



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

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

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

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

© 2020 Dell Inc. or its subsidiaries. All Rights Reserved. Dell, EMC, Dell Technologies and other trademarks are trademarks of Dell Inc. or its subsidiaries. Other trademarks may be trademarks of their respective owners.

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

Student Lab Guide



Table of Contents

Lab 1: Github	5
Module Objectives:.....	5
What is Git?.....	6
How to use .gitignore to avoid committing sensitive data	7
How to use environment variables to store and retrieve sensitive data.....	8
How to use Github and Github desktop	9
Lab Exercise: Create a new repo and publish a file with your name	10
Lab 2a: Deploy an App to PWS	18
Module Objectives:.....	18
What is Pivotal Web Services (PWS)?.....	19
Lab Exercise: Clone an example app and deploy to PWS	20
Reflect on what just happened	23
Retrospective: The results.....	24
Lab 2b: Deploy an App to PWS (Alternate “Go App”)	25
Module Objectives:.....	25
Lab Exercise: Clone an example app and deploy to PWS	26
Reflect on what just happened	29
Retrospective: The results.....	30
Lab 3: Create a Static HTML Website	31
Module Objectives:.....	31
Lab Exercise: Copy the static webpage provided and setup a new project.....	32
Retrospective: A basis for an MVP	38
Lab 4: Python and Flask	39
Module Objectives:.....	39
First a bit about Python & Flask	40
What's changed in Python 3?.....	41
Lab Exercise: Create a basic Python app to run the Flask web server	42
Retrospective: Applying concepts of Model, Views, Controllers	43
Lab 5: MongoDB	44
Module Objectives:.....	44
Lab Exercise: Copy the new code files and overwrite existing application files	45
Retrospective: Review the implementation of ‘models’ into our application	46



Lab 6: ECS	48
Module Objectives:.....	48
Lab Exercise:.....	49
Retrospective: Review code and note implementation of Boto3 & Pillow.....	55
Lab 7: Hosted TAS (PWS) & mLab	58
Module Objectives:.....	58
Lab Exercise (Part A): Create the pre-requisite platform files and upload the app to Hosted TAS (PWS)	59
Tanzu Services Marketplace	61
Pivotal Web Services Marketplace.....	62
Lab Exercise (Part B): Add the external MongoDB service from the PWS Marketplace	63
Retrospective: Review code and take note of the 'VCAP_SERVICES' section in models.py	65



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



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

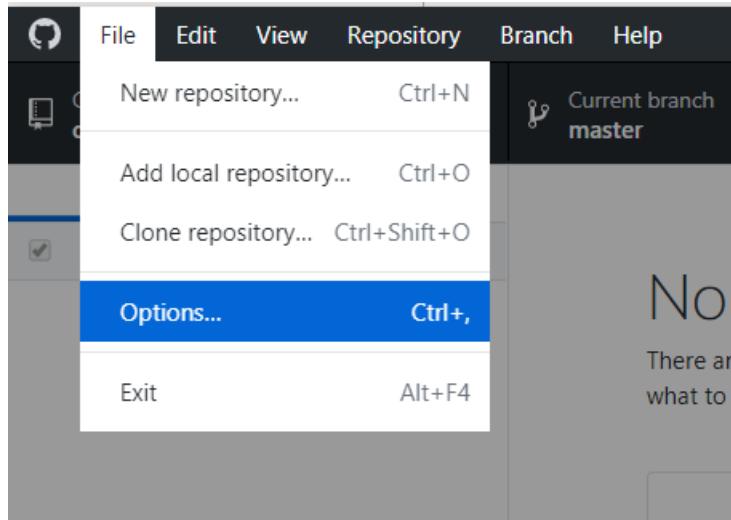
- If you do not know how to use Github, please complete the following tutorial
<https://guides.github.com/activities/hello-world/>
- If you do not know how to use Github Desktop GUI client, please complete the following tutorial
<https://help.github.com/desktop/guides/getting-started-with-github-desktop/>



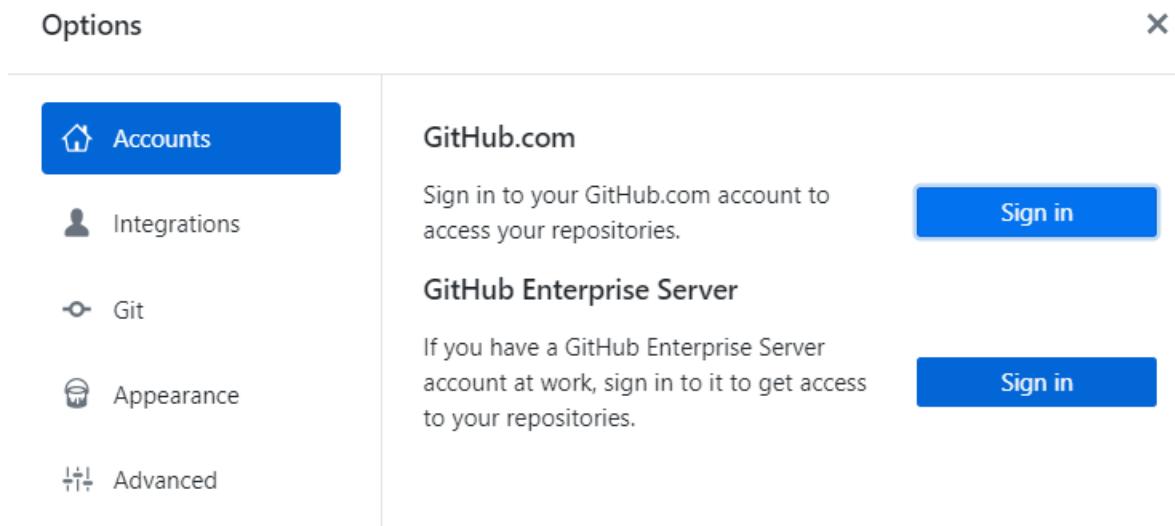
Lab Exercise: Create a new repo and publish a file with your name

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



Sign in

X

Sign in using your browser

or

Username or email address

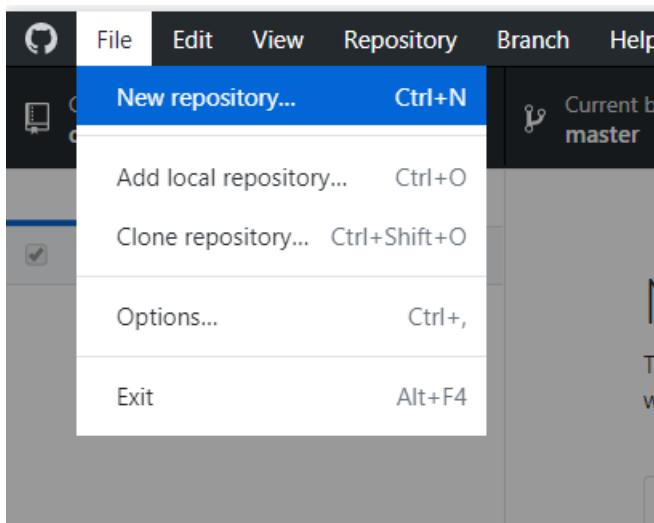
Password

[Forgot password?](#)

[Sign in](#)

[Cancel](#)

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

Create a new repository X

Name

Piper-GitHub-Repo

Description

Local path

D:\

Choose...

 Initialize this repository with a README

Git ignore

None

▼

License

None

▼

Create repository

Cancel

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

Error



Commit failed - exit code 128 received, with output: *** Please tell me who you are.

Run

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

to set your account's default identity.

Omit --global to set the identity only in this repository.

```
fatal: empty ident name (for <>) not allowed'
```

