# Pied Piper Technical Workshop

## Day 2: Containers and Kubernetes

Abstract

This document is provided to assist attendees with completing the appropriate labs to apply the concepts and knowledge learnt throughout the technical workshop program. It is not intended to be used or distributed in isolation and may not contain all required information.

May 2020

Student Lab Guide

# Revisions

| Version | Date | Description |
|---|---|---|
| 0.1 | May 2020 | Initial draft |
| 0.2 | June 2020 | Updates- Labs documented with detailed steps and additional screenshots |
| | | |

The information in this publication is provided "as is." Dell Inc. makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any software described in this publication requires an applicable software license.

Dell believes the information in this document is accurate as of its publication date. The information is subject to change without notice.

# Table of Contents

# Lab 1: Docker

## Module Objectives:

- □ Learn about Docker and Docker Desktop
- □ Build a Docker image
- □ Run a Docker container
- □ Test its functionality

Student Lab Guide

## Tutorial- Docker

- The link below offers a quick start to docker container
  - https://hub.docker.com/?overlay=onboarding&step=clone

## Lab Exercise: Build a docker image and run as a docker container

Complete the below steps to demonstrate your understanding of the tools and concepts required for the remaining lab exercises;

1. Clone the following repository (if not already);

   https://github.com/theocrithary/Piper-2020/tree/master/Day%202/Lab%2001%20-%20Docker

2. Review the files contained in the "Day-2/Lab 01 – Docker" directory

   - app.py
   - DockerFile
   - requirements.txt

3. Open a command prompt and cd into the local directory containing your files

   ```
   > cd C:\Users\{user}\Documents\Piper-2020\Day 2\Lab 01 - Docker
   ```

4. Type the following command and observe the output {NOTE- don't miss the dot (.) at the end of the command below}

   ```
   > docker build -t helloworld .
   ```

5. Type this command to check that the container image was built successfully

   ```
   > docker image ls
   ```

6. Type this command to start the container and run the Python application

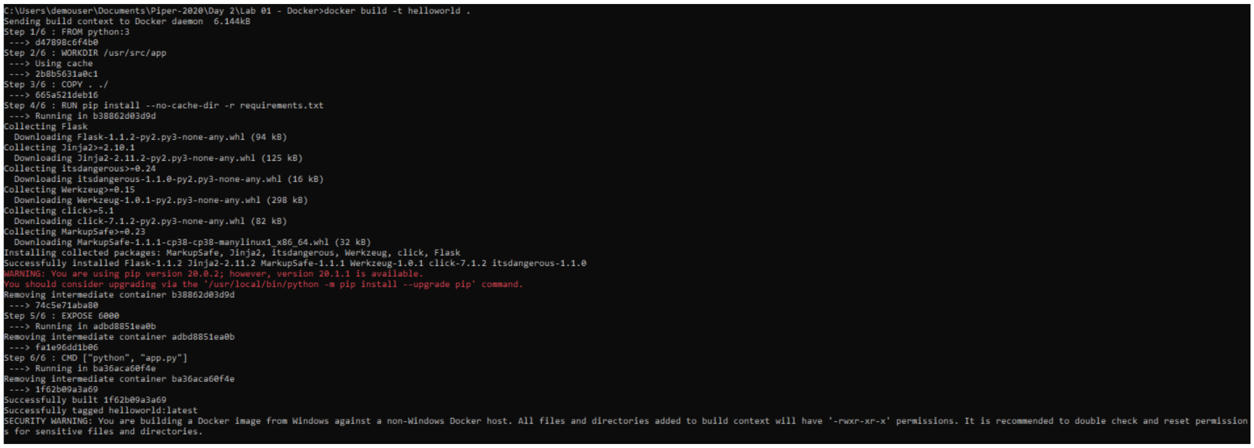
   ```
   > docker run -p 6000:6000 helloworld
   ```

7. Open a web browser and go to the following URL: http://localhost:6000
8. Press CTRL +C on command prompt to quit

## Retrospective: The results

❑ Built a Docker image

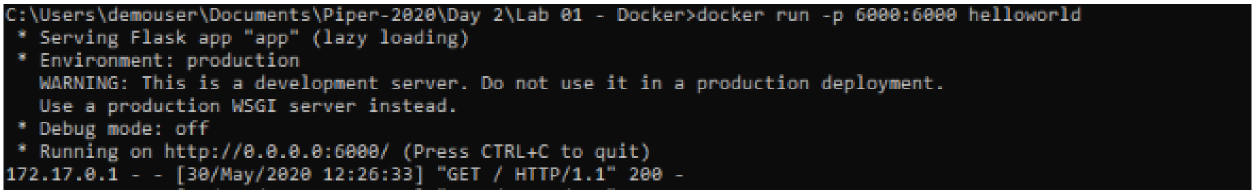
If you successfully built a docker image, you should see the below result in the command prompt



```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 01 - Docker>docker build -t helloworld .
Sending build context to Docker daemon  6.144kB
Step 1/6 : FROM python:3
 ---> d47898c6f4b0
Step 2/6 : WORKDIR /usr/src/app
 ---> Using cache
 ---> 2b8b5631a0c1
Step 3/6 : COPY . ./
 ---> 665a521deb16
Step 4/6 : RUN pip install --no-cache-dir -r requirements.txt
 ---> Running in b38862d03d9d
Collecting Flask
  Downloading Flask-1.1.2-py2.py3-none-any.whl (94 kB)
Collecting Jinja2>=2.10.1
  Downloading Jinja2-2.11.2-py2.py3-none-any.whl (125 kB)
Collecting itsdangerous>=0.24
  Downloading itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
Collecting Werkzeug>=0.15
  Downloading Werkzeug-1.0.1-py2.py3-none-any.whl (298 kB)
Collecting click>=5.1
  Downloading click-7.1.2-py2.py3-none-any.whl (82 kB)
Collecting MarkupSafe>=0.23
  Downloading MarkupSafe-1.1.1-cp38-cp38-manylinux1_x86_64.whl (32 kB)
Installing collected packages: MarkupSafe, Jinja2, itsdangerous, Werkzeug, click, Flask
Successfully installed Flask-1.1.2 Jinja2-2.11.2 MarkupSafe-1.1.1 Werkzeug-1.0.1 click-7.1.2 itsdangerous-1.1.0
WARNING: You are using pip version 20.0.2; however, version 20.1.1 is available.
You should consider upgrading via the '/usr/local/bin/python -m pip install --upgrade pip' command.
Removing intermediate container b38862d03d9d
 ---> 74c5e71aba80
Step 5/6 : EXPOSE 6000
 ---> Running in adbd8851ea0b
Removing intermediate container adbd8851ea0b
 ---> fa1e96dd1b06
Step 6/6 : CMD ["python", "app.py"]
 ---> Running in ba36aca60f4e
Removing intermediate container ba36aca60f4e
 ---> 1f62b09a3a69
Successfully built 1f62b09a3a69
Successfully tagged helloworld:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permission
s for sensitive files and directories.
```

❑ Ran a container instance from a Docker image

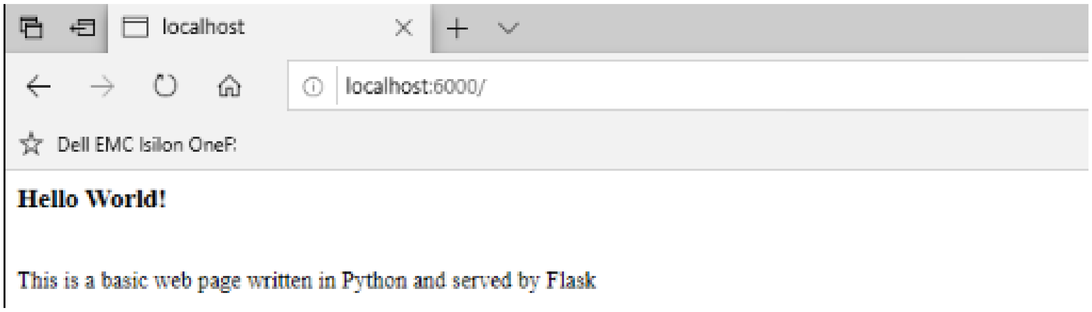
If you successfully ran the docker container, you should see the below results on the command prompt



```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 01 - Docker>docker run -p 6000:6000 helloworld
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:6000/ (Press CTRL+C to quit)
172.17.0.1 - - [30/May/2020 12:26:33] "GET / HTTP/1.1" 200 -
```

❑ Tested the web app

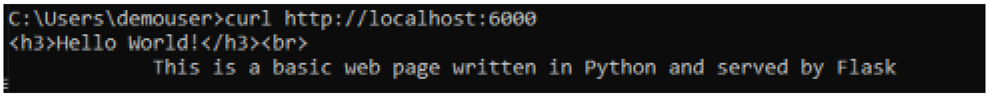
If you successfully ran the docker container, you should see the below results on the web browser or CLI



**Hello World!**

This is a basic web page written in Python and served by Flask

Or,



```
C:\Users\demouser>curl http://localhost:6000
<h3>Hello World!</h3><br>
             This is a basic web page written in Python and served by Flask
```

# Lab 2: Kubernetes

## Module Objectives:

- ☐ Learn about Kubernetes and Docker Desktop Kubernetes
- ☐ Create a Kubernetes deployment from a Docker image
- ☐ Create a service to expose the port
- ☐ Test its functionality

## Tutorial- Standalone Kubernetes with Docker for Windows

- The link below offers a quick start to standalone Kubernetes server that runs on Windows host running Docker Desktop
  - https://docs.docker.com/docker-for-windows/#kubernetes

DELL Technologies

## Lab Exercise: Create a Kubernetes deployment from a Docker image and expose a service

Complete the below steps to demonstrate your understanding of the tools and concepts required for the remaining lab exercises;

1. Clone the following repository (if not already);

   https://github.com/theocrithary/Piper-2020/tree/master/Day%202/Lab%2002%20-%20Kubernetes

2. Review the files contained in the Docker directory

   - app.py
   - DockerFile
   - requirements.txt

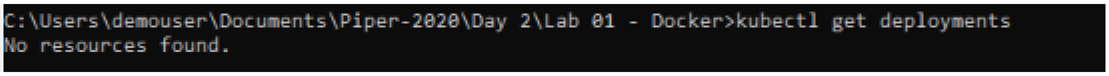
   Note: This version has changed port to 6001 in app.py and DockerFile

