DELL Technologies

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.

June 2020

Student Lab Guide

DELL Technologies

Revisions

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

Student Lab Guide

DELL Technologies

Table of Contents

DELL Technologies

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….

DELL Technologies

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"
```

.gitignore

```
setenv.py
```

main.py

```
from setenv import user,
password

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

DELL Technologies

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')
```

DELL Technologies

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

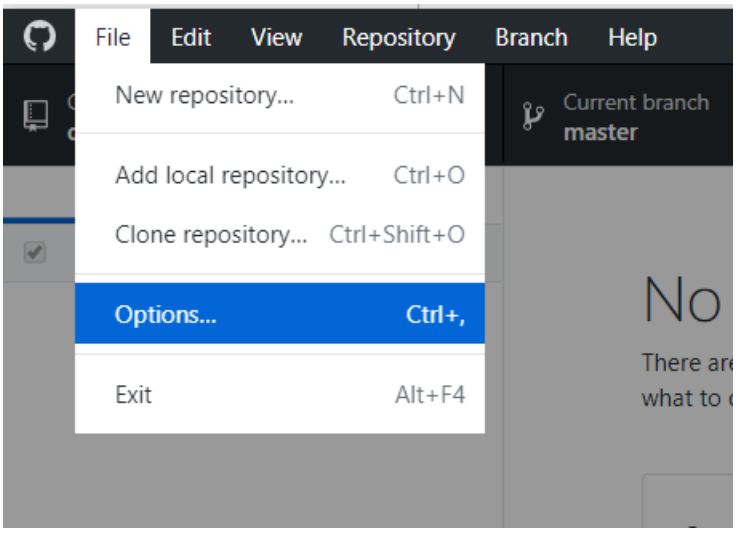
  https://help.github.com/desktop/guides/getting-started-with-github-desktop/

DELL Technologies

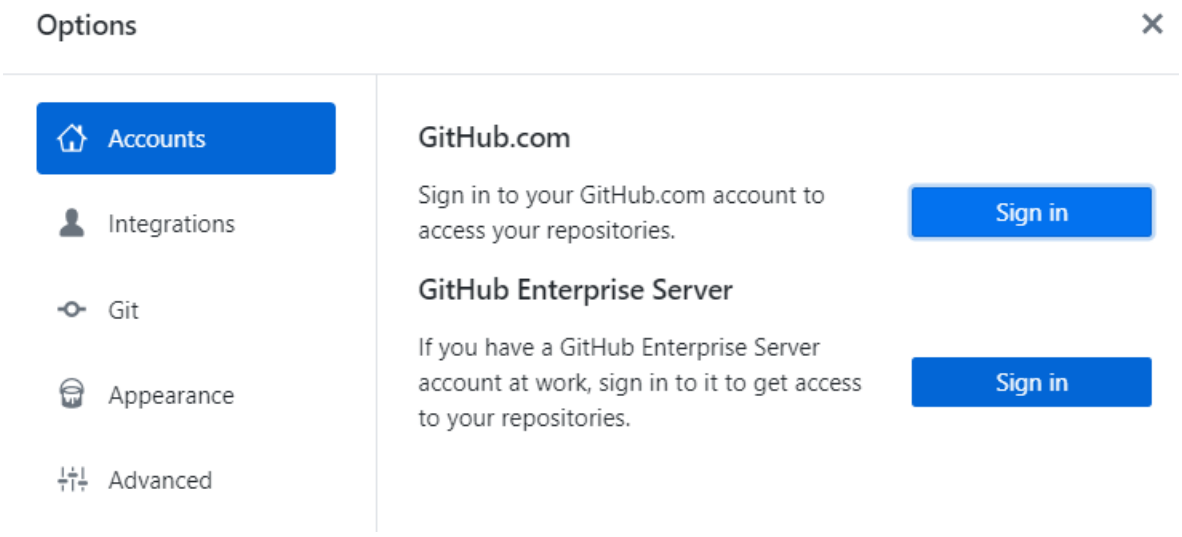
Lab Exercise: Create a new repo and publish a file with your name

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

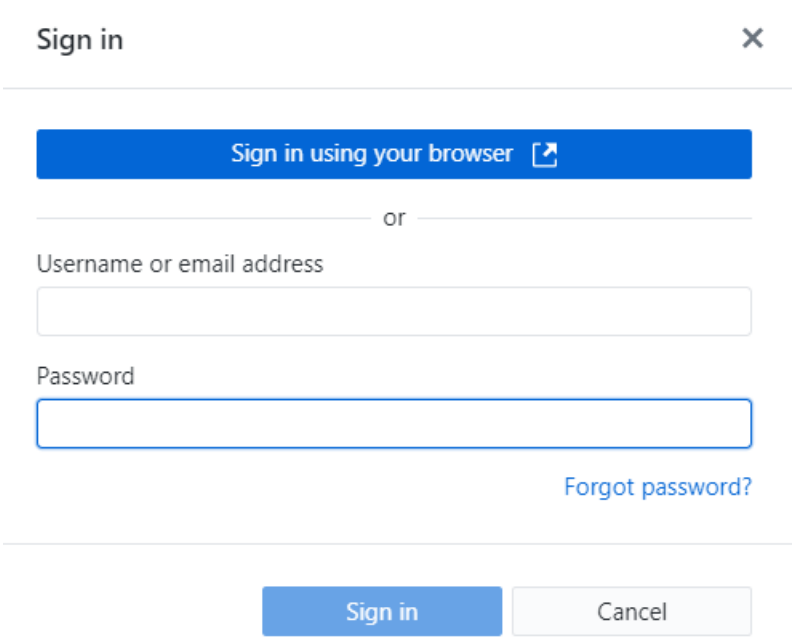
1. Open the GitHub Desktop Client (from the shortcut on the desktop)
2. Go to “File > Options” to login to GitHub



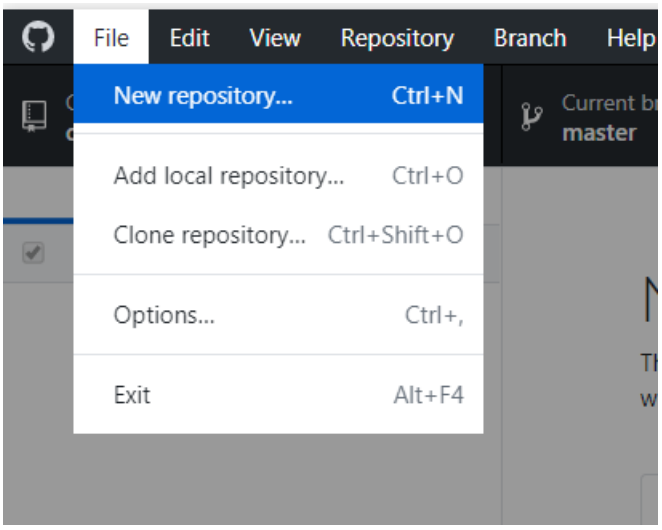
3. Click on “Accounts > Sign in” in order to Login to GitHub.com



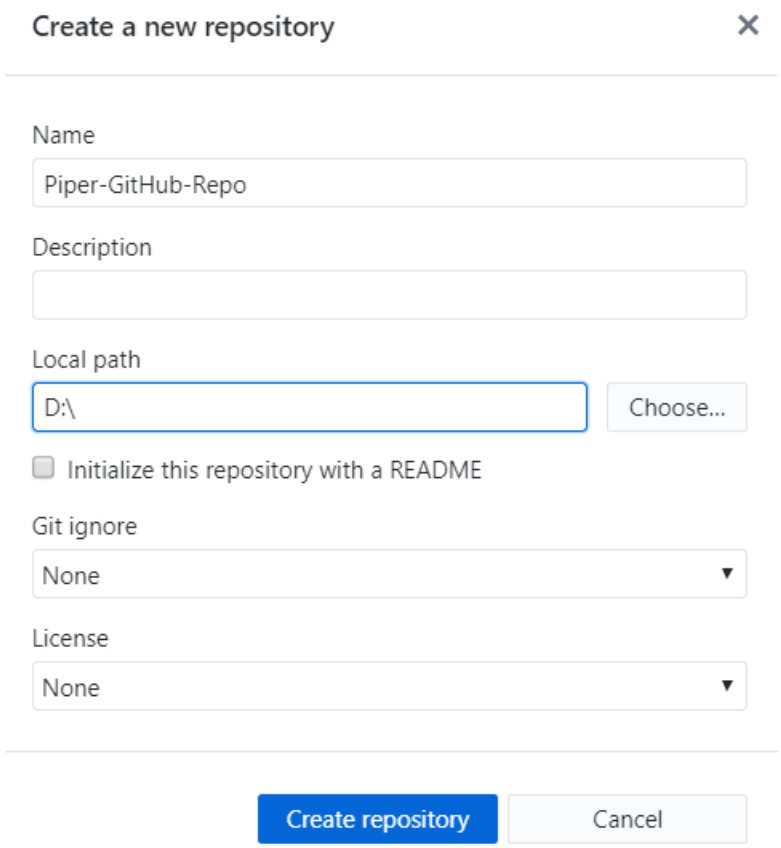
4. Type in your personal email ID and password used to sign-up the GitHub account



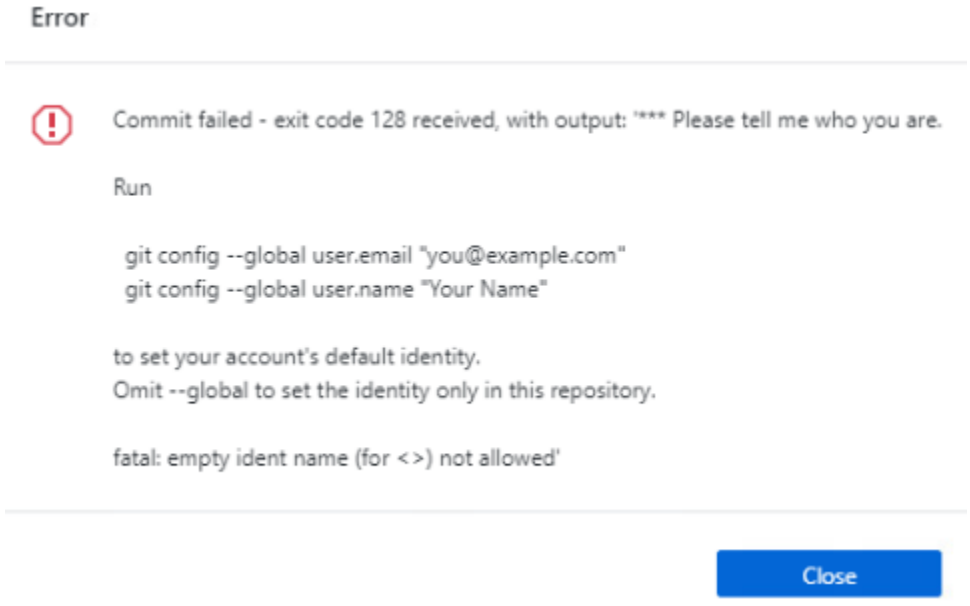
5. Create a new repository from “File> New Repository” option



6. Type in the Name as Piper-GitHub-Repo and change the path to D:\ and click on create repository

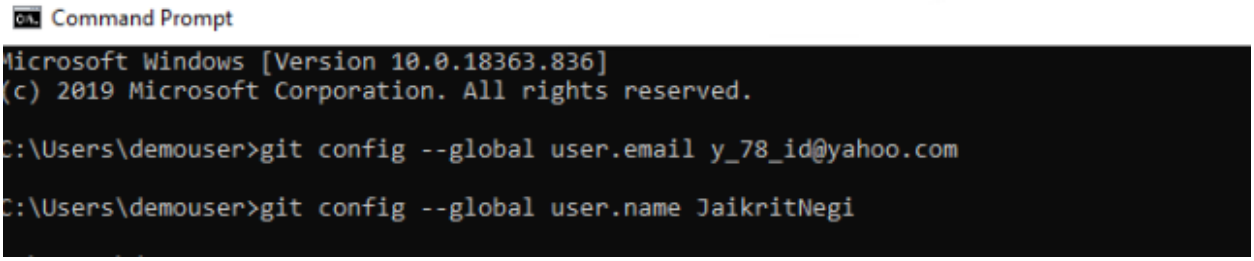


7. **NOTE- If you receive below error code 128, follow the Step-5 else go to Step-6**

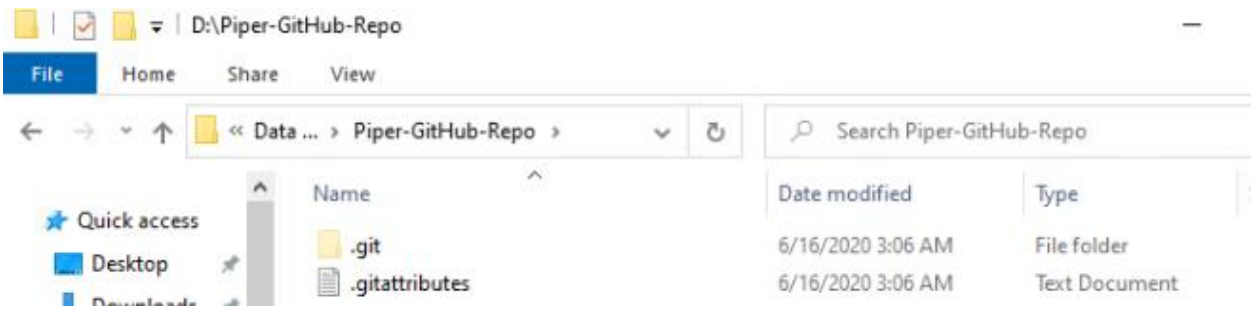


8. Open the Windows command prompt and type in above 2 commands (example in below screenshot) by:
    1. replacing last field in 1st command with your GitHub account &

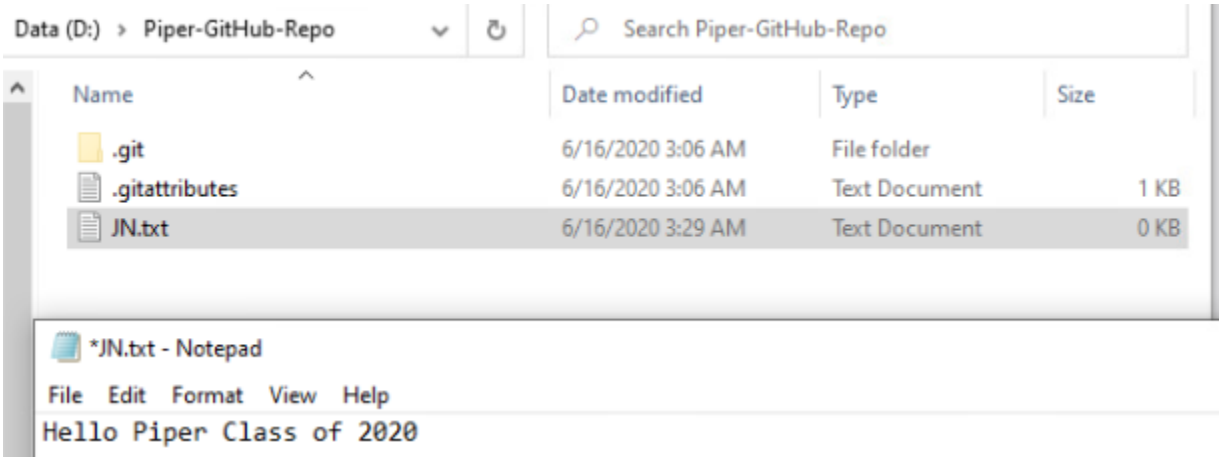
2. replacing last field in 2nd command with your own Name



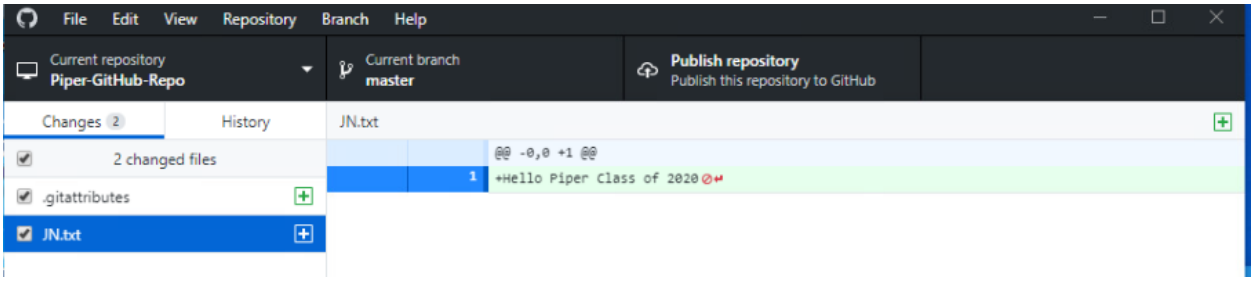
9. Open Windows Explorer and find the folder in the D:\ drive created by GitHub Desktop Client with the folder name chosen as repository name (Piper-GitHub-Repo) with below structure



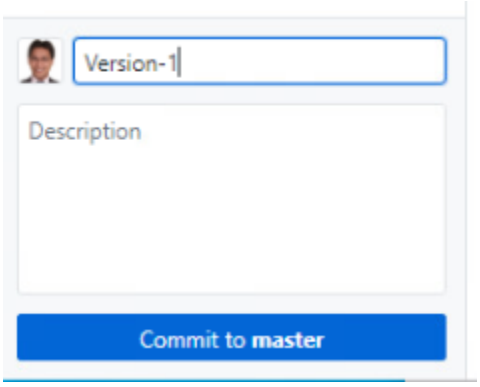
10. Create a text file with your name as the filename and add the content of the text file with the words “Hello Piper class of 2020!” and save the file



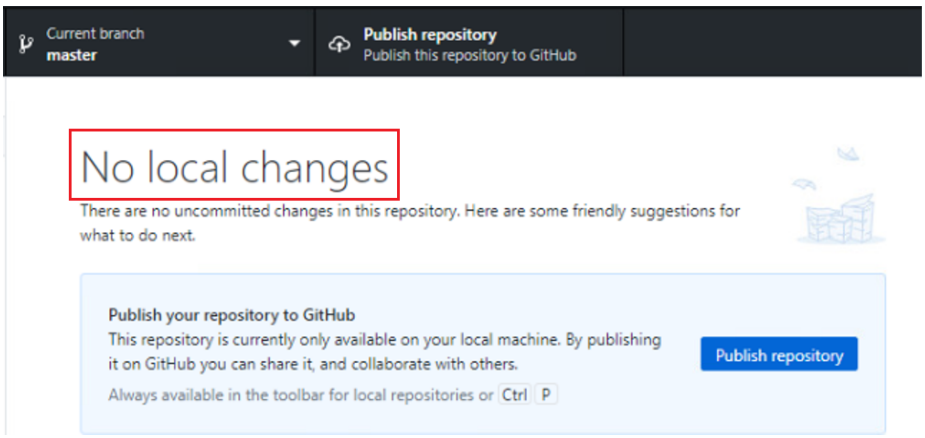
11. Go to already opened instance of GitHub Desktop Client and notice the newly created file and content and shows the changed files.



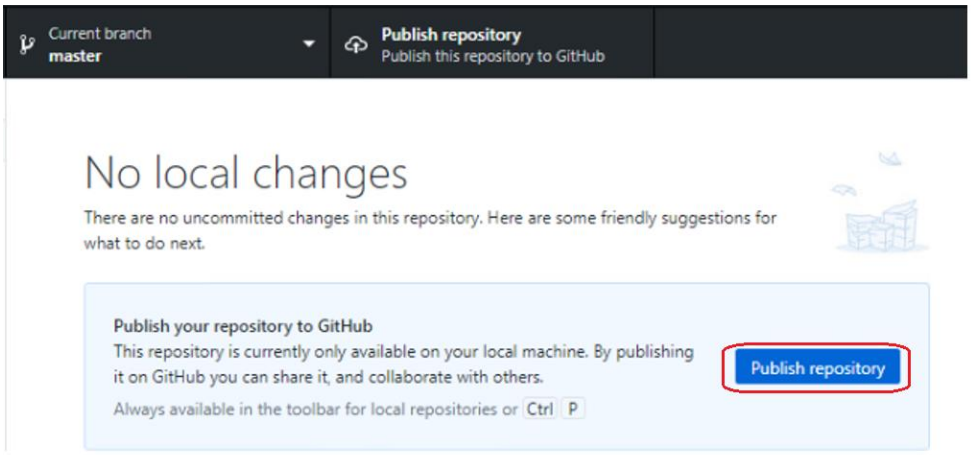