Close

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 &

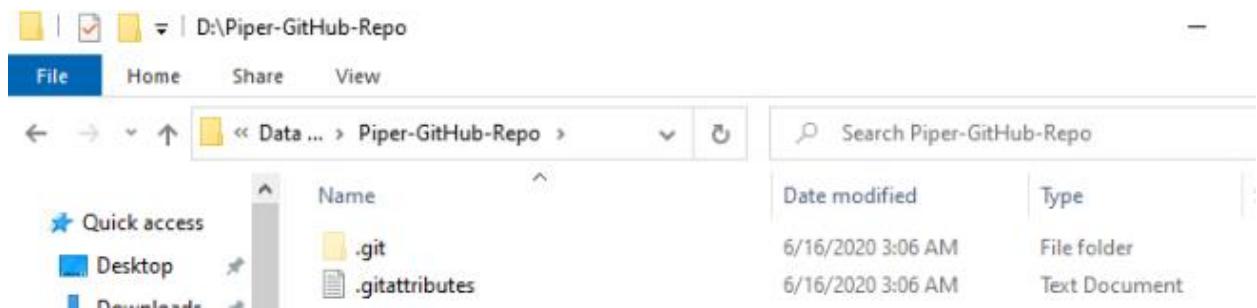


2. replacing last field in 2nd command with your own Name

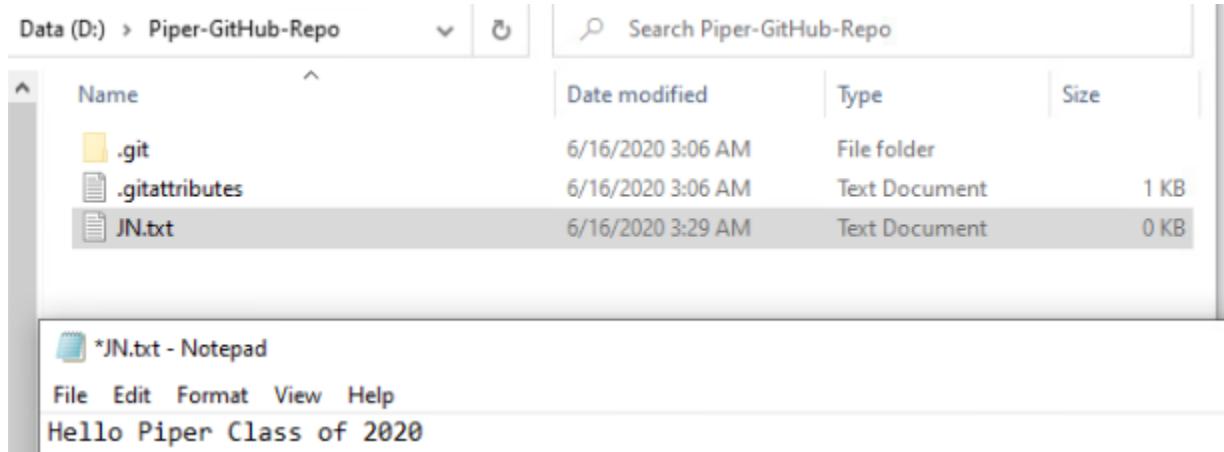
```
Command Prompt
Microsoft Windows [Version 10.0.18363.836]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\demouser>git config --global user.email y_78_id@yahoo.com
C:\Users\demouser>git config --global user.name JaikritNegi
```

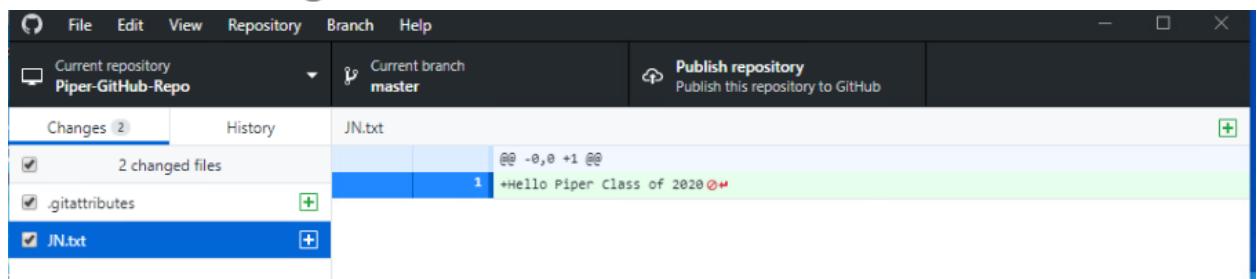
9. Open Windows Explorer and find the folder in the D:\ drive created by GitHub Desktop Client with the folder name chosen as repository name (Piper-GitHub-Repo) with below structure



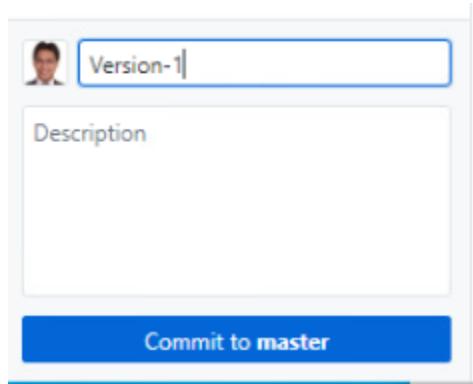
10. Create a text file with your name as the filename and add the content of the text file with the words “Hello Piper class of 2020!” and save the file



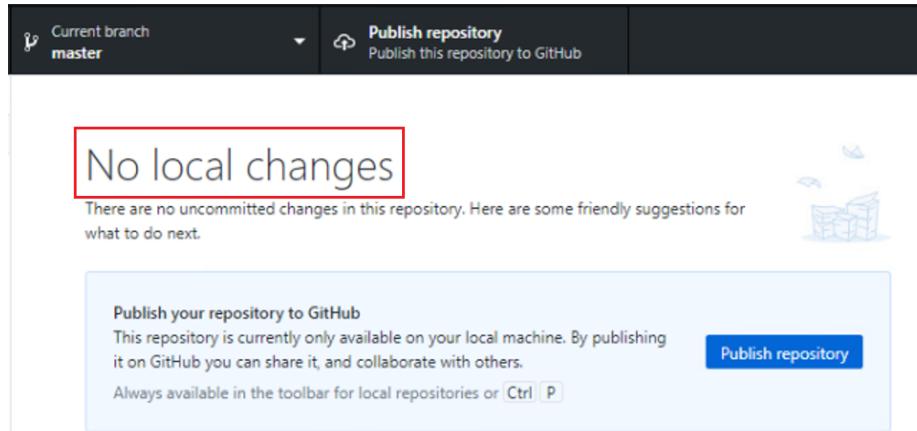
11. Go to already opened instance of GitHub Desktop Client and notice the newly created file and content and shows the changed files.



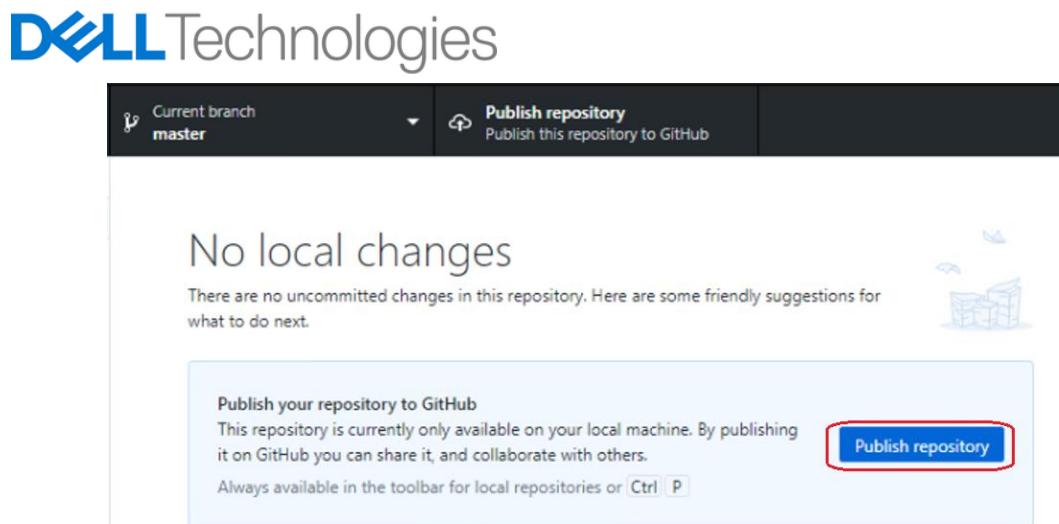
12. The current change in file is in working directory and tracked by GitHub client. In order to commit the changes to local directory, type the comment in the summary field and click on commit to master



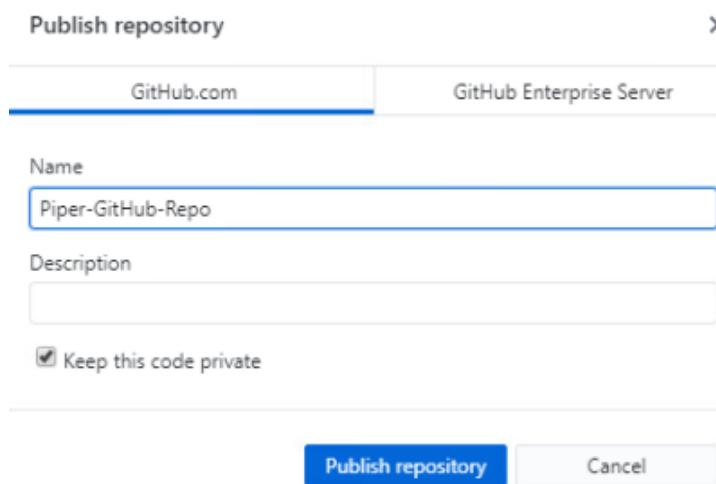
13. Above action should show below screen confirming commit with display as No local changes



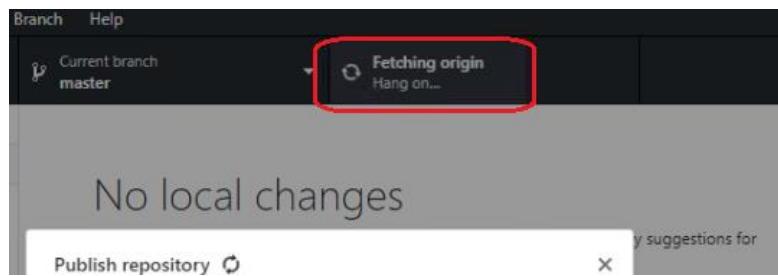
14. Publish the local repository to central repository by clicking on Publish repository

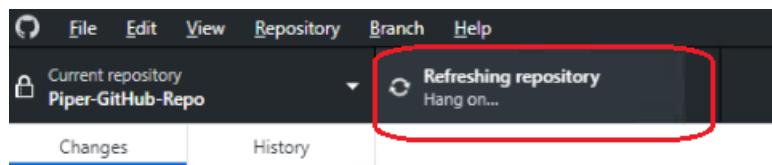


15. Upon which choose to keep it Private (visible to self only) or Public (visible to all on GitHub) and Click on Publish repository.