3. Open a command prompt and cd into the local directory containing your files (note that it's changed to "Lab 02 – Kubernetes" folder)

   > cd C:\Users\{user}\Documents\Piper-2020\Day 2\Lab 02 - Kubernetes

1. Check that the Kubernetes cluster is running and available

   > kubectl get deployment



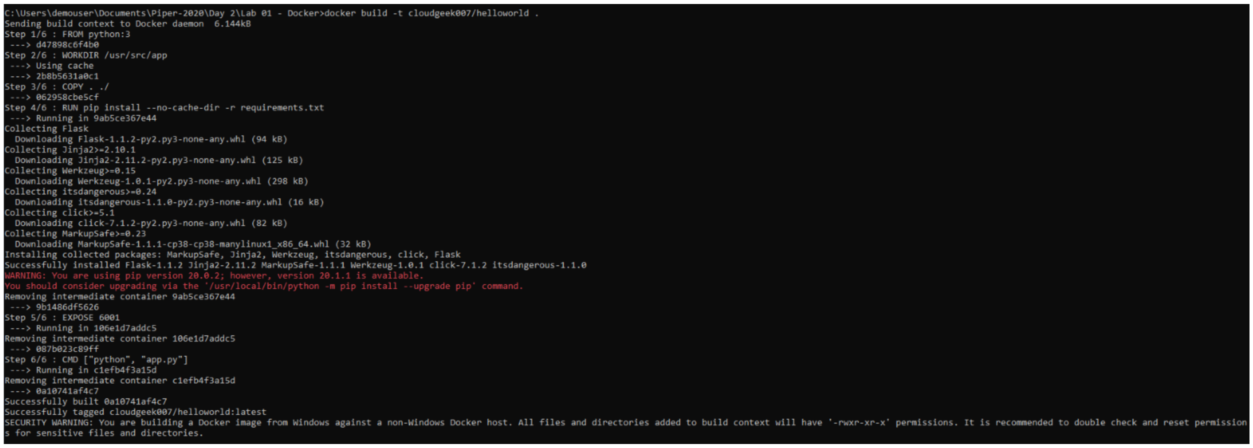
```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 01 - Docker>kubectl get deployments
No resources found.
```

2. Login to docker with your DockerHub credentials

   > docker login

3. Build docker container with your DockerHub credentials {NOTE- don't miss the dot (.) at the end of the command below}

   > docker build -t "YourDockerHubID"/helloworld .

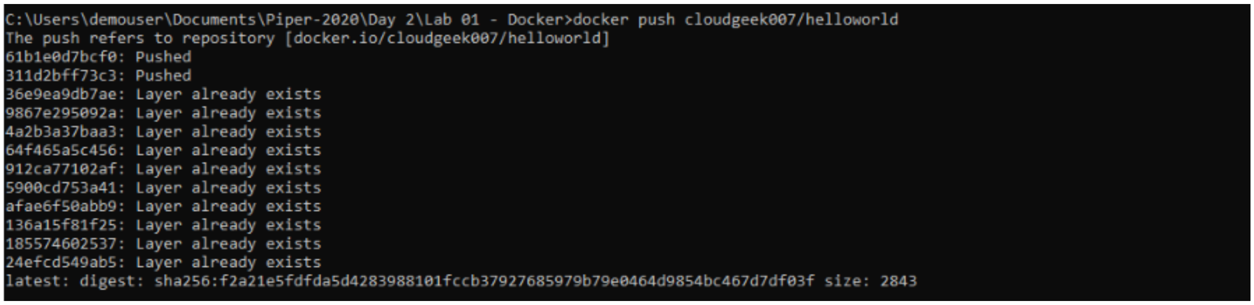


```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 01 - Docker>docker build -t cloudgeek007/helloworld .
Sending build context to Docker daemon  6.144kB
Step 1/6 : FROM python:3
 ---> d47898c6f4b0
Step 2/6 : WORKDIR /usr/src/app
 ---> Using cache
 ---> 2b8b5631a0c1
Step 3/6 : COPY . ./
 ---> 062958cbe5cf
Step 4/6 : RUN pip install --no-cache-dir -r requirements.txt
 ---> Running in 9ab5ce367e44
Collecting Flask
  Downloading Flask-1.1.2-py2.py3-none-any.whl (94 kB)
Collecting Jinja2>=2.10.1
  Downloading Jinja2-2.11.2-py2.py3-none-any.whl (125 kB)
Collecting Werkzeug>=0.15
  Downloading Werkzeug-1.0.1-py2.py3-none-any.whl (298 kB)
Collecting itsdangerous>=0.24
  Downloading itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
Collecting click>=5.1
  Downloading click-7.1.2-py2.py3-none-any.whl (82 kB)
Collecting MarkupSafe>=0.23
  Downloading MarkupSafe-1.1.1-cp38-cp38-manylinux1_x86_64.whl (32 kB)
Installing collected packages: MarkupSafe, Jinja2, Werkzeug, itsdangerous, click, Flask
Successfully installed Flask-1.1.2 Jinja2-2.11.2 MarkupSafe-1.1.1 Werkzeug-1.0.1 click-7.1.2 itsdangerous-1.1.0
WARNING: You are using pip version 20.0.2; however, version 20.1.1 is available.
You should consider upgrading via the '/usr/local/bin/python -m pip install --upgrade pip' command.
Removing intermediate container 9ab5ce367e44
 ---> 9b1486df5626
Step 5/6 : EXPOSE 6001
 ---> Running in 106e1d7addc5
Removing intermediate container 106e1d7addc5
 ---> 087b023c89ff
Step 6/6 : CMD ["python", "app.py"]
 ---> Running in c1efb4f3a15d
Removing intermediate container c1efb4f3a15d
 ---> 0a10741af4c7
Successfully built 0a10741af4c7
Successfully tagged cloudgeek007/helloworld:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permission
s for sensitive files and directories.
```

Note: we are using the same image we built in Lab 01, except this version has changed port to 6001 and has to be pushed to a repository on Docker Hub. i.e. https://hub.docker.com/repository/docker/"YourDockerhubID"/helloworld

4. Push the docker image to repository on Docker Hub

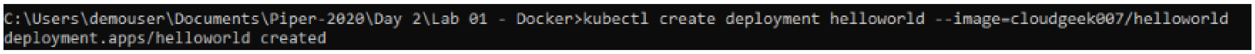
   > docker push YourDockerHubID"/helloworld



```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 01 - Docker>docker push cloudgeek007/helloworld
The push refers to repository [docker.io/cloudgeek007/helloworld]
61b1e0d7bcf0: Pushed
311d2bff73c3: Pushed
36e9ea9db7ae: Layer already exists
9867e295092a: Layer already exists
4a2b3a37baa3: Layer already exists
64f465a5c456: Layer already exists
912ca77102af: Layer already exists
5900cd753a41: Layer already exists
afae6f50abb9: Layer already exists
136a15f81f25: Layer already exists
185574602537: Layer already exists
24efcd549ab5: Layer already exists
latest: digest: sha256:f2a21e5fdfda5d4283988101fccb37927685979b79e0464d9854bc467d7df03f size: 2843
```

5. Type the following command and observe the output

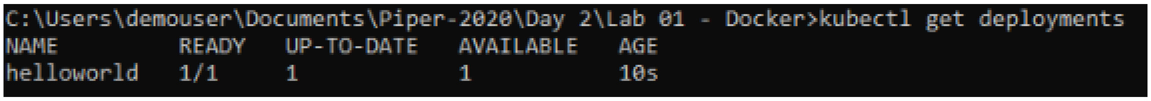
   > kubectl create deployment helloworld --image="YourDockerHubID"/helloworld



```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 01 - Docker>kubectl create deployment helloworld --image=cloudgeek007/helloworld
deployment.apps/helloworld created
```

6. Type this command to check that the container was deployed to the Kubernetes cluster

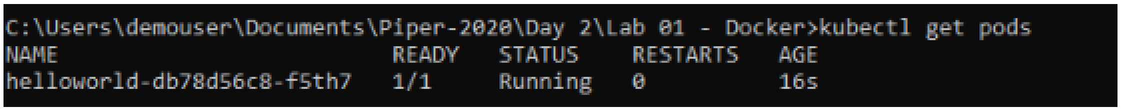
   > kubectl get deployment



```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 01 - Docker>kubectl get deployments
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
helloworld   1/1     1            1           10s
```

7. Type this command to check that a pod was built and is running successfully

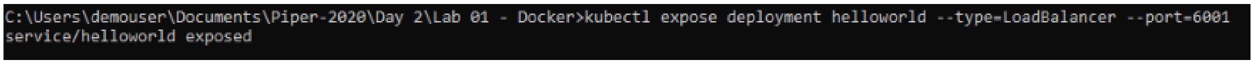
   > kubectl get pods



```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 01 - Docker>kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
helloworld-db78d56c8-f5th7    1/1     Running   0          16s
```

8. Type this command to expose port 6001 and allow external access into the Kubernetes cluster

   > kubectl expose deployment helloworld --type=LoadBalancer --port=6001



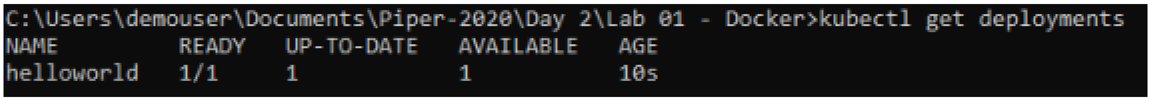
```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 01 - Docker>kubectl expose deployment helloworld --type=LoadBalancer --port=6001
service/helloworld exposed
```

9. Open a web browser and go to the following URL: http://localhost:6001

## Retrospective: The results

□ Created a Kubernetes deployment from a Docker hub image

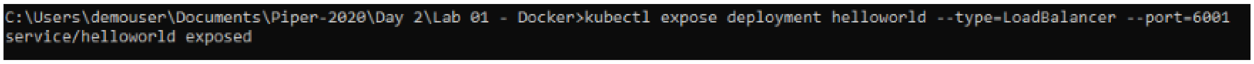
If you successfully created a deployment, you should see the below result in the command prompt



```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 01 - Docker>kubectl get deployments
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
helloworld   1/1     1            1           10s
```