12. The current change in file is in working directory and tracked by GitHub client. In order to commit the changes to local directory, type the comment in the summary field and click on commit to master



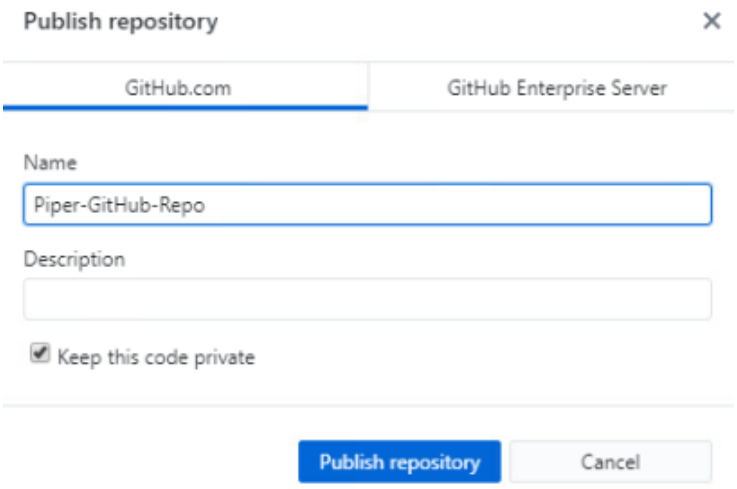
13. Above action should show below screen confirming commit with display as No local changes



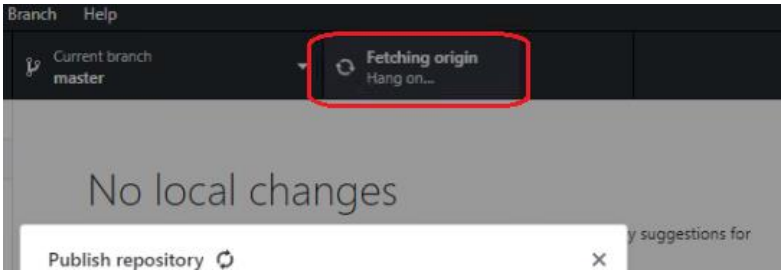
14. Publish the local repository to central repository by clicking on Publish repository

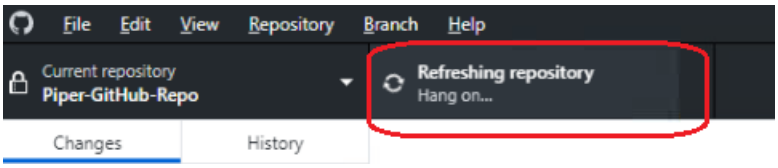


15. Upon which choose to keep it Private (visible to self only) or Public (visible to all on GitHub) and Click on Publish repository.

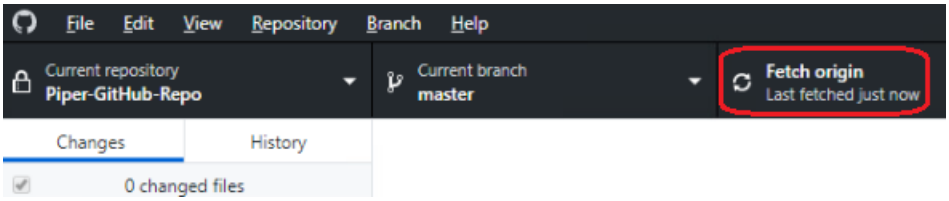


16. Notice the activity after publishing the repository to central GitHub repo.

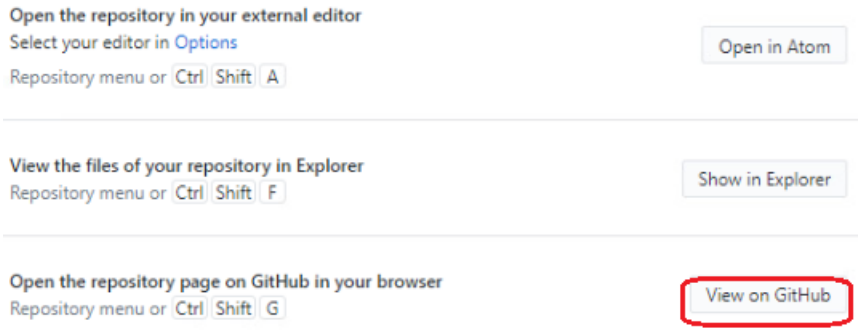




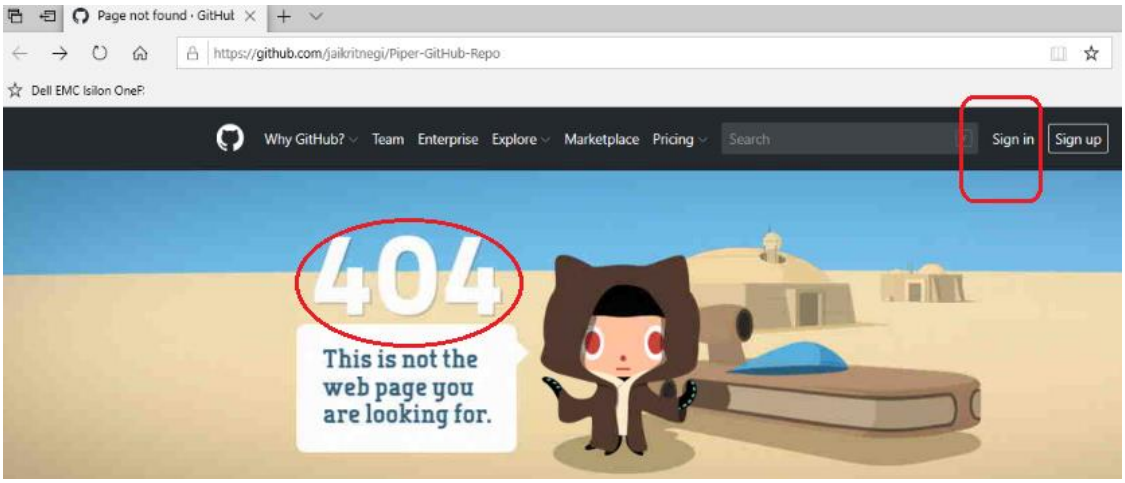
17. Once the repository is published to central GitHub repo, notice the message as Last fetched just now.

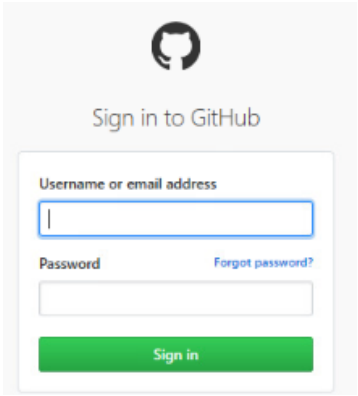


18. Once the repository is published, click on View on GitHub option that opens a browser window to verify the repository on GitHub in browser

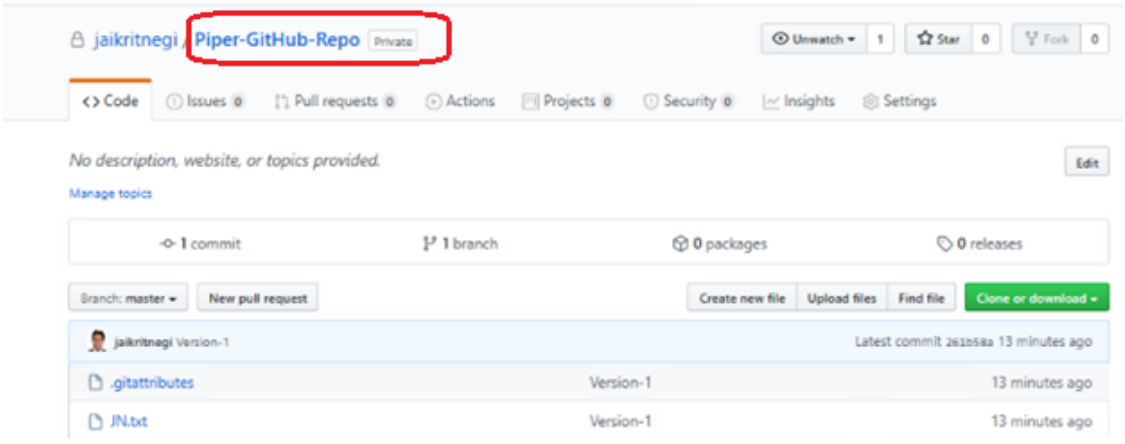


19. Should you encounter error code 404 on the web browser as below (only when checked as “Private” when publishing repository, click on sign in to login with your GutHub credentials to view the published repository.





20. Upon successful login, you can verify the published repository and content on central GitHub repository



DELL Technologies

Lab 2a: Deploy an App to PWS

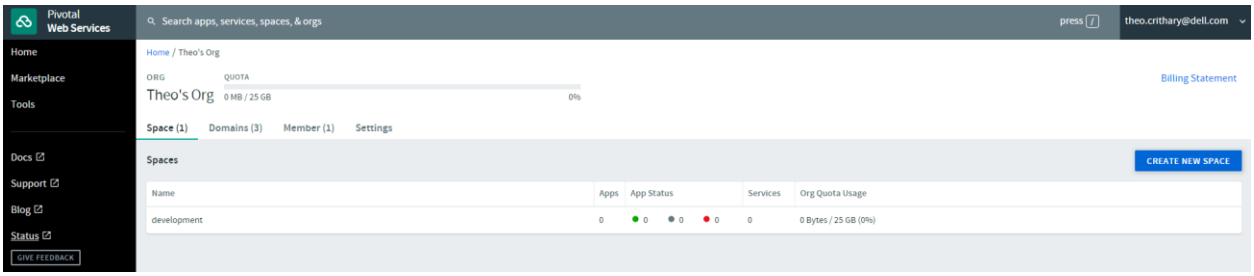
Module Objectives:

- ❑ 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.



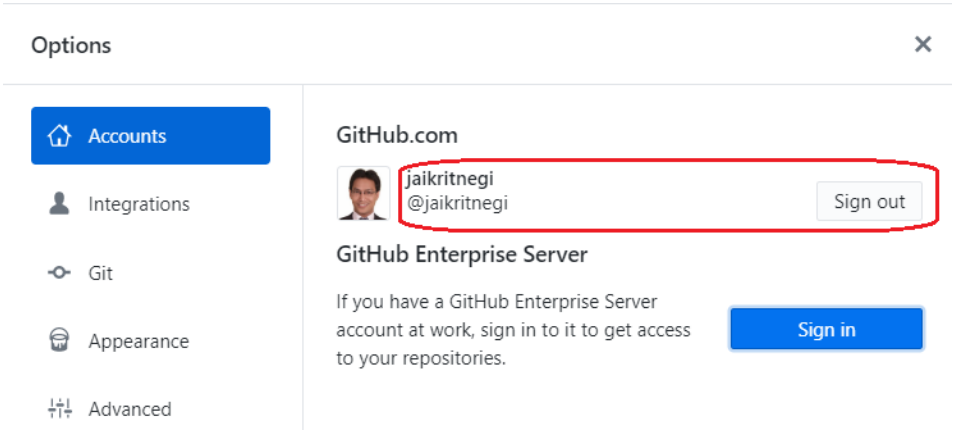
Important note: Please create an organisation before continuing with the lab

DELL Technologies

Lab Exercise: Clone an example app and deploy to PWS

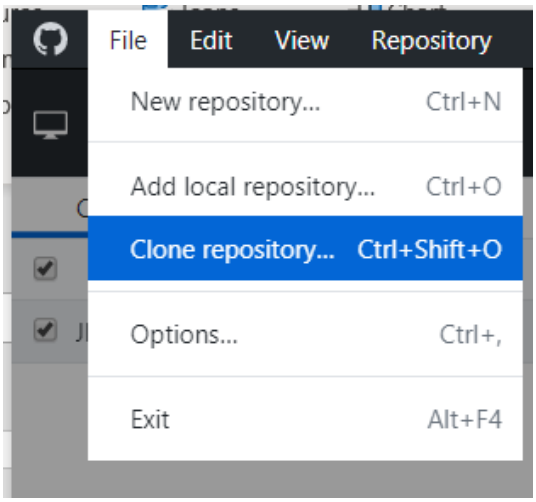
Complete the below steps to demonstrate your understanding of the tools and concepts required for the remaining lab exercises;

1. Open the GitHub Desktop Client (from the shortcut on the desktop) and ensure that you are log in with your personal GitHub account

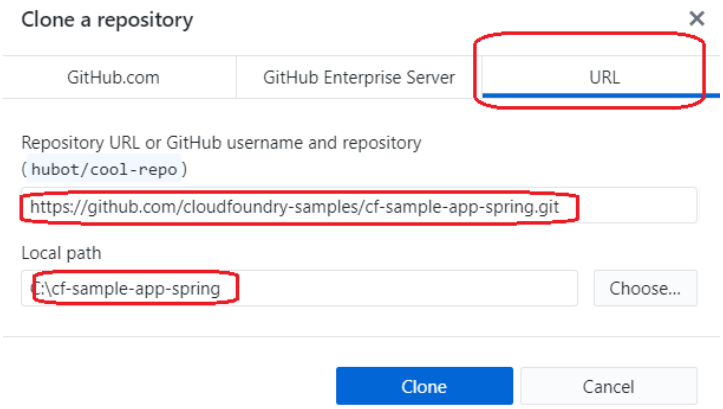


2. Clone the following repository –"Go to File > Clone repository" to clone it from GitHub

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



3. Click on the URL tab and copy above github URL and paste to the Repository/ URL field and select the local path on D:\ drive.



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

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

5. Login to PWS with your PWS account credentials

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