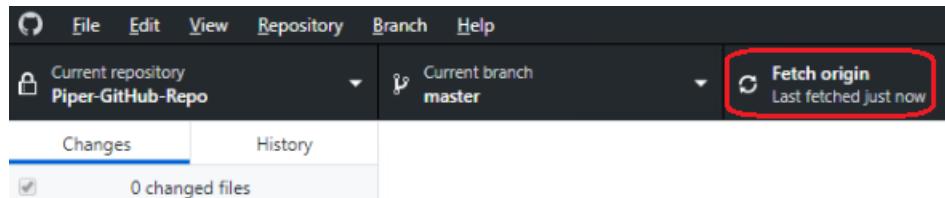


16. Notice the activity after publishing the repository to central GitHub repo.

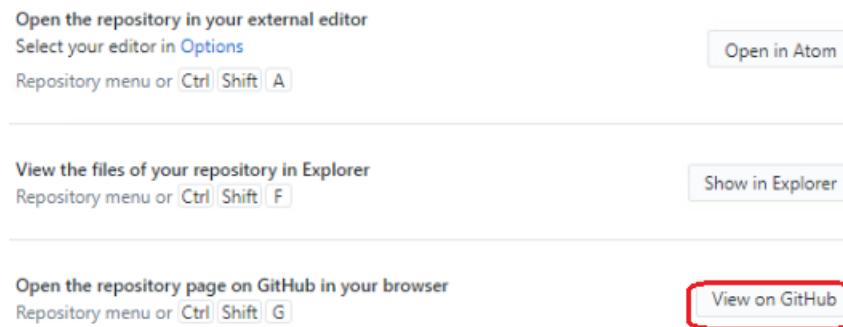




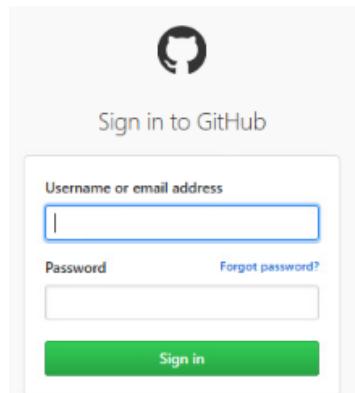
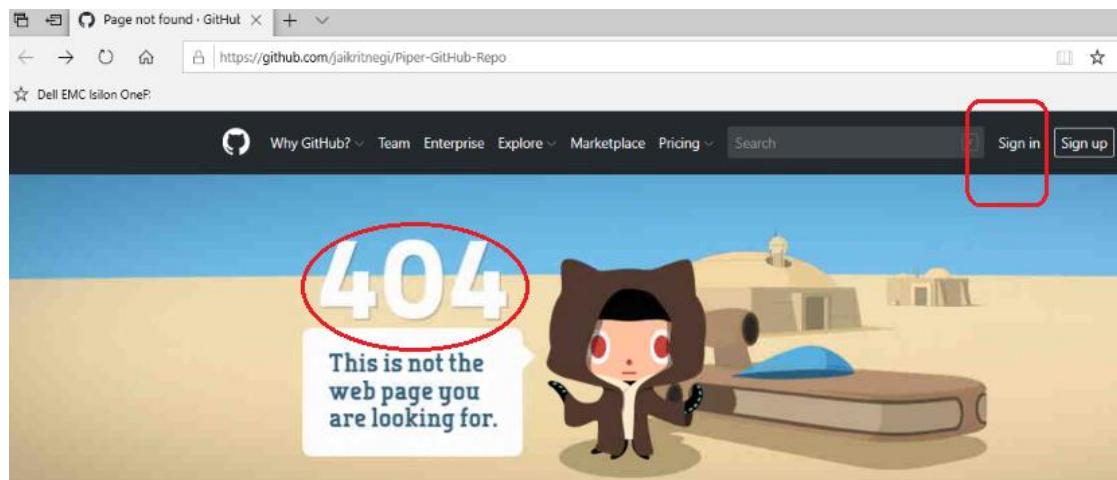
17. Once the repository is published to central GitHub repo, notice the message as Last fetched just now.



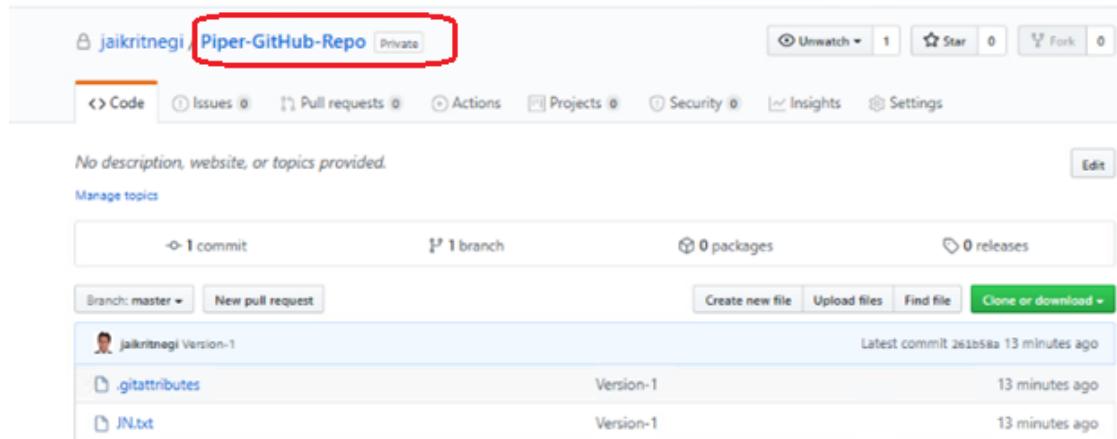
18. Once the repository is published, click on View on GitHub option that opens a browser window to verify the repository on GitHub in browser



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 GitHub credentials to view the published repository.



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





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



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.

The screenshot shows the Pivotal Web Services (PWS) dashboard. On the left, there is a sidebar with links: Home, Marketplace, Tools, Docs, Support, Blog, Status, and a GIVE FEEDBACK button. The main area has a search bar at the top. Below it, there's an ORG section for "Theo's Org" showing a QUOTA of 0 MB / 25 GB (0%). There are tabs for Space (1), Domains (3), Member (1), and Settings. Under the Space tab, there's a "Spaces" section with a table. The table has columns: Name, Apps, App Status, Services, and Org Quota Usage. It lists one space named "development" with 0 Apps, 0 Green status, 0 Black status, 0 Red status, 0 Services, and 0 Bytes / 25 GB (0%). There is a "CREATE NEW SPACE" button at the bottom right of the table. At the top right of the dashboard, there are links for "press" and "theo.critchary@dell.com".

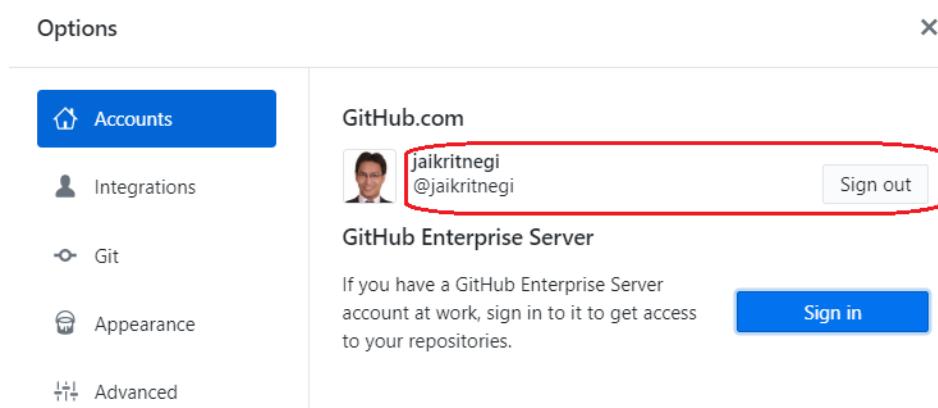
Important note: Please create an organisation before continuing with the lab



Lab Exercise: Clone an example app and deploy to PWS

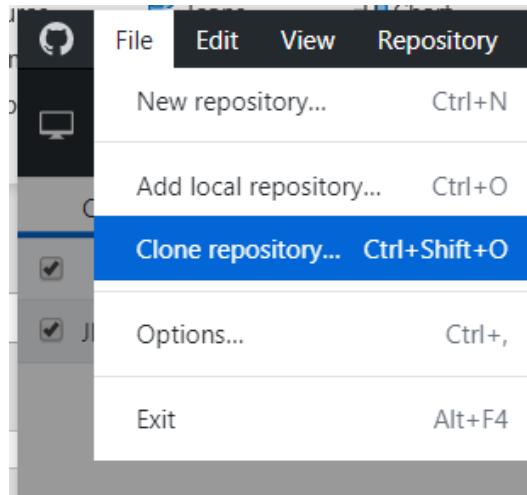
Complete the below steps to demonstrate your understanding of the tools and concepts required for the remaining lab exercises;

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.



