Pied Piper Technical Workshop

Day 1: Cloud Native Applications

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

DELL Technologies

Revisions

Version	Date	Description
0.1	May 2020	Initial draft

Table of Contents

Lab 1: Github

Module Objectives:

- ❑ Learn about git, git tools, Github and Github Desktop GUI
- ❑ Setup a new personal repo for our project
- ❑ Use git tools to learn the skills to commit code
- ❑ Upload a file to Github

DELL Technologies

What is Git?

- *Distributed* version control system for tracking changes in ANY set of files
  - Created by Linus Torvalds in 2005 for Linux kernel development
- Important to note that Git is an open standard and licensed under the GNU GPL v2
- Here are a few of the popular SaaS based implementations;
  - Github
  - Bitbucket
  - AWS Code Commit
  - Gitlab
  - Etc….

How to use .gitignore to avoid committing sensitive data

- Posting passwords to services like AWS costs **$$$$**
- Best Practice: When publishing code use **.gitignore** to exclude any files containing sensitive data
- Below is an example of using an external file containing credentials and using python code to reference the variables rather than embedding the username and password directly into the main.py application code. You can then use the .gitignore file to exclude the setenv.py file that contains sensitive data.
- This approach is more secure than embedding credentials into application code, but is still not 12-factor compliant and can easily be mistakenly uploaded to a git repository or hosting provider with your source code.

setenv.py

```
user = "joeblogs"
password = "whatever"
```

main.py

```
from setenv import user,
password

print "user is : " + user
print "pass is : " + password
```

.gitignore

```
setenv.py
```

How to use environment variables to store and retrieve sensitive data

- Environment variables are considered ‘best practice’ when working with sensitive data such as user credentials and environment configuration. See ‘The Twelve-Factor App: III. Config’
- If you have a python-based application and want to manage your environment using a separate script that can be ran before the application is deployed, you might have a setup file similar to the below and exclude that file from being uploaded to Github and Tanzu Application Service.

setenv.py

```
import os
os.environ["user"]="joeblogs"
os.environ["pass"]="whatever"
```

.gitignore and .cfignore

```
setenv.py
```

- The main application would retrieve the credentials from the environment as shown below;

main.py

```
import os

user = os.environ.get('user’)
pass = os.environ.get('pass')
```

How to use Github and Github desktop

- If you do not know how to use Github, please complete the following tutorial

  https://guides.github.com/activities/hello-world/

- If you do not know how to use Github Desktop GUI client, please complete the following tutorial

  https://help.github.com/desktop/guides/getting-started-with-github-desktop/

DELL Technologies

Lab Exercise: Create a new repo and publish a file with your name

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

1. Create a new private repo on Github using your personal account
2. Create a text file with your name as the filename
3. Add the content of the text file with the words “Hello Piper class of 2020!”
4. Commit your file to the above repo using the Github Desktop GUI tool and publish it to Github

DELL Technologies

Lab 2: Deploy an App to PWS

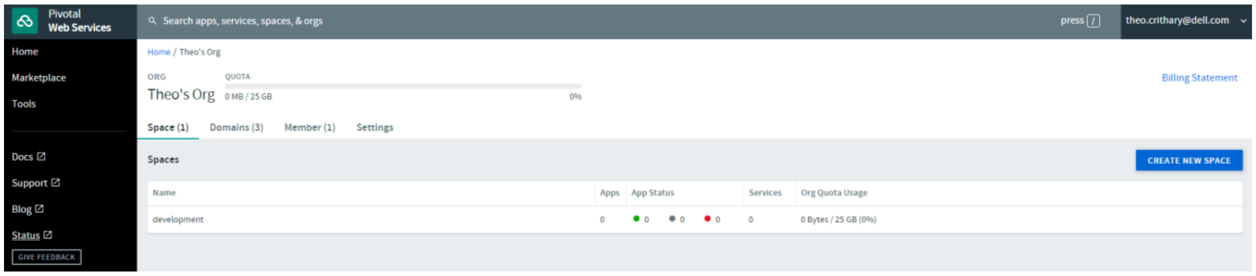
Module Objectives:

- ❑ Learn about PWS & cf cli tools
- ❑ Confirm cf cli is installed and working
- ❑ Login to PWS via cf cli
- ❑ Push a demo app to PWS and confirm we have the skills and tools required to publish a web app to the PWS PaaS platform

DELL Technologies

What is Pivotal Web Services (PWS)?

- PWS is a hosted environment of Tanzu Application Service. PWS is hosted on AWS in the US-East region. PWS utilizes two availability zones for redundancy.
- Your free trial org includes 2GB of memory quota and $87 of Trial credit. A trial org expires after one year or when you have used up all of your Trial credits.



ORG: Theo's Org
QUOTA: 0 MB / 25 GB 0%

Spaces

Name	Apps	App Status	Services	Org Quota Usage
development	0	● 0 ● 0 ● 0	0	0 Bytes / 25 GB (0%)

Important note: Please create an organisation before continuing with the lab

DELL Technologies

Lab Exercise: Clone an example app and deploy to PWS

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

1. Clone the following repository;

   https://github.com/cloudfoundry-samples/cf-sample-app-spring.git

2. Open a command prompt and navigate to the directory cloned above

   > cd <path of cloned repo's>\cf-sample-app-spring

3. Login to PWS with your account creds

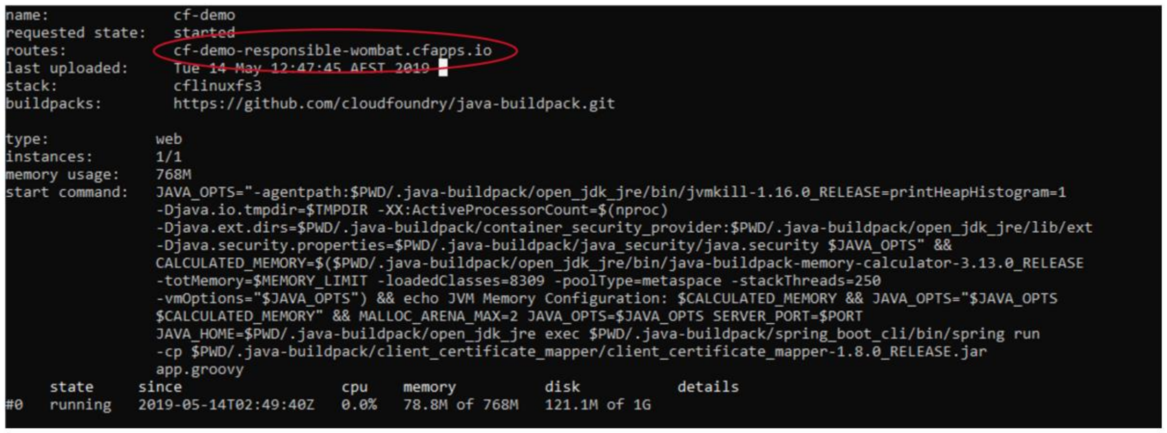
   > cf login -a https://api.run.pivotal.io

4. Push the app to PWS

   > cf push

5. Test app by opening in browser

   Use the URL shown in the 'routes:' section of the output displayed in your command prompt.



```
name:              cf-demo
requested state:   started
routes:            cf-demo-responsible-wombat.cfapps.io
last uploaded:     Tue 14 May 12:47:45 AEST 2019
stack:             cflinuxfs3
buildpacks:        https://github.com/cloudfoundry/java-buildpack.git

type:            web
instances:       1/1
memory usage:    768M
start command:   JAVA_OPTS="-agentpath:$PWD/.java-buildpack/open_jdk_jre/bin/jvmkill-1.16.0_RELEASE=printHeapHistogram=1
                 -Djava.io.tmpdir=$TMPDIR -XX:ActiveProcessorCount=$(nproc)
                 -Djava.ext.dirs=$PWD/.java-buildpack/container_security_provider:$PWD/.java-buildpack/open_jdk_jre/lib/ext
                 -Djava.security.properties=$PWD/.java-buildpack/java_security/java.security $JAVA_OPTS" &&
                 CALCULATED_MEMORY=$($PWD/.java-buildpack/open_jdk_jre/bin/java-buildpack-memory-calculator-3.13.0_RELEASE
                 -totMemory=$MEMORY_LIMIT -loadedClasses=8309 -poolType=metaspace -stackThreads=250
                 -vmOptions="$JAVA_OPTS") && echo JVM Memory Configuration: $CALCULATED_MEMORY && JAVA_OPTS="$JAVA_OPTS
                 $CALCULATED_MEMORY" && MALLOC_ARENA_MAX=2 JAVA_OPTS=$JAVA_OPTS SERVER_PORT=$PORT
                 JAVA_HOME=$PWD/.java-buildpack/open_jdk_jre exec $PWD/.java-buildpack/spring_boot_cli/bin/spring run
                 -cp $PWD/.java-buildpack/client_certificate_mapper/client_certificate_mapper-1.8.0_RELEASE.jar
                 app.groovy
     state     since                  cpu    memory         disk           details
#0   running   2019-05-14T02:49:40Z   0.0%   78.8M of 768M   121.1M of 1G
```

   e.g. https://cf-demo-responsible-wombat.cfapps.io (your URL will be different)

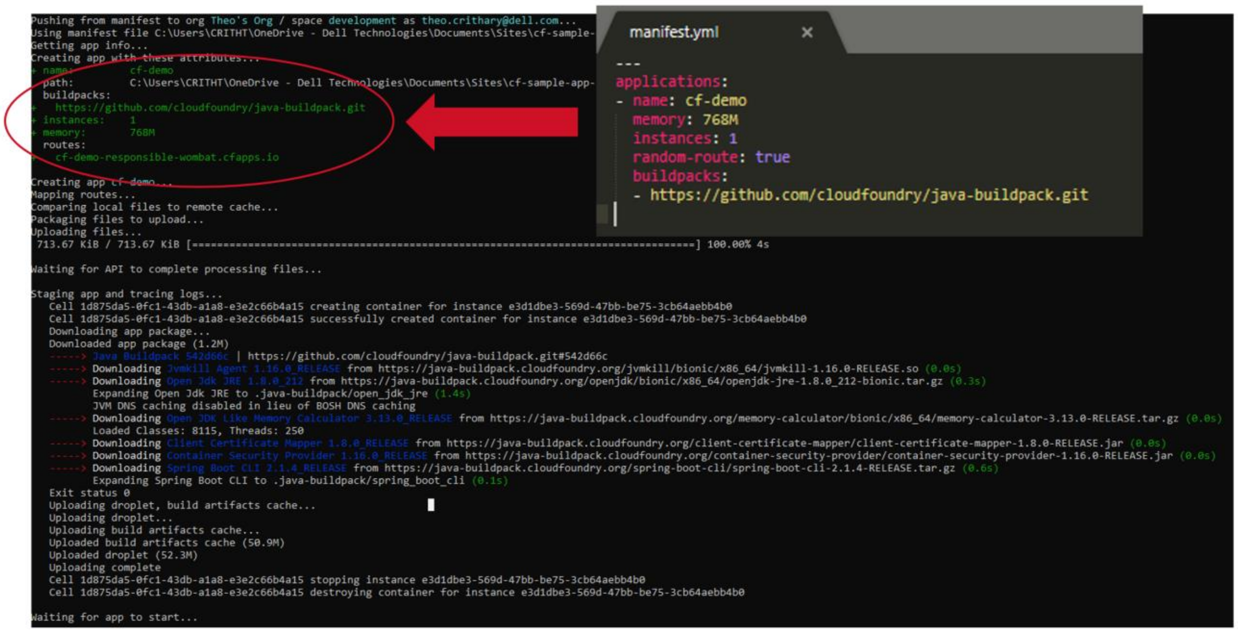
Reflect on what just happened

Take a look at the application name, number of instances, memory allocation, routes and build-pack details.