```
D:\cf-sample-app-spring>cf login -a https://api.run.pivotal.io
API endpoint: https://api.run.pivotal.io

Email: jaikrit.negi@dell.com

Password:
Authenticating...
OK

Targeted org Jaikrit

Targeted space development



API endpoint:   https://api.run.pivotal.io (API version: 3.82.0)
User:           jaikrit.negi@dell.com
Org:            Jaikrit
Space:          development
```

6. Push the app to PWS

   > cf push

DELL Technologies

```
D:\cf-sample-app-spring>cf push
Pushing from manifest to org Jaikrit / space development as jaikrit.negi@dell.com...
Using manifest file D:\cf-sample-app-spring\manifest.yml
Getting app info...
Updating app with these attributes...
  name:               cf-demo
  path:               D:\cf-sample-app-spring
  buildpacks:
    https://github.com/cloudfoundry/java-buildpack.git
  command:            JAVA_OPTS="-agentpath:$PWD/.java-buildpack/open_jdk_jre/bin/jvmkill-1.16.0_RELEASE=printHeapHistogram=1 -Djava.io.tmpdir=$TMPDIR -XX:A
ctiveProcessorCount=$(nproc) -Djava.ext.dirs=$PWD/.java-buildpack/container_security_provider:$PWD/.java-buildpack/open_jdk_jre/lib/ext -Djava.security.prope
rties=$PWD/.java-buildpack/java_security/java.security $JAVA_OPTS" && CALCULATED_MEMORY=$($PWD/.java-buildpack/open_jdk_jre/bin/java-buildpack-memory-calcula
tor-3.13.0_RELEASE -totMemory=$MEMORY_LIMIT -loadedClasses=8354 -poolType=metaspace -stackThreads=250 -vmOptions="$JAVA_OPTS") && echo JVM Memory Configurati
on: $CALCULATED_MEMORY && JAVA_OPTS="$JAVA_OPTS $CALCULATED_MEMORY" && MALLOC_ARENA_MAX=2 JAVA_OPTS=$JAVA_OPTS SERVER_PORT=$PORT JAVA_HOME=$PWD/.java-buildpa
ck/open_jdk_jre exec $PWD/.java-buildpack/spring_boot_cli/bin/spring run -cp $PWD/.java-buildpack/client_certificate_mapper/client_certificate_mapper-1.11.0_
RELEASE.jar app.groovy
  disk quota:         1G
  health check type:  port
  instances:          1
  memory:             768M
  stack:              cflinuxfs3
  routes:
    cf-demo-bright-gorilla-np.cfapps.io

Updating 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% 1s
```

7. 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 (NOTE- your URL will be different)

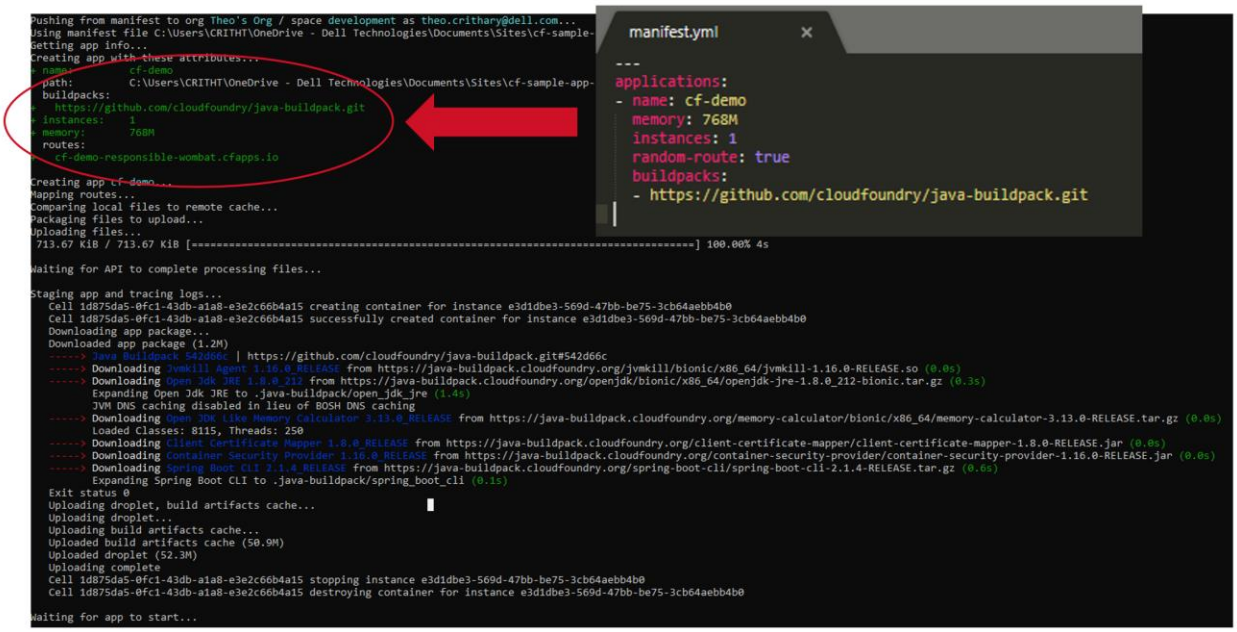
Reflect on what just happened

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.

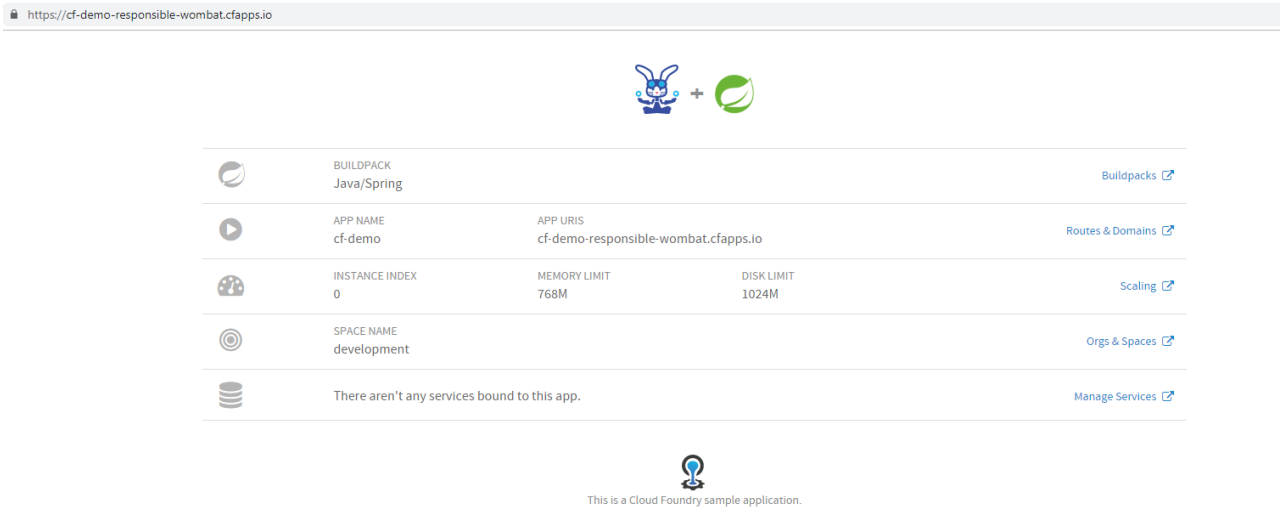
Random-route: true is a Cloud Foundry function that randomly generates a unique DNS based on selecting 2 words from a provided dictionary. This is used only in testing to ensure that DNS routes are not conflicting when deploying the same application many times.

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.



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;



DELL Technologies

Lab 2b: Deploy an App to PWS (Alternate “Go App”)

Module Objectives:

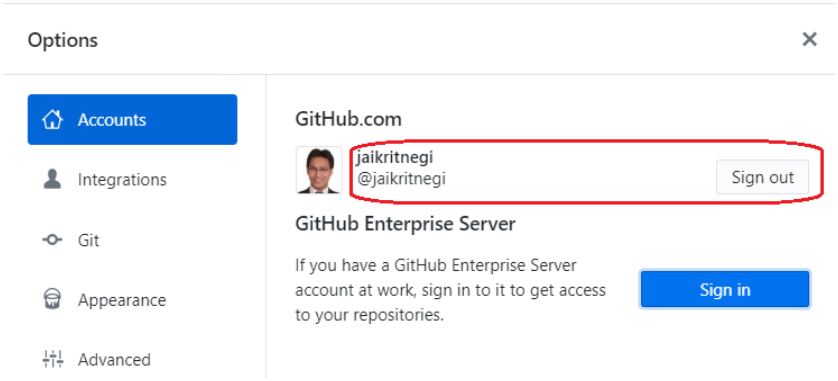
- ❑ Learn about PWS & cf cli tools
- ❑ Confirm cf cli is installed and working
- ❑ Login to PWS via cf cli
- ❑ Push a demo app to PWS and confirm we have the skills and tools required to publish a web app to the PWS PaaS platform

DELL Technologies

Lab Exercise: Clone an example app and deploy to PWS

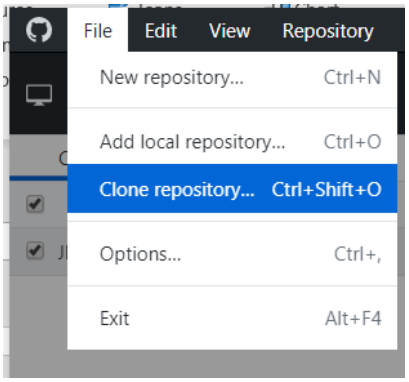
Complete the below steps to demonstrate your understanding of the tools and concepts required for the remaining lab exercises;

1. Open the GitHub Desktop Client (from the shortcut on the desktop) and ensure that you are log in with your personal GitHub account

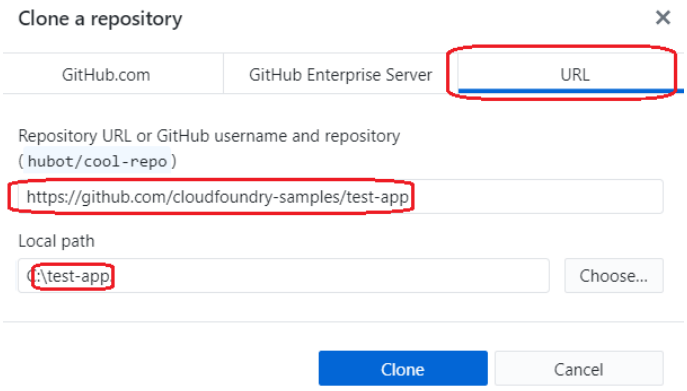


2. Clone the following repository – "Go to File > Clone repository" to clone it from GitHub

   https://github.com/cloudfoundry-samples/test-app.git



3. Click on the URL tab and copy above github URL and paste to the Repository/ URL field and select the local path on D:\ drive.



DELL Technologies

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

   > cd <path of cloned repo>\test-app

5. Login to PWS with your PWS account credentials

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

```
C:\test-app>cf login -a https://api.run.pivotal.io
API endpoint: https://api.run.pivotal.io

Email: jaikrit.negi@dell.com

Password:
Authenticating...
OK

Targeted org Jaikrit

Targeted space development



API endpoint:   https://api.run.pivotal.io (API version: 3.81.0)
User:           jaikrit.negi@dell.com
Org:            Jaikrit
Space:          development
```

6. Push the app to PWS

   > cf push

DELL Technologies

```
C:\test-app>cf push
Pushing from manifest to org Jaikrit / space development as jaikrit.negi@dell.com...
Using manifest file C:\test-app\manifest.yml
Getting app info...
Creating app with these attributes...
+ name:        test-app
  path:        C:\test-app
+ instances:   1
+ memory:      256M
  routes:
+   test-app-patient-llama-au.cfapps.io

Creating app test-app...
Mapping routes...
Comparing local files to remote cache...
Packaging files to upload...
Uploading files...
 37.28 KiB / 37.28 KiB [==========================================================================================================

Waiting for API to complete processing files...

