



Part 1 - Cloud Native Technical Workshop

Lab Manual

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.

February 2021

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
0.3	July 2020	Lab release 2.0
0.4	July 2020	Removed requirement for S3 browser application
1.0	Feb 2021	Refresh of all content for 2021 courseware

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.

© 2021 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



Contents

Pre-requisites	5
Accounts:.....	5
Lab 1: Github	6
Module Objectives:.....	6
What is Git?	7
How to use .gitignore to avoid committing sensitive data	8
More Tutorials (if required)	9
Lab Exercise: Create a new repo and publish a file with your name	10
Retrospective: What was achieved	19
Lab 2: Deploy an App to Tanzu Application Service.....	20
Module Objectives:.....	20
Lab Exercise: Clone an example app and deploy to TAS.....	21
Retrospective: Reflect on what just happened.....	25
Lab 3: Clone the GitHub Repository for all Piper Labs.....	26
Module Objectives:.....	26
Lab Exercise: Clone the source repository	27
Retrospective: A read-only master source repository	29
Lab 4: Create a Static HTML Website	30
Module Objectives:.....	30
Lab Exercise: Copy the static webpage provided and setup a new project.....	31
Retrospective: A basis for an MVP	36
Lab 5: Python and Flask	37
Module Objectives:.....	37
Lab Exercise: Create a basic Python app to run the Flask web server	38
Retrospective: Applying concepts of Model, Views, Controllers	39
Lab 6: MongoDB	40
Module Objectives:.....	40
Lab Exercise: Copy the new code files and overwrite existing application files	41
Retrospective: Review the implementation of 'models' into our application	43
Lab 7: ECS	45
Module Objectives:.....	45
Lab Exercise: Use Boto3 Python module to upload images to ECS	46



Retrospective: Review code and note implementation of Boto3 & Pillow.....	50
Lab 8: TAS & MongoDB.....	53
Module Objectives:.....	53
Lab Exercise (Part A): Create the pre-requisite platform files and upload the app to Hosted TAS ..	54
Lab Exercise (Part B): Add the MongoDB database connection environment variables.....	56
Retrospective: Review code and take note of the 'DB_URI' section in models.py	62



Pre-requisites

Accounts:

Please ensure you have created the following accounts before commencing these labs.

- **GitHub** https://github.com/join?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home
- **ECS Test Drive** <https://portal.ecstestdrive.com/Account/Register>

Note: ECS Test Drive accounts typically require **24-48 business hours** to be approved and should use your corporate email (e.g. @dell.com). The approval email may also be blocked by spam filters and may need to be released from quarantine. If you are a Dell employee, search for "Email Digest:" in your inbox and release the email from ECS Test Drive.

These accounts are required to complete parts of the lab and you will not be able to proceed without these account credentials.



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
```



More Tutorials (if required)

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

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

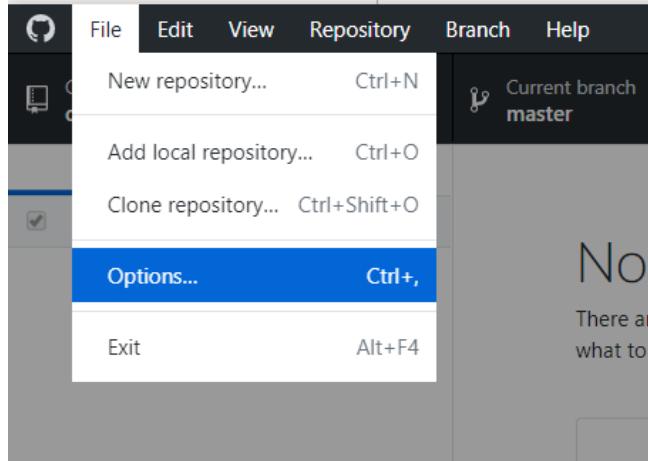
- If you do not know how to use Github Desktop GUI client and want to learn more, please complete the following tutorial

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

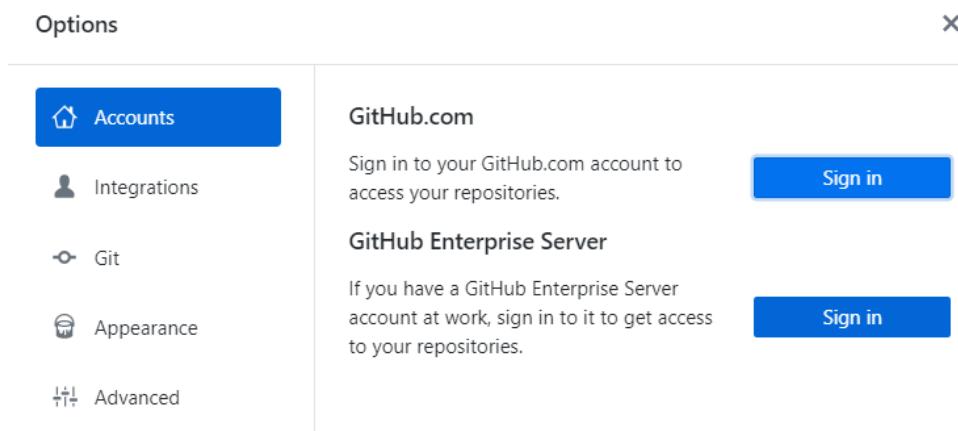


Lab Exercise: Create a new repo and publish a file with your name

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

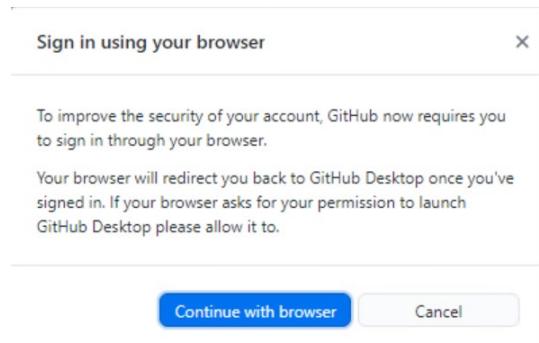


3. Click on “Accounts > Sign in” in order to Login to GitHub.com



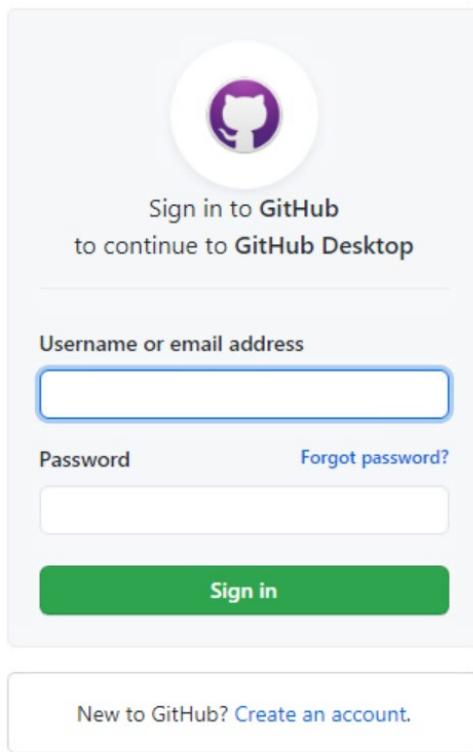


4. Click the button to ‘continue with browser’

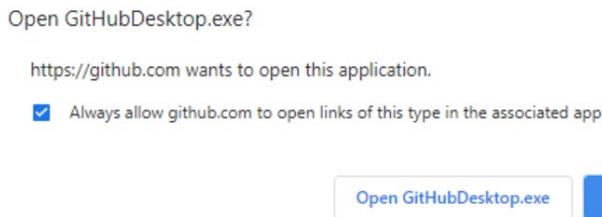


5. A browser window should open and prompt you to sign in to GitHub.

Sign in using your personal account with the email ID and password you created in the pre-requisites.

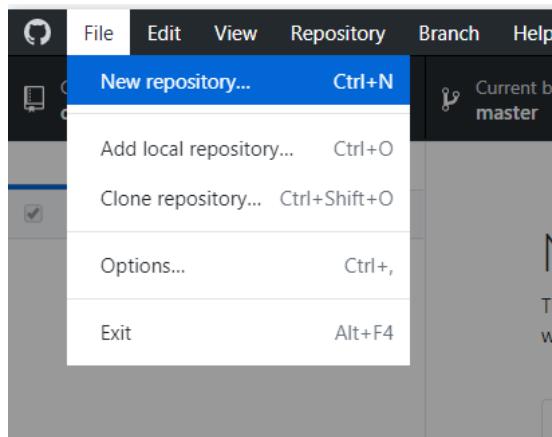


6. You may be prompted to ‘Open GitHubDesktop.exe’, as shown below. Click the open button.





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



8. Type in a name for your personal project (e.g. MyProject) and change the path to your preferred path (i.e. D:\) before clicking on create repository (leave the remaining options as default)

Create a new repository ×

Name

Description

Local path
 Choose...

Initialize this repository with a README

Git ignore

License

Create repository Cancel



9. After the repository has been created, click on ‘Publish repository’

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.



Publish your repository to GitHub

This repository is currently only available on your local machine. By publishing it on GitHub you can share it, and collaborate with others.

Publish repository

Always available in the toolbar for local repositories or **Ctrl P**

10. NOTE- If you receive below error code 128, follow the steps below;

This is a one time configuration of the GitHub Desktop application and will not need to be performed again for any other labs.

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

Open the Windows command prompt and type in below 2 commands (example in below screenshot) by:

```
> git config --global user.email "you@example.com"  
(replacing last field in the command with your GitHub account email address)
```

```
> git config --global user.name "Your Name"  
(replacing last field in the command with your username)
```

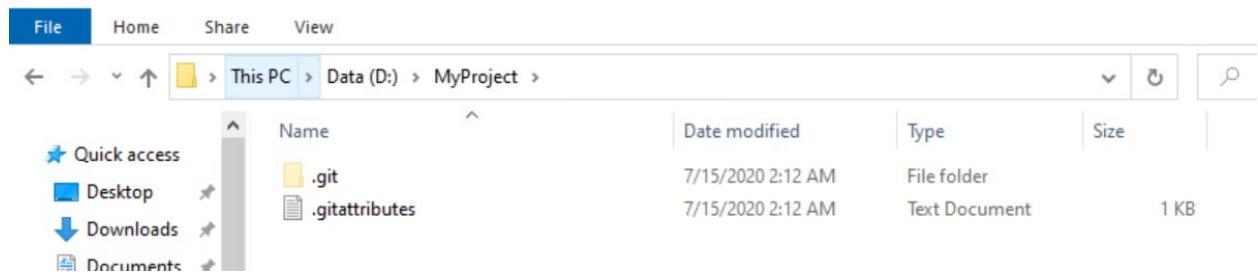


```
| Command Prompt
| Microsoft Windows [Version 10.0.18363.1316]
| (c) 2019 Microsoft Corporation. All rights reserved.