These are all provided by the **manifest.yml** file as application configuration options.

Random-route: true is a Cloud Foundry function that randomly generates a unique DNS based on selecting 2 words from a provided dictionary. This is used only in testing to ensure that DNS routes are not conflicting when deploying the same application many times.

buildpacks: are used to define the required libraries and runtimes to setup an environment for the application to run successfully. If you do not define a buildpack, Cloud Foundry will try to determine a default buildpack based on the language of the code provided.



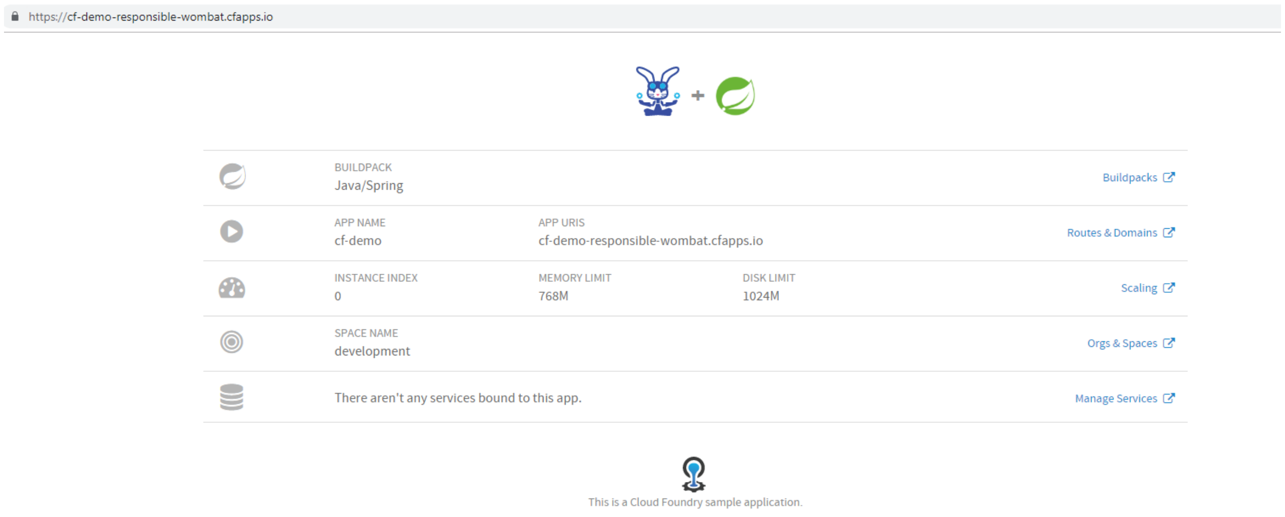
```
Pushing from manifest to org Theo's Org / space development as theo.crithary@dell.com...
Using manifest file C:\Users\CRITHT\OneDrive - Dell Technologies\Documents\Sites\cf-sample-
Getting app info...
Creating app with these attributes...
+ name:         cf-demo
  path:         C:\Users\CRITHT\OneDrive - Dell Technologies\Documents\Sites\cf-sample-app-
  buildpacks:
+   https://github.com/cloudfoundry/java-buildpack.git
+ instances:    1
+ memory:       768M
  routes:
+   cf-demo-responsible-wombat.cfapps.io

Creating app cf-demo...
Mapping routes...
Comparing local files to remote cache...
Packaging files to upload...
Uploading files...
 713.67 KiB / 713.67 KiB [=================================================================================] 100.00% 4s

Waiting for API to complete processing files...

Staging app and tracing logs...
   Cell 1d875da5-0fc1-43db-a1a8-e3e2c66b4a15 creating container for instance e3d1dbe3-569d-47bb-be75-3cb64aebb4b0
   Cell 1d875da5-0fc1-43db-a1a8-e3e2c66b4a15 successfully created container for instance e3d1dbe3-569d-47bb-be75-3cb64aebb4b0
   Downloading app package...
   Downloaded app package (1.2M)
   -----> Java Buildpack 542d66c | https://github.com/cloudfoundry/java-buildpack.git#542d66c
   -----> Downloading Jvmkill Agent 1.16.0_RELEASE from https://java-buildpack.cloudfoundry.org/jvmkill/bionic/x86_64/jvmkill-1.16.0-RELEASE.so (0.0s)
   -----> Downloading Open Jdk JRE 1.8.0_212 from https://java-buildpack.cloudfoundry.org/openjdk/bionic/x86_64/openjdk-jre-1.8.0_212-bionic.tar.gz (0.3s)
          Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.4s)
          JVM DNS caching disabled in lieu of BOSH DNS caching
   -----> Downloading Open JDK Like Memory Calculator 3.13.0_RELEASE from https://java-buildpack.cloudfoundry.org/memory-calculator/bionic/x86_64/memory-calculator-3.13.0-RELEASE.tar.gz (0.0s)
          Loaded Classes: 8115, Threads: 250
   -----> Downloading Client Certificate Mapper 1.8.0_RELEASE from https://java-buildpack.cloudfoundry.org/client-certificate-mapper/client-certificate-mapper-1.8.0-RELEASE.jar (0.0s)
   -----> Downloading Container Security Provider 1.16.0_RELEASE from https://java-buildpack.cloudfoundry.org/container-security-provider/container-security-provider-1.16.0-RELEASE.jar (0.0s)
   -----> Downloading Spring Boot CLI 2.1.4_RELEASE from https://java-buildpack.cloudfoundry.org/spring-boot-cli/spring-boot-cli-2.1.4-RELEASE.tar.gz (0.6s)
          Expanding Spring Boot CLI to .java-buildpack/spring_boot_cli (0.1s)
   Exit status 0
   Uploading droplet, build artifacts cache...
   Uploading droplet...
   Uploading build artifacts cache...
   Uploaded build artifacts cache (50.9M)
   Uploaded droplet (52.3M)
   Uploading complete
   Cell 1d875da5-0fc1-43db-a1a8-e3e2c66b4a15 stopping instance e3d1dbe3-569d-47bb-be75-3cb64aebb4b0
   Cell 1d875da5-0fc1-43db-a1a8-e3e2c66b4a15 destroying container for instance e3d1dbe3-569d-47bb-be75-3cb64aebb4b0

Waiting for app to start...
```

manifest.yml

```
---
applications:
- name: cf-demo
  memory: 768M
  instances: 1
  random-route: true
  buildpacks:
  - https://github.com/cloudfoundry/java-buildpack.git
```

Retrospective: The results

If your app was successfully deployed and you were able to open the correct URL in a web browser, you should see the below webpage;



https://cf-demo-responsible-wombat.cfapps.io

BUILDPACK
Java/Spring
Buildpacks

APP NAME
cf-demo
APP URIS
cf-demo-responsible-wombat.cfapps.io
Routes & Domains

INSTANCE INDEX
0
MEMORY LIMIT
768M
DISK LIMIT
1024M
Scaling

SPACE NAME
development
Orgs & Spaces

There aren't any services bound to this app.
Manage Services

This is a Cloud Foundry sample application.

Lab 3: Create a Static HTML Website

Module Objectives:

- ❑ Learn a little bit about HTML & CSS
- ❑ Create a basic website that will be used as a starting point for our application
- ❑ Review the code and become familiar with the layout and design
- ❑ Test the functionality of the default website
- ❑ Save and upload to Github