Staging app and tracing logs...
   Downloading web_config_transform_buildpack...
   Downloading binary_buildpack...
   Downloading staticfile_buildpack...
   Downloading dotnet_core_buildpack_beta...
   Downloading go_buildpack...
   Downloaded web_config_transform_buildpack
   Downloading java_buildpack...
   Downloaded binary_buildpack
   Downloading ruby_buildpack...
   Downloaded go_buildpack
   Downloading dotnet_core_buildpack...
   Downloaded dotnet_core_buildpack_beta
   Downloading nodejs_buildpack...
   Downloaded java_buildpack
   Downloading python_buildpack...
   Downloaded nodejs_buildpack
   Downloading php_buildpack...
   Downloaded ruby_buildpack
   Downloaded dotnet_core_buildpack
   Downloaded python_buildpack
   Downloaded php_buildpack
   Downloaded staticfile_buildpack
   Cell 732975b5-a95c-4e37-b595-a0a3c3a9e2ea creating container for instance 8296c487-ed49-4981-8104-b1144db770f4
```

7. Test app by opening in browser- Use the URL shown in the 'routes:' section of the output displayed in your command prompt.

```
Waiting for app to start...

name:              test-app
requested state:   started
routes:            test-app-patient-llama-au.cfapps.io
last uploaded:     Tue 16 Jun 16:36:49 +08 2020
stack:             cflinuxfs3
buildpacks:        go