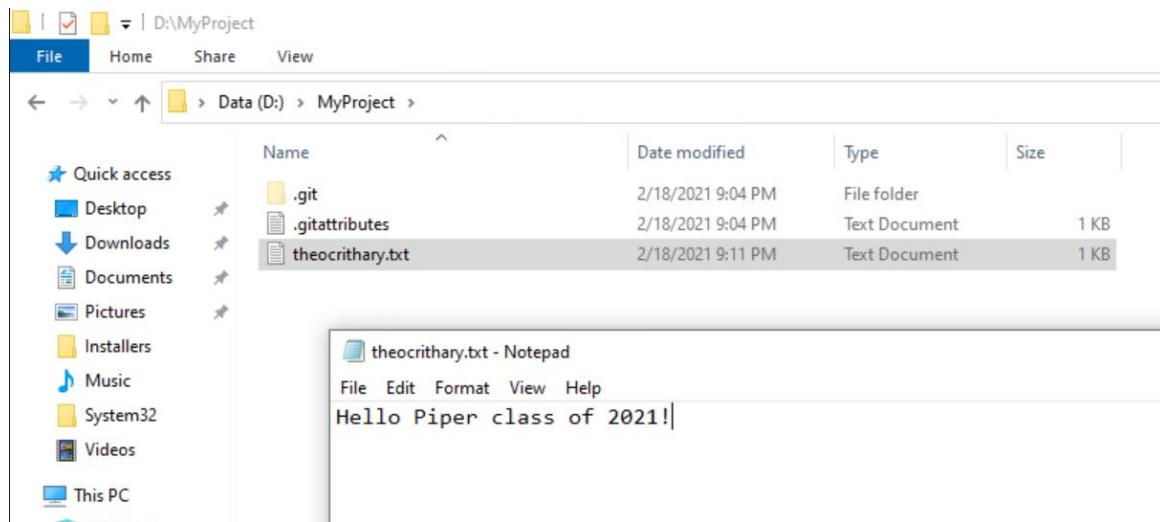
C:\Users\demouser>git config --global user.email theo@apacit.com.au
C:\Users\demouser>git config --global user.name theocrithary
```



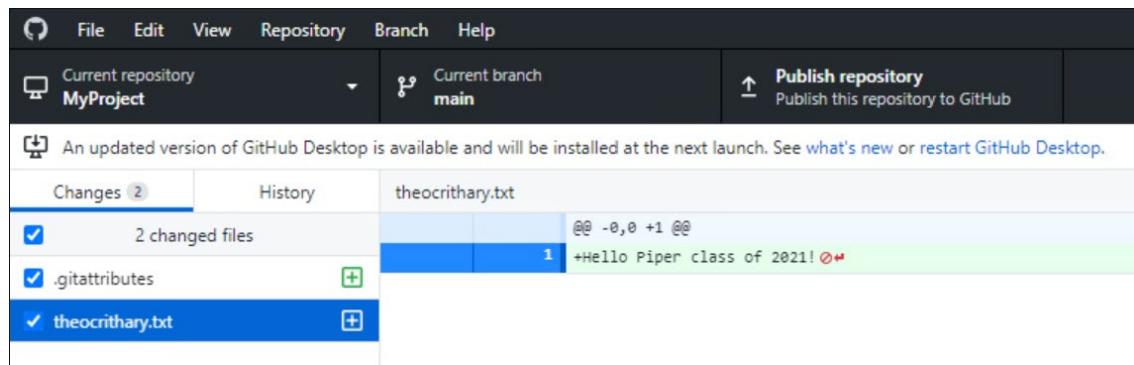
11. Open Windows Explorer and find the folder in the path created by GitHub Desktop Client with the folder name chosen as repository name (e.g. MyProject) with below structure



12. 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 2021!” and save the file

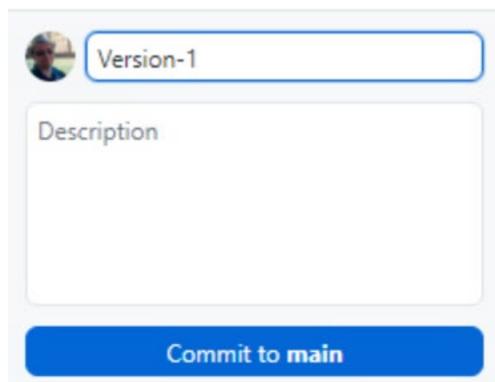


13. Go to already opened instance of GitHub Desktop Client and notice the newly created file and content and shows the changed files.

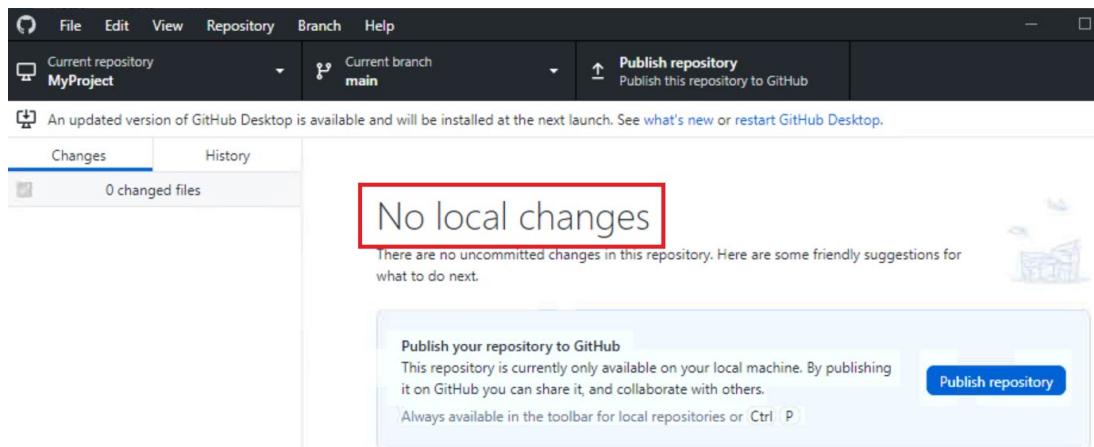




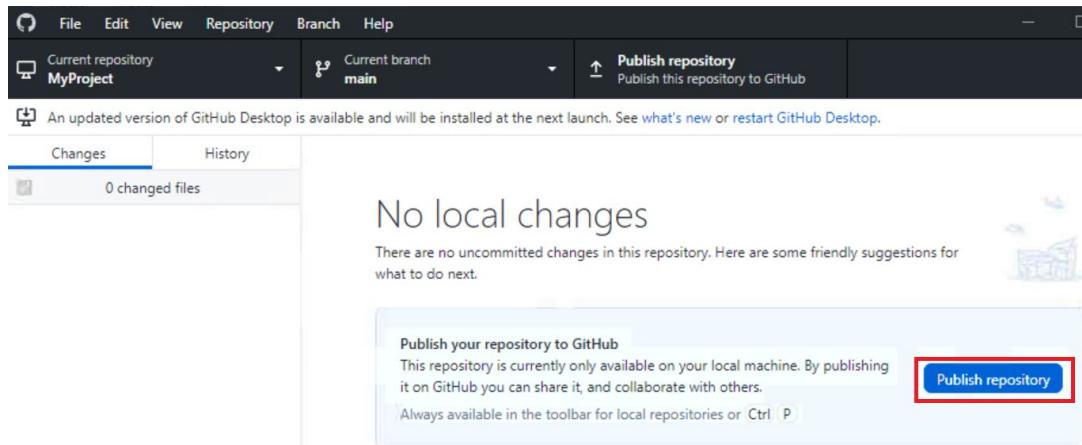
14. The current changes are only local and tracked by the git local database at this point. In order to commit the changes to the local git repository, type a comment in the summary field on the bottom left of the GitHub Desktop client and click on “Commit to main”



15. Above action should show below screen confirming commit with display as “No local changes”

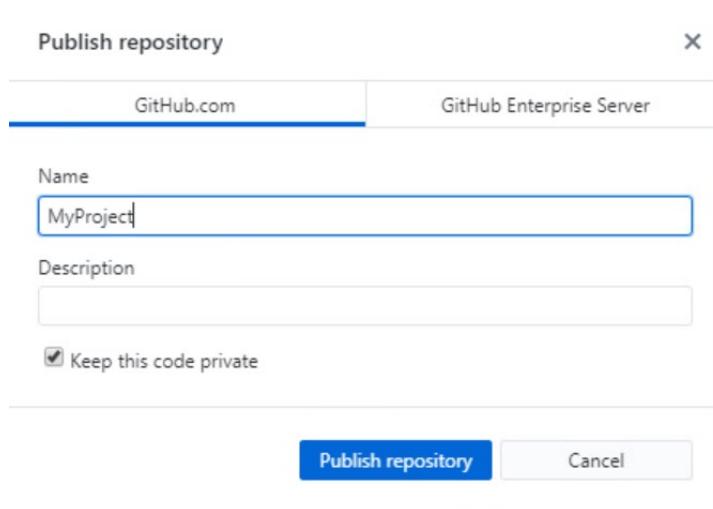


16. Publish the local repository to GitHub.com repository by clicking on “Publish repository”

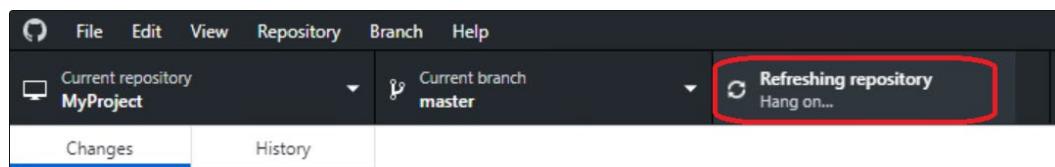
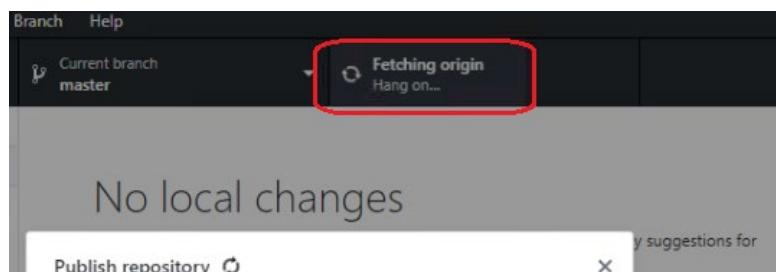




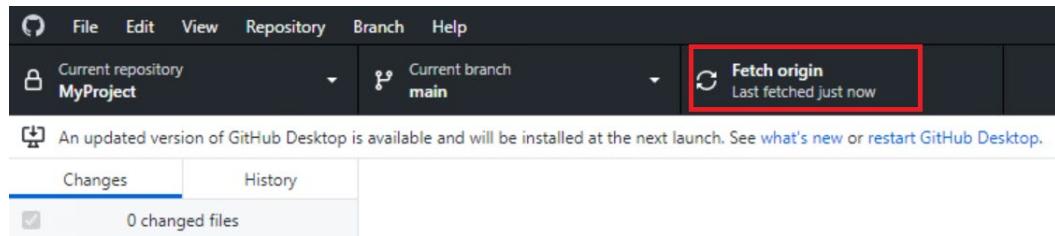
17. Upon which choose to keep it Private (visible to self only) or Public (visible to all on GitHub) and Click on Publish repository.



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



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





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

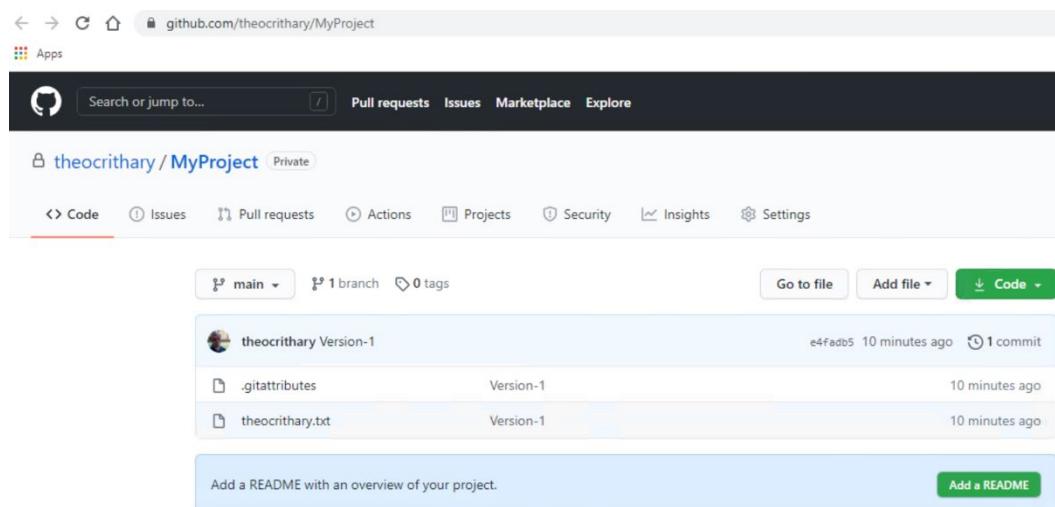
Open the repository in your external editor
Select your editor in [Options](#)

Repository menu or [Ctrl](#) [Shift](#) [A](#) [Open in Atom](#)

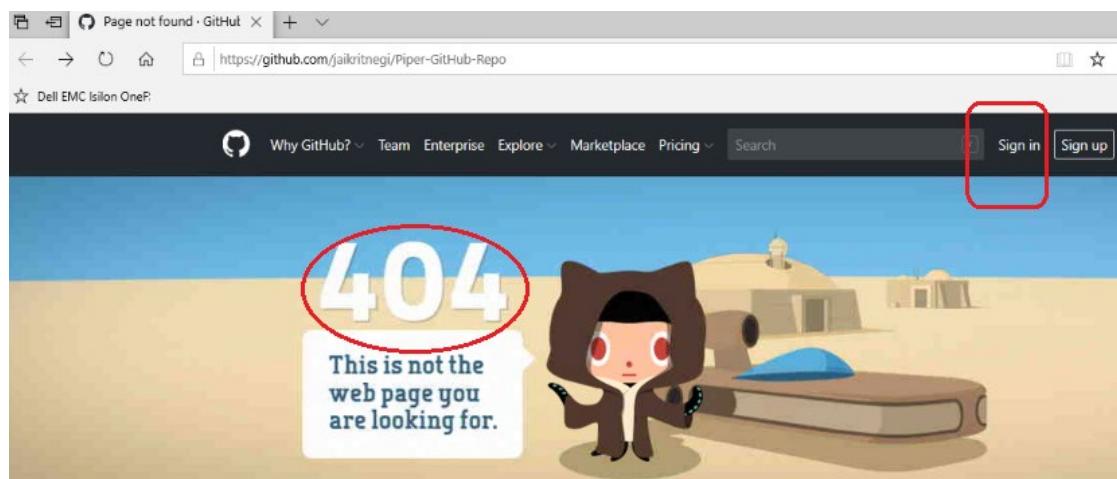
View the files of your repository in Explorer
Repository menu or [Ctrl](#) [Shift](#) [F](#) [Show in Explorer](#)

Open the repository page on GitHub in your browser
Repository menu or [Ctrl](#) [Shift](#) [G](#) [View on GitHub](#)

21. You should be directed to a browser and automatically logged in to GitHub as you have already signed in via the browser earlier.



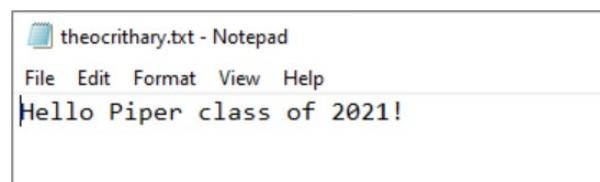
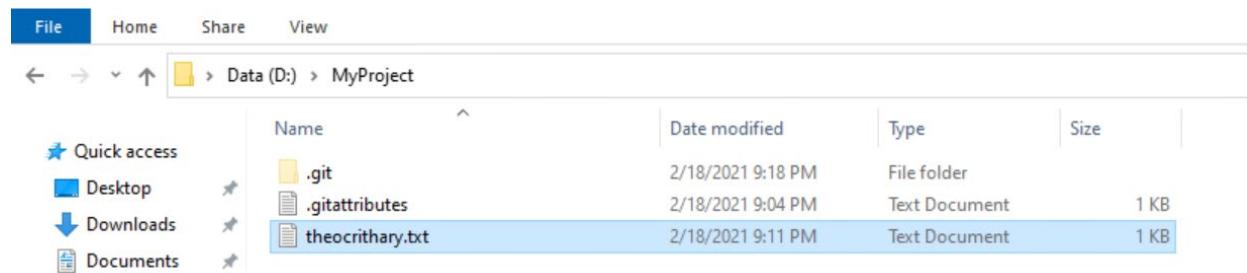
Should you encounter error code 404 on the web browser as below, click on 'sign in' to login with your GitHub credentials to view the published repository.





Retrospective: What was achieved

- Learnt about git, git tools, Github and Github Desktop UI
- Setup a new personal repo for our project
- Confirmed git tools are working and we have the skills to commit code
- Confirmed files uploaded to Github



theocritrary / MyProject (Private)

Code Issues Pull requests Actions Projects Security Insights Settings

main MyProject / theocritrary.txt

theocritrary Version-1

1 contributor

1 lines (1 sloc) | 26 Bytes