□ Created a service to expose port 6001

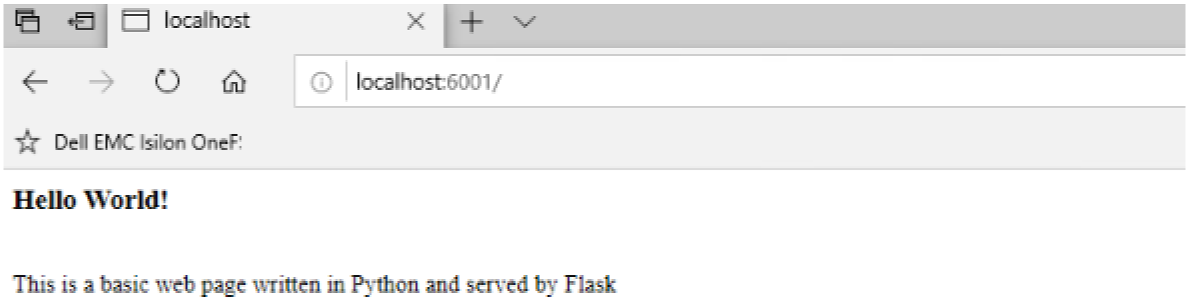
If you successfully created a service and exposed to port, you should see the below results on the command prompt



```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 01 - Docker>kubectl expose deployment helloworld --type=LoadBalancer --port=6001
service/helloworld exposed
```

□ Tested the web app

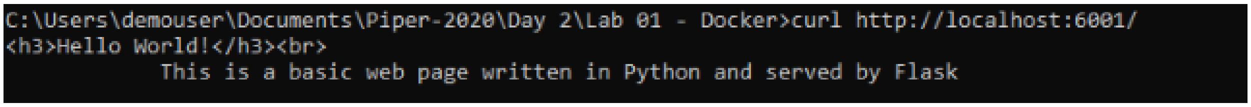
If you created deployment and exposed service to port successfully, you should see the below results on the web browser or CLI



**Hello World!**

This is a basic web page written in Python and served by Flask

Or,



```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 01 - Docker>curl http://localhost:6001/
<h3>Hello World!</h3><br>
            This is a basic web page written in Python and served by Flask
```

# Lab 3: Containerising an App

## Module Objectives:

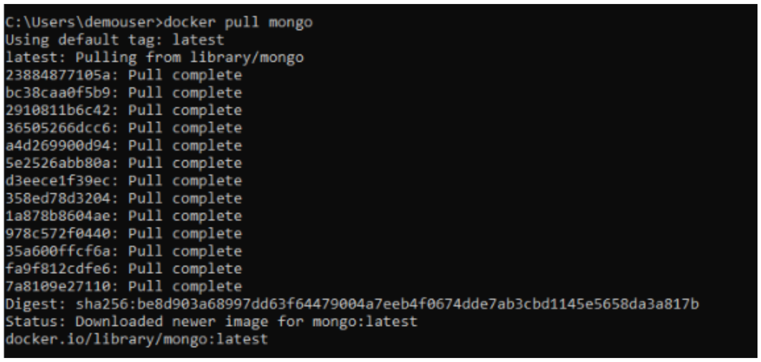
- ❑ Apply knowledge we learnt in previous labs
- ❑ Deploy a 2-tier app to Docker
- ❑ Forward port to allow local testing
- ❑ Deploy a 2-tier app to a Kubernetes cluster
- ❑ Expose the service port
- ❑ Test its functionality

## Lab Exercise Part 1: Create a 2-tier docker app and forward port for local testing

Complete the below steps to demonstrate your understanding of the tools and concepts required for the remaining lab exercises;

1. Get the mongodb image from Docker Hub

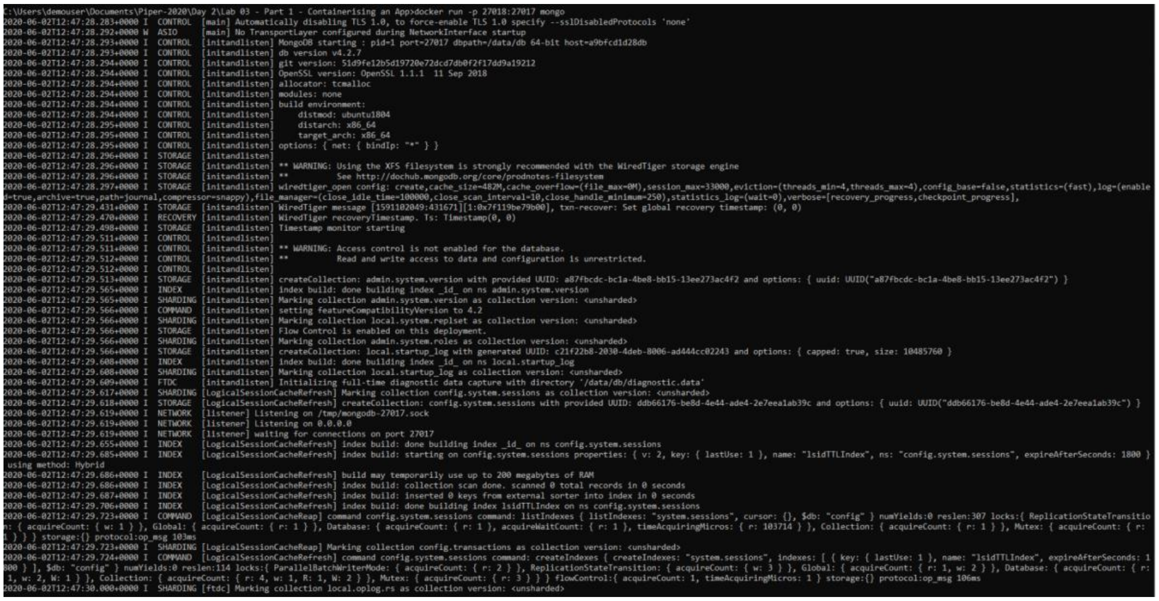
   > docker pull mongo



```
C:\Users\demouser>docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
23884877105a: Pull complete
bc38caa0f5b9: Pull complete
2910811b6c42: Pull complete
36505266dcc6: Pull complete
a4d269900d94: Pull complete
5e2526abb80a: Pull complete
d3eece1f39ec: Pull complete
358ed78d3204: Pull complete
1a878b8604ae: Pull complete
978c572f0440: Pull complete
35a600ffcf6a: Pull complete
fa9f812cdfe6: Pull complete
7a8109e27110: Pull complete
Digest: sha256:be8d903a68997dd63f64479004a7eeb4f0674dde7ab3cbd1145e5658da3a817b
Status: Downloaded newer image for mongo:latest
docker.io/library/mongo:latest
```