Clone a repository

GitHub.com GitHub Enterprise Server URL

Repository URL or GitHub username and repository
(`hubot/cool-repo`)

Local path

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
```



```
D:\cf-sample-app-spring>cf push [REDACTED]
Pushing from manifest to org Jaikrit / space development as jaikrit.negi@dell.com...
Using manifest file D:\cf-sample-app-spring\manifest.yml
Setting 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: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=8354 -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/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
#0   running    2019-05-14T02:49:40Z  0.0%   78.8M of 768M  121.1M of 1G  details
```

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.

```

Pushing from manifest to org Theo's Org / space development as theo.critchary@dell.com...
Using manifest file C:\Users\CRITCHY\OneDrive - Dell Technologies\Documents\Sites\cf-sample-app\manifest.yml
Setting app info...
Creating app with these attributes...
  name: cf-demo
  path: C:\Users\CRITCHY\OneDrive - Dell Technologies\Documents\Sites\cf-sample-app
  buildpacks:
    - https://github.com/cloudfoundry/java-buildpack.git
  instances: 1
  memory: 768M
  routes:
    - cf-demo-responsome.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 1d875das5-0fc1-43db-a1a8-e3e2c66b4a15 creating container for instance e3d1dbe3-569d-47bb-be75-3cb64aeabb4b0
Cell 1d875das5-0fc1-43db-a1a8-e3e2c66b4a15 successfully created container for instance e3d1dbe3-569d-47bb-be75-3cb64aeabb4b0
Downloading app package...
Downloaded app package (1.2M)
----> Downloading buildpack 542d66c | https://github.com/cloudfoundry/java-buildpack.git#542d66c
----> Downloading client-certificate-mapper-1.8.0_RELEASE.jar from https://java-buildpack.cloudfoundry.org/jvmkill/bionic/x86_64/jvmkill-1.16.0-RELEASE.so (0.0s)
----> Downloading openjdk_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 BOOSH 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/1.8.0-RELEASE.jar (0.0s)
----> Downloading Container Security Provider 1.10.0_RELEASE from https://java-buildpack.cloudfoundry.org/container-security-provider/container-security-provider-1.10.0-RELEASE.jar (0.0s)
----> Downloading Spring Boot CLI 2.1.4 RELEASE from https://java-buildpack.cloudfoundry.org/spring-boot-cli-2.1.4-RELEASE.tar.gz (0.0s)
  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 1d875das5-0fc1-43db-a1a8-e3e2c66b4a15 stopping instance e3d1dbe3-569d-47bb-be75-3cb64aeabb4b0
Cell 1d875das5-0fc1-43db-a1a8-e3e2c66b4a15 destroying container for instance e3d1dbe3-569d-47bb-be75-3cb64aeabb4b0

Waiting for app to start...

```



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

The screenshot shows the Cloud Foundry application details for 'cf-demo'. At the top, there's a header with the application name and a 'Wombat' icon. Below the header is a table with the following data:

Buildpack	Java/Spring	Buildpacks
App Name	cf-demo	Routes & Domains
Instance Index	0	Scaling
Space Name	development	Orgs & Spaces
There aren't any services bound to this app.		Manage Services

At the bottom, there's a note: "This is a Cloud Foundry sample application." with a small icon.



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

Module Objectives:

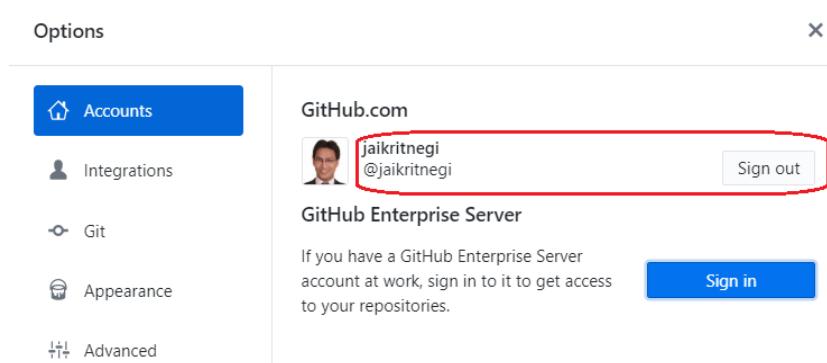
- Learn about PWS & cf cli tools
- Confirm cf cli is installed and working
- Login to PWS via cf cli
- Push a demo app to PWS and confirm we have the skills and tools required to publish a web app to the PWS PaaS platform



Lab Exercise: Clone an example app and deploy to PWS

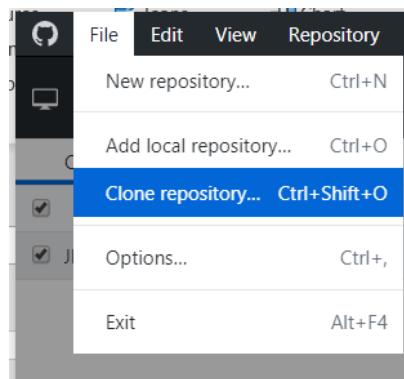
Complete the below steps to demonstrate your understanding of the tools and concepts required for the remaining lab exercises;

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

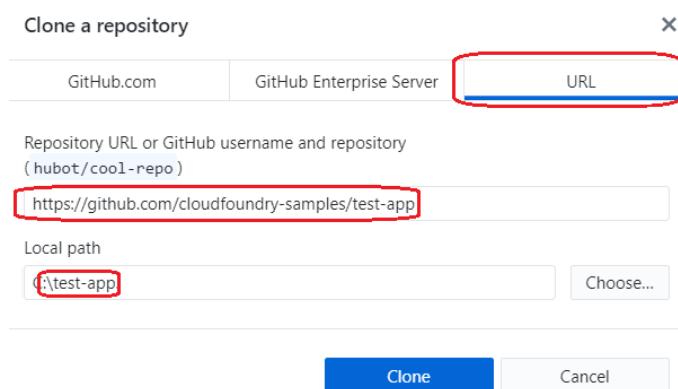


2. Clone the following repository – “Go to File > Clone repository” to clone it from GitHub

<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.





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
```



```
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)



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.

```
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-ua.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...

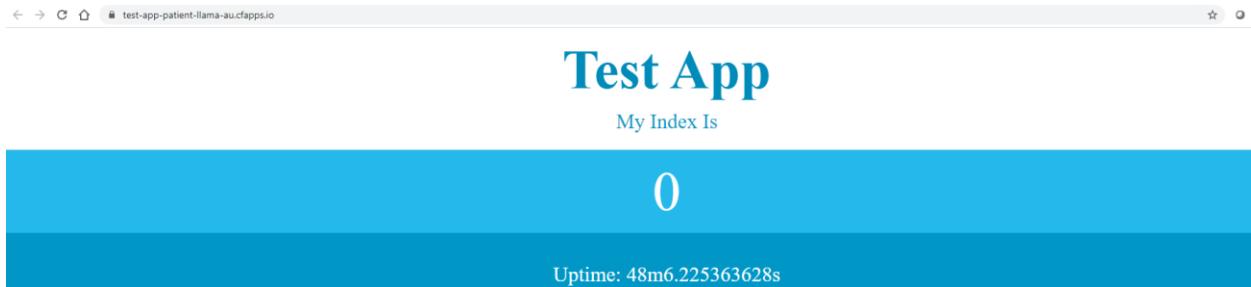
Staging app and tracing logs...
Downloading web_config_transform_buildpack...
Downloading binary_buildpack...
Downloading staticfile_buildpack...
Downloading dotnet_core_buildpack_beta...
Downloading go_buildpack...
Downloaded web_config_transform_buildpack
Downloading java_buildpack...
Downloaded binary_buildpack
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
Downloaded 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
Cell 732975b5-a95c-4e37-b595-a0a3c3a9e2ea successfully created container for instance 8296c487-ed49-4981-8104-b1144db770f4
Downloading app package...
Downloaded app package (37.3K)
ERROR: No buildpack groups passed detection.
```