DELL Technologies

Lab Exercise: Copy the static webpage provided and setup a new project

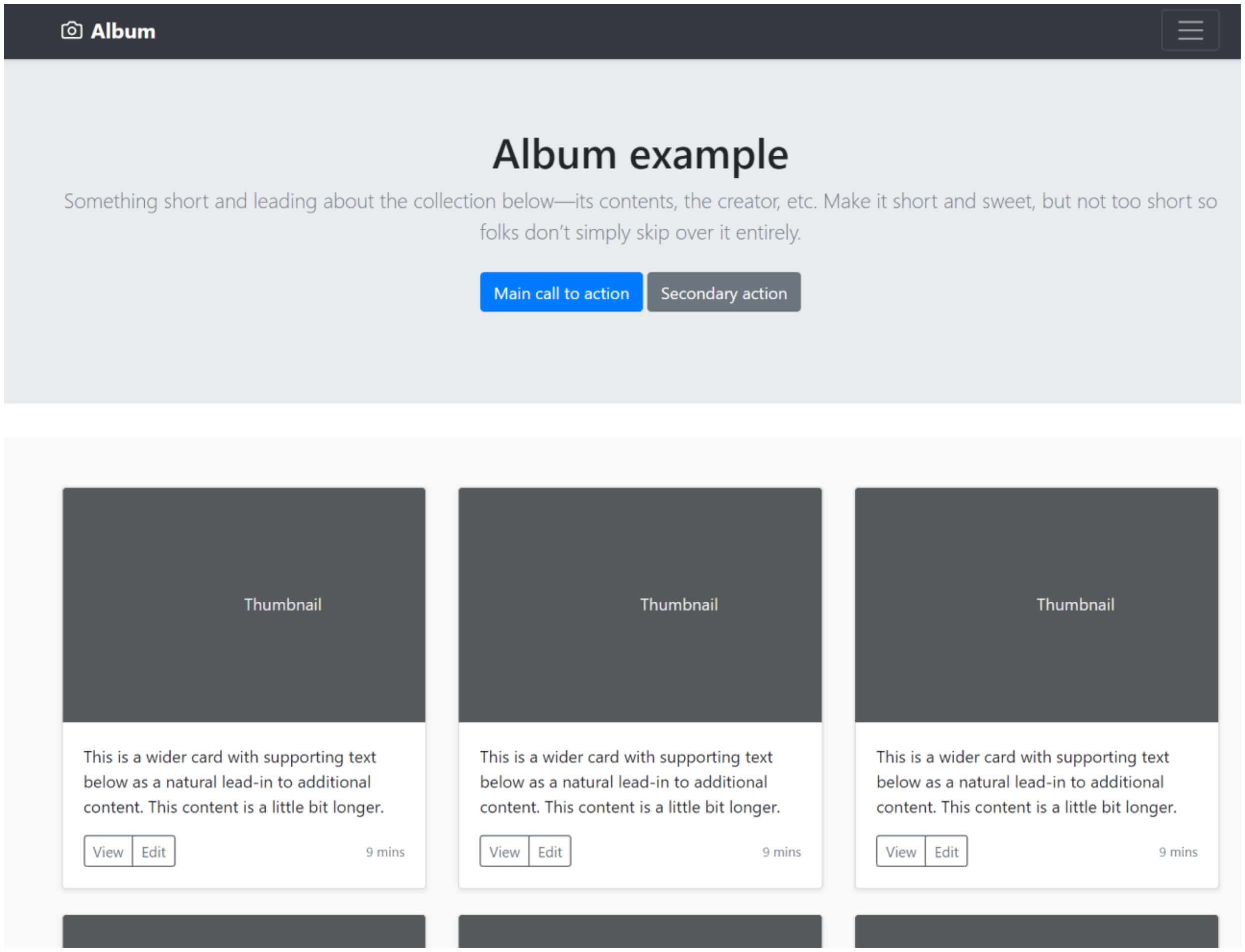
1. Create a new repo on GitHub to store your project – call it whatever you want
2. Clone the following repo and copy the file into your new working folder as created above

   https://github.com/theocrithary/Piper-2020/tree/master/Day%201/Lab%2003%20-%20CSS
3. Open the 'default.html' file in a browser to test its 'look and feel'
4. Open the 'default.html' file in the Atom text editor provided and review the code
5. Commit the file to your personal repo and push to GitHub

DELL Technologies

Retrospective: A basis for an MVP

You now have a HTML web page that can be used as a template design for our new application.


Album
Album example
Something short and leading about the collection below—its contents, the creator, etc. Make it short and sweet, but not too short so folks don't simply skip over it entirely.
Main call to action
Secondary action
Thumbnail
This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.
View
Edit
9 mins

If you opened the default.html file in the Atom text editor, you might have noticed the below lines of code in the <head> section at the top;

```html
<!-- Add support for mobile devices -->
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

```html
<!-- Bootstrap core CSS -->
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
```

These are the Bootstrap responsive design and external CSS functionality that provides the look and feel of the webpage as discussed in the theory module earlier.

Try commenting out these lines of code and reload the page in the browser to see what happens.

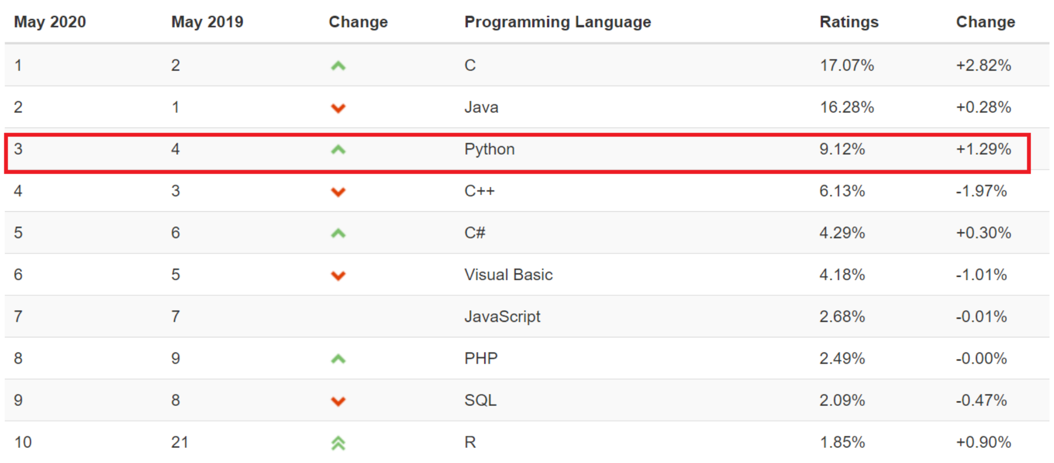
Lab 4: Python and Flask

Module Objectives:

- ❑ Learn about Python & Flask
- ❑ Write a Python app using Flask web server to run the ‘default.html’ website we created earlier
- ❑ Review the code and become familiar with the Python code
- ❑ Test the functionality of the web app
- ❑ Save and upload to Github

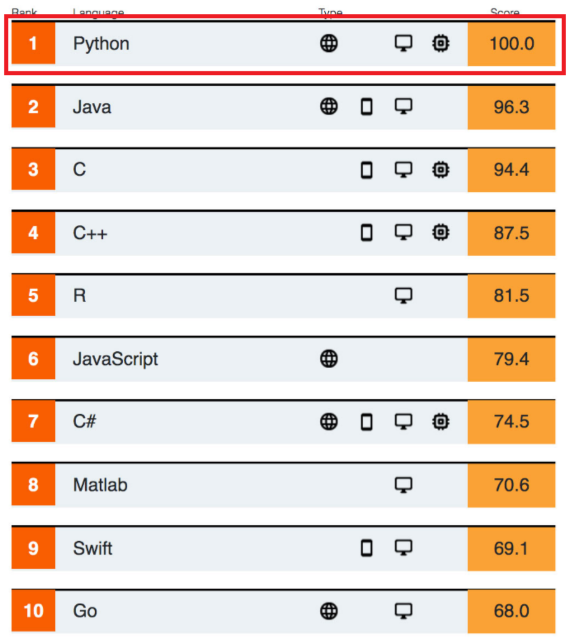
First a bit about Python & Flask

- Python is consistently in the top 4 languages YoY
- Python is embedded in many IAC tools (e.g. Ansible)
- Python is easy to implement with Raspberry Pi based IoT projects
- Python is becoming a leader in AI/ML applications
- Python is rich in community support
- Flask is a lightweight web server that easily runs with Python



May 2020	May 2019	Change	Programming Language	Ratings	Change
1	2	▲	C	17.07%	+2.82%
2	1	▼	Java	16.28%	+0.28%
3	4	▲	Python	9.12%	+1.29%
4	3	▼	C++	6.13%	-1.97%
5	6	▲	C#	4.29%	+0.30%
6	5	▼	Visual Basic	4.18%	-1.01%
7	7		JavaScript	2.68%	-0.01%
8	9	▲	PHP	2.49%	-0.00%
9	8	▼	SQL	2.09%	-0.47%
10	21	▲▲	R	1.85%	+0.90%

Source: https://www.tiobe.com/tiobe-index/



Rank	Language	Type	Score
1	Python	🌐 🖥 ⚙	100.0
2	Java	🌐 📱 🖥	96.3
3	C	📱 🖥 ⚙	94.4
4	C++	📱 🖥 ⚙	87.5
5	R	🖥	81.5
6	JavaScript	🌐	79.4
7	C#	🌐 📱 🖥 ⚙	74.5
8	Matlab	🖥	70.6
9	Swift	📱 🖥	69.1
10	Go	🌐 🖥	68.0

Source: https://spectrum.ieee.org/computing/software/the-top-programming-languages-2019

What's changed in Python 3?

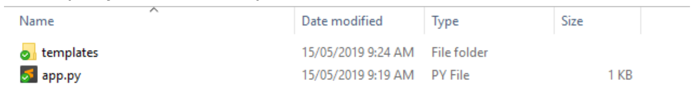
Today's Python community largely supports Python 3 code and it is rare to find new code written in Python 2 anymore. However, if you do come across older code written in Python 2, it is important to understand the changes and what is required to port the code to Python 3.

- Print function
- Integer division
- Strings have changed
- Xrange is deprecated
- Error and exception handling has changed
- Banker's rounding

Python 2	**Python 3**
print 'Hello, World!'	print ('Hello, World!')
print 3 / 2	print (3 / 2)
1	1.5
Strings are ASCII	Strings are Unicode (utf-8)
	NameError: not defined
raise IOError, "file error"	raise IOError("file error")
except NameError, err:	except NameError **as** err:
round(16.5)	round(16.5)
17.0	16

Lab Exercise: Create a basic Python app to run the Flask web server

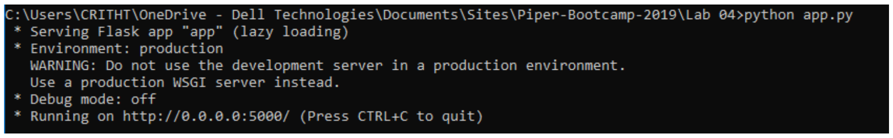
1. Create a new directory in your project folder called 'templates'
2. Move your static HTML page created in the previous exercise to this new folder
3. Download the files from Lab 04 – Python & Flask directory of the repository and place into your working project folder
4. Your project directory should look like this;



Name	Date modified	Type	Size
templates	15/05/2019 9:24 AM	File folder	
app.py	15/05/2019 9:19 AM	PY File	1 KB

5. Open a command prompt and navigate to your project directory
6. Run the following command

   > python app.py



```
C:\Users\CRITHT\OneDrive - Dell Technologies\Documents\Sites\Piper-Bootcamp-2019\Lab 04>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