```
Hello Piper class of 2021!
```



Lab 2: Deploy an App to Tanzu Application Service

Module Objectives:

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

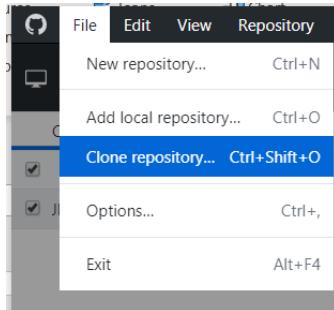


Lab Exercise: Clone an example app and deploy to TAS

1. Open the GitHub Desktop Client (from the shortcut on the desktop)

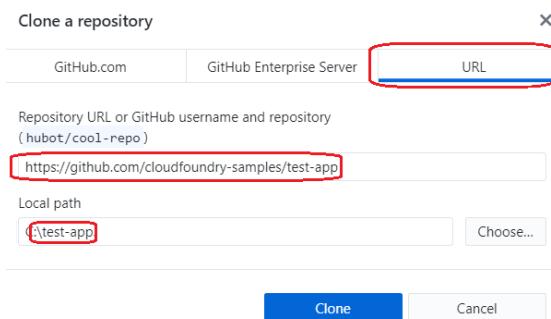


2. Clone a repository – “Go to File > Clone repository” to clone it from GitHub

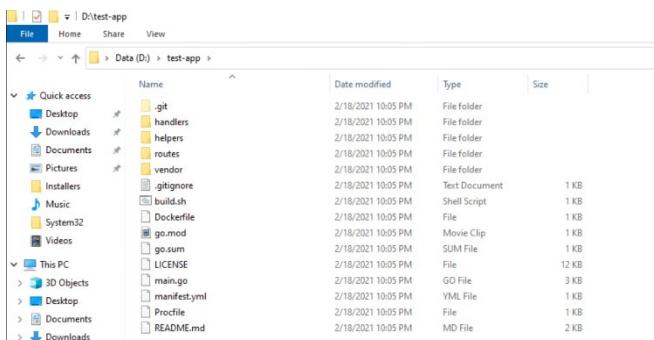


3. Click on the URL tab and copy the below GitHub URL and paste to the Repository URL field and select the local path on D:\ drive.

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



4. When it completes the clone process, you should have a new directory in D:\ with the ‘test-app’ files



5. Open the ‘manifest.yml’ file with any text editor (e.g. Atom or Notepad)



Add your initials or something unique to the end of the application name to avoid conflicts with other students.

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

Save and close the file.

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

> d:

> cd test-app

7. Login to TAS with the following account credentials

> cf login -a <https://api.sys.tan-zu.com> --skip-ssl-validation

```
D:\test-app>cf login -a https://api.sys.tan-zu.com --skip-ssl-validation
API endpoint: https://api.sys.tan-zu.com

Email: student

Password:
Authenticating...
OK

Targeted org piper

Targeted space development

API endpoint: https://api.sys.tan-zu.com (API version: 3.85.0)
User: student
Org: piper
Space: development
```

8. Push the app to TAS and observe the output

> cf push



```
D:\test-app>cf push
Pushing from manifest to org piper / space development as student...
Using manifest file D:\test-app\manifest.yml
Getting app info...
Creating app with these attributes...
+ name:      test-app-tc
  path:      D:\test-app
+ instances:  1
+ memory:    256M
  routes:
+   test-app-tc-anxious-crocodile-pm.apps.tan-zu.com
```

```
Creating app test-app-tc...
Mapping routes...
Comparing local files to remote cache...
Packaging files to upload...
Uploading files...
37.28 Kib / 37.28 Kib [=====] 100.00% 1s
Waiting for API to complete processing files...

Staging app and tracing logs...
  Downloading binary_buildpack...
  Downloading ruby_buildpack...
  Downloading staticfile_buildpack...
  Downloading nginx_buildpack...
  Downloaded staticfile_buildpack
  Downloaded nodejs_buildpack...
  Downloaded nginx_buildpack
  Downloading go_buildpack...
  Downloading r_buildpack...
  Downloaded ruby_buildpack
  Downloading python_buildpack...
  Downloaded binary_buildpack
  Downloading php_buildpack...
  Downloaded nodejs_buildpack
  Downloading dotnet_core_buildpack...
  Downloaded go_buildpack
  Downloading java_buildpack_offline...
  Downloaded r_buildpack
  Downloaded dotnet_core_buildpack
  Downloaded php_buildpack
  Downloaded python_buildpack
  Downloaded java_buildpack_offline
Cell fc54c244-5559-4ad0-bdc6-640588cd62f creating container for instance 40aeef8a1-2075-427d-8562-276cf83debe0
Cell fc54c244-5559-4ad0-bdc6-640588cd62f successfully created container for instance 40aeef8a1-2075-427d-8562-276cf83debe0
Downloading app package...
Downloaded app package (37.3K)
----> Go Buildpack version 1.9.23
  **WARNING** [DEPRECATION WARNING]:
  **WARNING** Please use AppDynamics extension buildpack for Golang Application instrumentation
  **WARNING** for more details: https://docs.pivotal.io/partners/appdynamics/multibuildpack.html
----> Installing godep 0.8
  Copy [/tmp/buildpacks/e7519bcc6c1b20873544186748154f7//dependencies/52a892f00e80ca4fdfc27d9828c7aba1/godep-v80-linux-x64-cflinuxfs3-b60ac947.tgz]
----> Installing glide 0.13.3
  Copy [/tmp/buildpacks/e7519bcc6c1b20873544186748154f7//dependencies/f5e4affa54f8cf8e22cf524de5165a6e/glide-v0.13.3-linux-x64-cflinuxfs3-ef07acb5.tgz]
----> Installing dep 0.5.4
  Copy [/tmp/buildpacks/e7519bcc6c1b20873544186748154f7//dependencies/f1900fc2de60a12ea6743ecf05e14d2/dep-v0.5.4-linux-x64-cflinuxfs3-79b3ab9e.tgz]
----> Installing go 1.15.5
  Copy [/tmp/buildpacks/e7519bcc6c1b20873544186748154f7//dependencies/0de0eebec31838aa4d1621b581cc3c31af/go_1.15.5_linux_x64_cflinuxfs3_fd04494f.tgz]
  **WARNING** Installing package '.' (default)
----> Running: go install -tags cloudfoundry -buildmode pie .
Exit status 0
Uploading droplet, build artifacts cache...
Uploading droplet...
Uploading build artifacts cache...
Uploading build artifacts cache (9.3M)
```

- When the upload completes and the application has been deployed, you should see the below state 'running'.