```
manifest.yml
1  ---
2  applications:
3    - name: test-app
4      memory: 256M
5      instances: 1
6      random-route: true
7
```



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;





Lab 3: Create a Static HTML Website

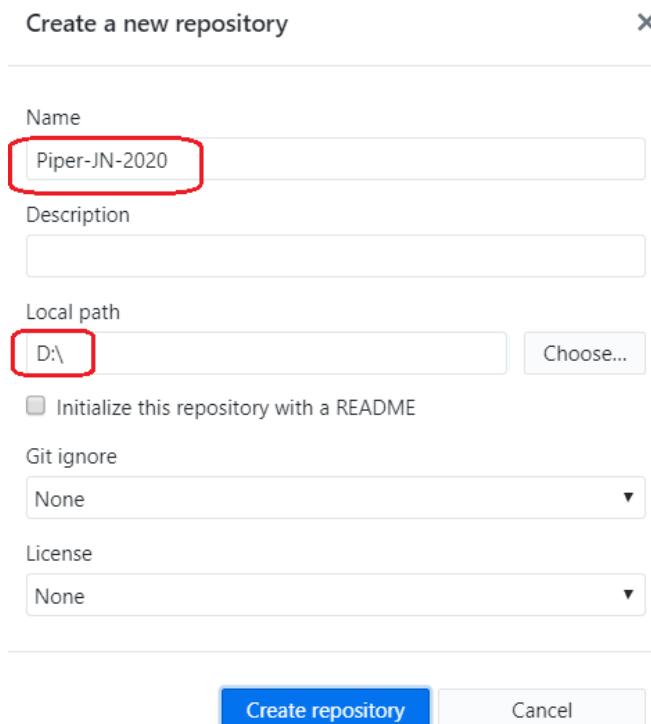
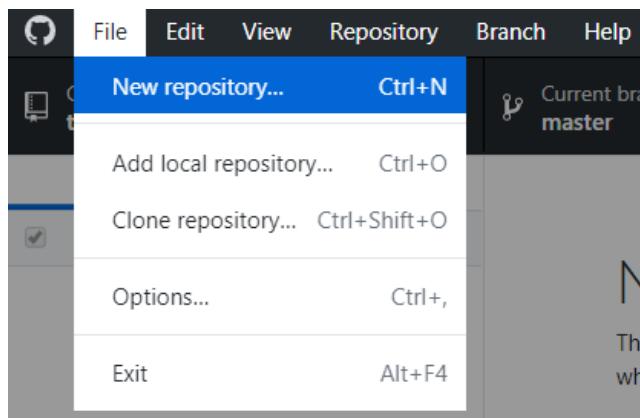
Module Objectives:

- 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



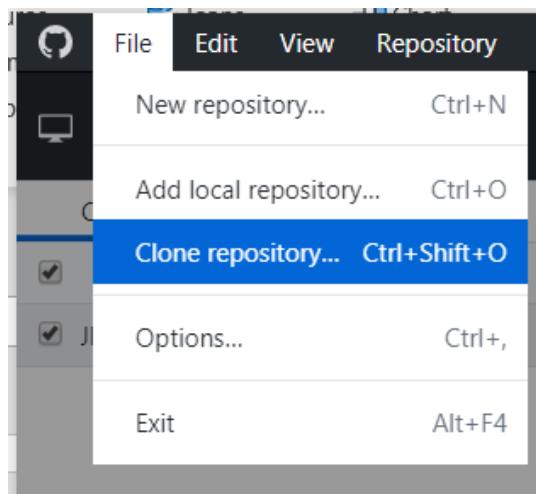
Lab Exercise: Copy the static webpage provided and setup a new project

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

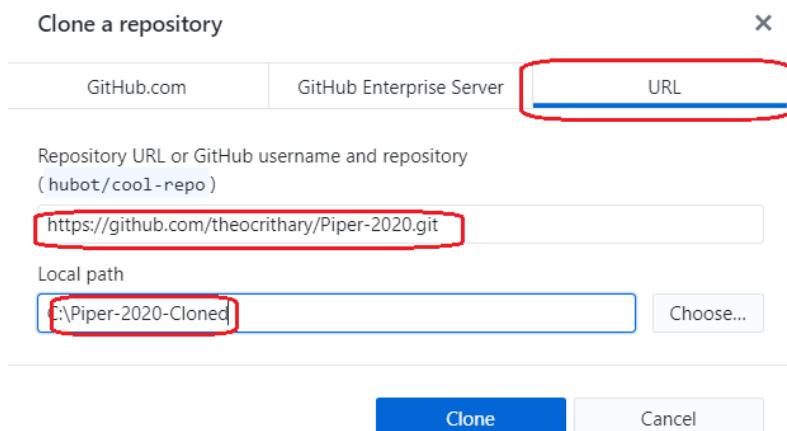


3. Clone the following Piper repository – “Go to File > Clone repository” to clone it from GitHub

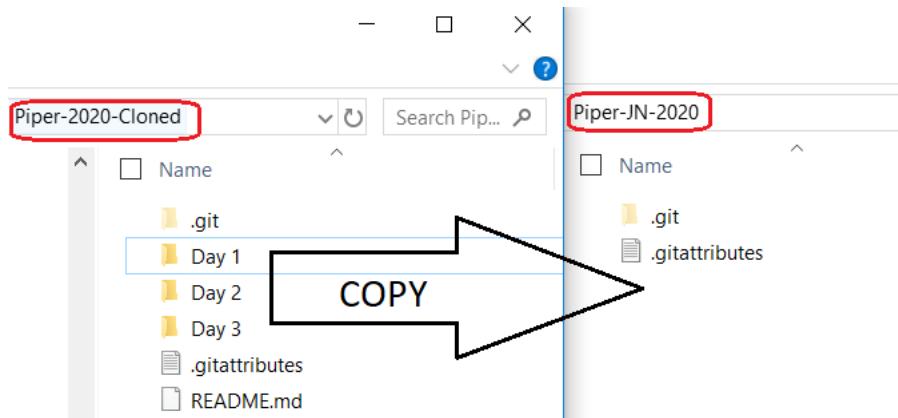
<https://github.com/theocrithary/Piper-2020.git>



- 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.

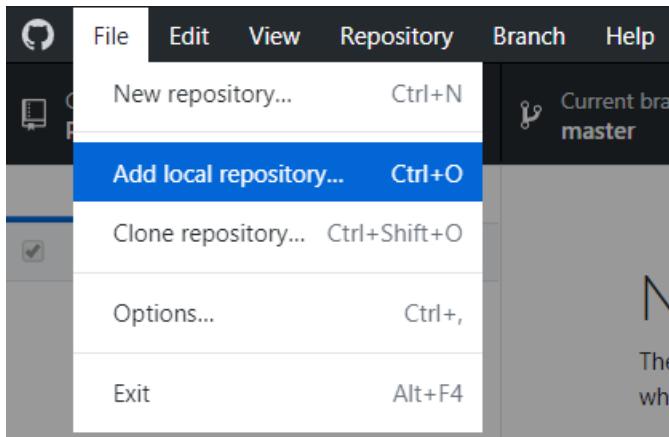


- Copy the Day 1, Day 2 and Day 3 folders from cloned directory (created above in Step 4) to your Project folder as created above in Step-2

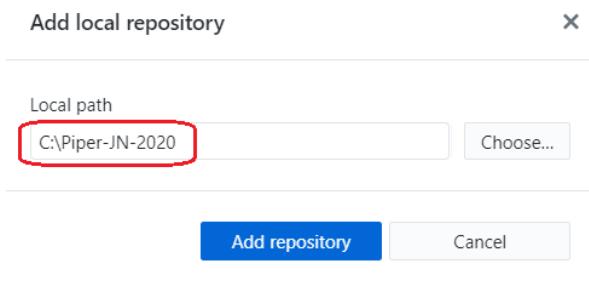




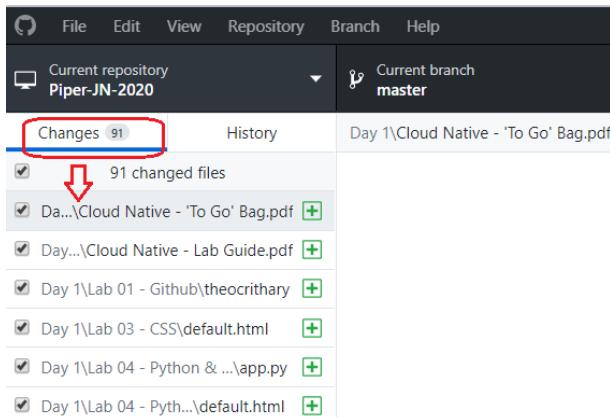
6. Go to your Project folder “Day 1 > Lab 03 – CSS”
7. Open the ‘default.html’ file in a browser to test its ‘look and feel’
8. Open the ‘default.html’ file in the Atom text editor provided and review the code
9. Go to the GitHub Desktop Client and Click on “File > Add local repository” (if not already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

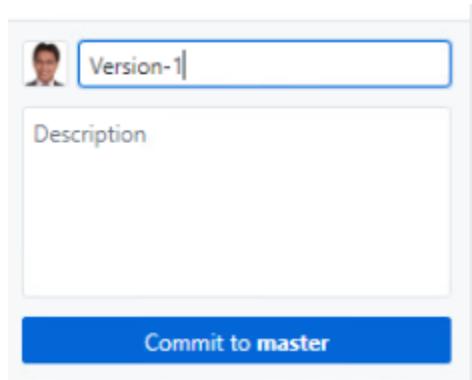


11. Notice all the changes on GitHub Desktop Client

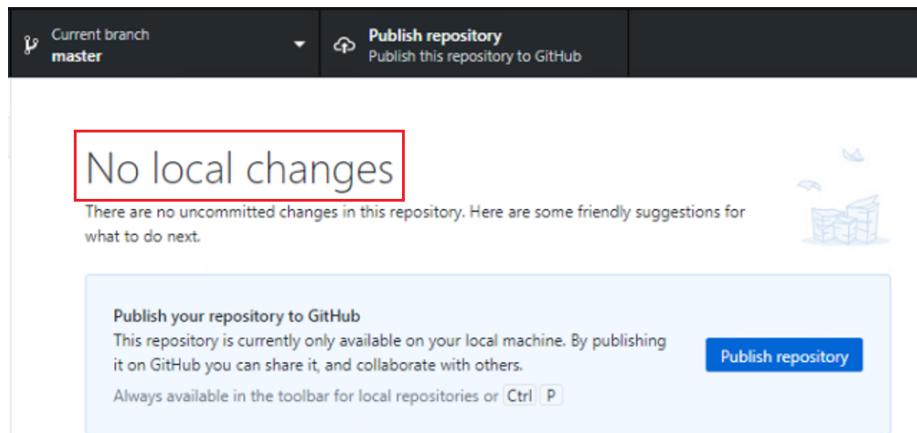




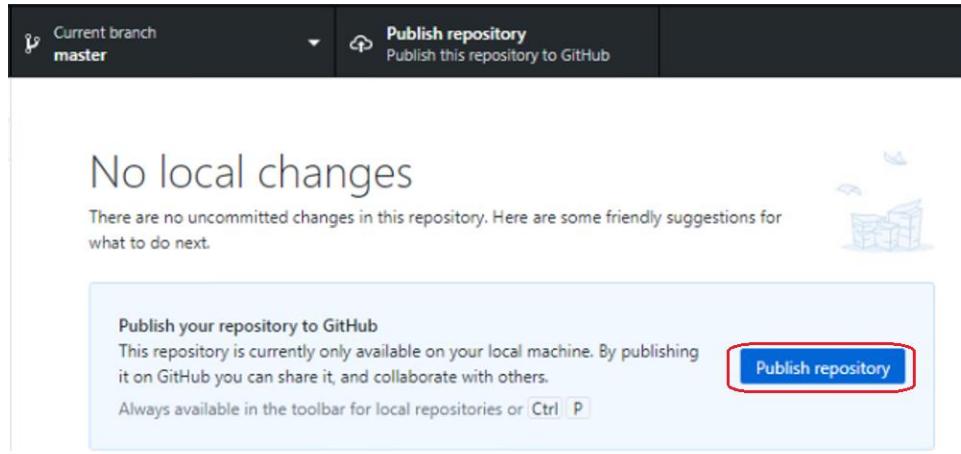
12. The current changes in directory structure is in working directory and tracked by GitHub client.
In order to commit the changes to local directory, type the comment in the summary field and click on commit to master



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

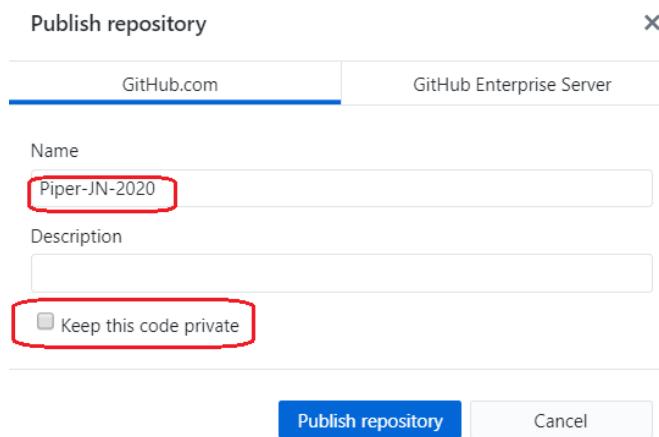


14. Publish the local repository to central repository by clicking on Publish repository

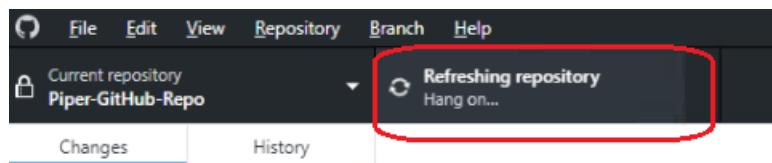
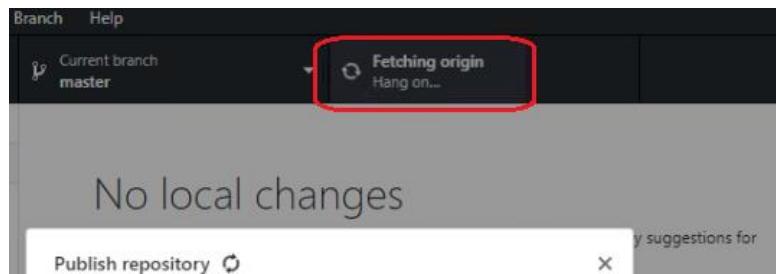




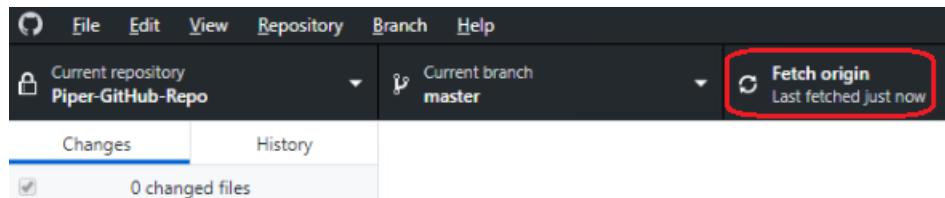
15. Upon which choose to keep it Public (visible to all on GitHub) and Click on Publish repository.



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



Open the repository in your external editor

Select your editor in Options

Repository menu or **Ctrl** **Shift** **A**

[Open in Atom](#)

View the files of your repository in Explorer

Repository menu or **Ctrl** **Shift** **F**

[Show in Explorer](#)

Open the repository page on GitHub in your browser

Repository menu or **Ctrl** **Shift** **G**

[View on GitHub](#)

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

jaikritnegi / **Piper-JN-2020**

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

No description, website, or topics provided. Edit

Manage topics

-o 2 commits 1 branch 0 packages 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

Commit	Author	Time
Day 1	jaikritnegi Version-1	9 minutes ago
Day 2	jaikritnegi Version-1	9 minutes ago
Day 3	jaikritnegi Version-1	9 minutes ago
.gitattributes	Initial commit	4 hours ago

Latest commit 515583e 9 minutes ago

Help people interested in this repository understand your project by adding a README. Add a README



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.

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

This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

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 View Edit 9 mins 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;