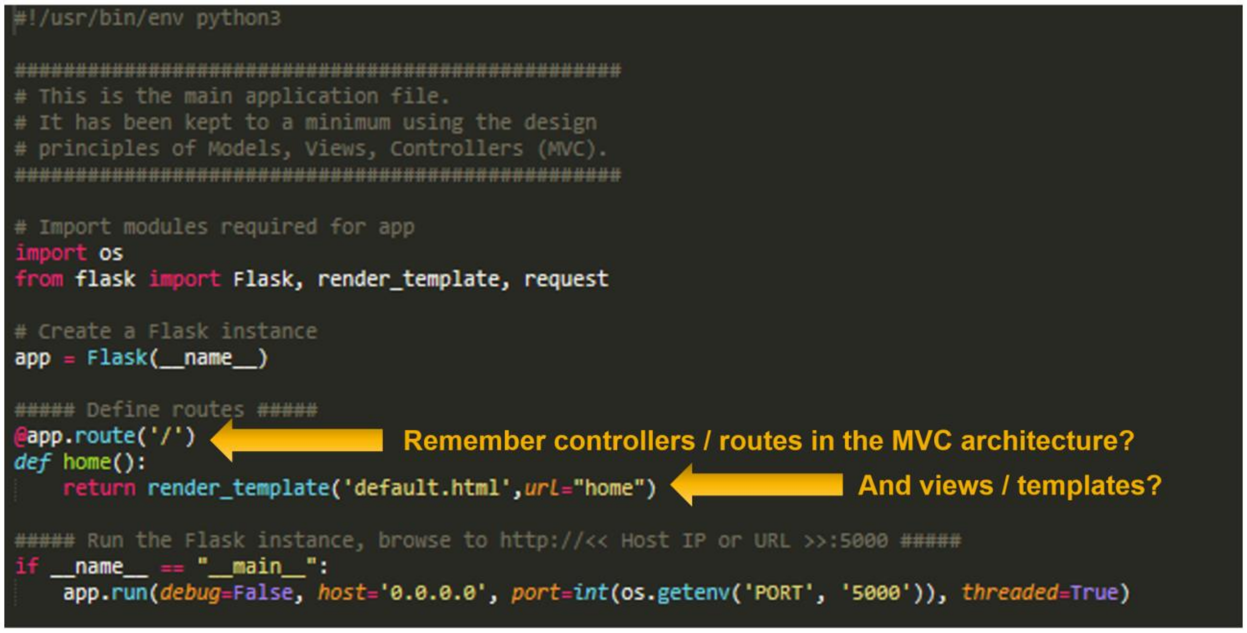
7. Open a browser with the following URL

   http://localhost:5000

DELL Technologies

Retrospective: Applying concepts of Model, Views, Controllers

Review the app.py code to see how views and controllers are used to implement a template with a default route controller;



```
#!/usr/bin/env python3

####################################################
# This is the main application file.
# It has been kept to a minimum using the design
# principles of Models, Views, Controllers (MVC).
####################################################

# Import modules required for app
import os
from flask import Flask, render_template, request

# Create a Flask instance
app = Flask(__name__)

##### Define routes #####
@app.route('/')
def home():
    return render_template('default.html',url="home")

##### Run the Flask instance, browse to http://<< Host IP or URL >>:5000 #####
if __name__ == "__main__":
    app.run(debug=False, host='0.0.0.0', port=int(os.getenv('PORT', '5000')), threaded=True)
```

Also take note of the port on the last line of code that determines which port the web server will listen on. For testing purposes we are using port 5000, but this can easily be changed to port 80 at any time.

Lab 5: MongoDB

Module Objectives:

- ❑ Setup local MongoDB server & install pymongo module (if not already)
- ❑ Use Compass UI to connect to DB and test it's functionality
- ❑ Edit our Python app we created earlier to add dynamic functionality with MongoDB
- ❑ Review the code and become familiar with the new Python code
- ❑ Test the functionality of the web app
- ❑ Save and upload to Github

Lab Exercise: Copy the new code files and overwrite existing application files

1. Copy the following files from Lab 05 - MongoDB into your project directory

   - /templates
     - default.html
     - submit-photo.html
   - app.py
   - models.py

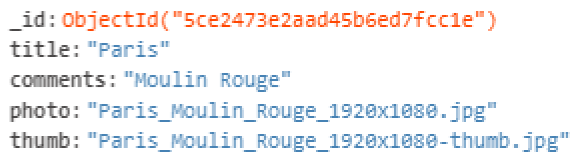
2. Open a command prompt, navigate to your project directory and run the app

   > python app.py

3. Open a browser with the following URL and test uploading an image

   http://localhost:5000

4. Use the MongoDB UI (Compass) to check that the title, comments and filename fields are written to the database collection 'photos'

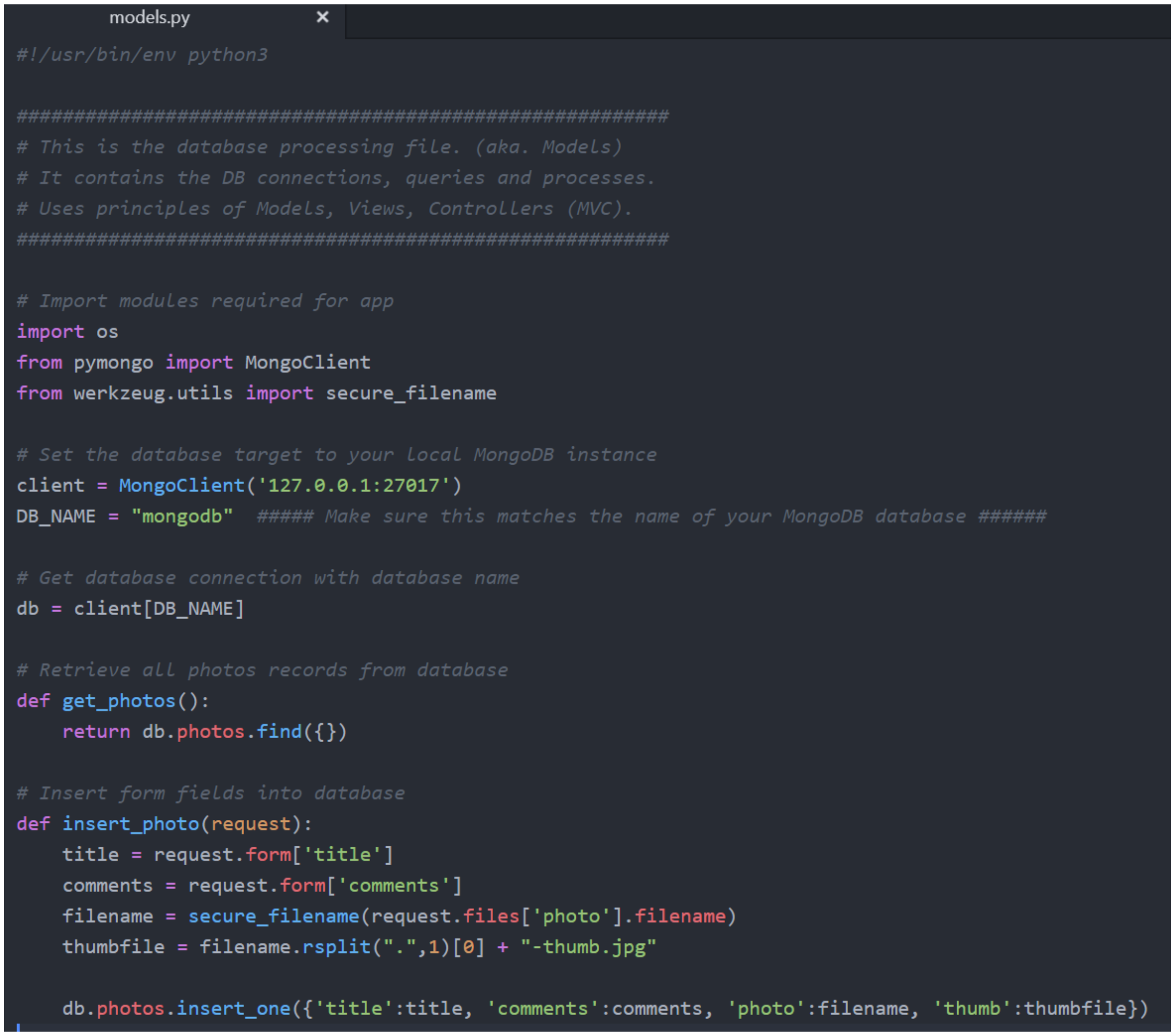


```
_id: ObjectId("5ce2473e2aad45b6ed7fcc1e")
title: "Paris"
comments: "Moulin Rouge"
photo: "Paris_Moulin_Rouge_1920x1080.jpg"
thumb: "Paris_Moulin_Rouge_1920x1080-thumb.jpg"
```

