance.

```
docker create --name Mymongodb mongo
```

The following command allows you to view all container instances in your environment, running or stopped.

```
docker ps -a
```

The following command starts your container instance in interactive mode so you can view operations in the Terminal.

```
docker start -i Mymongodb
```

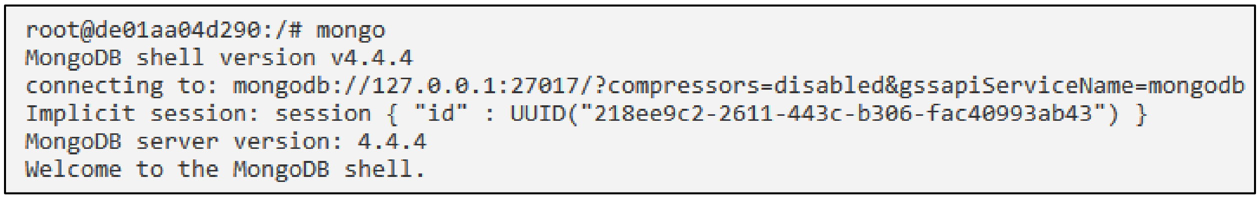
Verify MongoDB container functionality

5. Connect to the Mongodb container instance.
   - Launch a new Terminal.
     - *Start* > ***Docker Quickstart Terminal***
   - Enter the following command:

     `docker exec -it Mymongodb bash`

   - Enter `mongo` at the prompt.

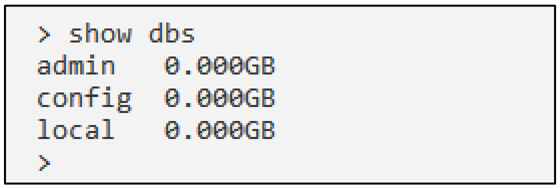
     `root@###########:/#`**`mongo`**



```
root@de01aa04d290:/# mongo
MongoDB shell version v4.4.4
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("218ee9c2-2611-443c-b306-fac40993ab43") }
MongoDB server version: 4.4.4
Welcome to the MongoDB shell.
```

   - Verify that you are connected to the instance.

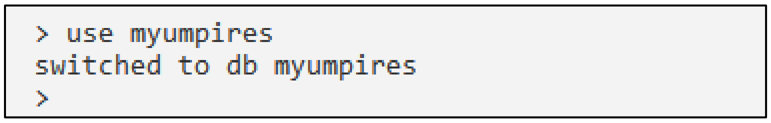
     `show dbs`



```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
>
```

6. Create a database with the name of myumpires.
   - Enter the following command:

     `use myumpires`

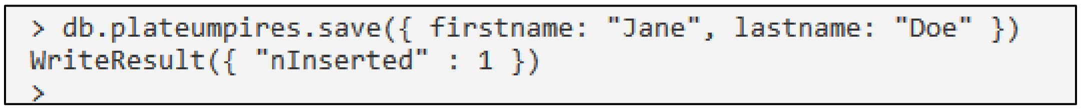


```
> use myumpires
switched to db myumpires
>
```

7. Insert data into the database myumpires.
   - Enter the following command:

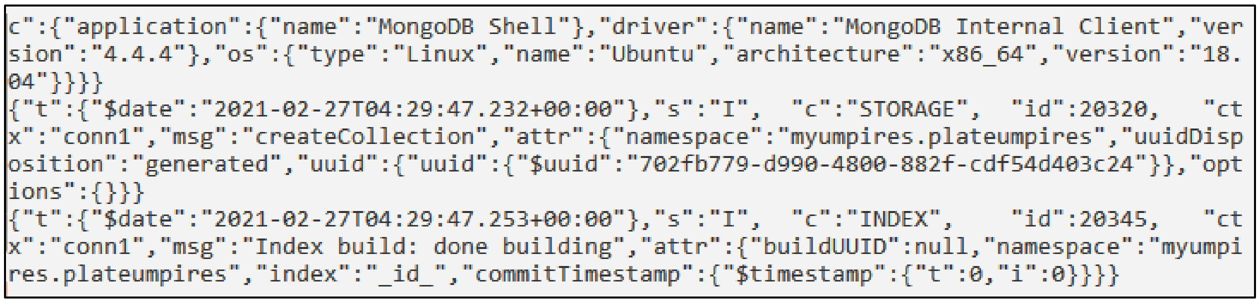
     `db.plateumpires.save({ firstname: "Jane", lastname: "Doe" })`

   - Observe the WriteResult message in the Terminal.



```
> db.plateumpires.save({ firstname: "Jane", lastname: "Doe" })
WriteResult({ "nInserted" : 1 })
>
```

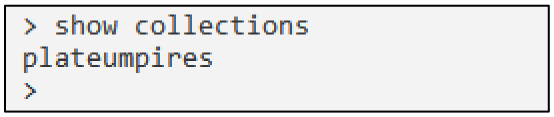
   - Observe the MongoDB collection messages in the container Terminal.



```
c":{"application":{"name":"MongoDB Shell"},"driver":{"name":"MongoDB Internal Client","ver
sion":"4.4.4"},"os":{"type":"Linux","name":"Ubuntu","architecture":"x86_64","version":"18.
04"}}}}
{"t":{"$date":"2021-02-27T04:29:47.232+00:00"},"s":"I",  "c":"STORAGE",  "id":20320,   "ct
x":"conn1","msg":"createCollection","attr":{"namespace":"myumpires.plateumpires","uuidDisp
osition":"generated","uuid":{"uuid":{"$uuid":"702fb779-d990-4800-882f-cdf54d403c24"}},"opt
ions":{}}}
{"t":{"$date":"2021-02-27T04:29:47.253+00:00"},"s":"I",  "c":"INDEX",    "id":20345,   "ct
x":"conn1","msg":"Index build: done building","attr":{"buildUUID":null,"namespace":"myumpi
res.plateumpires","index":"_id_","commitTimestamp":{"$timestamp":{"t":0,"i":0}}}}
```

   - Verify the creation of plateumpires.

     `show collections`

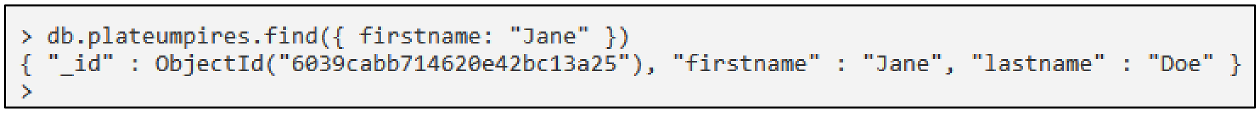


```
> show collections
plateumpires
>
```

8. Retrieve data from the collection plateumpires.
   - Enter the following command:

     `db.plateumpires.find({ firstname: "Jane" })`

   - Observe the results in the Terminal.



```
> db.plateumpires.find({ firstname: "Jane" })
{ "_id" : ObjectId("6039cabb714620e42bc13a25"), "firstname" : "Jane", "lastname" : "Doe" }
>
```

Congratulations! You have completed your first Docker lab exercise.