```
<!-- 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-ggOyR0iXcQFqyfzGVyDZSeWy2H4fFJ0/7+Q1lZ/t2uPeguKJqcyu5U2uCQVlU+tX" 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

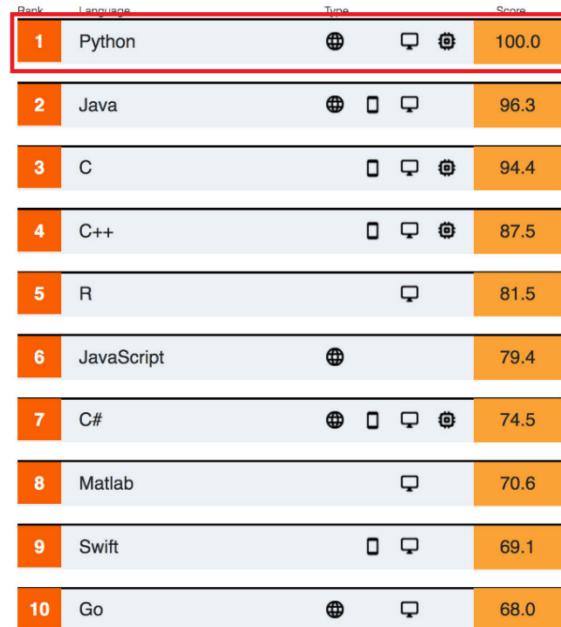


First a bit about Python & Flask

- 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/>



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

<u>Python 2</u>	<u>Python 3</u>
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. (If not already done in Lab 3), Download the files from [Lab 04 – Python & Flask](https://github.com/theocrithary/Piper-2020/tree/master/Day%201/Lab%2004%20-%20Python%20%26%20Flask) 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;

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

3. Open a command prompt and navigate to your working project folder
4. Run the following command

> python app.py

```
C:\Piper-JN-2020\Day 1\Lab 04 - Python & Flask>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]
```

5. Open a browser with the following URL
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)



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)
```

Remember controllers / routes in the MVC architecture? ←
And views / templates? ←

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; 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



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;

This PC > Data (D:) > Piper-GitHub-Repo > Day 1 > Lab 05 - MongoDB >

Name	Date modified	Type
__pycache__	6/17/2020 12:20 AM	File folder
templates	6/17/2020 12:20 AM	File folder
app.py	6/17/2020 12:18 AM	Python File
models.py	6/17/2020 12:18 AM	Python File

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

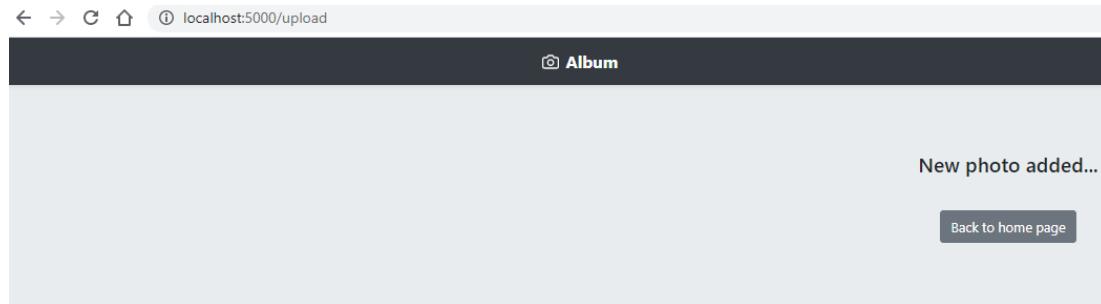
> python app.py