```
Waiting for app to start...

name:      test-app-tc
requested state:  started
routes:    test-app-tc-anxious-crocodile-pm.apps.tan-zu.com
last uploaded: Thu 18 Feb 22:59:19 GMT 2021
stack:     cflinuxfs3
buildpacks: go

type:      web
instances: 1/1
memory usage: 256M
start command: test-app
#0  state  since          cpu    memory   disk    details
#0  running  2021-02-18T22:59:28Z  0.0%  0 of 256M  0 of 1G
```



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

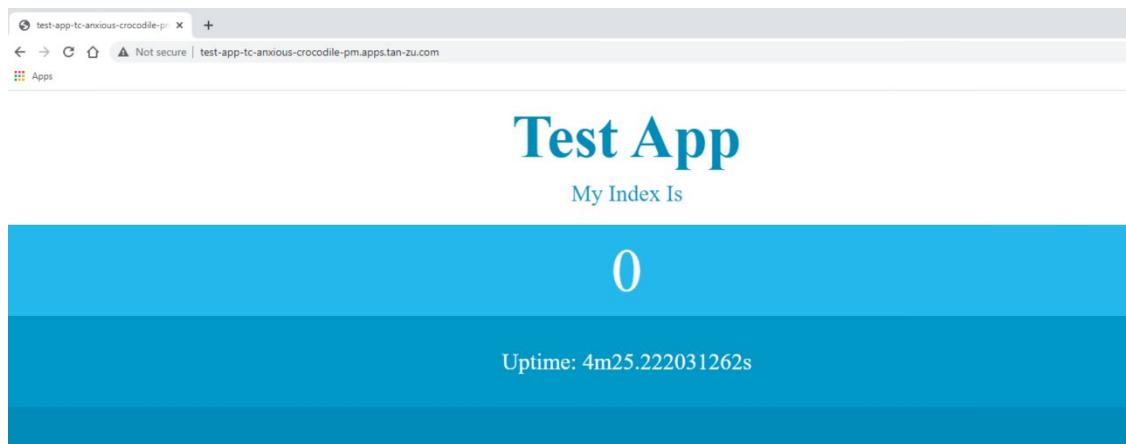
```
Waiting for app to start...

name:          test-app-tc
requested state: started
routes:        test-app-tc-anxious-crocodile-pm.apps.tan-zu.com
last uploaded: Thu 18 Feb 22:59:19 GMT 2021
stack:         cflinuxfs3
buildpacks:    go

type:          web
instances:     1/1
memory usage:  256M
start command: test-app
      state   since           cpu   memory   disk   details
#0  running  2021-02-18T22:59:28Z  0.0%  0 of 256M  0 of 1G
```

e.g. <https://test-app-tc-anxious-crocodile-pm.apps.tan-zu.com> (NOTE- your URL will be different)

11. Confirm your app is working as available from a public facing website by opening in a browser



12. Clean up and delete your app by running the following in the command prompt

> cf delete test-app-xx -r

Note: use your unique application name as specified in the manifest file and don't forget the '-r' option at the end to remove the route for your application

Type 'yes' when prompted to confirm deletion.



Retrospective: 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.

```
D:\test-app>cf push
Pushing from manifest to org piper / space development as student...
Using manifest file D:\test-app\manifest.yml
Getting app info...
Creating app with these attributes...
+ name: test-app-tc
  path: D:\test-app
+ instances: 1
+ memory: 256M
  routes:
+ test-app-tc-fearless-cat-as.apps.tan-zu.com

Creating app test-app-tc...
Mapping routes...
Comparing local files to remote cache...
Packaging files to upload...
Uploading files...
 37.28 KiB / 37.28 KiB [=====] 100.00% 1s

Waiting for API to complete processing files...

Staging app and tracing logs...

Waiting for app to start...

name: test-app-tc
requested state: started
routes: test-app-tc-fearless-cat-as.apps.tan-zu.com
last uploaded: Mon 01 Mar 22:51:59 GMT 2021
stack: cflinuxfs3
buildpacks: go

type: web
instances: 1/1
memory usage: 256M
start command: test-app
  state since          cpu    memory      disk   details
#0  running 2021-03-01T22:51:44Z  0.2%  14.6M of 256M  9.8M of 1G

D:\test-app>cf delete test-app-tc -r

Really delete the app test-app-tc?> yes
Deleting app test-app-tc in org piper / space development as student...
OK

D:\test-app>
```

Note: You did not allocate any IP addresses, build a VM / container, deploy an OS, install middleware / libraries, create a load balancer, assign a DNS entry, open a firewall rule or create any IP routing or VLANs to get this application running in a public facing website. The platform did all of that for you with one single command “**cf push**”!



Lab 3: Clone the GitHub Repository for all Piper Labs

Module Objectives:

- Clone the master repository for all labs provided in the Piper workshop
- Explore the folder structure
- Acknowledge the Piper-TH-Workshop folder as a read only copy of the source
- Establish a habit of copying files from the source to your personal project folder

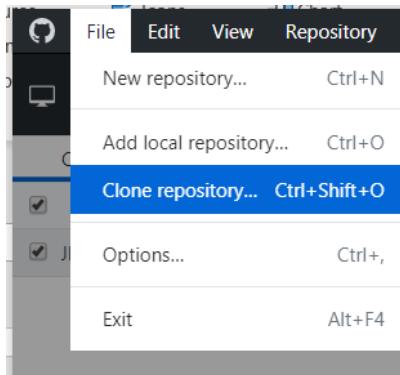


Lab Exercise: Clone the source repository

1. Open the GitHub Desktop Client (from the shortcut on the desktop)

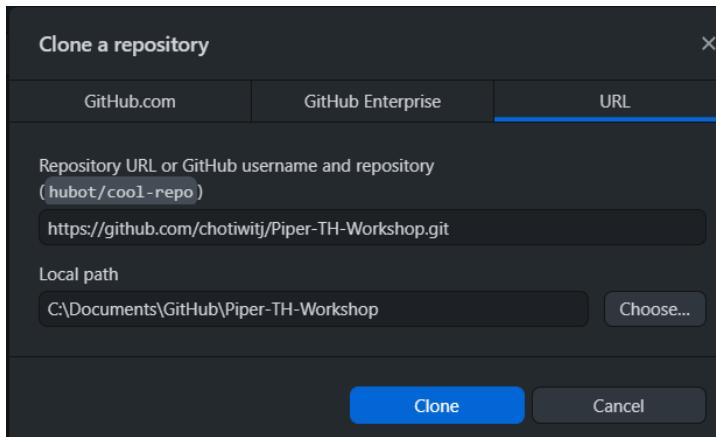


2. Clone a repository – “Go to File > Clone repository” to clone it from GitHub

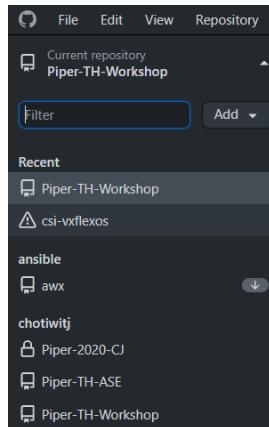


3. Click on the URL tab and copy the below GitHub URL and paste to the Repository URL field and select the local path on D:\ drive.

<https://github.com/chotiwitj/Piper-TH-Workshop.git>



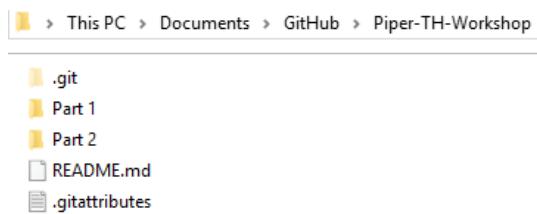
4. Make sure the correct repository is set as the current repository by clicking the drop-down arrow on the top left of the GitHub Desktop client and selecting “chotiwitj --> Piper-TH-Workshop”



5. Click on the “Show in Explorer” button to open the downloaded files in a windows explorer



6. All of the labs for this manual and corresponding labs are contained in the “Part 1” and “Part 2” folders.





Retrospective: A read-only master source repository

Now that we have copied all of the source files to our local folder, we can now use these files for any labs going forward.

It is important to complete this lab as it will become a pre-requisite for all labs throughout this course.

When using the source files, ensure you copy the files to your personal project folder (e.g. D:\MyProject, or whatever you setup in Lab 1).

DO NOT edit or update the files directly in original folder as this will prompt the GitHub Desktop client to commit the changes and you may try to push them to GitHub by mistake.



Lab 4: 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

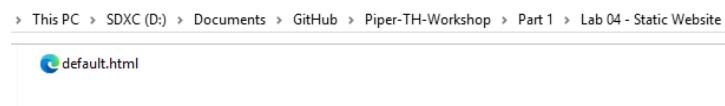


Lab Exercise: Copy the static webpage provided and setup a new project

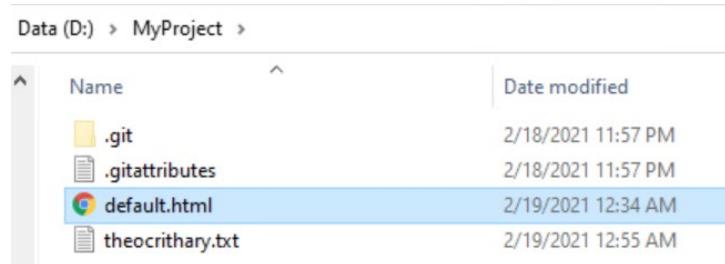
1. (If not already done in Lab 3), download the files from Lab 04 - Static Website directory of the repository (link as below) and place into your working project folder

<https://github.com/chotiwitj/Piper-TH-Workshop/tree/master/Part%201/Lab%2004%20-%20Static%20Website>

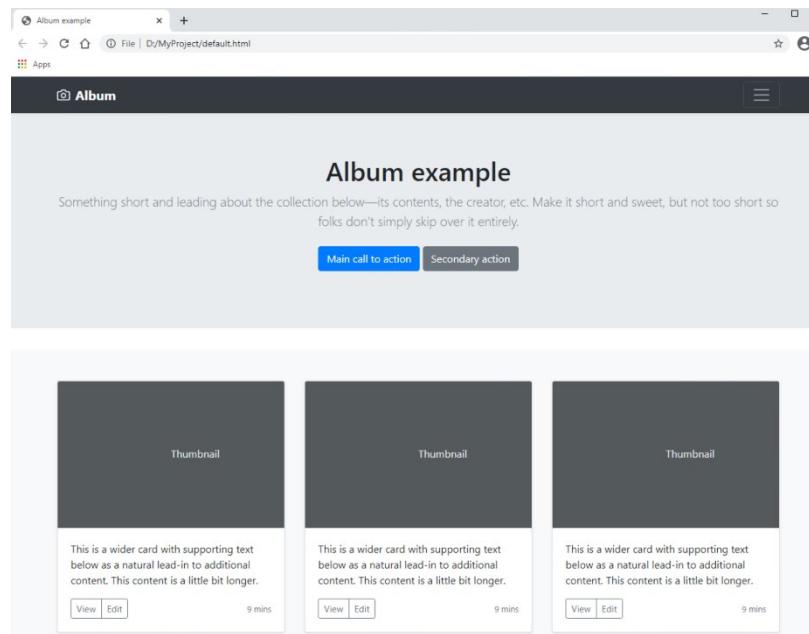
2. Open the D:\Piper-TH-Workshop folder and browse to the “Part 1 > Lab 04 – Basic Website” folder



3. Copy the ‘default.html’ file and paste into your personal working directory



4. Open the ‘default.html’ file in a browser to test its ‘look and feel’



5. Open the ‘default.html’ file in the Atom text editor and review the code

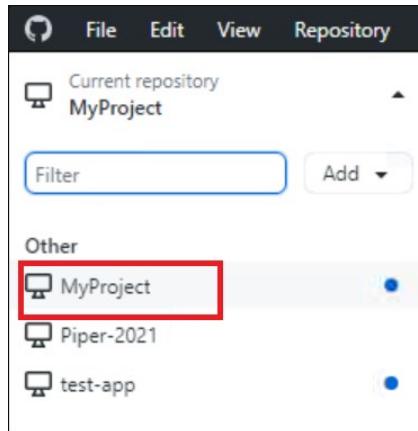