DELL Technologies

Retrospective: Review the implementation of ‘models’ into our application

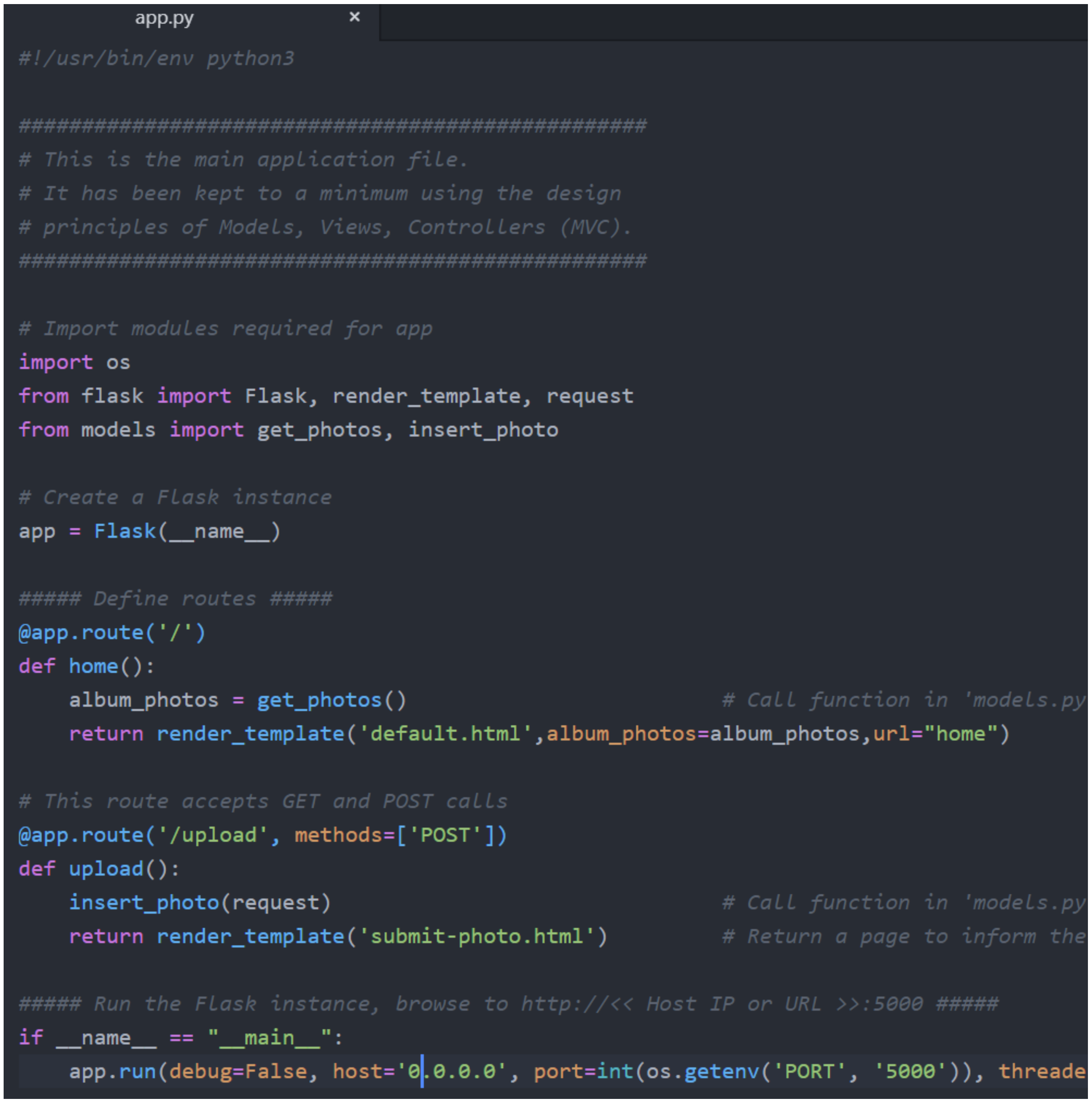
Review the models.py code to understand how the form data is being collected and inserted into the MongoDB database.



models.py ×

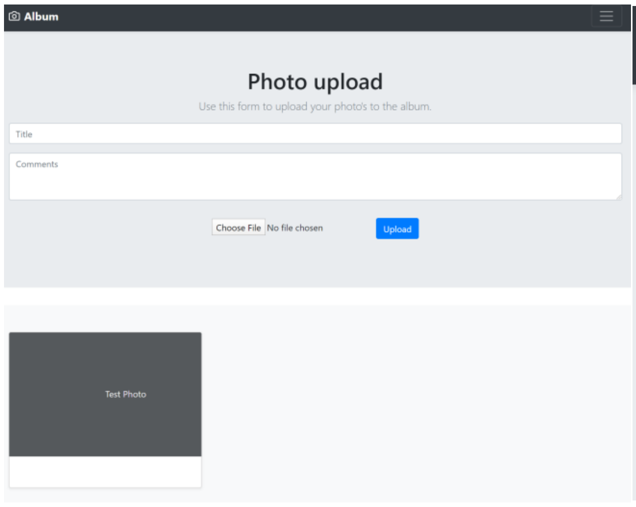
```
#!/usr/bin/env python3

##########################################################
# This is the database processing file. (aka. Models)
# It contains the DB connections, queries and processes.
# Uses principles of Models, Views, Controllers (MVC).
##########################################################

# Import modules required for app
import os
from pymongo import MongoClient
from werkzeug.utils import secure_filename

# Set the database target to your local MongoDB instance
client = MongoClient('127.0.0.1:27017')
DB_NAME = "mongodb"  ##### Make sure this matches the name of your MongoDB database ######

# Get database connection with database name
db = client[DB_NAME]

# Retrieve all photos records from database
def get_photos():
    return db.photos.find({})

# Insert form fields into database
def insert_photo(request):
    title = request.form['title']
    comments = request.form['comments']
    filename = secure_filename(request.files['photo'].filename)
    thumbfile = filename.rsplit(".",1)[0] + "-thumb.jpg"

    db.photos.insert_one({'title':title, 'comments':comments, 'photo':filename, 'thumb':thumbfile})
```

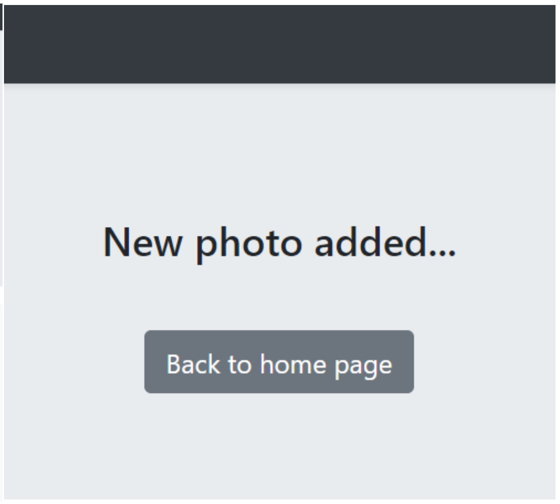
Also take note of the additional route added to the app.py code that provides a new application function to confirm the upload and provide the user with a message before returning to the main page.



app.py ×

```python
#!/usr/bin/env python3

###################################################
# This is the main application file.
# It has been kept to a minimum using the design
# principles of Models, Views, Controllers (MVC).
###################################################

# Import modules required for app
import os
from flask import Flask, render_template, request
from models import get_photos, insert_photo

# Create a Flask instance
app = Flask(__name__)

##### Define routes #####
@app.route('/')
def home():
    album_photos = get_photos()                          # Call function in 'models.py
    return render_template('default.html',album_photos=album_photos,url="home")

# This route accepts GET and POST calls
@app.route('/upload', methods=['POST'])
def upload():
    insert_photo(request)                                # Call function in 'models.py
    return render_template('submit-photo.html')          # Return a page to inform the

##### Run the Flask instance, browse to http://<< Host IP or URL >>:5000 #####
if __name__ == "__main__":
    app.run(debug=False, host='0.0.0.0', port=int(os.getenv('PORT', '5000')), threade
```


Album
Photo upload
Use this form to upload your photo's to the album.
Title
Comments
Choose File No file chosen
Upload
Test Photo



At this point, we still do not have anywhere to store the images and are only inserting the name of the image into the database. This is why you do not see the actual image on the web page yet.

In our next lab we will add some object storage to upload the users images and use the database to store the URL link to the images.

Lab 6: ECS

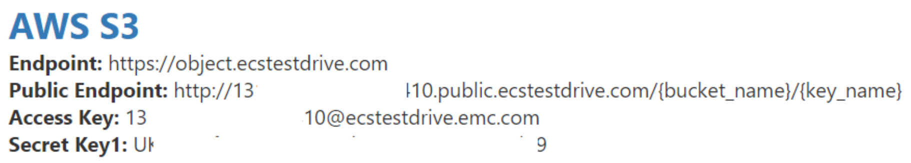
Module Objectives:

- ❏ Setup object storage with ECS test drive
- ❏ Use S3 Browser to connect to ECS and create a bucket
- ❏ Edit our Python app we created earlier to add object storage functionality with S3 compliant Python module
- ❏ Review the code and become familiar with the Python code
- ❏ Test the functionality of the web app
- ❏ Save and upload to Github

DELL Technologies

Lab Exercise:

1. Copy the following files from “Lab 06 - ECS” into your project directory
   - /templates
     - default.html
     - photo.html
     - submit-photo.html
   - app.py
   - models.py
   - config.py (rename the file from the provided config-example.py file)
   - /uploads (create this as a new directory)
   - .gitignore
2. Retrieve your ECS credentials from the following URL; https://portal.ecstestdrive.com/
3. Edit the config.py file with your credentials



AWS S3

Endpoint: https://object.ecstestdrive.com
Public Endpoint: http://13[redacted]10.public.ecstestdrive.com/{bucket_name}/{key_name}
Access Key: 13[redacted]10@ecstestdrive.emc.com
Secret Key1: Uk[redacted]9

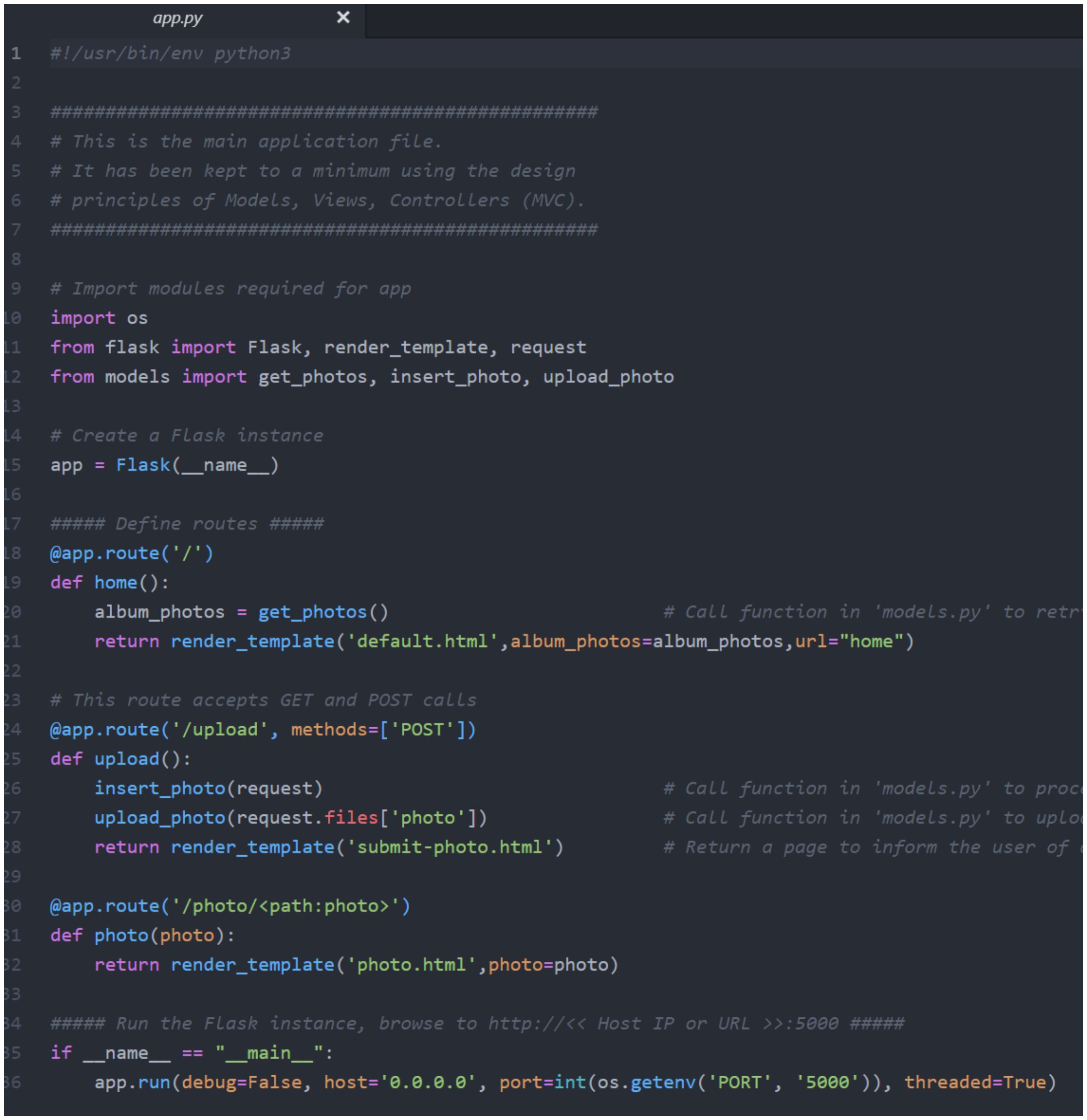
4. Install Boto3 & Pillow (if not already installed)
   > pip install boto3
   > pip install pillow
5. Run your app