2. Type this command to start the mongo DB container

   > docker run -p 27018:27017 mongo



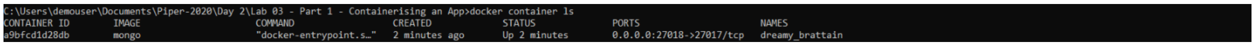
```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 03 - Part 1 - Containerising an App>docker run -p 27018:27017 mongo
2020-06-02T12:47:28.283+0000 I  CONTROL  [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2020-06-02T12:47:28.292+0000 W  ASIO     [main] No TransportLayer configured during NetworkInterface startup
2020-06-02T12:47:28.293+0000 I  CONTROL  [initandlisten] MongoDB starting : pid=1 port=27017 dbpath=/data/db 64-bit host=a9bfcd1d28db
2020-06-02T12:47:28.293+0000 I  CONTROL  [initandlisten] db version v4.2.7
2020-06-02T12:47:28.294+0000 I  CONTROL  [initandlisten] git version: 51d9fe12b5d19720e72dcd7db0f2f17dd9a19212
2020-06-02T12:47:28.294+0000 I  CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.1.1  11 Sep 2018
2020-06-02T12:47:28.294+0000 I  CONTROL  [initandlisten] allocator: tcmalloc
2020-06-02T12:47:28.294+0000 I  CONTROL  [initandlisten] modules: none
2020-06-02T12:47:28.294+0000 I  CONTROL  [initandlisten] build environment:
2020-06-02T12:47:28.294+0000 I  CONTROL  [initandlisten]     distmod: ubuntu1804
2020-06-02T12:47:28.295+0000 I  CONTROL  [initandlisten]     distarch: x86_64
2020-06-02T12:47:28.295+0000 I  CONTROL  [initandlisten]     target_arch: x86_64
2020-06-02T12:47:28.295+0000 I  CONTROL  [initandlisten] options: { net: { bindIp: "*" } }
2020-06-02T12:47:28.296+0000 I  STORAGE  [initandlisten]
2020-06-02T12:47:28.296+0000 I  STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2020-06-02T12:47:28.296+0000 I  STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2020-06-02T12:47:28.297+0000 I  STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=482M,cache_overflow=(file_max=0M),session_max=33000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enable
d=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000,close_scan_interval=10,close_handle_minimum=250),statistics_log=(wait=0),verbose=[recovery_progress,checkpoint_progress],
2020-06-02T12:47:29.431+0000 I  STORAGE  [initandlisten] WiredTiger message [1591102049:431671][1:0x7f119be79b00], txn-recover: Set global recovery timestamp: (0, 0)
2020-06-02T12:47:29.470+0000 I  RECOVERY [initandlisten] WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
2020-06-02T12:47:29.498+0000 I  STORAGE  [initandlisten] Timestamp monitor starting
2020-06-02T12:47:29.511+0000 I  CONTROL  [initandlisten]
2020-06-02T12:47:29.511+0000 I  CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-06-02T12:47:29.512+0000 I  CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2020-06-02T12:47:29.512+0000 I  CONTROL  [initandlisten]
2020-06-02T12:47:29.513+0000 I  STORAGE  [initandlisten] createCollection: admin.system.version with provided UUID: a87fbcdc-bc1a-4be8-bb15-13ee273ac4f2 and options: { uuid: UUID("a87fbcdc-bc1a-4be8-bb15-13ee273ac4f2") }
2020-06-02T12:47:29.565+0000 I  INDEX    [initandlisten] index build: done building index _id_ on ns admin.system.version
2020-06-02T12:47:29.565+0000 I  SHARDING [initandlisten] Marking collection admin.system.version as collection version: <unsharded>
2020-06-02T12:47:29.566+0000 I  COMMAND  [initandlisten] setting featureCompatibilityVersion to 4.2
2020-06-02T12:47:29.566+0000 I  SHARDING [initandlisten] Marking collection local.system.replset as collection version: <unsharded>
2020-06-02T12:47:29.566+0000 I  STORAGE  [initandlisten] Flow Control is enabled on this deployment.
2020-06-02T12:47:29.566+0000 I  SHARDING [initandlisten] Marking collection admin.system.roles as collection version: <unsharded>
2020-06-02T12:47:29.566+0000 I  STORAGE  [initandlisten] createCollection: local.startup_log with generated UUID: c21f22b8-2030-4deb-8006-ad444cc02243 and options: { capped: true, size: 10485760 }
2020-06-02T12:47:29.608+0000 I  INDEX    [initandlisten] index build: done building index _id_ on ns local.startup_log
2020-06-02T12:47:29.608+0000 I  SHARDING [initandlisten] Marking collection local.startup_log as collection version: <unsharded>
2020-06-02T12:47:29.609+0000 I  FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory '/data/db/diagnostic.data'
2020-06-02T12:47:29.617+0000 I  SHARDING [LogicalSessionCacheRefresh] Marking collection config.system.sessions as collection version: <unsharded>
2020-06-02T12:47:29.618+0000 I  STORAGE  [LogicalSessionCacheRefresh] createCollection: config.system.sessions with provided UUID: ddb66176-be8d-4e44-ade4-2e7eea1ab39c and options: { uuid: UUID("ddb66176-be8d-4e44-ade4-2e7eea1ab39c") }
2020-06-02T12:47:29.619+0000 I  NETWORK  [listener] Listening on /tmp/mongodb-27017.sock
2020-06-02T12:47:29.619+0000 I  NETWORK  [listener] Listening on 0.0.0.0
2020-06-02T12:47:29.619+0000 I  NETWORK  [listener] waiting for connections on port 27017
2020-06-02T12:47:29.655+0000 I  INDEX    [LogicalSessionCacheRefresh] index build: done building index _id_ on ns config.system.sessions
2020-06-02T12:47:29.685+0000 I  INDEX    [LogicalSessionCacheRefresh] index build: starting on config.system.sessions properties: { v: 2, key: { lastUse: 1 }, name: "lsidTTLIndex", ns: "config.system.sessions", expireAfterSeconds: 1800 }
 using method: Hybrid
2020-06-02T12:47:29.686+0000 I  INDEX    [LogicalSessionCacheRefresh] build may temporarily use up to 200 megabytes of RAM
2020-06-02T12:47:29.686+0000 I  INDEX    [LogicalSessionCacheRefresh] index build: collection scan done. scanned 0 total records in 0 seconds
2020-06-02T12:47:29.687+0000 I  INDEX    [LogicalSessionCacheRefresh] index build: inserted 0 keys from external sorter into index in 0 seconds
2020-06-02T12:47:29.706+0000 I  INDEX    [LogicalSessionCacheRefresh] index build: done building index lsidTTLIndex on ns config.system.sessions
2020-06-02T12:47:29.723+0000 I  COMMAND  [LogicalSessionCacheReap] command config.system.sessions command: listIndexes { listIndexes: "system.sessions", cursor: {}, $db: "config" } numYields:0 reslen:307 locks:{ ReplicationStateTransitio
n: { acquireCount: { w: 1 } }, Global: { acquireCount: { r: 1 } }, Database: { acquireCount: { r: 1 }, acquireWaitCount: { r: 1 }, timeAcquiringMicros: { r: 103714 } }, Collection: { acquireCount: { r: 1 } }, Mutex: { acquireCount: { r:
1 } } } storage:{} protocol:op_msg 103ms
2020-06-02T12:47:29.723+0000 I  SHARDING [LogicalSessionCacheReap] Marking collection config.transactions as collection version: <unsharded>
2020-06-02T12:47:29.724+0000 I  COMMAND  [LogicalSessionCacheRefresh] command config.system.sessions command: createIndexes { createIndexes: "system.sessions", indexes: [ { key: { lastUse: 1 }, name: "lsidTTLIndex", expireAfterSeconds: 1
800 } ], $db: "config" } numYields:0 reslen:114 locks:{ ParallelBatchWriterMode: { acquireCount: { r: 2 } }, ReplicationStateTransition: { acquireCount: { w: 3 } }, Global: { acquireCount: { r: 1, w: 2 } }, Database: { acquireCount: { r:
 1, w: 2, W: 1 } }, Collection: { acquireCount: { r: 4, w: 1, R: 1, W: 2 } }, Mutex: { acquireCount: { r: 3 } } } flowControl:{ acquireCount: 1, timeAcquiringMicros: 1 } storage:{} protocol:op_msg 106ms
2020-06-02T12:47:30.000+0000 I  SHARDING [ftdc] Marking collection local.oplog.rs as collection version: <unsharded>
```

   > Ctrl + C to break from command and run in the background

   **Note: The change in exposed port to 27018. This is to avoid conflicts with the existing MongoDB server running locally at 27017**

3. Type these commands to check that the container was deployed to Docker

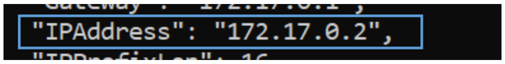
   > docker container ls



```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 03 - Part 1 - Containerising an App>docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                      NAMES
a9bfcd1d28db        mongo               "docker-entrypoint.s…"   2 minutes ago       Up 2 minutes        0.0.0.0:27018->27017/tcp   dreamy_brattain
```

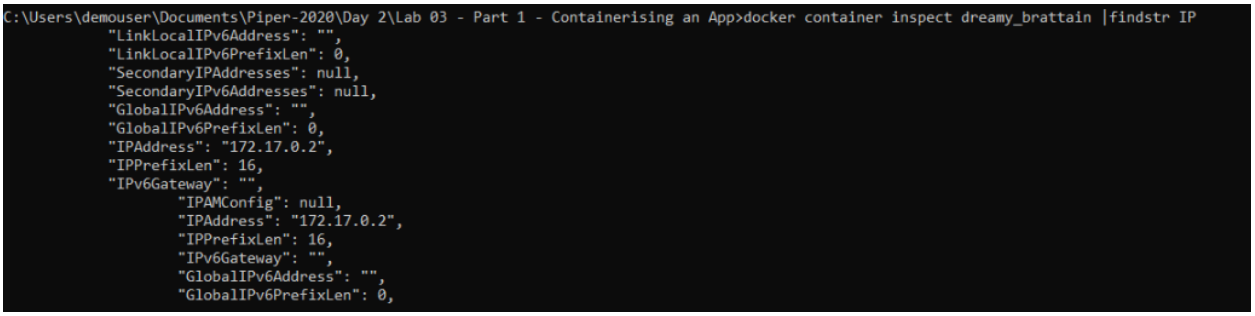
4. Type this command to get the IP address of the docker container running mongodb

   > docker container inspect {container name}



```
"IPAddress": "172.17.0.2",
```

**NOTE- Take note of this IP address, we will use it later in our code to create a database connection**



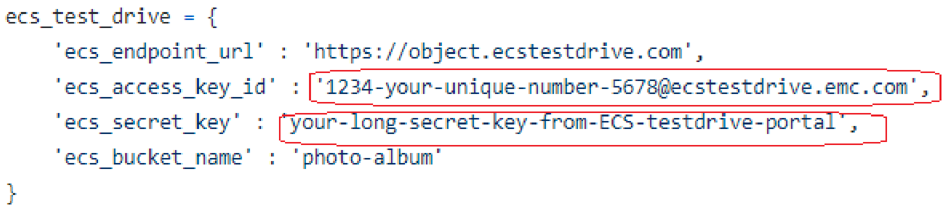
```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 03 - Part 1 - Containerising an App>docker container inspect dreamy_brattain |findstr IP
            "LinkLocalIPv6Address": "",
            "LinkLocalIPv6PrefixLen": 0,
            "SecondaryIPAddresses": null,
            "SecondaryIPv6Addresses": null,
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "IPAddress": "172.17.0.2",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
                    "IPAMConfig": null,
                    "IPAddress": "172.17.0.2",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
```

5. Clone the following repository (if not already); https://github.com/theocrithary/Piper-2020/tree/master/Day%202/Lab%2003%20-%20Part%201%20-%20Containerising%20an%20App

6. Check that the config file contains your ECS credentials

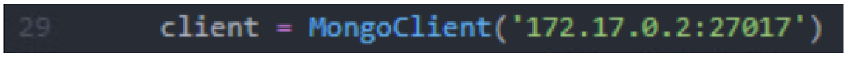
   - config.py

   - Rename config-example.py to config.py and replace with your ECS account credentials if required



```
ecs_test_drive = {
    'ecs_endpoint_url' : 'https://object.ecstestdrive.com',
    'ecs_access_key_id' : '1234-your-unique-number-5678@ecstestdrive.emc.com',
    'ecs_secret_key' : 'your-long-secret-key-from-ECS-testdrive-portal',
    'ecs_bucket_name' : 'photo-album'
}
```

7. Edit the models.py file on line 29 to reflect the IP address of your mongodb container



```
29        client = MongoClient('172.17.0.2:27017')
```