```
:\Piper-JN-2020\Day 1\Lab 05 - MongoDB>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)
```

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)



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

1  #!/usr/bin/env python3
2
3  ##### This is the database processing file. (aka. Models)
4  # It contains the DB connections, queries and processes.
5  # Uses principles of Models, Views, Controllers (MVC).
6  #####
7
8
9  # Import modules required for app
10 import os
11 from pymongo import MongoClient
12 from werkzeug.utils import secure_filename
13
14 ### Application configuration settings ###
15 # Set the database target to your Local MongoDB instance
16 client = MongoClient('127.0.0.1:27017')
17 DB_NAME = "mongodb" # This will be the name of your database
18 COL_NAME = "photos" # This will be the name of your collection
19
20
21 ##### Main body code #####
22
23 db = client[DB_NAME] # Create the database using the name provided and client connection to the MongoDB server
24 db_collection = db[COL_NAME] # Create the collection using the name provided and database connection
25
26 # Retrieve all photos records from database
27 def get_photos():
28     return db_collection.find({})
29
30 # Insert form fields into database
31 def insert_photo(request):
32     title = request.form['title']
33     comments = request.form['comments']
34     filename = secure_filename(request.files['photo'].filename)
35     thumbfile = filename.rsplit(".",1)[0] + "-thumb.jpg"
36
37     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
x

1 #!/usr/bin/env python3
2
3 ##### This is the main application file.
4 # It has been kept to a minimum using the design
5 # principles of Models, Views, Controllers (MVC).
6 #####
7
8
9 # Import modules required for app
10 import os
11 from flask import Flask, render_template, request
12 from models import get_photos, insert_photo
13
14 # Create a Flask instance
15 app = Flask(__name__)
16
17 ##### Define routes #####
18 @app.route('/')
19 def home():
20     album_photos = get_photos() # Call function in 'models.py' to retrieve all photo's from database
21     return render_template('default.html', album_photos=album_photos, url="home")
22
23 # This route accepts GET and POST calls
24 @app.route('/upload', methods=['POST'])
25 def upload():
26     insert_photo(request) # Call function in 'models.py' to process the database transaction
27     return render_template('submit-photo.html') # Return a page to inform the user of a successful upload
28
29 ##### Run the Flask instance, browse to http://<> Host IP or URL >>:5000 #####
30 if __name__ == "__main__":
31     app.run(debug=False, host='0.0.0.0', port=int(os.getenv('PORT', '5000')), threaded=True)

```

The screenshot shows a web application interface. On the left, there is a form titled "Photo upload" with instructions: "Use this form to upload your photo's to the album." It has fields for "Title" and "Comments", and a file upload section with a "Choose File" button and an "Upload" button. A preview area shows a thumbnail labeled "Test Photo". On the right, a message "New photo added..." is displayed above a "Back to home page" button.

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.



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



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>

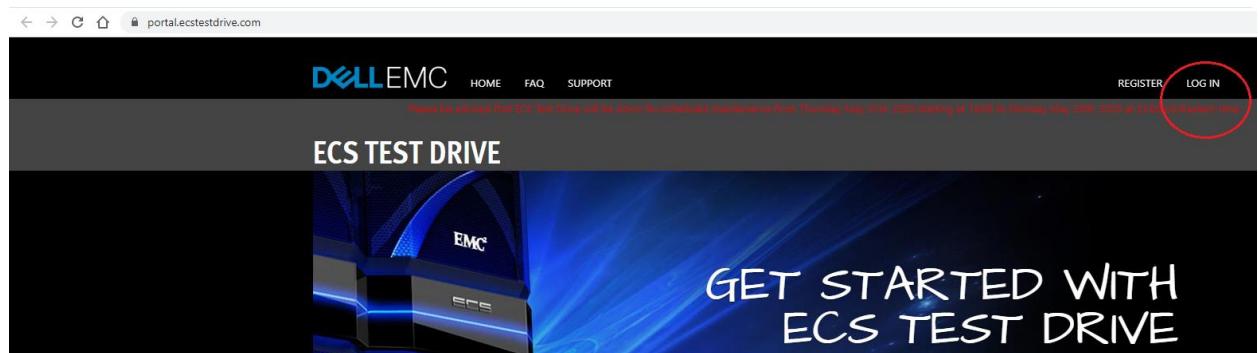
2. Your project directory should look like this;

Name	Date modified	Type
templates	6/16/2020 11:11 PM	File folder
.gitignore	6/16/2020 10:54 PM	Text Document
app.py	6/16/2020 10:54 PM	Python File
config-example.py	6/16/2020 10:54 PM	Python File
models.py	6/16/2020 10:54 PM	Python File

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



DELL EMC HOME FAQ SUPPORT CREDENTIALS LOG OFF

Please be advised that ECS Test Drive will be down for scheduled maintenance from Thursday May 21st, 2020 starting at 19:00 to Monday May 25th, 2020 at 10:00 UTC Eastern time.

ECS TEST DRIVE

- Take note of the Access key and secret key1 as following under the “AWS S3” section:

AWS S3

Endpoint: <https://object.ecstestdrive.com>

Public Endpoint: http://132276173696851222.public.ecstestdrive.com/{bucket_name}/{key_name}

Access Key: 132276173696851222@ecstestdrive.emc.com

Secret Key1: 0KbZxM5Wcj...otWlxJkMqahXbejMx1WBhp

Secret Key2:

- 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

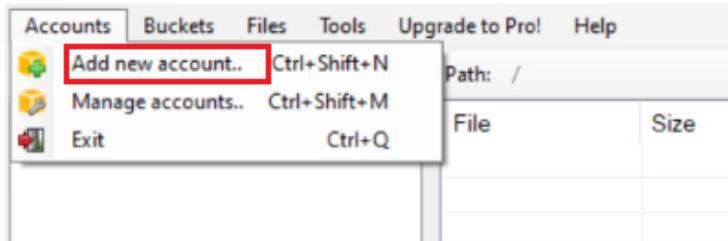
```

config.py

1 ecs_test_drive = {
2     'ecs_endpoint_url' : 'https://object.ecstestdrive.com',
3     'ecs_access_key_id' : '1234-your-unique-number-5678@ecstestdrive.emc.com',
4     'ecs_secret_key' : 'your-long-secret-key-from-ECS-testdrive-portal',
5     'ecs_bucket_name' : 'photo-album'
6 }
```

- 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”

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





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”

Add New Account

Add New Account

Enter new account details and click Add new account

online help

Account Name:
JaikritNegi

Assign any name to your account.

Account Type:
S3 Compatible Storage

Choose the storage you want to work with. Default is Amazon S3 Storage.

REST Endpoint:
object.ecstestdrive.com

Specify S3-compatible API endpoint. It can be found in storage documentation. Example: rest.server.com:8080

Signature Version:
Signature V2

Choose the supported signature version. Default value is Signature V2.

Access Key ID:
132276173696851222@ecstestdrive.emc.com

Required to sign the requests you send to Amazon S3, see more details at <https://s3browser.com/keys>

Secret Access Key:
[REDACTED]

Required to sign the requests you send to Amazon S3, see more details at <https://s3browser.com/keys>

Encrypt Access Keys with a password:
[REDACTED]

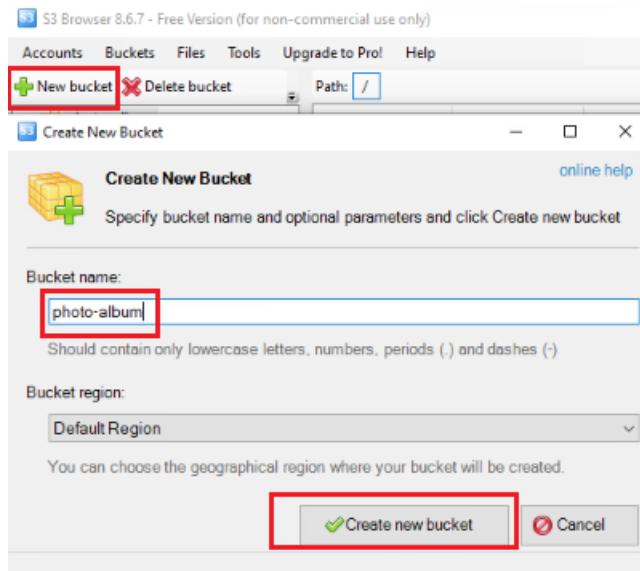
Turn this option on if you want to protect your Access Keys with a master password.

Use secure transfer (SSL/TLS)
If checked, all communications with the storage will go through encrypted SSL/TLS channel

Click here to sign up for Amazon S3..