type:            web
instances:       1/1
memory usage:    256M
start command:   test-app
     state     since                  cpu    memory      disk      details
#0   running   2020-06-16T08:36:59Z   0.0%   0 of 256M   0 of 1G
```

e.g. https://test-app-patient-llama-au.cfapps.io (NOTE- your URL will be different)

DELL Technologies

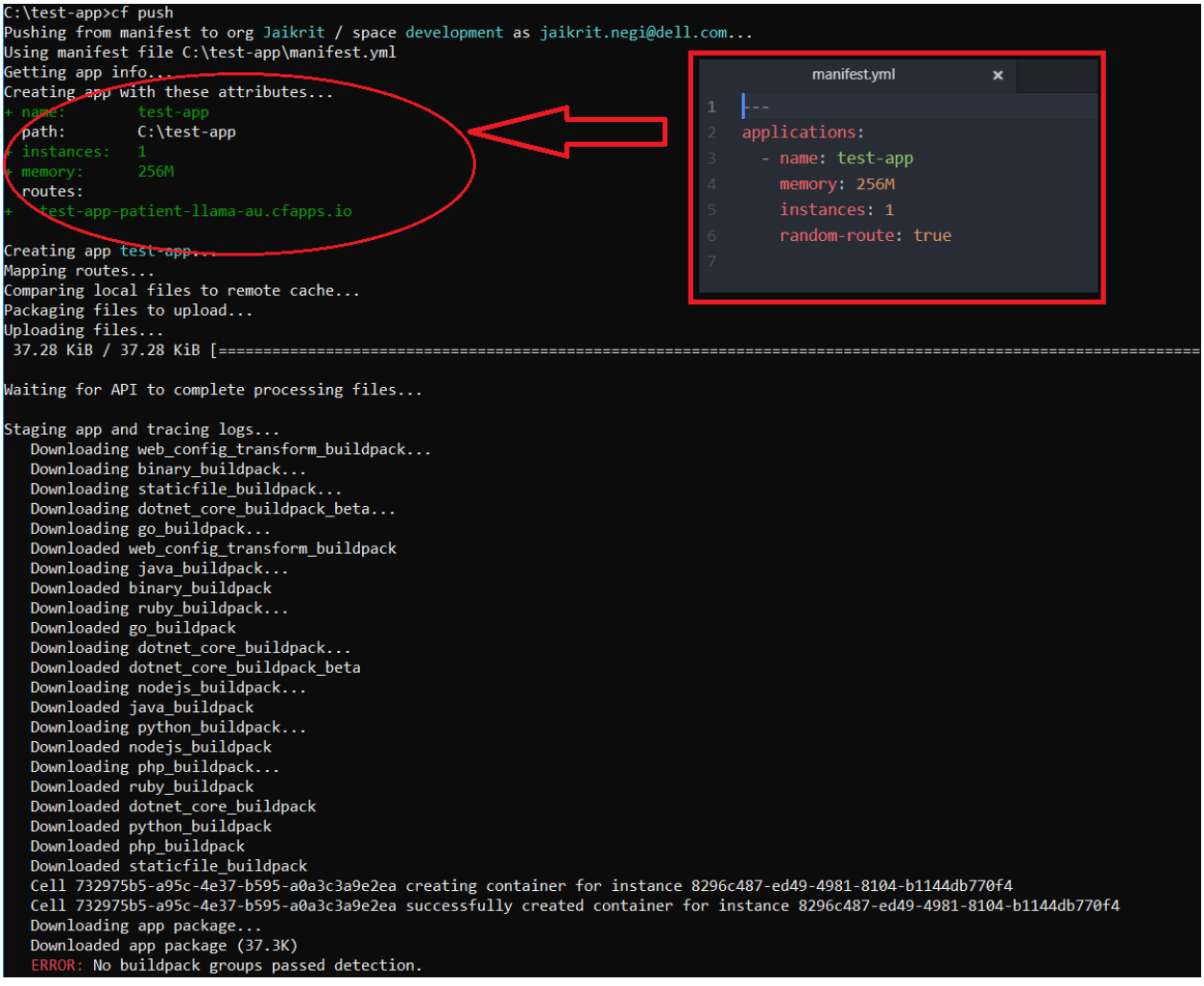
Reflect on what just happened

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.

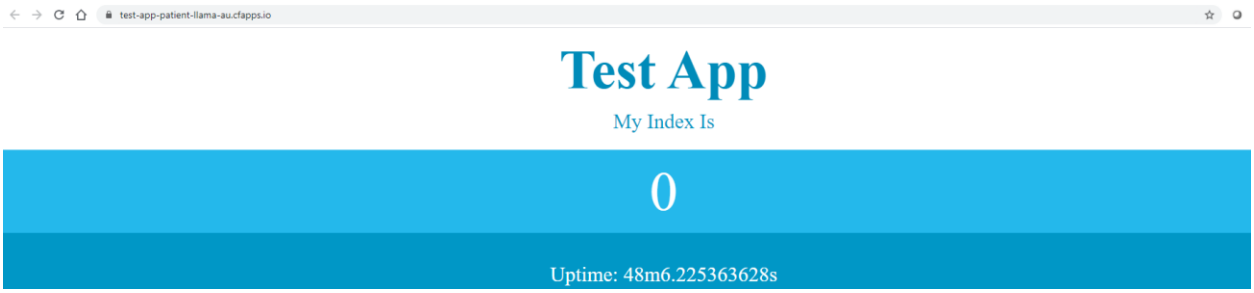
Random-route: true is a Cloud Foundry function that randomly generates a unique DNS based on selecting 2 words from a provided dictionary. This is used only in testing to ensure that DNS routes are not conflicting when deploying the same application many times.

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.



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;



DELL Technologies

Lab 3: Create a Static HTML Website

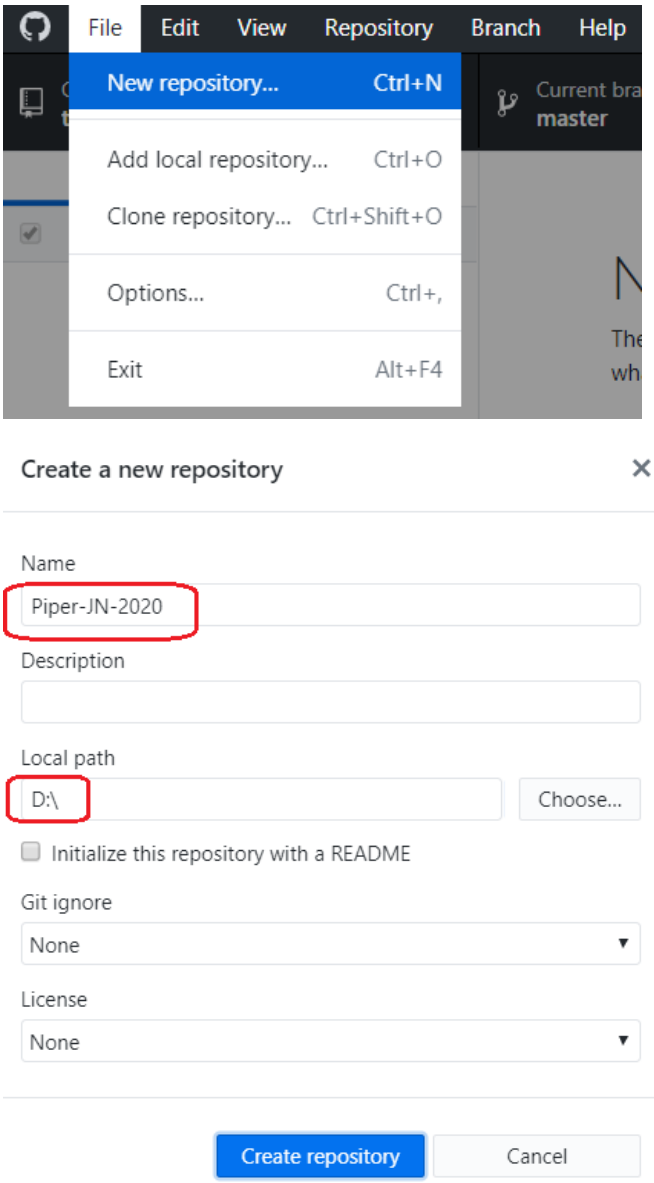
Module Objectives:

- ❑ 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

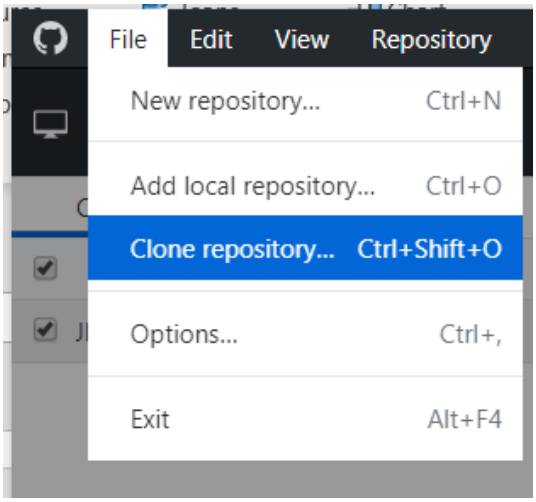
Lab Exercise: Copy the static webpage provided and setup a new project

1. Open the GitHub Desktop Client (from the shortcut on the desktop)
2. Go to File > New repository to create a new repo to store your project – call it whatever you want and click on Create repository button

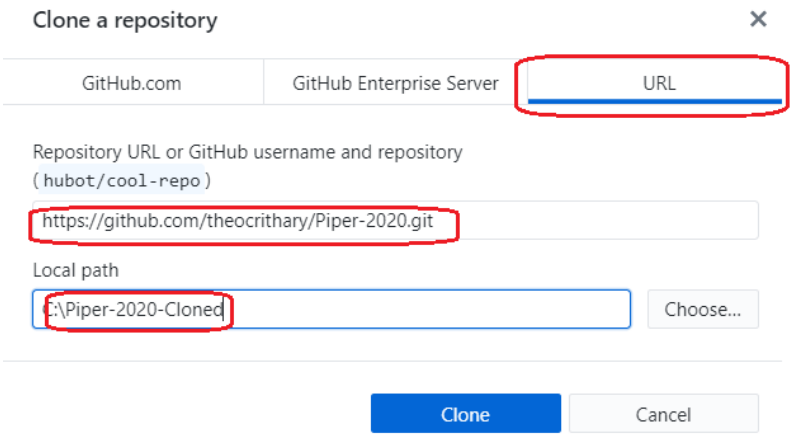


3. Clone the following Piper repository – "Go to File > Clone repository" to clone it from GitHub

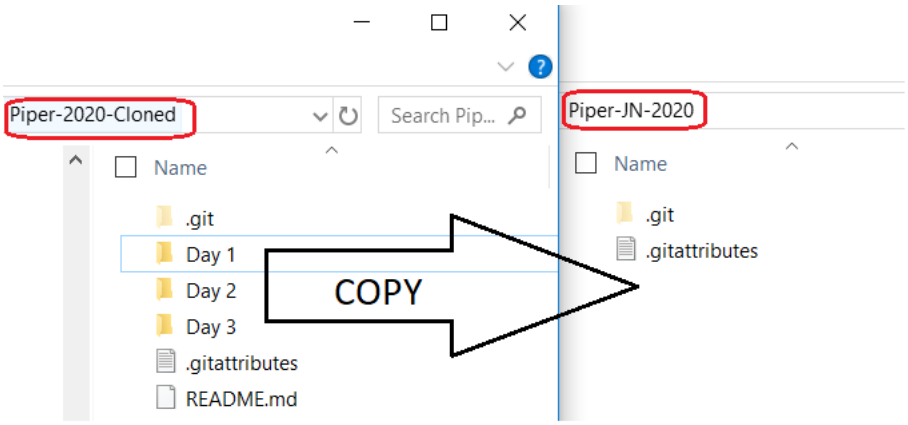
   https://github.com/theocrithary/Piper-2020.git



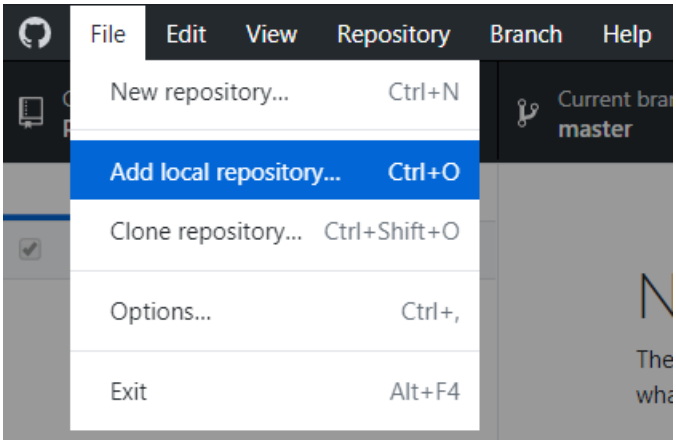
4. Click on the URL tab and copy above GitHub URL and paste to the Repository/ URL field and select the local path on D:\ drive.



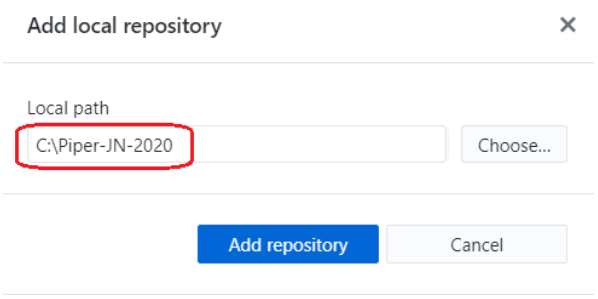
5. Copy the Day 1, Day 2 and Day 3 folders from cloned directory (created above in Step 4) to your Project folder as created above in Step-2



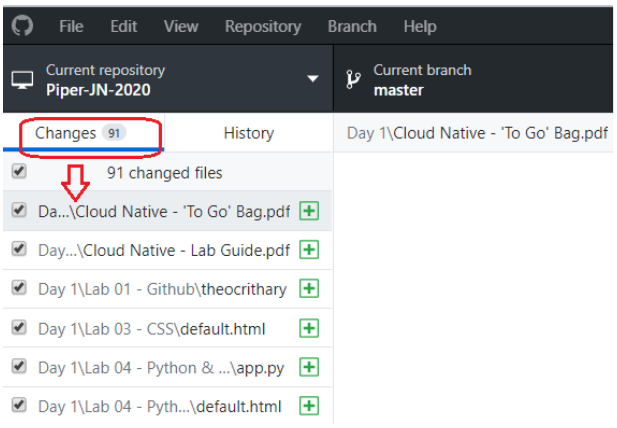
6. Go to your Project folder “Day 1 > Lab 03 – CSS”
7. Open the ‘default.html’ file in a browser to test its ‘look and feel’
8. Open the ‘default.html’ file in the Atom text editor provided and review the code
9. Go to the GitHub Desktop Client and Click on “File > Add local repository” (if not already in your Project folder)



10. Type the local path as “D: \”Replace the name to the path of your Project folder” and click Add repository

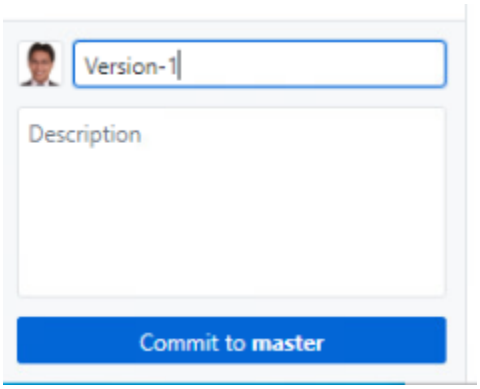


11. Notice all the changes on GitHub Desktop Client

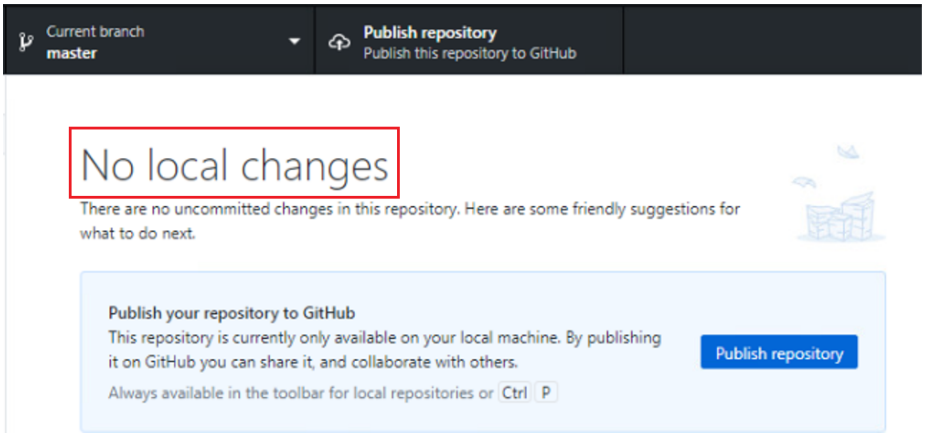


DELL Technologies

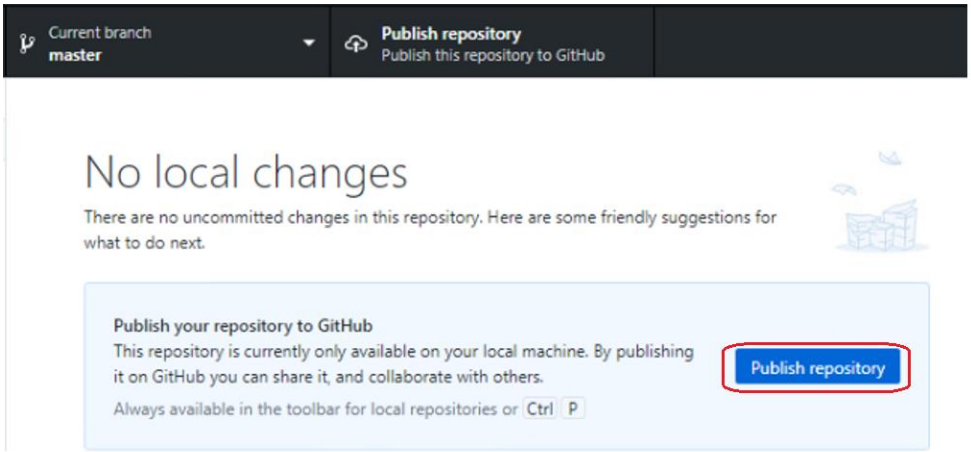
12. The current changes in directory structure is in working directory and tracked by GitHub client. In order to commit the changes to local directory, type the comment in the summary field and click on commit to master



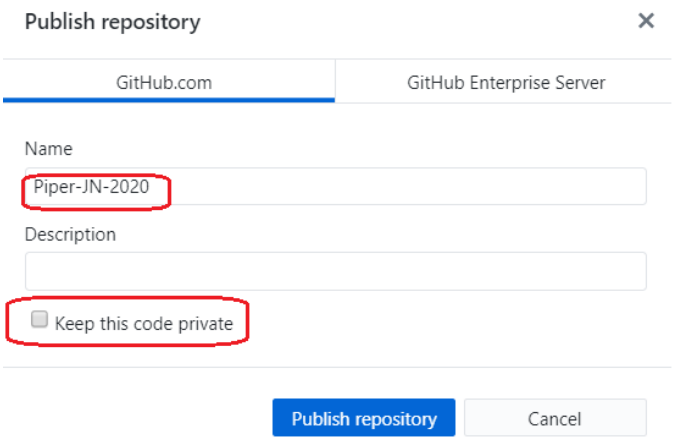
13. Above action should show below screen confirming commit with display as No local changes



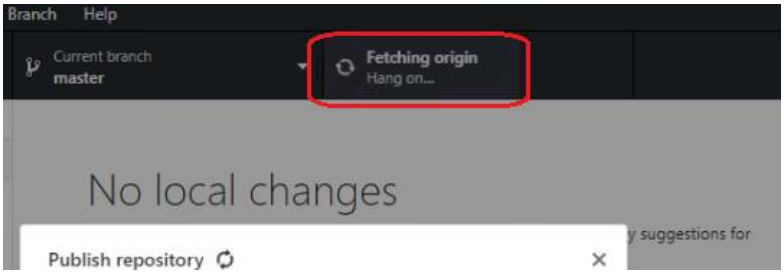
14. Publish the local repository to central repository by clicking on Publish repository

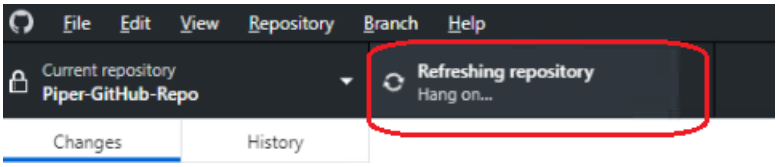


15. Upon which choose to keep it Public (visible to all on GitHub) and Click on Publish repository.

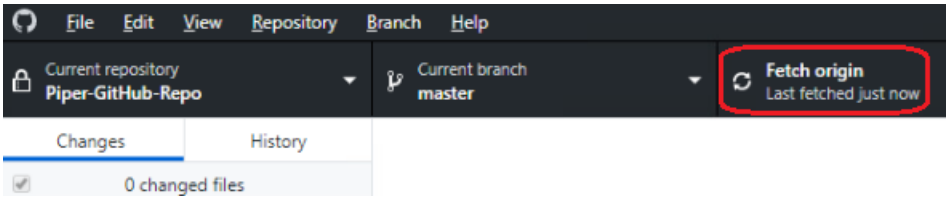


16. Notice the activity after publishing the repository to central GitHub repo.

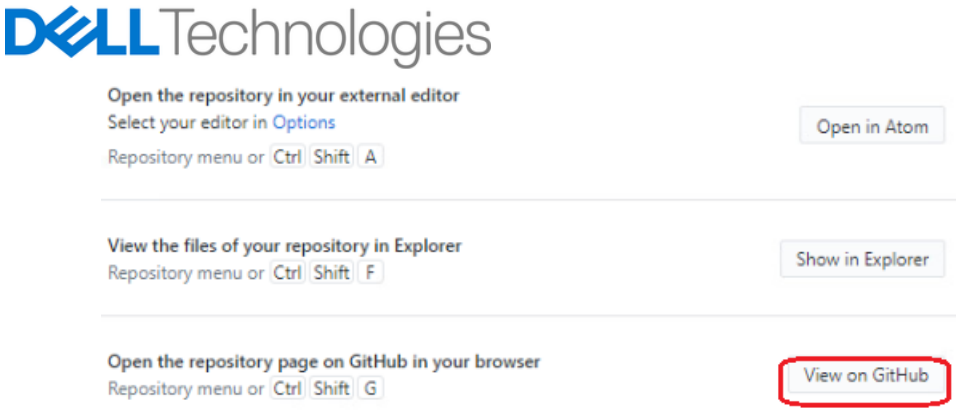




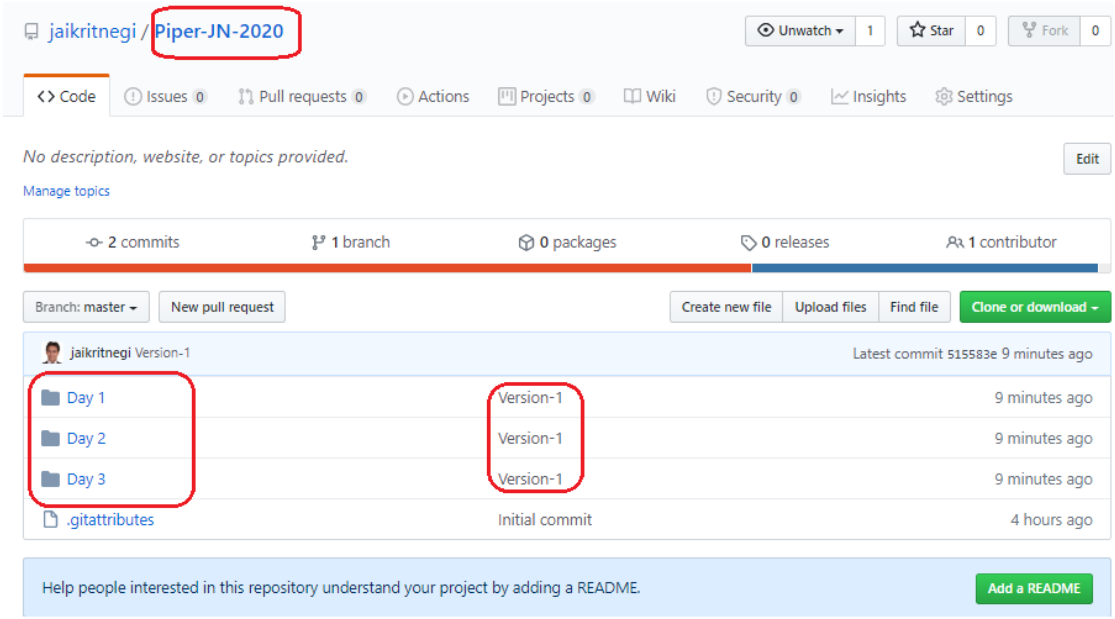
17. Once the repository is published to central GitHub repo, notice the message as Last fetched just now.



18. Once the repository is published, click on View on GitHub option that opens a browser window to verify the repository on GitHub in browser



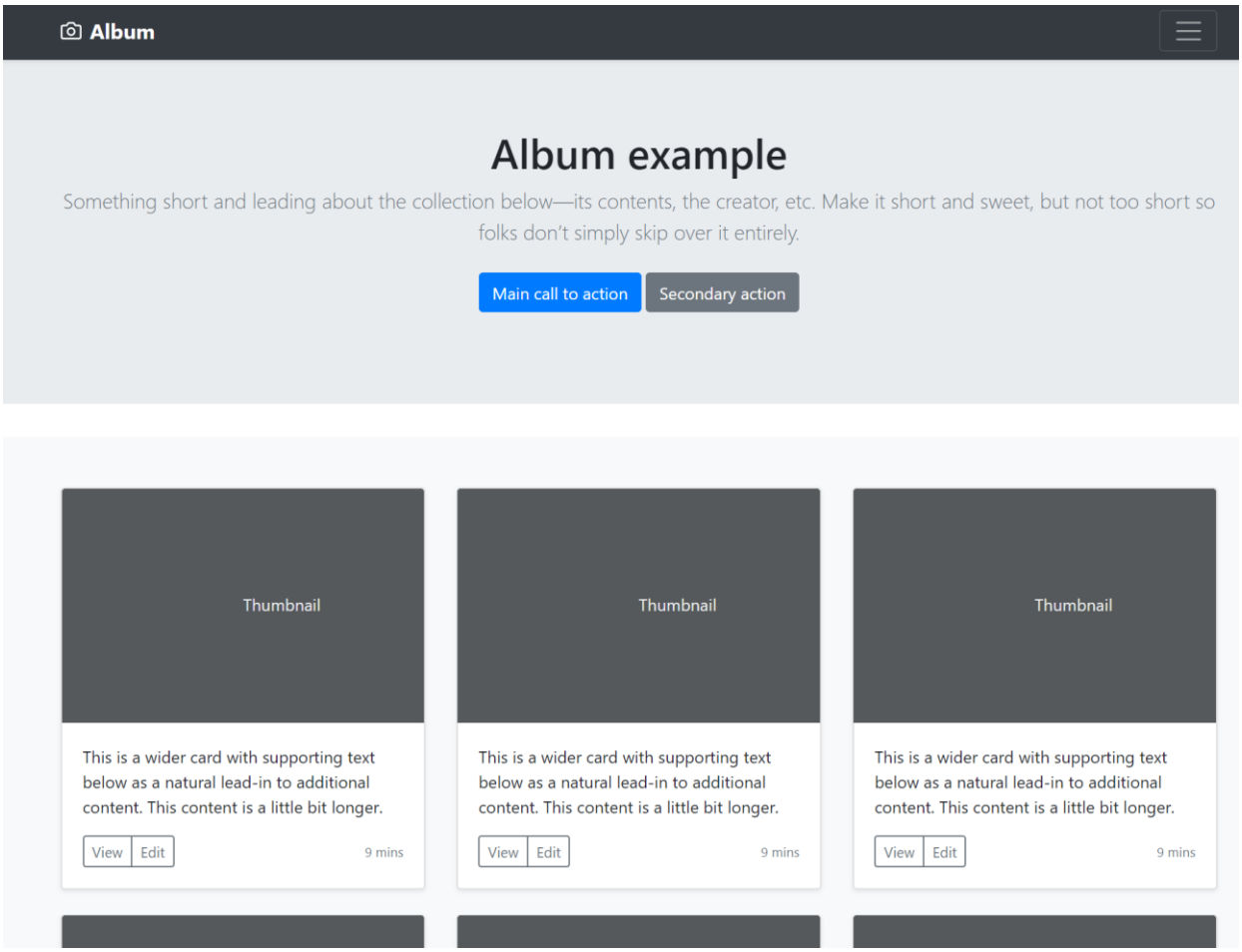
19. Since this is a public repository, you can verify the published repository and content on central GitHub repository without even login to GitHub account



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.



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;

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

```
<!-- 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.