```

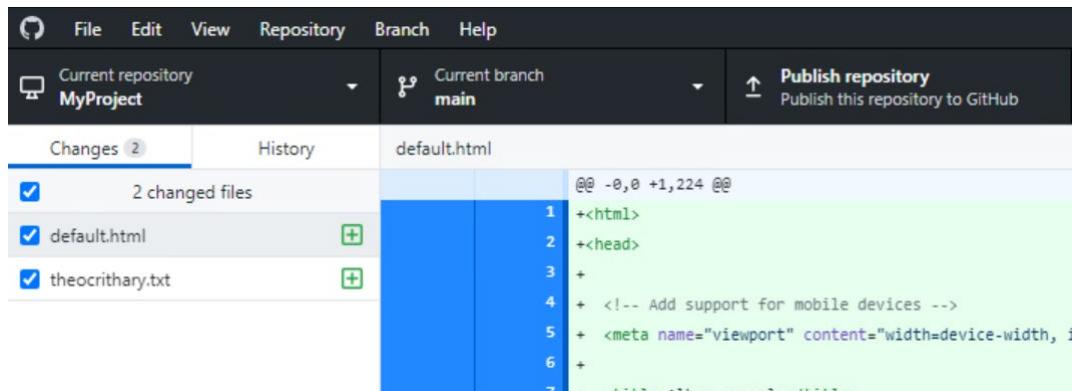
default.html
1 <html>
2 <head>
3
4     <!-- Add support for mobile devices --&gt;
5     &lt;meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"&gt;
6
7     &lt;title&gt;Album example&lt;/title&gt;
8
9     &lt;!-- Bootstrap core CSS --&gt;
10    &lt;link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1143x" crossorigin="anonymous">
11
12 </head>
13 <body>
14
15     <!-- Top Navbar Section -->
16     <header>
17         <div class="collapse bg-dark" id="navbarHeader">
18             <div class="container">

```

6. Open the GitHub Desktop client and select the “Current repository” as your personal project using the drop-down button and selecting the project you setup in Lab 1

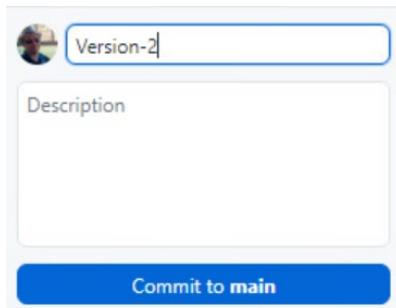


7. Notice the updated changes and new file that has been added to your working directory





8. The current changes in the working directory is still only local and has not yet been committed to the local git repository. In order to commit the changes, type a comment in the summary field and click on commit to main



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

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

10. Now we need to publish the local changes to the github.com repository by clicking on “Push origin”



No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.





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

No local changes

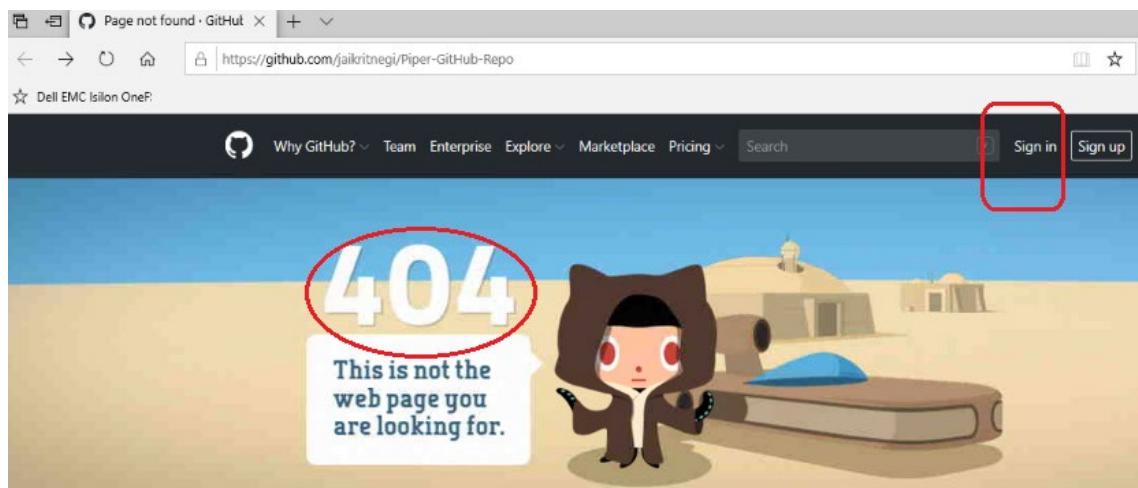
There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next:

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](#)

12. Should you encounter error code 404 on the web browser as below, click on sign in to login with your GitHub credentials to view the published repository.



Sign in to GitHub

Username or email address

Password [Forgot password?](#)

[Sign in](#)



13. Upon successful login, you can verify the published repository and content on GitHub.com repository

The screenshot shows a GitHub repository page for 'theocrithary/MyProject'. The repository is private. The commit history is as follows:

File	Commit Message	Time Ago
.gitattributes	Initial commit	3 hours ago
default.html	Version-2	6 minutes ago
theocrithary.txt	.	4 minutes ago

At the bottom, there is a note: "Add a README with an overview of your project." and a green "Add a README" button.



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

Thumbnail

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

[View](#) [Edit](#) 9 mins

Thumbnail

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

[View](#) [Edit](#) 9 mins

Thumbnail

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

[View](#) [Edit](#) 9 mins

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