Add new account Cancel

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



11. Pay close attention to bucket and object permissions tab

photo-album

- + New bucket - Delete bucket
- Path: /
- Tasks Permissions URL: https://photo-album.object.ecstestdrive.com/
- Properties Ctrl+P

Bucket Actions

- Create New Bucket... Ctrl+N
- Delete Bucket Del
- Refresh Buckets list F5
- Copy all files to...
- Download all files to...
- Edit Permissions (ACL) Ctrl+L
- Add External Bucket... Ctrl+E
- Mount as Windows Drive... Ctrl+M
- Bucket Sharing Wizard...
- Edit Bucket Policy... Ctrl+O
- Edit Logging Settings... Ctrl+G
- Edit Website Configuration... Ctrl+W
- Edit Versioning Settings... Ctrl+Shift+V
- Transfer Acceleration... Ctrl+Shift+T
- Cross-Region Replication... Ctrl+Shift+R
- Lifecycle Configuration... Ctrl+I
- Cost Allocation Tags... Ctrl+T
- CORS Configuration... Ctrl+R
- Generate Web URL(s)... Ctrl+U
- Change Storage Class to ...
- Server Side Encryption ...
- CloudFront ...

Tasks Permissions URL: https://photo-album.object.ecstestdrive.com/

User Name	Full Control	Read	Write	Read Permissions	Write Permissions
Owner (13227617369685...)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Any AWS Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
All Users	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>



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 (from boto3) (0.9.5)
Requirement already satisfied: docutils<0.16,>=0.10 in c:\users\demouser\appdata\local\programs\python\python37\lib\site-packages (from boto3) (0.15.2)
Requirement already satisfied: urllib3<1.26,>=1.20; python_version != "3.4" in c:\users\demouser\appdata\local\programs\python\python37\lib\site-packages (from 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-packages (from boto3) (2.8.1)
Requirement already satisfied: six<1.5 in c:\users\demouser\appdata\local\programs\python\python37\lib\site-packages (from python-dateutil<3.0.0,>=2.1->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 pip' 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 pip' command.
```

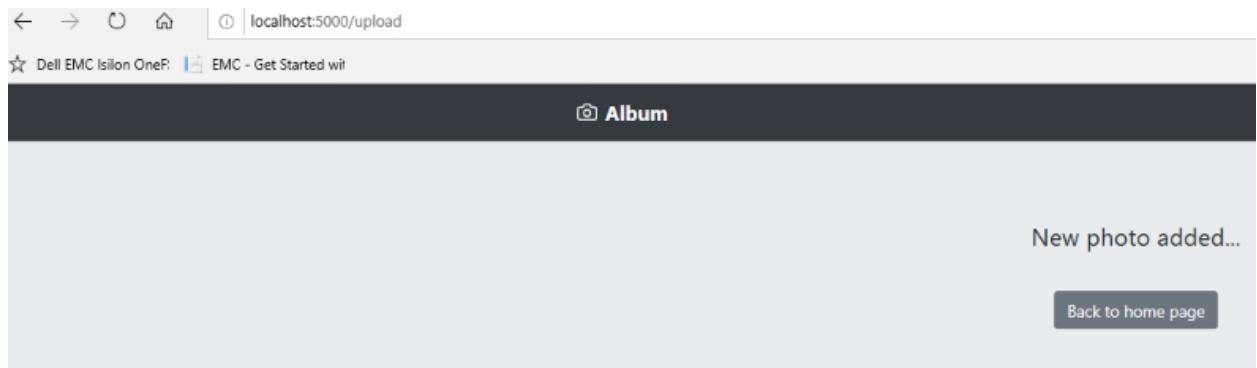
13. Run your app

> 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)



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
1  #!/usr/bin/env python3
2
3  #####
4  # This is the main application file.
5  # It has been kept to a minimum using the design
6  # principles of Models, Views, Controllers (MVC).
7  #####
8
9  # Import modules required for app
10 import os
11 from flask import Flask, render_template, request
12 from models import get_photos, insert_photo, upload_photo
13
14 # Create a Flask instance
15 app = Flask(__name__)
16
17 ##### Define routes #####
18 @app.route('/')
19 def home():
20     album_photos = get_photos()                      # Call function in 'models.py' to retrieve
21     return render_template('default.html',album_photos=album_photos,url="home")
22
23 # This route accepts GET and POST calls
24 @app.route('/upload', methods=['POST'])
25 def upload():
26     insert_photo(request)                          # Call function in 'models.py' to process
27     upload_photo(request.files['photo'])           # Call function in 'models.py' to upload
28     return render_template('submit-photo.html')    # Return a page to inform the user of
29
30 @app.route('/photo<path:photo>')
31 def photo(photo):
32     return render_template('photo.html',photo=photo)
33
34 ##### Run the Flask instance, browse to http://<< Host IP or URL >>:5000 #####
35 if __name__ == "__main__":
36     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.



```

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=ecs_secret_key)

    # 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 accidentally to your Github repository.

<pre> config-example.py x 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' } </pre>	<pre> .gitignore 1 config.py 2 __pycache__ </pre>
--	---

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

No file chosen

Test Photo

Test Photo



New Photo



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

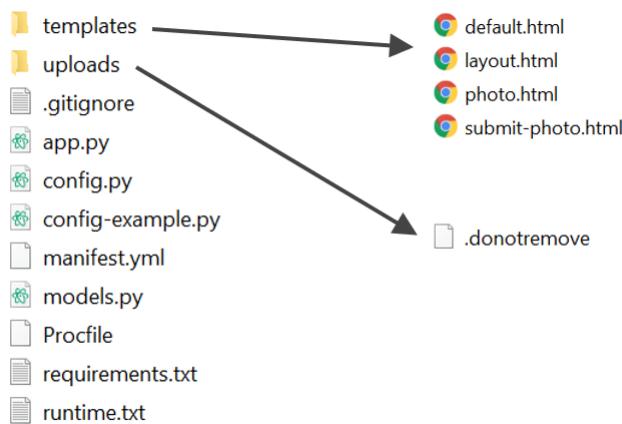


Lab Exercise (Part A): Create the pre-requisite platform files and upload the app to Hosted TAS (PWS)

1. (If not already done in Lab 3), Download the following files from [Lab 07 - PWS & mLab](https://github.com/theocrithary/Piper-2020/tree/master/Day%201/Lab%2007%20-%20PWS%20%26%20mLab) 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 -t photo-album --recent' for more information
FAILED
```



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>

VMware Tanzu Why Tanzu Products Services Customers Resources Contact Us Support Sign In Q

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 ... [Read more](#)

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

Student Lab Guide



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.

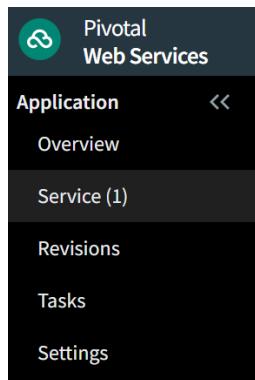
For our application, we will deploy a MongoDB service provided by mLab as a hosted multi-tenanted database instance.

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



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

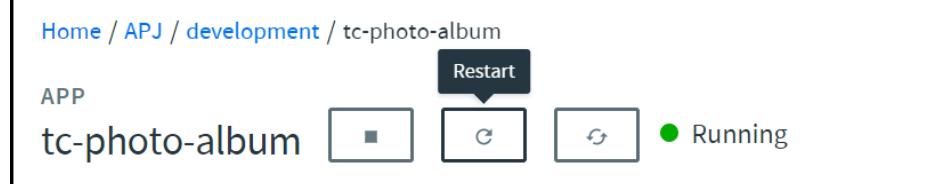


8. Click on the 'NEW SERVICE' button and create an mLab instance to bind to your app

<https://console.run.pivotal.io/>

The screenshot shows the PWS console interface. At the top, it displays the application 'tc-photo-album' with a status of 'Crashed'. Below this, the 'Services' tab is selected. A modal window is open for adding a new service. In the 'Create a new service' section, 'mLab' is selected from the marketplace. The 'Instance Name' field contains 'mongodb'. A red annotation with the text 'Must be unique' is placed above this field. The 'Bind to App (Optional)' dropdown is set to 'tc-photo-album'. The 'CREATE' button is visible at the bottom right of the modal.

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

A screenshot of a web-based interface for managing routes. At the top, there are three tabs: "Overview", "Service (1)", and "Route (1)", with "Route (1)" being the active tab. Below the tabs, the word "Routes" is displayed. A single route entry is shown in a table:

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

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)



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.

<pre>Lab 07 - PWS & mLab > __pycache__ > templates > uploads .gitignore app.py config-example.py config.py manifest.yml models.py Procfile requirements.txt runtime.txt</pre>	<pre>manifest.yml 1 --- 2 applications: 3 - name: photo-album 4 memory: 64M 5 instances: 1 6 random-route: true</pre>	<pre>Procfile 1 web: python3 app.py</pre>	<pre>requirements.txt 1 pip==20.0.2 2 Flask==1.1.1 3 boto3==1.11.12 4 Pillow==7.0.0 5 pymongo==3.10.1 6 Werkzeug==0.16.1</pre>
		<pre>runtime.txt 1 python-3.8.1</pre>	