   > python app.py
6. Test your app and upload an image;

   http://localhost:5000

Retrospective: Review code and note implementation of Boto3 & Pillow

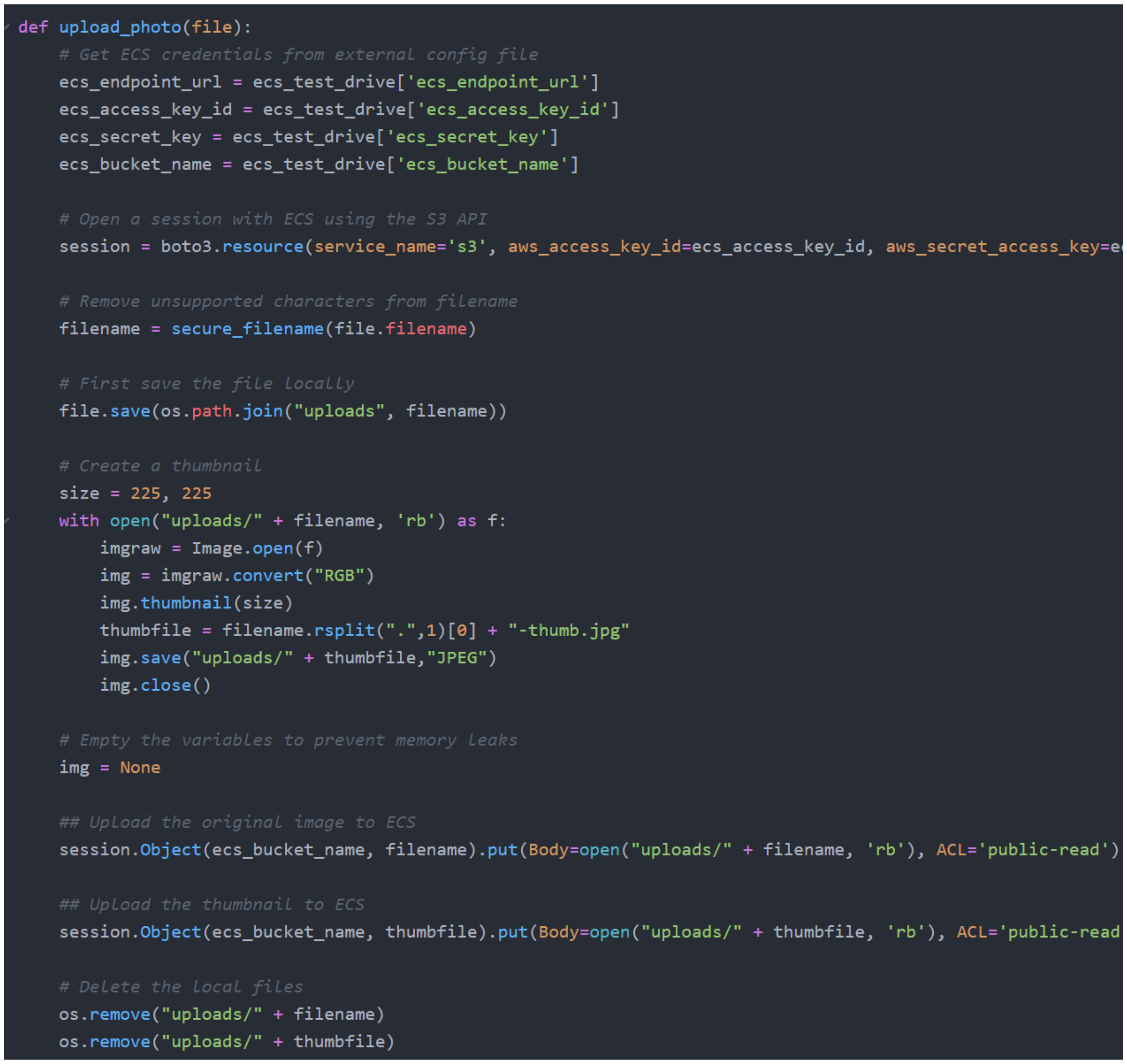
Take a look at the app.py code and notice the additional route used for displaying a full resolution image on a separate page and the additional call to the upload_photo function provided by the models.py file.



app.py

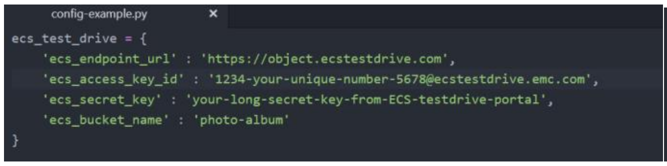
```
#!/usr/bin/env python3

##################################################
# This is the main application file.
# It has been kept to a minimum using the design
# principles of Models, Views, Controllers (MVC).
##################################################

# Import modules required for app
import os
from flask import Flask, render_template, request
from models import get_photos, insert_photo, upload_photo

# Create a Flask instance
app = Flask(__name__)

##### Define routes #####
@app.route('/')
def home():
    album_photos = get_photos()                      # Call function in 'models.py' to retr
    return render_template('default.html',album_photos=album_photos,url="home")

# This route accepts GET and POST calls
@app.route('/upload', methods=['POST'])
def upload():
    insert_photo(request)                            # Call function in 'models.py' to proc
    upload_photo(request.files['photo'])             # Call function in 'models.py' to uplo
    return render_template('submit-photo.html')      # Return a page to inform the user of

@app.route('/photo/<path:photo>')
def photo(photo):
    return render_template('photo.html',photo=photo)

##### Run the Flask instance, browse to http://<< Host IP or URL >>:5000 #####
if __name__ == "__main__":
    app.run(debug=False, host='0.0.0.0', port=int(os.getenv('PORT', '5000')), threaded=True)
```

Open the models.py file and note the additional upload_photos function that uses the Boto3 module to upload the image into ECS test drive using the standards based S3 protocol and the Pillow module that converts the full resolution image into a smaller thumbnail image to be used in the gallery page of our application.

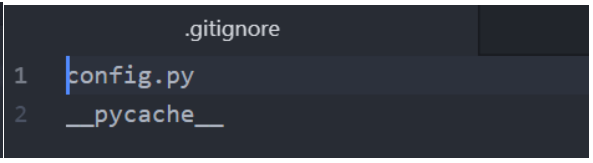


```python
def upload_photo(file):
    # Get ECS credentials from external config file
    ecs_endpoint_url = ecs_test_drive['ecs_endpoint_url']
    ecs_access_key_id = ecs_test_drive['ecs_access_key_id']
    ecs_secret_key = ecs_test_drive['ecs_secret_key']
    ecs_bucket_name = ecs_test_drive['ecs_bucket_name']

    # Open a session with ECS using the S3 API
    session = boto3.resource(service_name='s3', aws_access_key_id=ecs_access_key_id, aws_secret_access_key=e

    # Remove unsupported characters from filename
    filename = secure_filename(file.filename)

    # First save the file locally
    file.save(os.path.join("uploads", filename))

    # Create a thumbnail
    size = 225, 225
    with open("uploads/" + filename, 'rb') as f:
        imgraw = Image.open(f)
        img = imgraw.convert("RGB")
        img.thumbnail(size)
        thumbfile = filename.rsplit(".",1)[0] + "-thumb.jpg"
        img.save("uploads/" + thumbfile,"JPEG")
        img.close()

    # Empty the variables to prevent memory leaks
    img = None

    ## Upload the original image to ECS
    session.Object(ecs_bucket_name, filename).put(Body=open("uploads/" + filename, 'rb'), ACL='public-read')

    ## Upload the thumbnail to ECS
    session.Object(ecs_bucket_name, thumbfile).put(Body=open("uploads/" + thumbfile, 'rb'), ACL='public-read

    # Delete the local files
    os.remove("uploads/" + filename)
    os.remove("uploads/" + thumbfile)
```

The credentials are kept as an external file and the .gitignore file ensures that the credentials are not uploaded accidently to your Github repository.



```python
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'
}
```



```
config.py
__pycache__
```

When you test your new app, it should link your image and display the thumbnail similar to the below;