```
<!-- 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-ggOyR0iXcbMQQ3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
```

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

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



Lab 5: Python and Flask

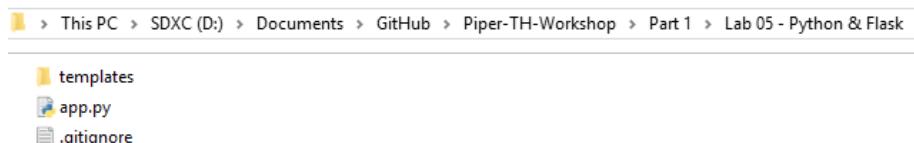
Module Objectives:

- 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

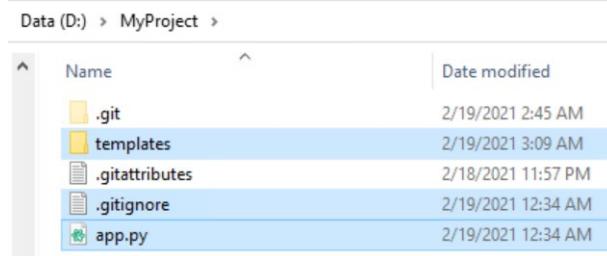


Lab Exercise: Create a basic Python app to run the Flask web server

1. (If not already done in Lab 3), download the files from Lab 05 – Python & Flask directory of the repository (link as below) and place into your working project folder
<https://github.com/chotiwitj/Piper-TH-Workshop/tree/master/Part%201/Lab%2005%20-%20Python%20%26%20Flask>
2. Delete the existing ‘default.html’ and ‘theocrithary.txt’ files from Labs 1 & 4
3. Copy all the files and folders from the Lab 05 directory above and paste them into your working project directory (e.g. D:\MyProject)



4. Your project directory should look like this;



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

```
> d:
```

```
> cd MyProject
```

6. Run the following command to start your application

```
> python app.py
```

```
D:\MyProject>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://localhost:5000/ (Press CTRL+C to quit)
```

7. Open a browser with the following URL

<http://localhost:5000>

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

9. Save work in your working project folder and upload to GitHub



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;

```

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 from flask import Flask, render_template, request
11
12 # Create a Flask instance
13 app = Flask(__name__)
14
15 ##### Define routes #####
16 @app.route('/')
17 def home():
18     return render_template('default.html',url="home") ← Remember controllers / routes from MVC
19                                         architecture principles?
20
21 ##### Run the Flask instance, browse to http://localhost:5000 #####
22 if __name__ == "__main__":
23     app.run(debug=False, host='localhost', port='5000', threaded=True) ← And views / templates?
24

```

← Port bindings
from 12-factor
apps?

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 6: MongoDB

Module Objectives:

- Setup local MongoDB server & install pymongo module (this is 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 files from Lab 06 - MongoDB directory of the repository (link as below) and place into your working project folder
<https://github.com/chotiwitj/Piper-TH-Workshop/tree/master/Part%201/Lab%2006%20-%20MongoDB>
2. Copy all files in the Lab 06 - MongoDB folder and place into your working directory (overwrite any existing files in your working directory)
3. Your project directory should now look like this;

Data (D:) > MyProject >		
	Name	Date modified
	.git	2/19/2021 2:45 AM
	templates	2/19/2021 3:30 AM
	.gitattributes	2/18/2021 11:57 PM
	.gitignore	2/19/2021 12:34 AM
	app.py	2/19/2021 12:34 AM
	models.py	2/19/2021 12:34 AM

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

```
> d:  
> cd MyProject  
> python app.py
```

```
D:\MyProject>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://localhost:5000/ (Press CTRL+C to quit)
```

5. Open a browser with the following URL

<http://localhost:5000>

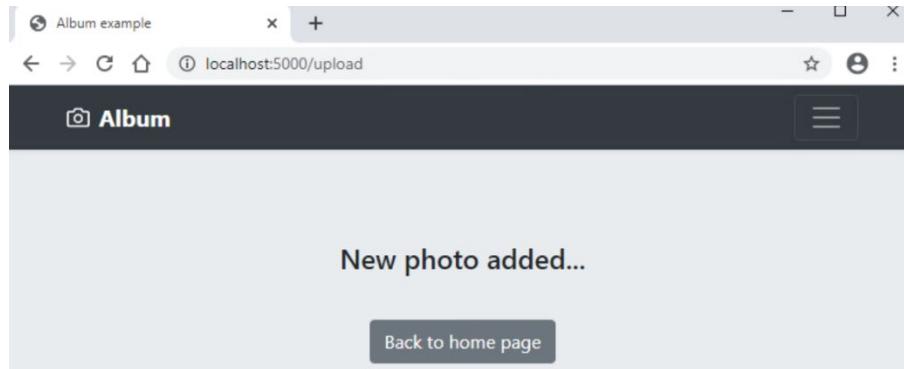
6. Test uploading an image and confirm the page loads correctly with the newly introduced functionality of a couple of text fields and an image upload function.

The screenshot shows a web page titled "Photo upload". The page instructs the user to "Use this form to upload your photo's to the album." It features two input fields: "Title" and "Comments", both currently empty. Below these is a file input field labeled "Choose File" which displays "No file chosen". To the right of the file input is a blue "Upload" button. The overall layout is clean and modern, typical of a web-based file upload interface.

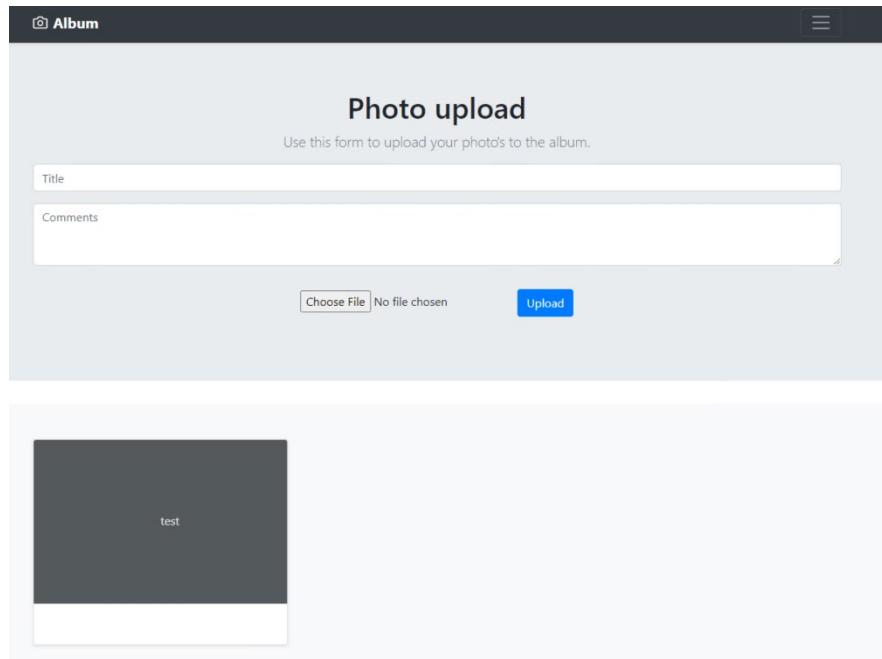


7. Test the functionality by entering some text in the 'title' and 'comments' field, then clicking the 'Choose File' button and selecting an image before clicking on 'Upload'.

If your upload is successful, you will see the below confirmation page.



8. Click on the 'Back to home page' function and you will be sent back to the default home page with a placeholder thumbnail representing your uploaded image.



9. Press CTRL + C on the command prompt to quit upon testing the webapp
10. Save work in your working project folder and upload to Github (if any changes to your local working directory)
 - Follow Lab 4- Steps 6 to 11 (if need guidance on the steps for pushing the changes from local repository to GitHub.com 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 modal window titled "Photo upload" with the sub-instruction "Use this form to upload your photos to the album." It contains fields for "Title" and "Comments", and a file upload section with "Choose File" and "Upload" buttons. On the right, the main content area displays the message "New photo added..." above a "Back to home page" button. Below this message, there is a thumbnail image labeled "Test Photo".

At this point, we still do not have anywhere to store the images and are only inserting the name of the image into the database. 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 7: ECS

Module Objectives:

- Setup object storage with ECS test drive
- 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: Use Boto3 Python module to upload images to ECS

1. (If not already done in Lab 3), download the following files from Lab 07 - ECS directory of the repository (link as below) and place into your working project folder
<https://github.com/chotiwitj/Piper-TH-Workshop/tree/master/Part%201/Lab%2007%20-%20ECS>
2. Copy all files in the Lab 07 - ECS folder and place into your working directory (overwrite any existing files in your working directory)
3. Your project directory should look like this;

Data (D:) > MyProject >		
	Name	Date modified
	.git	2/19/2021 2:45 AM
	__pycache__	2/19/2021 3:40 AM
	templates	2/19/2021 4:34 AM
	uploads	2/19/2021 4:34 AM
	.gitattributes	2/18/2021 11:57 PM
	.gitignore	2/19/2021 12:34 AM
	app.py	2/19/2021 12:34 AM
	config-example.py	2/19/2021 12:34 AM
	models.py	2/19/2021 12:34 AM

4. Rename the 'config-example.py' file to 'config.py'
5. Login to ECS Test Drive portal at <https://portal.ecstestdrive.com/> using the account you created during the pre-requisites before starting these labs.



6. Upon successful login, retrieve your ECS credentials from the credentials link at the top right corner of 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 23:00 (ET) Eastern Time

ECS TEST DRIVE

GET STARTED WITH 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://131.170.5410.public.ecstestdrive.com/{bucket_name}/{key_name}

Access Key: 131.170.5410@ecstestdrive.emc.com

Secret Key1: m4CrS1opCIKai+Ojpmbq

Secret Key2:

Note: if you do not have a secret key, go to ‘Manage Secret Keys’ and create a new secret key1

- Edit the ‘config.py file’ and replace the following sections with your ECS credentials

- 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 }
```