DELL Technologies

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

DELL Technologies

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/

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

DELL Technologies

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 1	print (3 / 2) 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) 17.0	round(16.5) 16

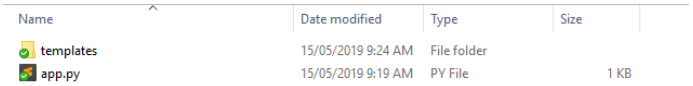
DELL Technologies

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

1. (If not already done in Lab 3), Download the files from Lab 04 – Python & Flask directory of the repository (link as below) and place into your working project folder

   https://github.com/theocrithary/Piper-2020/tree/master/Day%201/Lab%2004%20-%20Python%20%26%20Flask

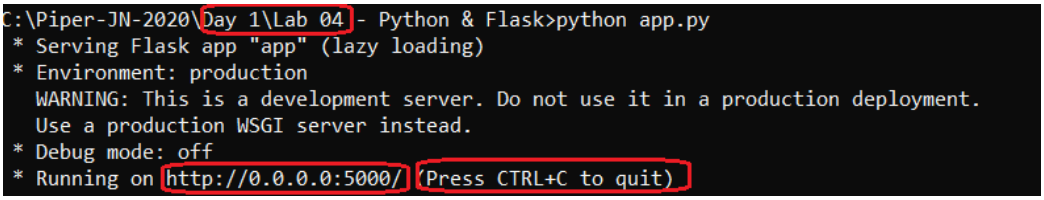
2. Your project directory should look like this;



3. Open a command prompt and navigate to your working project folder

4. Run the following command

   > python app.py



5. Open a browser with the following URL

   http://localhost:5000

6. Press CTRL + C to quit upon testing the webapp

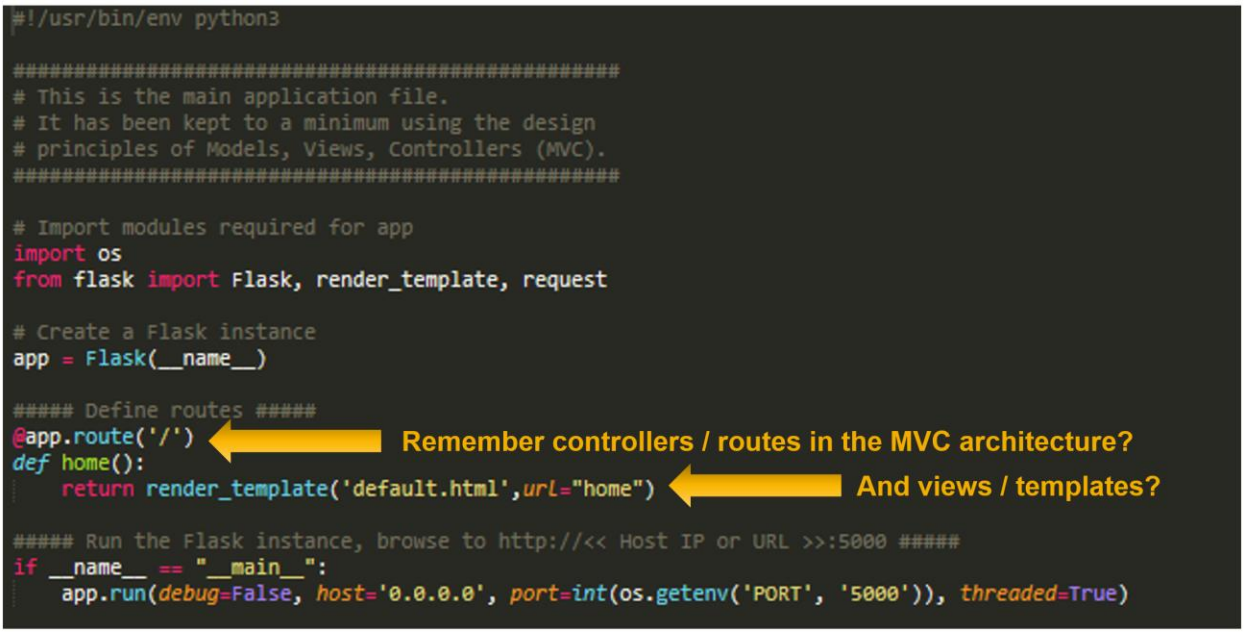
7. Save work in your working project folder and upload to Github

   - Follow Lab 3- Steps 9 to 19 (if need guidance on the steps for pushing the changes from local repository to central GitHub repository)

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.

DELL Technologies

Lab 5: MongoDB

Module Objectives:

- ❑ Setup local MongoDB server & install pymongo module (if not already; its pre-installed in the lab environment)
- ❑ MongoDB test for its 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

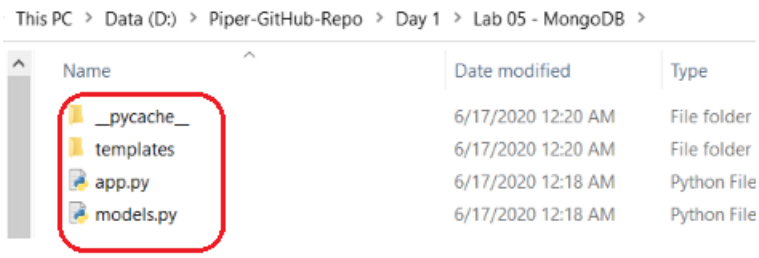
DELL Technologies

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

1. (If not already done in Lab 3), Download the following files from Lab 05 - MongoDB directory of the repository (link as below) and place into your working project folder

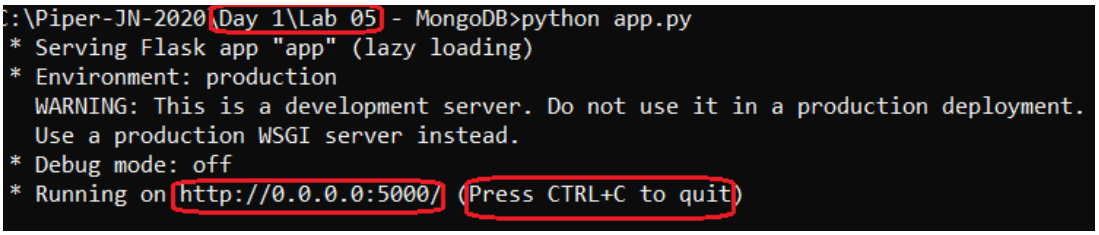
   https://github.com/theocrithary/Piper-2020/tree/master/Day%201/Lab%2005%20-%20MongoDB

2. Your project directory should look like this;



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

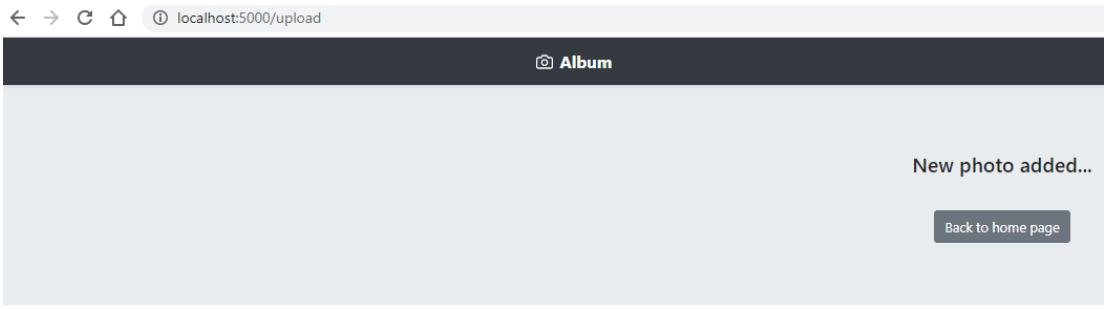
   > python app.py



4. Open a browser with the following URL

   http://localhost:5000

5. Test uploading an image and confirm upload



8. Press CTRL + C to quit upon testing the webapp

9. Save work in your working project folder and upload to Github (if any changes to your local working directory)

   - Follow Lab 3- Steps 9 to 19 (if need guidance on the steps for pushing the changes from local repository to central GitHub repository)

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.

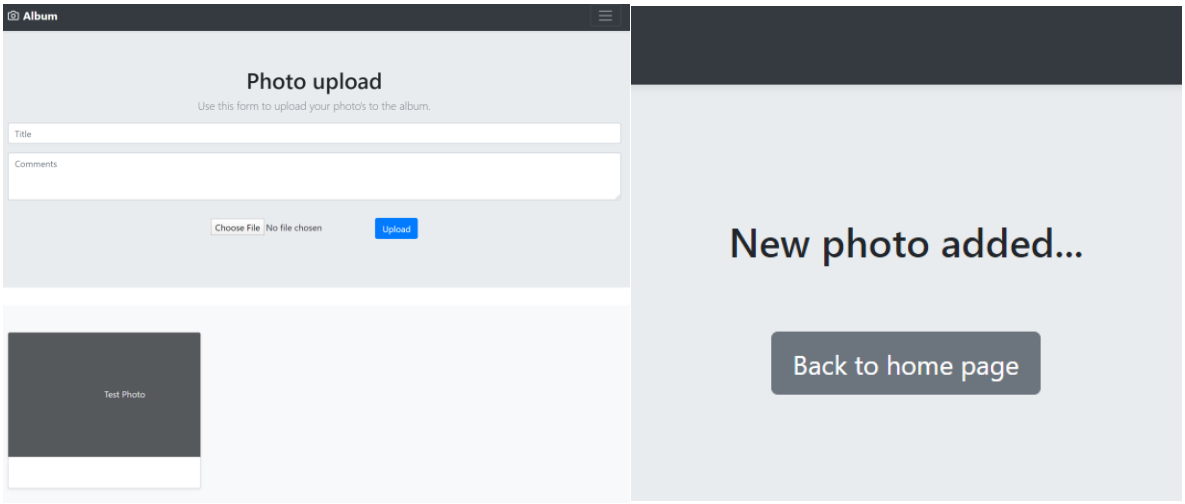
models.py

```
#!/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

### Applicaiton configuration settings ###
# Set the database target to your local MongoDB instance
client = MongoClient('127.0.0.1:27017')
DB_NAME = "mongodb"  # This will be the name of your database
COL_NAME = "photos"  # This will be the name of your collection


##### Main body code #####

db = client[DB_NAME] # Create the database using the name provided and client connection to the MongoDB server
db_collection = db[COL_NAME]   # Create the collection using the name provided and database connection

# Retrieve all photos records from database
def get_photos():
    return db_collection.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_collection.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' to retrieve all photo's from database
    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 process the database transaction
    return render_template('submit-photo.html')    # Return a page to inform the user of a successful upload

##### 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)
```



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. *That's 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 user's images and use the database to store the URL link to the images.

DELL Technologies

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

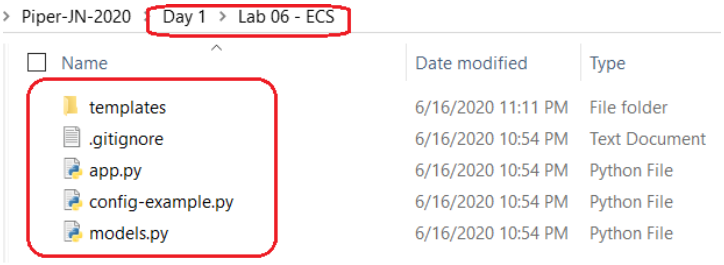
DELL Technologies

Lab Exercise:

1. (If not already done in Lab 3), Download the following files from Lab 06 - ECS directory of the repository (link as below) and place into your working project folder

   https://github.com/theocrithary/Piper-2020/tree/master/Day%201/Lab%2006%20-%20ECS

2. Your project directory should look like this;

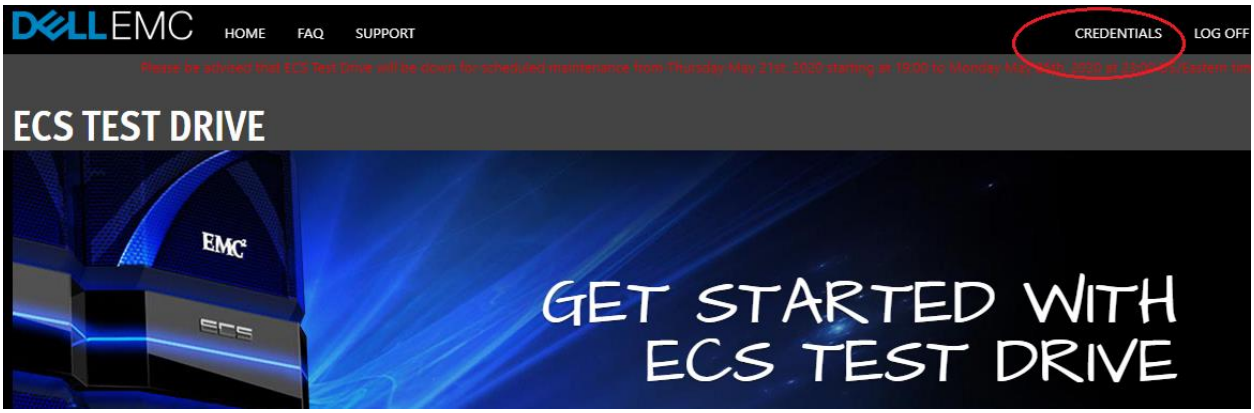


3. Verify and "make changes*" to the directory structure into your project directory as below

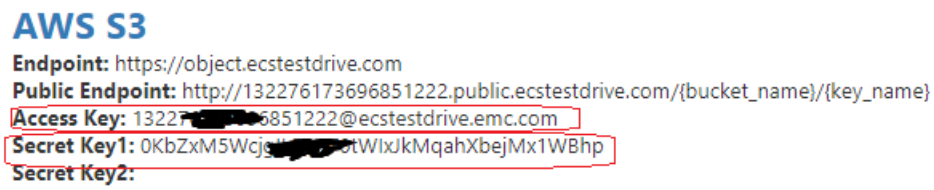
   - /templates
     - default.html
     - photo.html
     - submit-photo.html
   - app.py
   - models.py
   - **config.py (rename* the file from the provided config-example.py file)**
   - **/uploads (create* this as a new directory)**
   - .gitignore