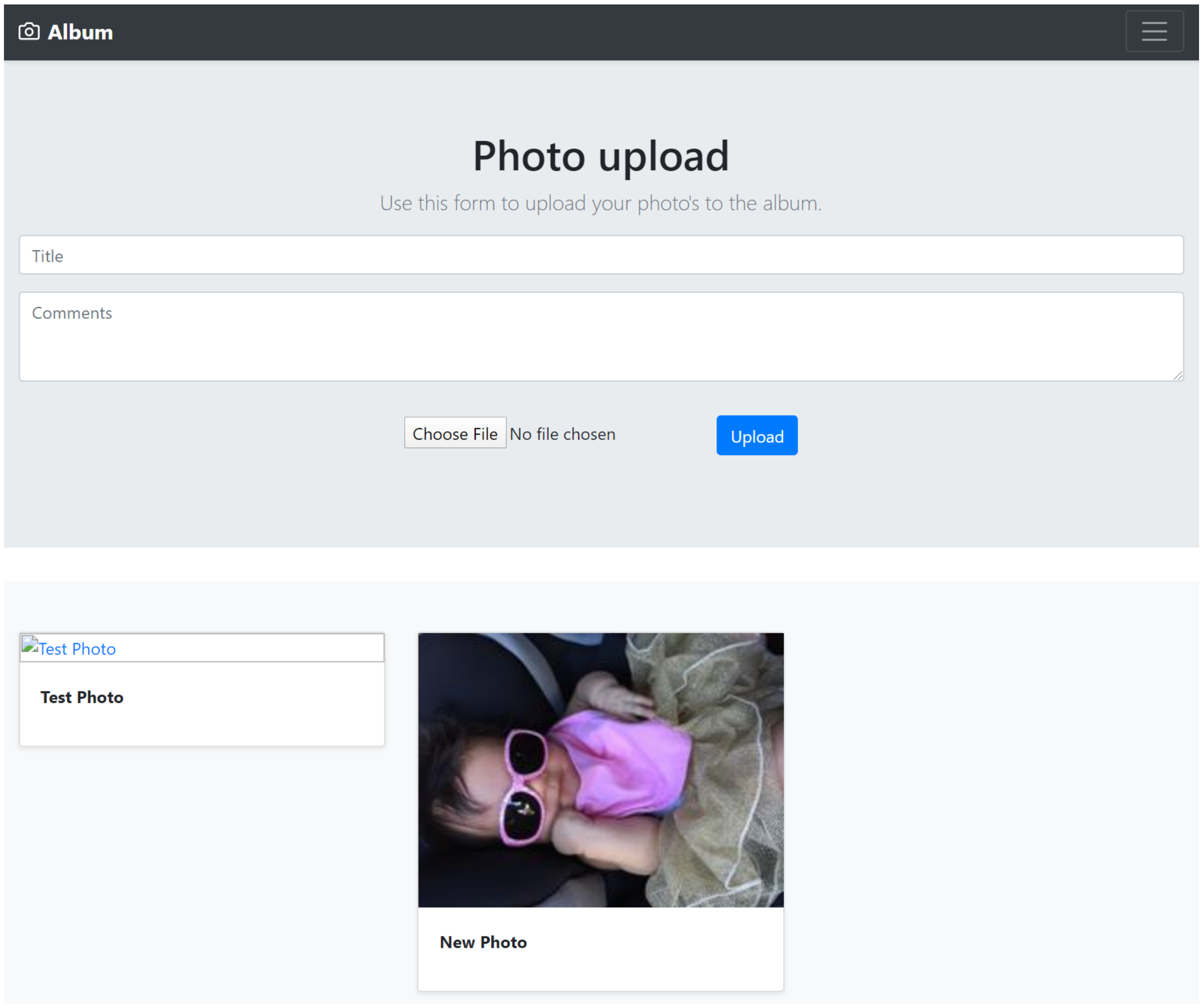


Photo upload

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

Title

Comments

Test Photo

Test Photo

New Photo

Lab 7: PWS & mLab

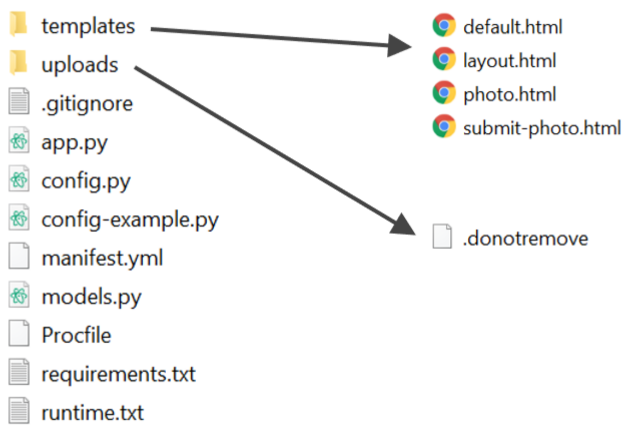
Module Objectives:

- ❑ Upload app to PWS PaaS platform
- ❑ Create a MongoDB as a Service instance
- ❑ Edit our Python app we created earlier to leverage mLab service
- ❑ Review the code and become familiar with the Python code
- ❑ Test the functionality of the web app
- ❑ Save and upload to Github

Lab Exercise (Part A): Create the pre-requisite platform files and upload the app to PWS

1. Copy all files from “Lab 07 – PWS & mLab” folder and place into your working directory

   Overwrite the existing files and confirm the folder looks like this;



2. Open a command prompt and navigate to your project

   > cd <path of your project>

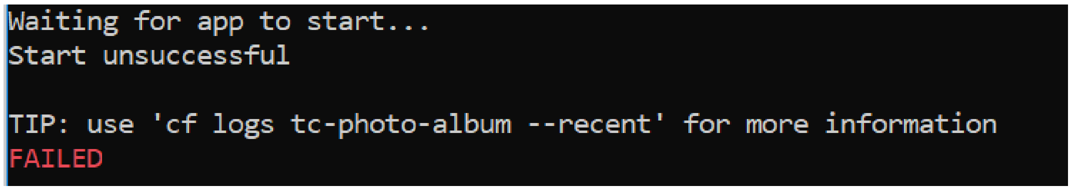
3. Login to PWS with your account creds

   > cf login -a https://api.run.pivotal.io

4. Push the app to PWS

   > cf push

5. Your app will fail to start and crash……. Why?



```
Waiting for app to start...
Start unsuccessful

TIP: use 'cf logs tc-photo-album --recent' for more information
FAILED
```

DELL Technologies

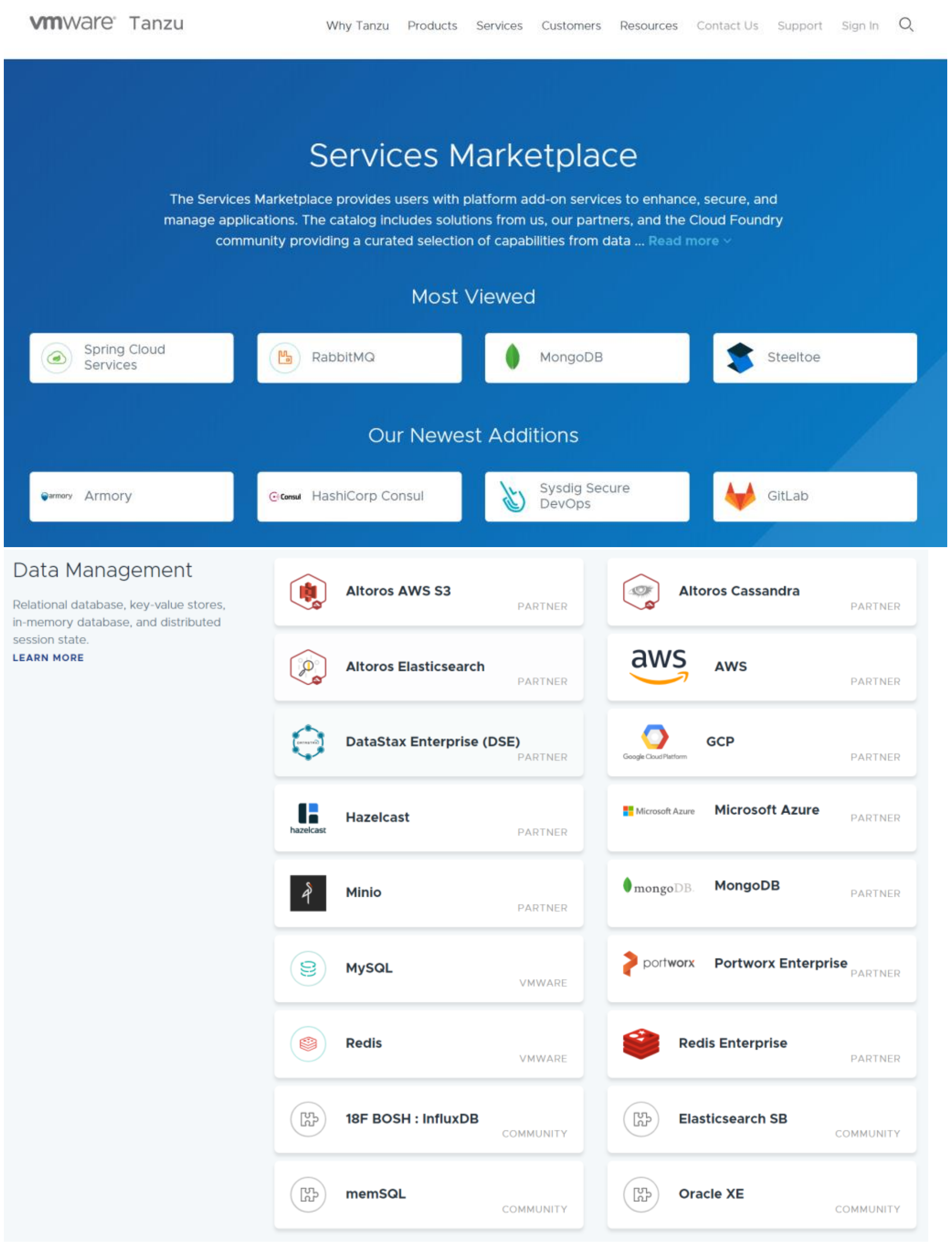
Tanzu Services Marketplace

If you haven't already guessed, the reason the application failed in the previous lab is because the code is expecting to connect to a MongoDB database that we have not provisioned into the PWS environment yet.

The Tanzu Services Marketplace is available for both the private deployment of Tanzu Application Services and the public hosted service provided by PWS.

Some of the packaged services that have been developed to be deployed at found at;

https://tanzu.vmware.com/services-marketplace



Services Marketplace

The Services Marketplace provides users with platform add-on services to enhance, secure, and manage applications. The catalog includes solutions from us, our partners, and the Cloud Foundry community providing a curated selection of capabilities from data ...