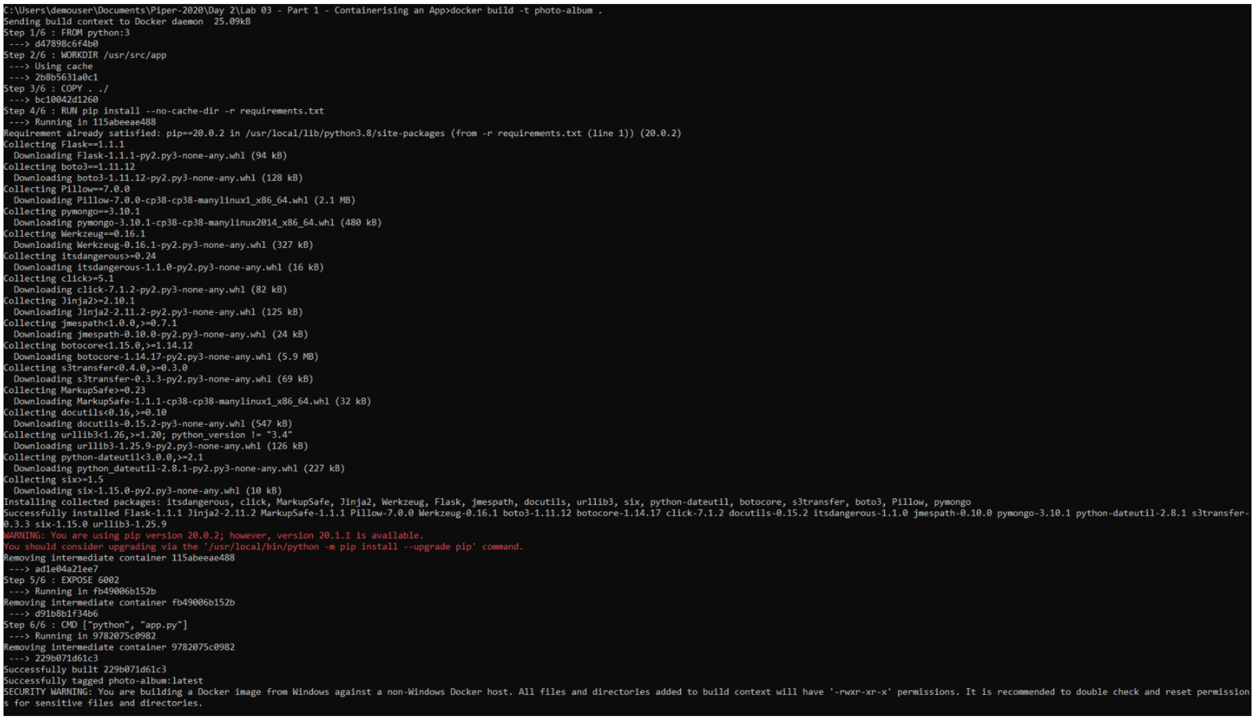
   Note: We use port 27017 as the internal accessible port within the Docker network.

8. Open a command prompt and cd into the local directory containing your files

   > cd {user project folder}\Piper-2020\Day 2\Lab 03 - Containerising an App\

9. Type the following commands and observe the outputs {NOTE- don’t miss the dot (.) at the end of the command below}

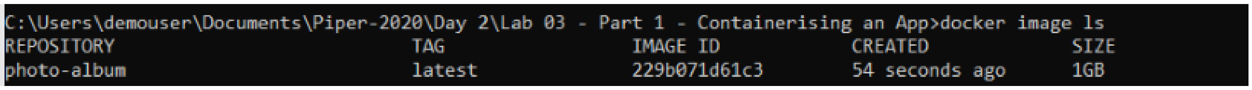
   > docker build -t photo-album .



```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 03 - Part 1 - Containerising an App>docker build -t photo-album .
Sending build context to Docker daemon  25.09kB
Step 1/6 : FROM python:3
 ---> d47898c6f4b0
Step 2/6 : WORKDIR /usr/src/app
 ---> Using cache
 ---> 2b8b5631a0c1
Step 3/6 : COPY . ./
 ---> bc10042d1260
Step 4/6 : RUN pip install --no-cache-dir -r requirements.txt
 ---> Running in 115abeeae488
Requirement already satisfied: pip==20.0.2 in /usr/local/lib/python3.8/site-packages (from -r requirements.txt (line 1)) (20.0.2)
Collecting Flask==1.1.1
  Downloading Flask-1.1.1-py2.py3-none-any.whl (94 kB)
Collecting boto3==1.11.12
  Downloading boto3-1.11.12-py2.py3-none-any.whl (128 kB)
Collecting Pillow==7.0.0
  Downloading Pillow-7.0.0-cp38-cp38-manylinux1_x86_64.whl (2.1 MB)
Collecting pymongo==3.10.1
  Downloading pymongo-3.10.1-cp38-cp38-manylinux2014_x86_64.whl (480 kB)
Collecting Werkzeug==0.16.1
  Downloading Werkzeug-0.16.1-py2.py3-none-any.whl (327 kB)
Collecting itsdangerous>=0.24
  Downloading itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
Collecting click>=5.1
  Downloading click-7.1.2-py2.py3-none-any.whl (82 kB)
Collecting Jinja2>=2.10.1
  Downloading Jinja2-2.11.2-py2.py3-none-any.whl (125 kB)
Collecting jmespath<1.0.0,>=0.7.1
  Downloading jmespath-0.10.0-py2.py3-none-any.whl (24 kB)
Collecting botocore<1.15.0,>=1.14.12
  Downloading botocore-1.14.17-py2.py3-none-any.whl (5.9 MB)
Collecting s3transfer<0.4.0,>=0.3.0
  Downloading s3transfer-0.3.3-py2.py3-none-any.whl (69 kB)
Collecting MarkupSafe>=0.23
  Downloading MarkupSafe-1.1.1-cp38-cp38-manylinux1_x86_64.whl (32 kB)
Collecting docutils<0.16,>=0.10
  Downloading docutils-0.15.2-py3-none-any.whl (547 kB)
Collecting urllib3<1.26,>=1.20; python_version != "3.4"
  Downloading urllib3-1.25.9-py2.py3-none-any.whl (126 kB)
Collecting python-dateutil<3.0.0,>=2.1
  Downloading python_dateutil-2.8.1-py2.py3-none-any.whl (227 kB)
Collecting six>=1.5
  Downloading six-1.15.0-py2.py3-none-any.whl (10 kB)
Installing collected packages: itsdangerous, click, MarkupSafe, Jinja2, Werkzeug, Flask, jmespath, docutils, urllib3, six, python-dateutil, botocore, s3transfer, boto3, Pillow, pymongo
Successfully installed Flask-1.1.1 Jinja2-2.11.2 MarkupSafe-1.1.1 Pillow-7.0.0 Werkzeug-0.16.1 boto3-1.11.12 botocore-1.14.17 click-7.1.2 docutils-0.15.2 itsdangerous-1.1.0 jmespath-0.10.0 pymongo-3.10.1 python-dateutil-2.8.1 s3transfer-
0.3.3 six-1.15.0 urllib3-1.25.9
WARNING: You are using pip version 20.0.2; however, version 20.1.1 is available.
You should consider upgrading via the '/usr/local/bin/python -m pip install --upgrade pip' command.
Removing intermediate container 115abeeae488
 ---> ad1e04a21ee7
Step 5/6 : EXPOSE 6002
 ---> Running in fb49006b152b
Removing intermediate container fb49006b152b
 ---> d91b8b1f34b6
Step 6/6 : CMD ["python", "app.py"]
 ---> Running in 9782075c0982
Removing intermediate container 9782075c0982
 ---> 229b071d61c3
Successfully built 229b071d61c3
Successfully tagged photo-album:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permission
s for sensitive files and directories.
```

10. Type this command to check that the container image photo-album was built and mongo was downloaded from docker hub

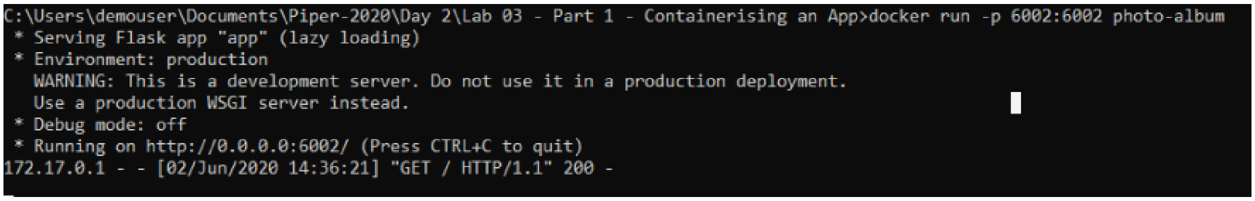
    > docker image ls



```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 03 - Part 1 - Containerising an App>docker image ls
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
photo-album                   latest              229b071d61c3        54 seconds ago      1GB
```

11. Type this command to start the mongo DB container

    > docker run -p 6002:6002 photo-album



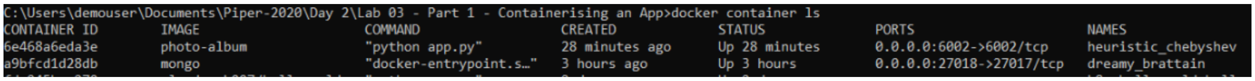
```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 03 - Part 1 - Containerising an App>docker run -p 6002:6002 photo-album
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:6002/ (Press CTRL+C to quit)
172.17.0.1 - - [02/Jun/2020 14:36:21] "GET / HTTP/1.1" 200 -
```

12. Open a web browser and go to the following URL: http://localhost:6002

## Retrospective: The results

❑ Deployed a 2-tier app to both Docker

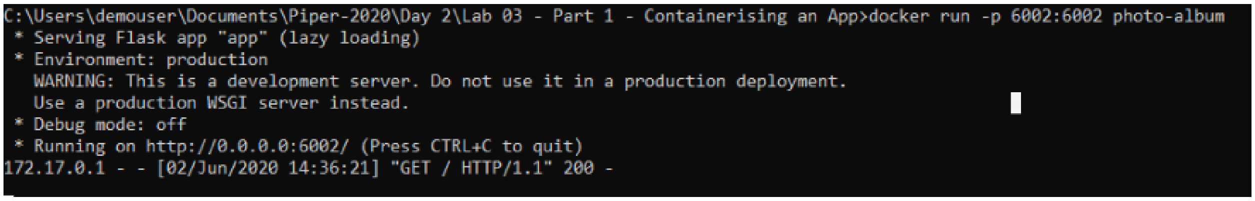
If you successfully ran docker containers for mongodb and photo-album, you should see the below result in the command prompt



```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 03 - Part 1 - Containerising an App>docker container ls
CONTAINER ID        IMAGE                          COMMAND                    CREATED             STATUS              PORTS                         NAMES
6e468a6eda3e        photo-album                    "python app.py"            28 minutes ago      Up 28 minutes       0.0.0.0:6002->6002/tcp        heuristic_chebyshev
a9bfcd1d28db        mongo                          "docker-entrypoint.s…"     3 hours ago         Up 3 hours          0.0.0.0:27018->27017/tcp      dreamy_brattain
```

❑ Forward port to allow local testing

If you successfully ran docker for photo-album on stated port, you should see the below result in the command prompt



```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 03 - Part 1 - Containerising an App>docker run -p 6002:6002 photo-album
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:6002/ (Press CTRL+C to quit)
172.17.0.1 - - [02/Jun/2020 14:36:21] "GET / HTTP/1.1" 200 -
```

❑ Tested the web app

If you are running photo-album container on stated port successfully, you should see the below results on the web browser or CLI