4. Login to ECS Testdrive portal on https://portal.ecstestdrive.com/

5. Upon successful login, Retrieve your ECS credentials from the credentials link the page as following



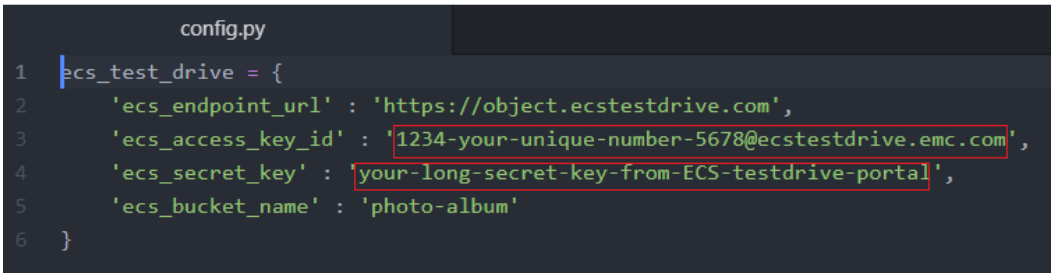
6. Take note of the Access key and secret key1 as following under the "AWS S3" section:



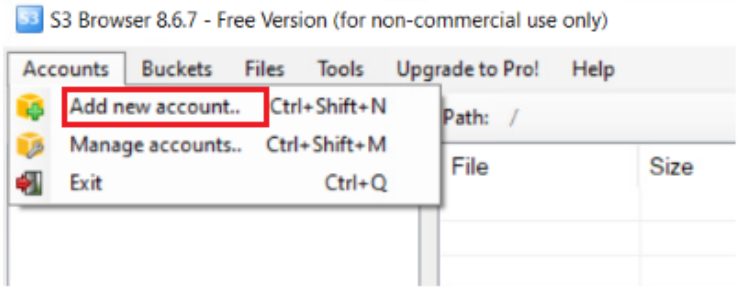
7. Edit the config.py file and replace

   - ecs_access_key_id with your access key (noted as above)
   - ecs_secret_key with your secret key1 (noted as above)

   NOTE- keep the opening and closing indents ('____') of the replaced values as-is



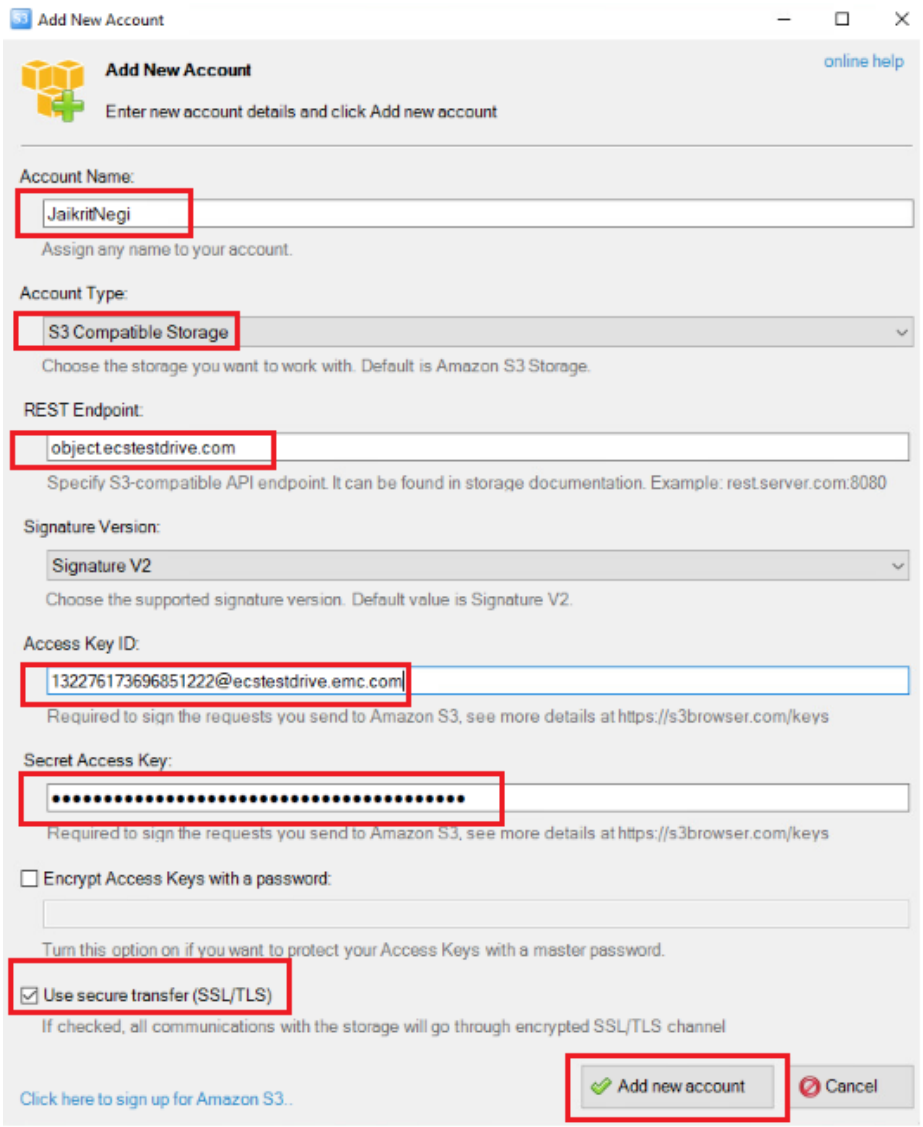
8. Now, In order to create the ecs_bucket_name as photo-album (as shown in above snapshot), Open the S3Browser from the shortcut icon in the Desktop and Click on "Accounts > Add new account"



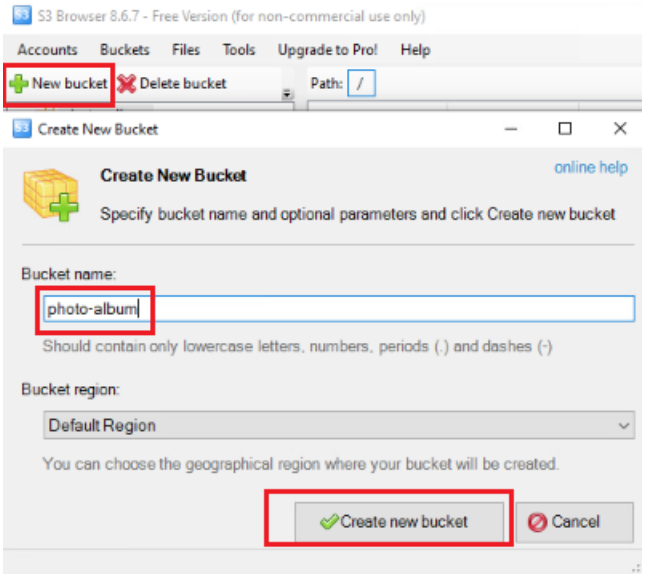
DELL Technologies

9. Key in the following fields as described in the snapshot below & Click on Add new account

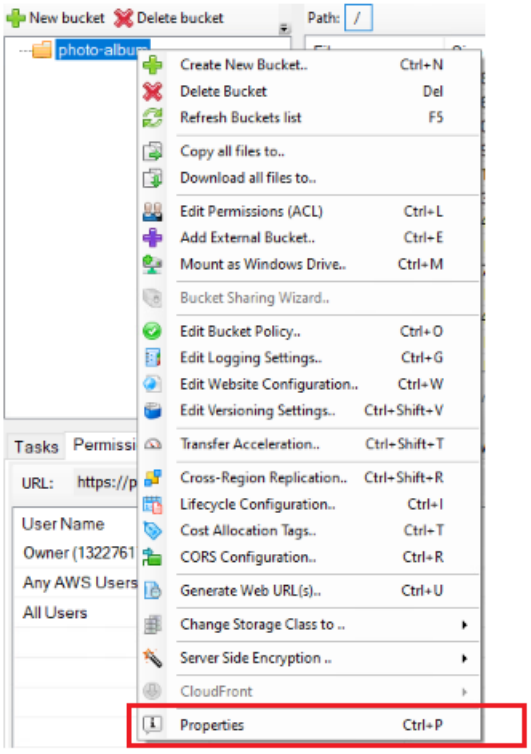
   - Account Name- Use “YourName”
   - Account Type- S3 Compatible Storage
   - REST Endpoint- object.ecstestdrive.com (NOTE- without ‘https://’)
   - Access Key ID- your access key from ecs testdrive portal (noted as above in Step-6)
   - Access Key ID- your access key from ecs testdrive portal (noted as above in Step-6)
   - Ensure Tick on “SSL/TLS”

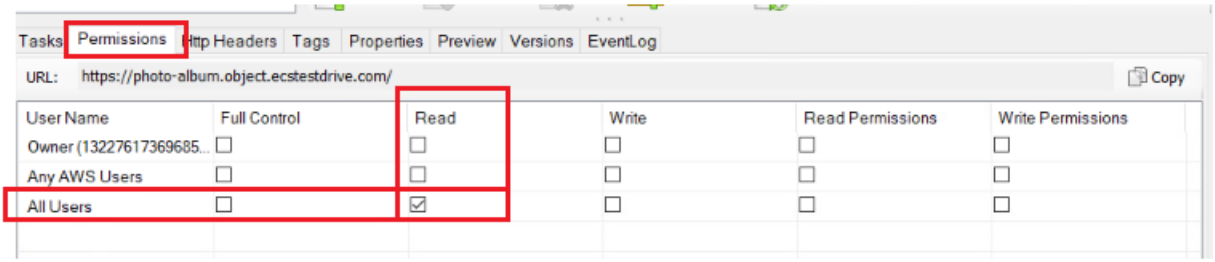


10. Create a bucket called photo-album. Use lowercase characters only.



11. Pay close attention to bucket and object permissions tab





DELL Technologies

12. Install Boto3 & Pillow (if not already installed)

    > pip install boto3

```
D:\Piper-GitHub-Repo\Day 1\Lab 06 - ECS>pip install boto3
Collecting boto3
  Downloading boto3-1.14.4-py2.py3-none-any.whl (128 kB)
     |████████████████████████████████| 128 kB 3.3 MB/s
Collecting botocore<1.18.0,>=1.17.4
  Downloading botocore-1.17.4-py2.py3-none-any.whl (6.3 MB)
     |████████████████████████████████| 6.3 MB 3.3 MB/s
Requirement already satisfied: s3transfer<0.4.0,>=0.3.0 in c:\users\demouser\appdata\local\programs\python\python37\lib\site-packages
(from boto3) (0.3.3)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in c:\users\demouser\appdata\local\programs\python\python37\lib\site-packages (f
rom boto3) (0.9.5)
Requirement already satisfied: docutils<0.16,>=0.10 in c:\users\demouser\appdata\local\programs\python\python37\lib\site-packages (fro
m botocore<1.18.0,>=1.17.4->boto3) (0.15.2)
Requirement already satisfied: urllib3<1.26,>=1.20; python_version != "3.4" in c:\users\demouser\appdata\local\programs\python\python3
7\lib\site-packages (from botocore<1.18.0,>=1.17.4->boto3) (1.25.9)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in c:\users\demouser\appdata\local\programs\python\python37\lib\site-packag
es (from botocore<1.18.0,>=1.17.4->boto3) (2.8.1)
Requirement already satisfied: six>=1.5 in c:\users\demouser\appdata\local\programs\python\python37\lib\site-packages (from python-dat
eutil<3.0.0,>=2.1->botocore<1.18.0,>=1.17.4->boto3) (1.14.0)
Installing collected packages: botocore, boto3
  Attempting uninstall: botocore
    Found existing installation: botocore 1.15.40
    Uninstalling botocore-1.15.40:
      Successfully uninstalled botocore-1.15.40
Successfully installed boto3-1.14.4 botocore-1.17.4
WARNING: You are using pip version 20.0.2; however, version 20.1.1 is available.
You should consider upgrading via the 'c:\users\demouser\appdata\local\programs\python\python37\python.exe -m pip install --upgrade pi
p' command.
```

    > pip install pillow

```
D:\Piper-GitHub-Repo\Day 1\Lab 06 - ECS>pip install pillow
Collecting pillow
  Downloading Pillow-7.1.2-cp37-cp37m-win_amd64.whl (2.0 MB)
     |████████████████████████████████| 2.0 MB 2.2 MB/s
Installing collected packages: pillow
Successfully installed pillow-7.1.2
WARNING: You are using pip version 20.0.2; however, version 20.1.1 is available.
You should consider upgrading via the 'c:\users\demouser\appdata\local\programs\python\python37\python.exe -m pip install --upgrade pi
p' command.
```

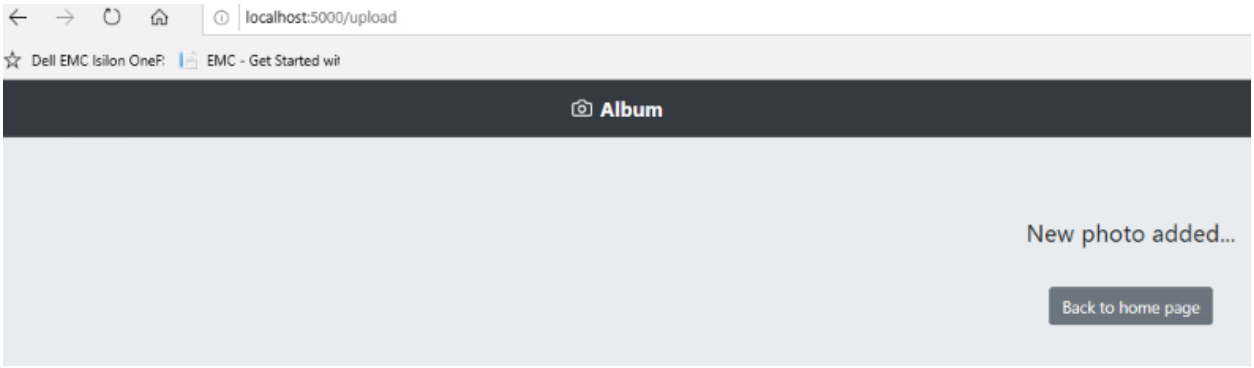
13. Run your app

    > python app.py

```
D:\Piper-GitHub-Repo\Day 1\Lab 06 - ECS>python app.py
 * 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://0.0.0.0:5000/ (Press CTRL+C to quit)
```

14. Test your app and upload an image;

    http://localhost:5000



15. Press CTRL + C to quit upon testing the webapp
16. Save work in your working project folder and upload to Github (if any changes to your local working directory)
    - Follow Lab 3- Steps 9 to 19 (if need guidance on the steps for pushing the changes from local repository to central GitHub repository)

DELL Technologies

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.