Most Viewed

Spring Cloud Services | RabbitMQ | MongoDB | Steeltoe

Our Newest Additions

Armory | HashiCorp Consul | Sysdig Secure DevOps | GitLab

Data Management

Relational database, key-value stores, in-memory database, and distributed session state.

LEARN MORE

Altoros AWS S3 — PARTNER	Altoros Cassandra — PARTNER
Altoros Elasticsearch — PARTNER	AWS — PARTNER
DataStax Enterprise (DSE) — PARTNER	GCP — PARTNER
Hazelcast — PARTNER	Microsoft Azure — PARTNER
Minio — PARTNER	MongoDB — PARTNER
MySQL — VMWARE	Portworx Enterprise — PARTNER
Redis — VMWARE	Redis Enterprise — PARTNER
18F BOSH : InfluxDB — COMMUNITY	Elasticsearch SB — COMMUNITY
memSQL — COMMUNITY	Oracle XE — COMMUNITY

Pivotal Web Services Marketplace

The services that are available in PWS are a subset of the Tanzu marketplace and are targeted at some of the most common services requested by customers.

To view all available services, login to https://console.run.pivotal.io/ and browse to the services section of your organization.

For our application, we will deploy a MongoDB service provided by mLab as a hosted multi-tenanted database instance.

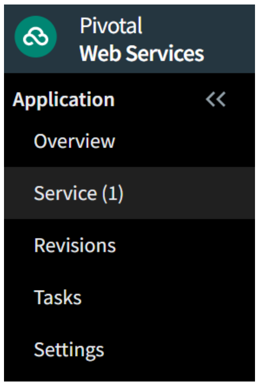
Continue to the next part of the lab (Part B) for instructions to deploy and bind the mLab service to our application.

DELL Technologies

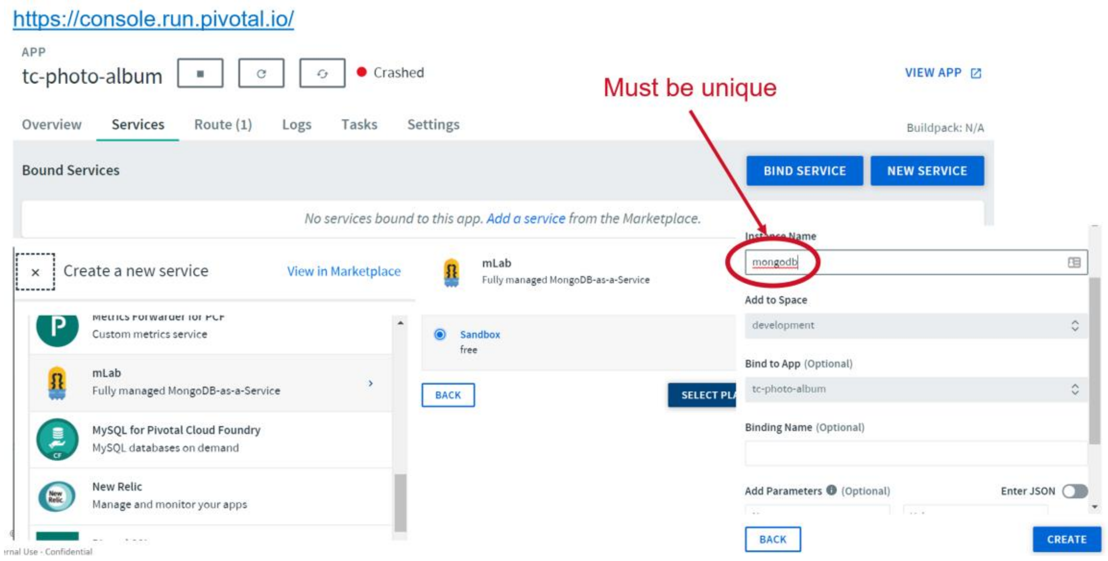
Lab Exercise (Part B): Add the external MongoDB service from the PWS Marketplace

6. Login to the PWS console at https://console.run.pivotal.io

7. Find your deployed application and click on the 'Service' menu option on the left



8. Click on the 'NEW SERVICE' button and create an mLab instance to bind to your app



No services bound to this app. Add a service from the Marketplace.

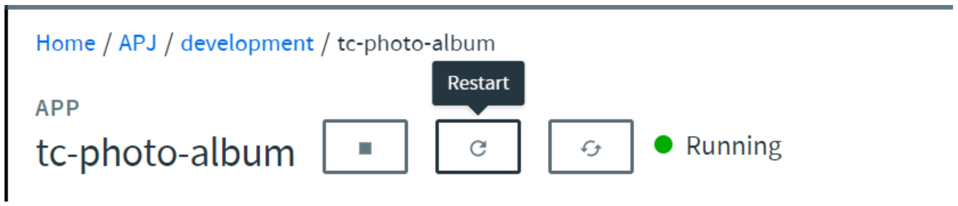
Instance Name: mongodb

Add to Space: development

Bind to App (Optional): tc-photo-album

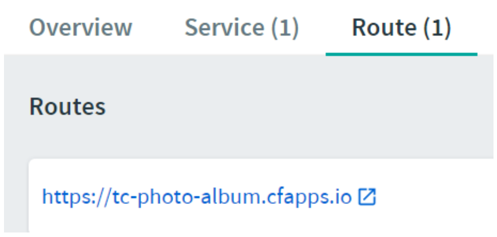
9. Use the default options for everything except the Instance Name, which needs to be unique

10. Restart your app from the console if required



11. Open the app

12. Test functionality and upload an image



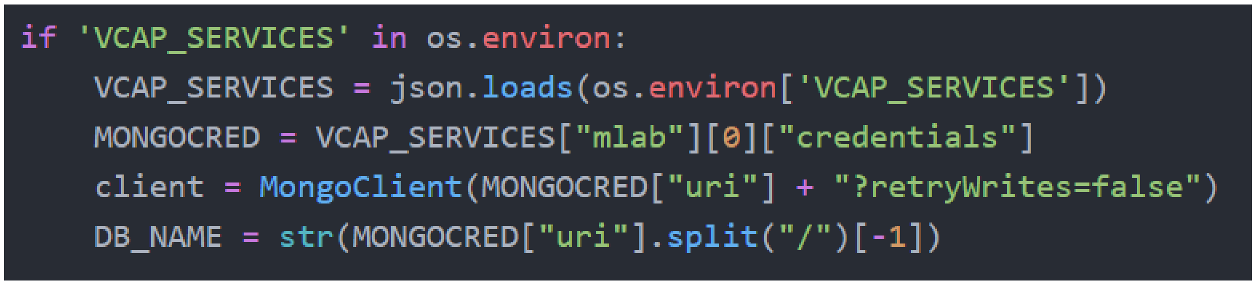
https://tc-photo-album.cfapps.io

DELL Technologies

Retrospective: Review code and take note of the 'VCAP_SERVICES' section in models.py

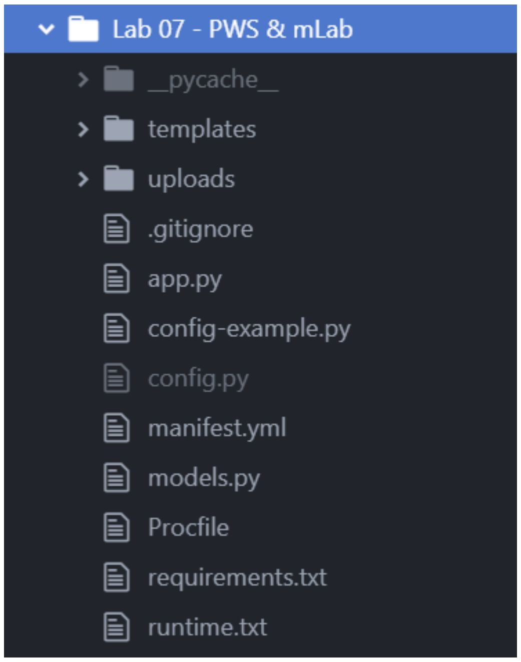
The application is now running in a resilient PaaS platform with the ability to scale on demand with autoscaling or manually with admin intervention.

Take note of the models.py application code to retrieve the environment variables with the MongoDB credentials setup in the previous Lab Part B section.

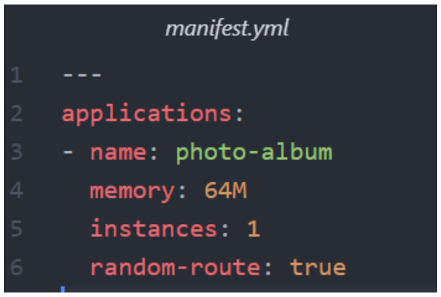


```python
if 'VCAP_SERVICES' in os.environ:
    VCAP_SERVICES = json.loads(os.environ['VCAP_SERVICES'])
    MONGOCRED = VCAP_SERVICES["mlab"][0]["credentials"]
    client = MongoClient(MONGOCRED["uri"] + "?retryWrites=false")
    DB_NAME = str(MONGOCRED["uri"].split("/")[-1])
```

Take note of the platform specific files required for the PaaS environment to configure and setup the containers with the right pre-requisites for the application to run.

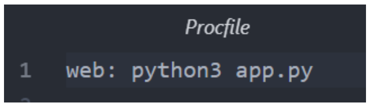


```
Lab 07 - PWS & mLab
    __pycache__
    templates
    uploads
    .gitignore
    app.py
    config-example.py
    config.py
    manifest.yml
    models.py
    Procfile
    requirements.txt
    runtime.txt
```



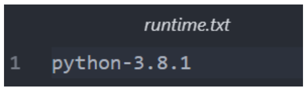
manifest.yml

```yaml
---
applications:
- name: photo-album
  memory: 64M
  instances: 1
  random-route: true
```



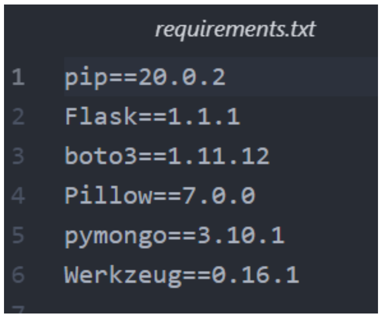
Procfile

```
web: python3 app.py
```



runtime.txt

```
python-3.8.1
```



requirements.txt

```
pip==20.0.2
Flask==1.1.1
boto3==1.11.12
Pillow==7.0.0
pymongo==3.10.1
Werkzeug==0.16.1
```