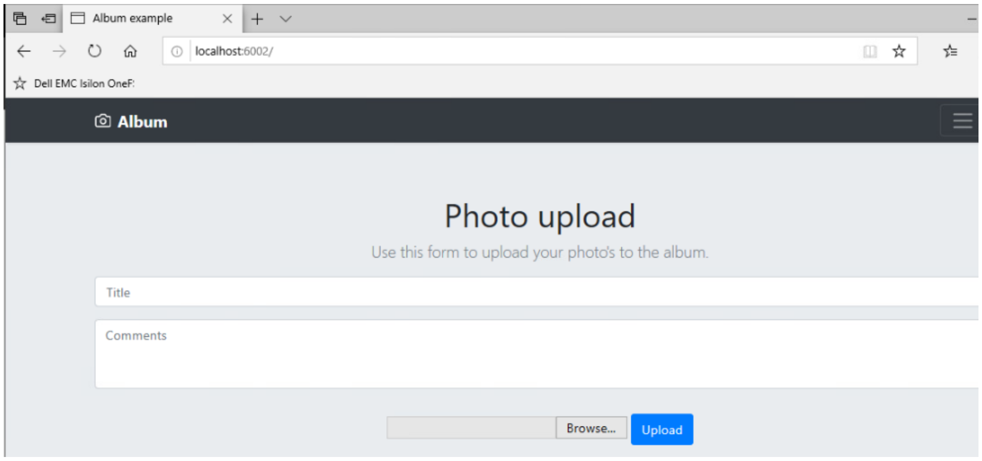
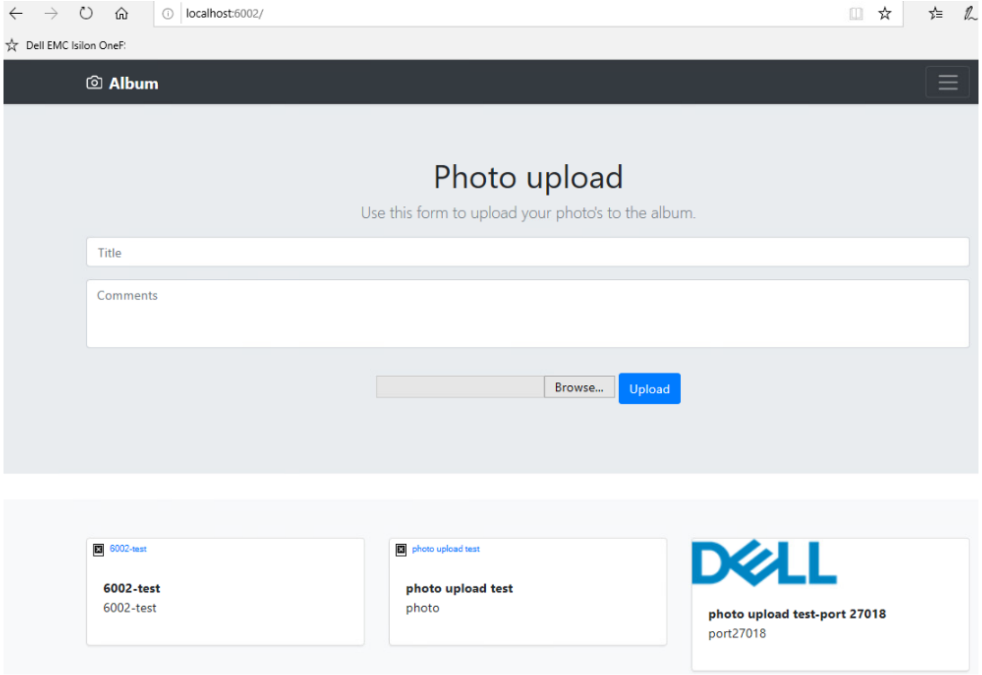


Photo upload

Use this form to upload your photo's to the album.

If you upload a photo, you should see similar results as below on the web browser or CLI



localhost:6002/

Dell EMC Isilon OneF:

Album

Photo upload

Use this form to upload your photo's to the album.

Title

Comments

Browse... Upload

6002-test

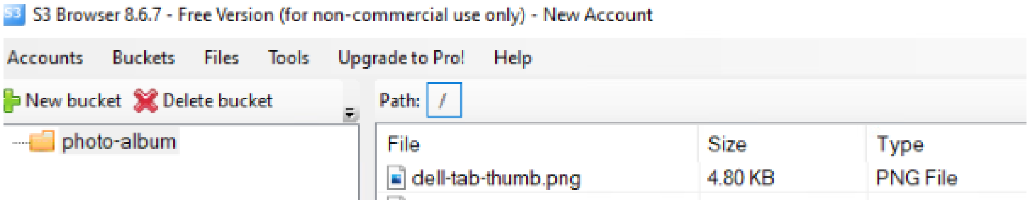
**6002-test**
6002-test

photo upload test

**photo upload test**
photo

**photo upload test-port 27018**
port27018

The photo image being uploaded can be verified using S3 browser



S3 Browser 8.6.7 - Free Version (for non-commercial use only) - New Account

photo-album

| File | Size | Type |
|---|---|---|
| dell-tab-thumb.png | 4.80 KB | PNG File |

## Lab Exercise Part-2: Create a 2-tier docker app for local testing and deploy the 2-tier app to kubernetes cluster

Clean up and delete any Docker container instances you deployed in the previous labs

Complete the below steps to demonstrate your understanding of the tools and concepts required for the remaining lab exercises;

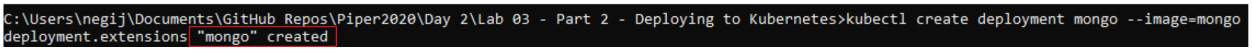
1. Check that the Kubernetes cluster is running and if there are any existing deployments

   > kubectl cluster-info

   > kubectl get deployment

2. Type the following command and observe the output

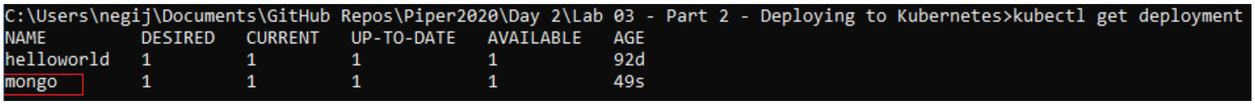
   > kubectl create deployment mongo --image=mongo



```
C:\Users\negij\Documents\GitHub Repos\Piper2020\Day 2\Lab 03 - Part 2 - Deploying to Kubernetes>kubectl create deployment mongo --image=mongo
deployment.extensions "mongo" created
```

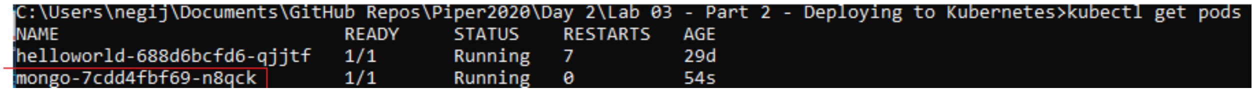
3. Type these commands to check that the container was deployed to the Kubernetes cluster

   > kubectl get deployment



```
C:\Users\negij\Documents\GitHub Repos\Piper2020\Day 2\Lab 03 - Part 2 - Deploying to Kubernetes>kubectl get deployment
NAME         DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
helloworld   1         1         1            1           92d
mongo        1         1         1            1           49s
```

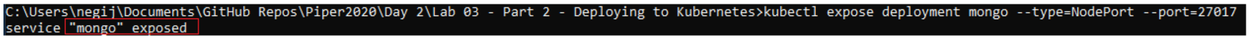
   > kubectl get pods



```
C:\Users\negij\Documents\GitHub Repos\Piper2020\Day 2\Lab 03 - Part 2 - Deploying to Kubernetes>kubectl get pods
NAME                          READY     STATUS    RESTARTS   AGE
helloworld-688d6bcfd6-qjjtf   1/1       Running   7          29d
mongo-7cdd4fbf69-n8qck        1/1       Running   0          54s
```

4. Type this command to expose port 27017 to allow our application to connect to the database port

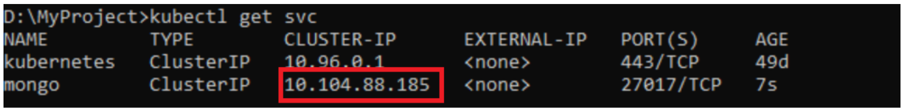
   > kubectl expose deployment mongo --port=27017



```
C:\Users\negij\Documents\GitHub Repos\Piper2020\Day 2\Lab 03 - Part 2 - Deploying to Kubernetes>kubectl expose deployment mongo --type=NodePort --port=27017
service "mongo" exposed
```

5. Type this command to retrieve the cluster IP address of the mongo deployment

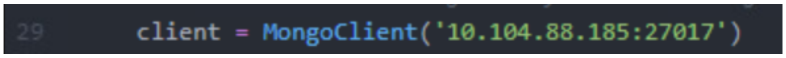
   > kubectl get svc



```
D:\MyProject>kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)     AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP     49d
mongo        ClusterIP   10.104.88.185   <none>        27017/TCP   7s
```

7. Clone the following repository (if not already); [https://github.com/theocrithary/Piper-2020/tree/master/Day%202/Lab%2003%20-%20Part%202%20-%20Deploying%20to%20Kubernetes](https://github.com/theocrithary/Piper-2020/tree/master/Day%202/Lab%2003%20-%20Part%202%20-%20Deploying%20to%20Kubernetes)