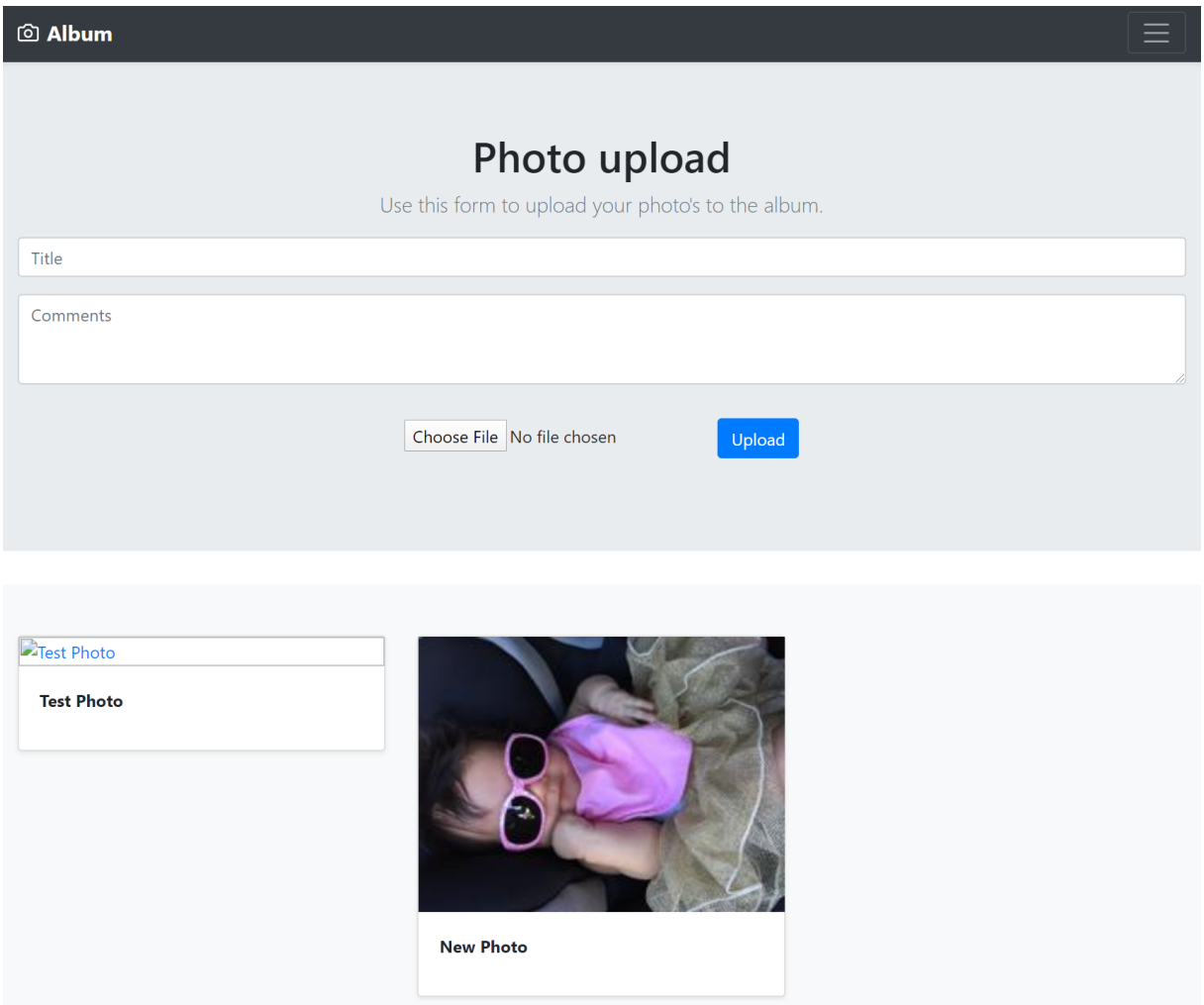
config-example.py

```
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'
}
```

.gitignore

```
1  config.py
2  __pycache__
```

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


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

DELL Technologies

Lab 7: Hosted TAS (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

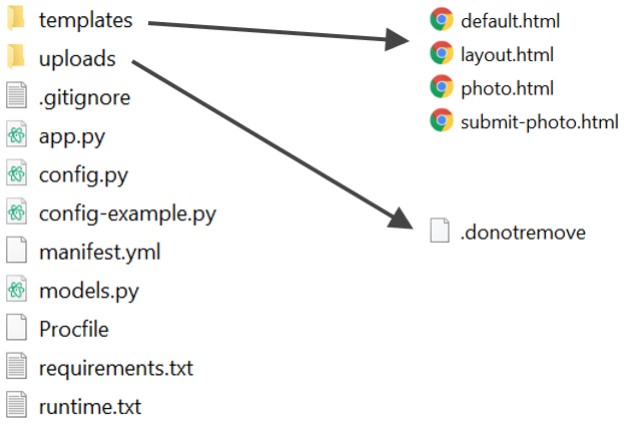
DELL Technologies

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

1. (If not already done in Lab 3), Download the following files from Lab 07 - PWS & mLab directory of the repository (link as below) and place into your working project folder

   https://github.com/theocrithary/Piper-2020/tree/master/Day%201/Lab%2007%20-%20PWS%20%26%20mLab

2. Your project directory should look 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

```
D:\Piper-GitHub-Repo\Day 1\Lab 07 - PWS & mLab>cf login -a https://api.run.pivotal.io
API endpoint: https://api.run.pivotal.io

Email: jaikrit.negi@dell.com

Password:
Authenticating...
OK

Targeted org Jaikrit

Targeted space development



API endpoint:   https://api.run.pivotal.io (API version: 3.82.0)
User:           jaikrit.negi@dell.com
Org:            Jaikrit
Space:          development
```

4. Push the app to PWS

> cf push

```
D:\Piper-GitHub-Repo\Day 1\Lab 07 - PWS & mLab>cf push
Pushing from manifest to org Jaikrit / space development as jaikrit.negi@dell.com...
Using manifest file D:\Piper-GitHub-Repo\Day 1\Lab 07 - PWS & mLab\manifest.yml
Getting app info...
Creating app with these attributes...
+ name:        photo-album
  path:        D:\Piper-GitHub-Repo\Day 1\Lab 07 - PWS & mLab
+ instances:   1
+ memory:      64M
  routes:
+   photo-album-quick-mandrill-pd.cfapps.io

Creating app photo-album...
Mapping routes...
Comparing local files to remote cache...
Packaging files to upload...
Uploading files...
 7.02 KiB / 7.02 KiB [================================================================================

Waiting for API to complete processing files...

Staging app and tracing logs...
   Downloading web_config_transform_buildpack...
   Downloading nodejs_buildpack...
   Downloading go_buildpack...
   Downloading staticfile_buildpack...
   Downloading python_buildpack...
   Downloaded nodejs_buildpack
   Downloading php_buildpack...
   Downloaded python_buildpack
   Downloading binary_buildpack...
   Downloaded go_buildpack
   Downloading dotnet_core_buildpack_beta...
   Downloaded web_config_transform_buildpack
   Downloading ruby_buildpack...
   Downloaded staticfile_buildpack
```

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

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

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

DELL Technologies

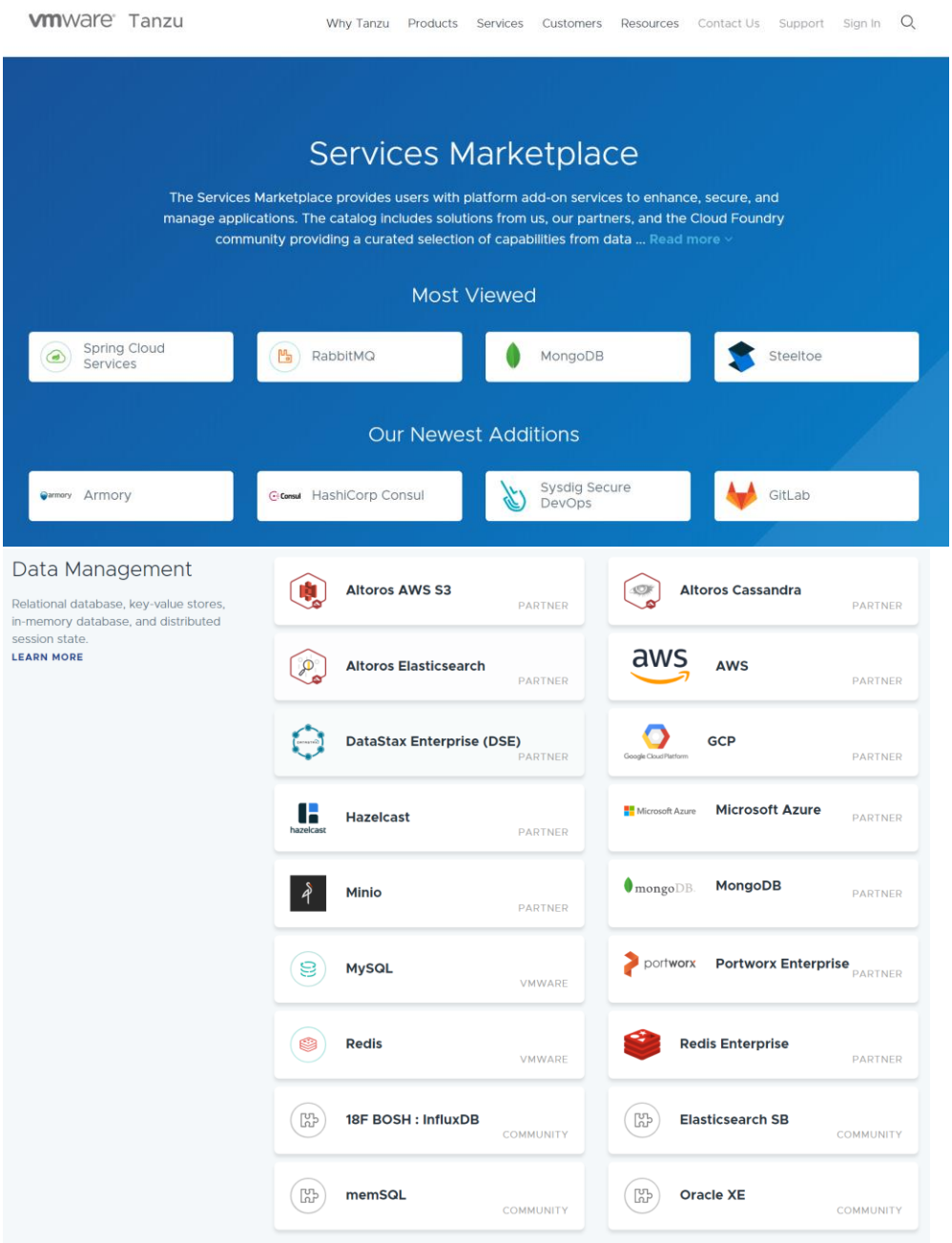
Tanzu Services Marketplace

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



Student Lab Guide

DELL Technologies

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.

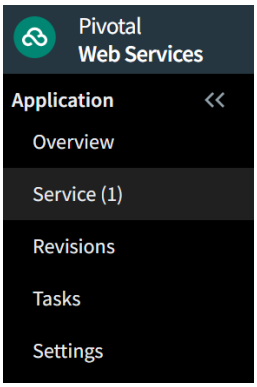
Continue to the next part of the lab (Part B) for instructions to deploy and bind the mLab service to our application.

DELL Technologies

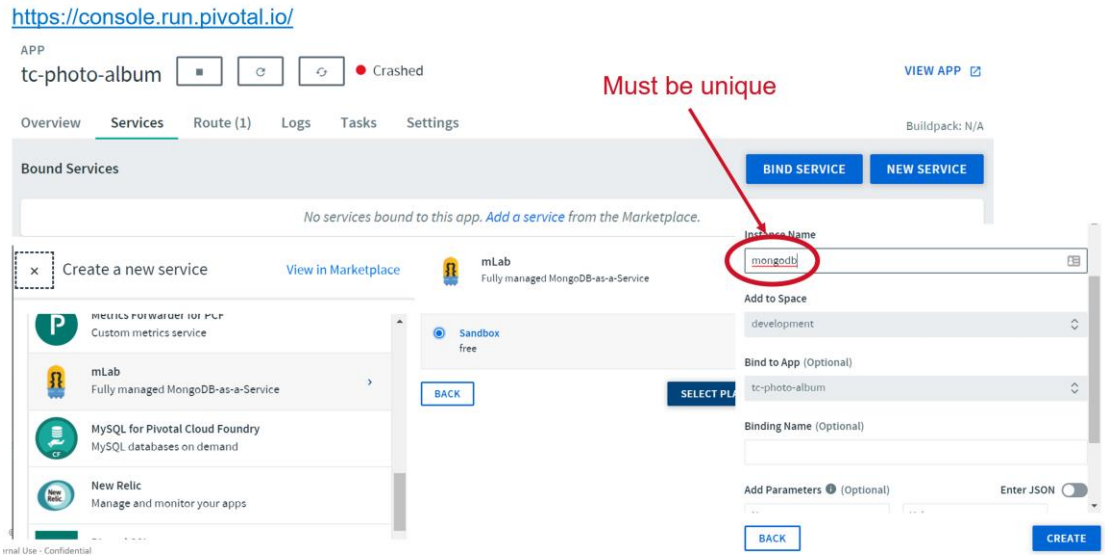
Lab Exercise (Part B): Add the external MongoDB service from the PWS Marketplace

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

7. Find your deployed application (photo-album) and click on the 'Service' menu option on the left

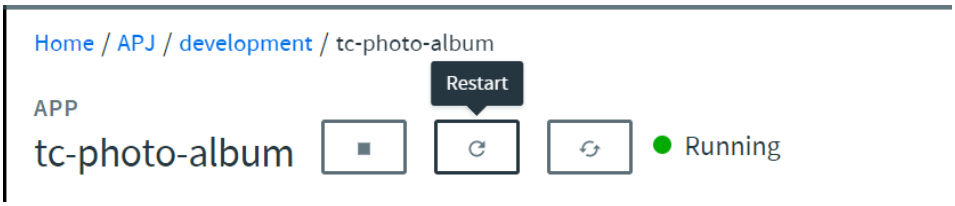


8. Click on the 'NEW SERVICE' button and create an mLab instance to bind to your app



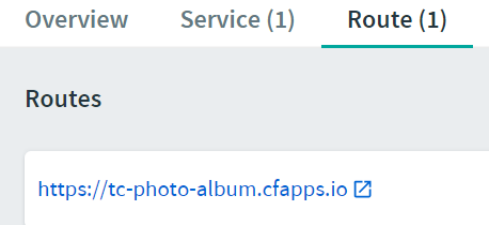
9. Use the default options for everything except the Instance Name (mongodb), which needs to be unique

10. Restart your app from the console if required



11. Open the app

12. Test functionality and upload an image



13. Save work in your working project folder and upload to Github (if any changes to your local working directory)
    - Follow Lab 3- Steps 9 to 19 (if need guidance on the steps for pushing the changes from local repository to central GitHub repository)

DELL Technologies

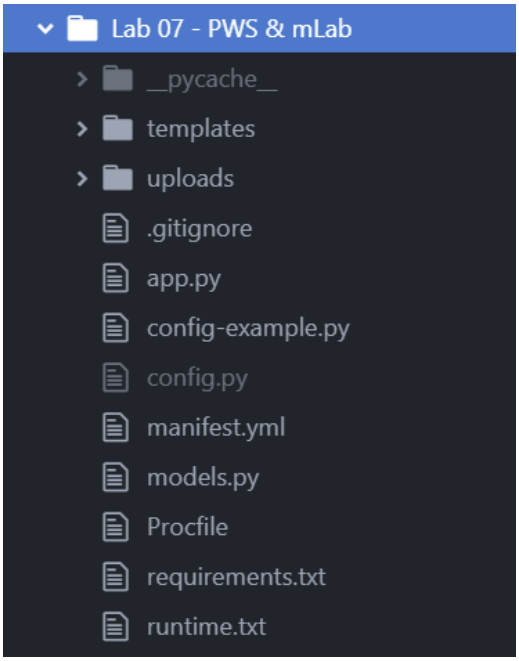
Retrospective: Review code and take note of the 'VCAP_SERVICES' section in models.py

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.

```
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.



manifest.yml

```
---
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
```