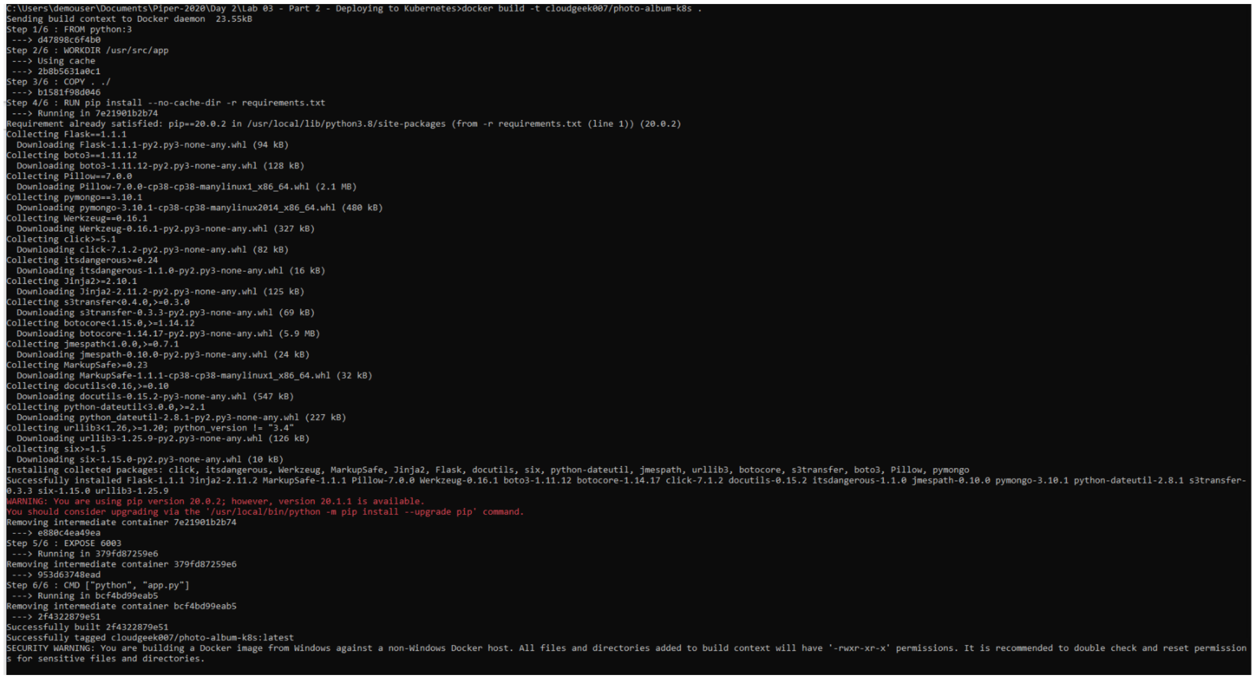
8. Edit the models.py file on line 29 and add the IP address you obtained from the mongo svc



```
29        client = MongoClient('10.104.88.185:27017')
```

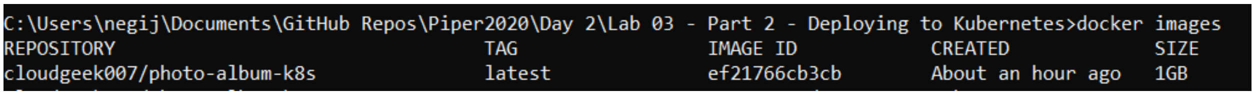
9. Build the docker image and push it to Docker Hub {NOTE- don't miss the dot (.) at the end of the command below}

> docker build -t {dockerhubID}/photo-album-k8s .



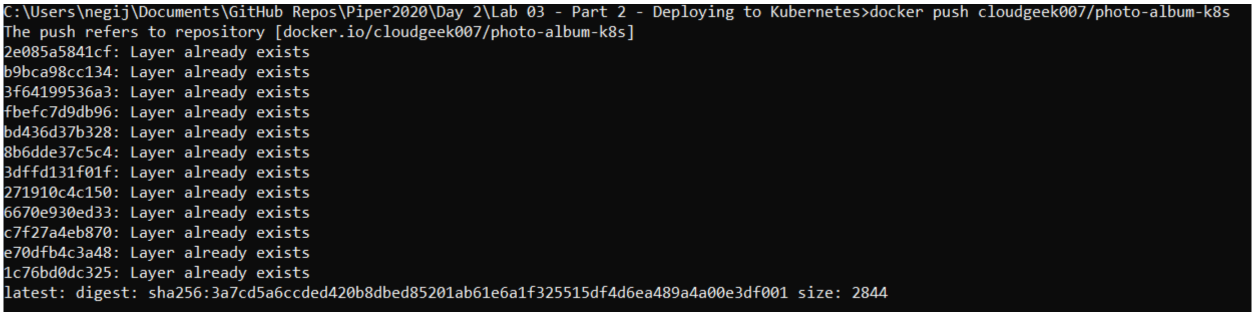
```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 03 - Part 2 - Deploying to Kubernetes>docker build -t cloudgeek007/photo-album-k8s .
Sending build context to Docker daemon  23.55kB
Step 1/6 : FROM python:3
 ---> d47898c6f4b0
Step 2/6 : WORKDIR /usr/src/app
 ---> Using cache
 ---> 2b8b5631a0c1
Step 3/6 : COPY . ./
 ---> b1581f98d046
Step 4/6 : RUN pip install --no-cache-dir -r requirements.txt
 ---> Running in 7e21901b2b74
Requirement already satisfied: pip==20.0.2 in /usr/local/lib/python3.8/site-packages (from -r requirements.txt (line 1)) (20.0.2)
Collecting Flask==1.1.1
  Downloading Flask-1.1.1-py2.py3-none-any.whl (94 kB)
Collecting boto3==1.11.12
  Downloading boto3-1.11.12-py2.py3-none-any.whl (128 kB)
Collecting Pillow==7.0.0
  Downloading Pillow-7.0.0-cp38-cp38-manylinux1_x86_64.whl (2.1 MB)
Collecting pymongo==3.10.1
  Downloading pymongo-3.10.1-cp38-cp38-manylinux2014_x86_64.whl (480 kB)
Collecting Werkzeug==0.16.1
  Downloading Werkzeug-0.16.1-py2.py3-none-any.whl (327 kB)
Collecting click>=5.1
  Downloading click-7.1.2-py2.py3-none-any.whl (82 kB)
Collecting itsdangerous>=0.24
  Downloading itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
Collecting Jinja2>=2.10.1
  Downloading Jinja2-2.11.2-py2.py3-none-any.whl (125 kB)
Collecting s3transfer<0.4.0,>=0.3.0
  Downloading s3transfer-0.3.3-py2.py3-none-any.whl (69 kB)
Collecting botocore<1.15.0,>=1.14.12
  Downloading botocore-1.14.17-py2.py3-none-any.whl (5.9 MB)
Collecting jmespath<1.0.0,>=0.7.1
  Downloading jmespath-0.10.0-py2.py3-none-any.whl (24 kB)
Collecting MarkupSafe>=0.23
  Downloading MarkupSafe-1.1.1-cp38-cp38-manylinux1_x86_64.whl (32 kB)
Collecting docutils<0.16,>=0.10
  Downloading docutils-0.15.2-py3-none-any.whl (547 kB)
Collecting python-dateutil<3.0.0,>=2.1
  Downloading python_dateutil-2.8.1-py2.py3-none-any.whl (227 kB)
Collecting urllib3<1.26,>=1.20; python_version != "3.4"
  Downloading urllib3-1.25.9-py2.py3-none-any.whl (126 kB)
Collecting six>=1.5
  Downloading six-1.15.0-py2.py3-none-any.whl (10 kB)
Installing collected packages: click, itsdangerous, Werkzeug, MarkupSafe, Jinja2, Flask, docutils, six, python-dateutil, jmespath, urllib3, botocore, s3transfer, boto3, Pillow, pymongo
Successfully installed Flask-1.1.1 Jinja2-2.11.2 MarkupSafe-1.1.1 Pillow-7.0.0 Werkzeug-0.16.1 boto3-1.11.12 botocore-1.14.17 click-7.1.2 docutils-0.15.2 itsdangerous-1.1.0 jmespath-0.10.0 pymongo-3.10.1 python-dateutil-2.8.1 s3transfer-
0.3.3 six-1.15.0 urllib3-1.25.9
WARNING: You are using pip version 20.0.2; however, version 20.1.1 is available.
You should consider upgrading via the '/usr/local/bin/python -m pip install --upgrade pip' command.
Removing intermediate container 7e21901b2b74
 ---> e880c4ea49ea
Step 5/6 : EXPOSE 6003
 ---> Running in 379fd87259e6
Removing intermediate container 379fd87259e6
 ---> 953d63748ead
Step 6/6 : CMD ["python", "app.py"]
 ---> Running in bcf4bd99eab5
Removing intermediate container bcf4bd99eab5
 ---> 2f4322879e51
Successfully built 2f4322879e51
Successfully tagged cloudgeek007/photo-album-k8s:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permission
s for sensitive files and directories.
```

> docker images



```
C:\Users\negij\Documents\GitHub Repos\Piper2020\Day 2\Lab 03 - Part 2 - Deploying to Kubernetes>docker images
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
cloudgeek007/photo-album-k8s        latest              ef21766cb3cb        About an hour ago   1GB
```

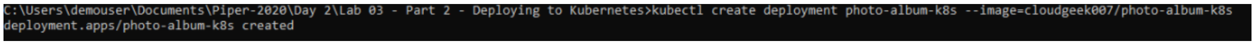
> docker push {dockerhubID}/photo-album-k8s



```
C:\Users\negij\Documents\GitHub Repos\Piper2020\Day 2\Lab 03 - Part 2 - Deploying to Kubernetes>docker push cloudgeek007/photo-album-k8s
The push refers to repository [docker.io/cloudgeek007/photo-album-k8s]
2e085a5841cf: Layer already exists
b9bca98cc134: Layer already exists
3f64199536a3: Layer already exists
fbefc7d9db96: Layer already exists
bd436d37b328: Layer already exists
8b6dde37c5c4: Layer already exists
3dffd131f01f: Layer already exists
271910c4c150: Layer already exists
6670e930ed33: Layer already exists
c7f27a4eb870: Layer already exists
e70dfb4c3a48: Layer already exists
1c76bd0dc325: Layer already exists
latest: digest: sha256:3a7cd5a6ccded420b8dbed85201ab61e6a1f325515df4d6ea489a4a00e3df001 size: 2844
```

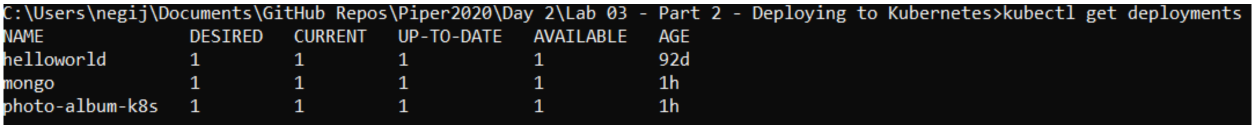
10. Deploy your image to the Kubernetes cluster and expose port 6003 for local access

> kubectl create deployment photo-album-k8s --image={dockerhub username}/photo-album-k8s



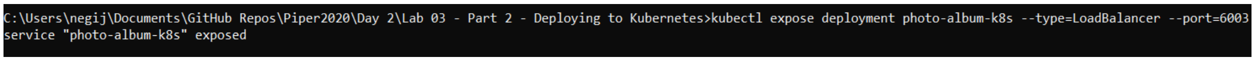
```
C:\Users\demouser\Documents\Piper-2020\Day 2\Lab 03 - Part 2 - Deploying to Kubernetes>kubectl create deployment photo-album-k8s --image=cloudgeek007/photo-album-k8s
deployment.apps/photo-album-k8s created
```

>kubectl get deployment



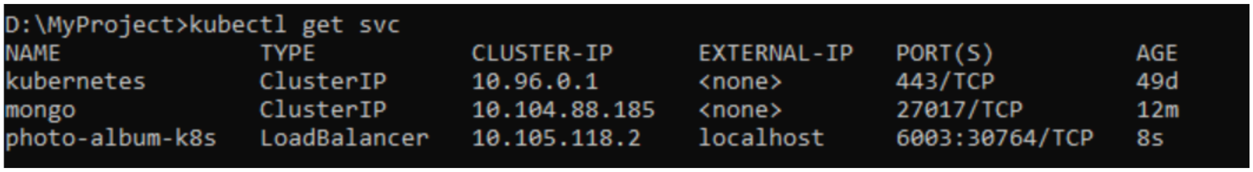
```
C:\Users\negij\Documents\GitHub Repos\Piper2020\Day 2\Lab 03 - Part 2 - Deploying to Kubernetes>kubectl get deployments
NAME              DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
helloworld        1         1         1            1           92d
mongo             1         1         1            1           1h
photo-album-k8s   1         1         1            1           1h
```

> kubectl expose deployment photo-album-k8s --type=LoadBalancer --port=6003



```
C:\Users\negij\Documents\GitHub Repos\Piper2020\Day 2\Lab 03 - Part 2 - Deploying to Kubernetes>kubectl expose deployment photo-album-k8s --type=LoadBalancer --port=6003
service "photo-album-k8s" exposed
```

> kubectl get service



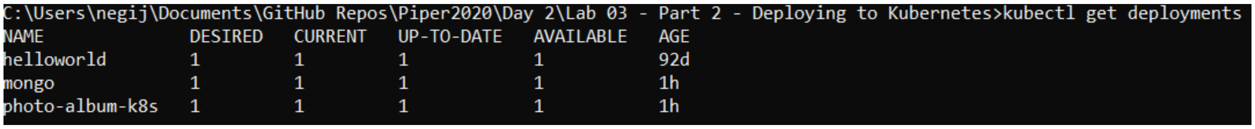
```
D:\MyProject>kubectl get svc
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes        ClusterIP      10.96.0.1       <none>        443/TCP          49d
mongo             ClusterIP      10.104.88.185   <none>        27017/TCP        12m
photo-album-k8s   LoadBalancer   10.105.118.2    localhost     6003:30764/TCP   8s
```

11. Open a web browser and go to the following URL: http://localhost:6003

## Retrospective: The results

□ Deployed a 2-tier app to Kubernetes

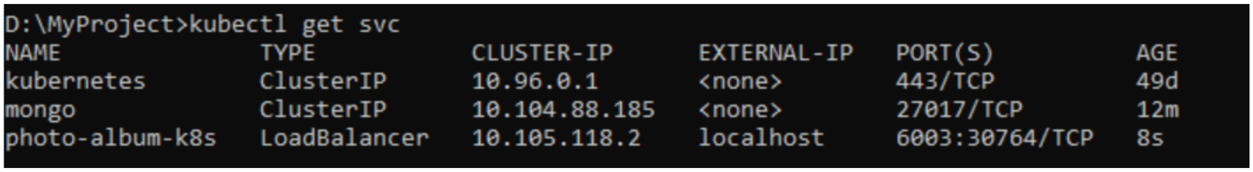
If you successfully deployed mongodb and photo-album deployments, you should see the below result in the command prompt



```
C:\Users\negij\Documents\GitHub Repos\Piper2020\Day 2\Lab 03 - Part 2 - Deploying to Kubernetes>kubectl get deployments
NAME              DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
helloworld        1         1         1            1           92d
mongo             1         1         1            1           1h
photo-album-k8s   1         1         1            1           1h
```

□ Exposed the service port to access application

If you successfully exposed the service port to access application photo-album-k8s on stated port, you should see the similar result in the command prompt



```
D:\MyProject>kubectl get svc
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes        ClusterIP      10.96.0.1       <none>        443/TCP          49d
mongo             ClusterIP      10.104.88.185   <none>        27017/TCP        12m
photo-album-k8s   LoadBalancer   10.105.118.2    localhost     6003:30764/TCP   8s
```

□ Tested the web app

If you are running photo-album-k8s deployment on stated port successfully, you should see the below results on the web browser or CLI

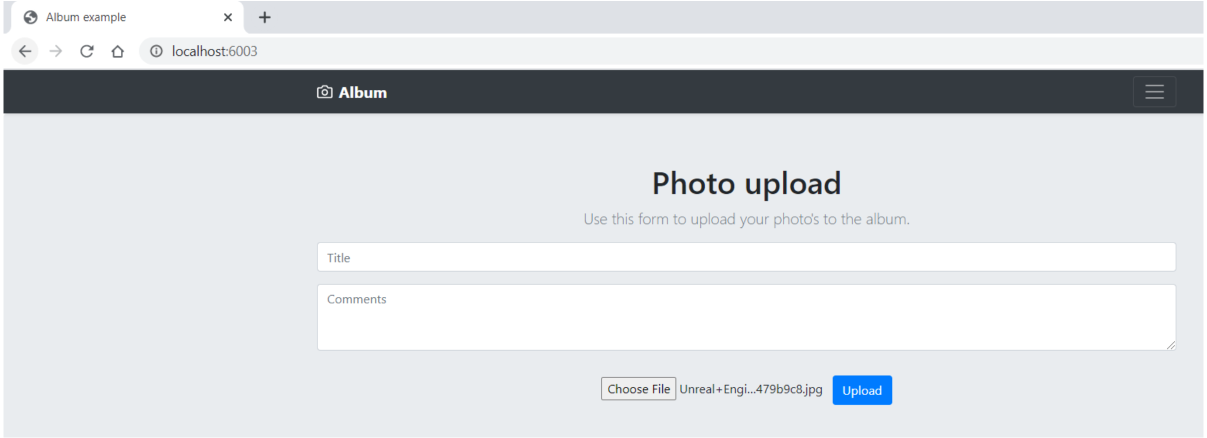
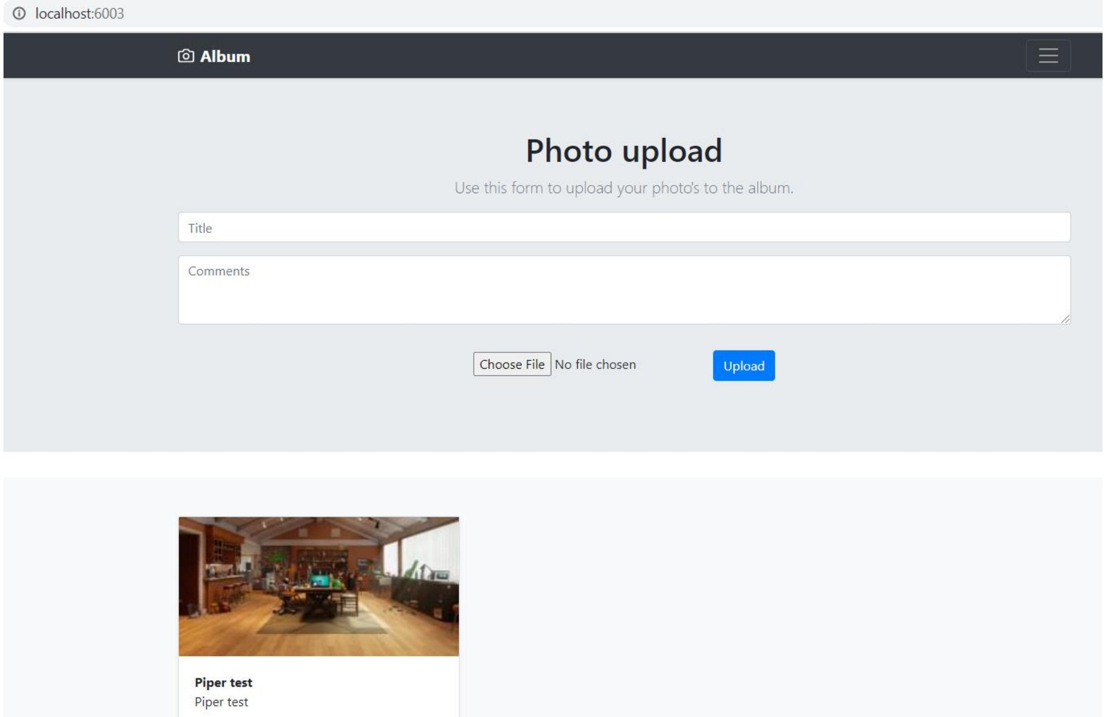


Photo upload

Use this form to upload your photo's to the album.

If you upload any photo, you should see similar results as below on the web browser



localhost:6003

Album

Photo upload

Use this form to upload your photo's to the album.

Title

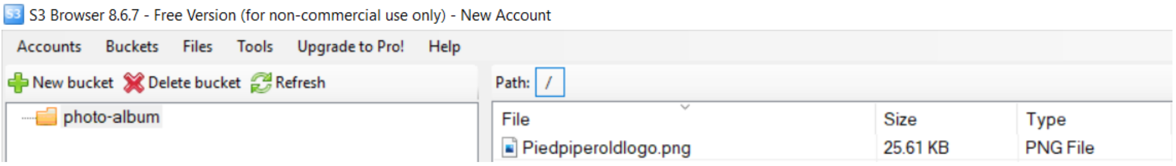
Comments

Choose File No file chosen

Upload

**Piper test**
Piper test

The photo image being uploaded can be verified using S3 browser



S3 Browser 8.6.7 - Free Version (for non-commercial use only) - New Account

| File | Size | Type |
|---|---|---|
| Piedpiperoldlogo.png | 25.61 KB | PNG File |