- Run your app

```
> d:  
> cd MyProject  
> python app.py
```

```
D:\MyProject>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://localhost:5000/ (Press CTRL+C to quit)
```



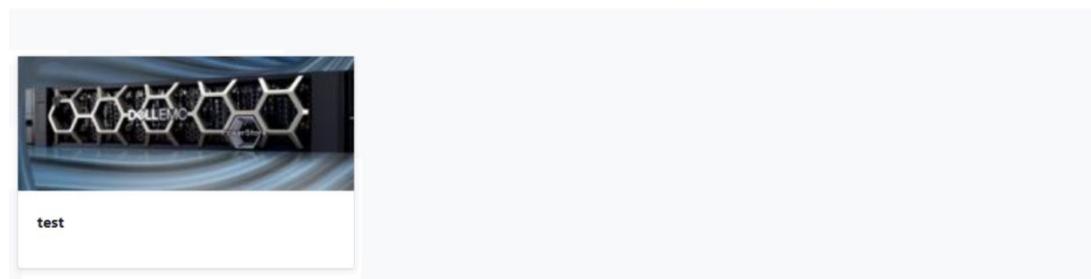
10. Test your app and upload an image;

<http://localhost:5000>

The screenshot shows a "Photo upload" form. It has fields for "Title" (containing "test") and "Comments". Below these is a "Choose File" button with "Dell-Intro.jpg" selected. A blue "Upload" button is also present.

The screenshot shows a confirmation message "New photo added..." after a successful upload. A "Back to home page" button is visible at the bottom.

The screenshot shows the same "Photo upload" form as the first one, but this time the file input field shows "No file chosen".





11. Press CTRL + C on the command prompt window to stop the python webapp after testing
12. Save work in your working project folder and upload to Github (if any changes to your local working directory)
 - Follow Lab 4- Steps 6 to 11 (if need guidance on the steps for pushing the changes from local repository to GitHub.com 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 from flask import Flask, render_template, request
11 from models import get_photos, insert_photo, upload_photo
12
13 # Create a Flask instance
14 app = Flask(__name__)
15
16 ##### Define routes #####
17 @app.route('/')
18 def home():
19     album_photos = get_photos()                      # Call function in 'mode
20     return render_template('default.html',album_photos=album_photos,url="home")
21
22 # This route accepts GET and POST calls
23 @app.route('/upload', methods=['POST'])
24 def upload():
25     insert_photo(request)                          # Call function in 'mode
26     upload_photo(request.files['photo'])          # Call function in 'mode
27     return render_template('submit-photo.html')    # Return a page to inform
28
29 @app.route('/photo/<path:photo>')
30 def photo(photo):
31     return render_template('photo.html',photo=photo)
32
33 ##### Run the Flask instance, browse to http://localhost:5000 #####
34 if __name__ == "__main__":
35     app.run(debug=False, host='localhost', 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 8: TAS & MongoDB

Module Objectives:

- Upload app to TAS PaaS platform
- Use a shared MongoDB instance
- Edit the TAS environment variables to connect the app to the MongoDB database
- 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

1. (If not already done in Lab 3), download the following files from Lab 08 - TAS & MongoDB directory of the repository (link as below) and place into your working project folder

<https://github.com/chotiwitj/Piper-TH-Workshop/tree/master/Part%201/Lab%2008%20-%20TAS%20%26%20MongoDB>

2. Copy all files in the Lab 08 – TAS & MongoDB folder and place into your working directory (overwrite any existing files in your working directory)
3. Your project directory should look like this;

Data (D:) > MyProject		
Name	Date modified	Type
.git	2/19/2021 2:45 AM	File folder
__pycache__	2/19/2021 5:00 AM	File folder
templates	2/19/2021 5:09 AM	File folder
uploads	2/19/2021 5:01 AM	File folder
.gitattributes	2/18/2021 11:57 PM	Text Document
.gitignore	2/19/2021 12:34 AM	Text Document
app.py	2/19/2021 12:34 AM	PY File
config.py	8/11/2020 9:22 AM	PY File
config-example.py	2/19/2021 12:34 AM	PY File
manifest.yml	2/19/2021 12:34 AM	YML File
models.py	2/19/2021 12:34 AM	PY File
Procfile	2/19/2021 12:34 AM	File
requirements.txt	2/19/2021 12:34 AM	Text Document
runtime.txt	2/19/2021 12:34 AM	Text Document

4. Edit the manifest.yaml file and make the following 2 changes
 - Add a unique name to the application name: line

Your file should look similar to the below;

```
manifest.yml
1  ---
2  applications:
3  - name: photo-album-tc
4    memory: 64M
5    instances: 1
6    random-route: true
7    buildpacks:
8      - https://github.com/cloudfoundry/python-buildpack.git
```

5. Open a command prompt and navigate to your project

```
> d:  
> cd MyProject
```

6. Login to TAS with the below account credentials

```
> cf login -a https://api.sys.tan-zu.com --skip-ssl-validation
```



```
D:\MyProject>cf login -a https://api.sys.tan-zu.com --skip-ssl-validation
API endpoint: https://api.sys.tan-zu.com

Email: student

Password:
Authenticating...
OK

Targeted org piper
Targeted space development

API endpoint: https://api.sys.tan-zu.com (API version: 3.85.0)
User: student
Org: piper
Space: development
```

7. Push the app to the TAS platform

```
> cf push
```

```
D:\MyProject>cf push
Pushing from manifest to org apples-pivotal-org / space development as theo.crithary@dell.com...
Using manifest file D:\MyProject\manifest.yml
Getting app info...
Creating app with these attributes...
+ name: photo-album
+ path: D:\MyProject
+ instances: 1
+ memory: 64M
routes:
+ photo-album-shiny-hippopotamus-dn.cfapps.io

Creating app photo-album...
Mapping routes...
Comparing local files to remote cache...
Packaging files to upload...
Uploading files...
90.00 KiB / 90.00 KiB [=====] 100.00% 1s
Waiting for API to complete processing files...
Staging app and tracing logs...
```

8. 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
```

This is because we have not provided the connection details for the MongoDB database. In the next section (Part B) we will use the platform environment variables to connect our application to the database as a more secure 12-factor compliant method of storing credentials than the previous examples of using plain text files and hard coded connection strings.



Lab Exercise (Part B): Add the MongoDB database connection environment variables

1. Open a browser and go to <https://apps.sys.tan-zu.com>
(accept the SSL certificates as they are self-signed, you will need to do this twice)



Your connection is not private

Attackers might be trying to steal your information from apps.sys.tan-zu.com (for example, passwords, messages, or credit cards). [Learn more](#)

NET::ERR_CERT_AUTHORITY_INVALID

To get Chrome's highest level of security, [turn on enhanced protection](#)

[Hide advanced](#)

[Back to safety](#)

This server could not prove that it is apps.sys.tan-zu.com; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

[Proceed to apps.sys.tan-zu.com \(unsafe\)](#)

2. Login with the same account credentials as above

3. Click on the org name – piper

Orgs			
Org Name	Quota	Spaces	Domains
piper	1% (64 MB / 10 GB)	1	0



Then the space named – development

Spaces					Org Quota Usage
Name	Apps	App Status	Services		Org Quota Usage
development	1	● 0 ■ 0 ▼ 0	0		0% (0 Bytes / 10 GB)

Then 'your' application, as named in the manifest file in Part A - e.g. photo-album-tc

Apps					
Status	Name	Instances	Memory	Last Update	Route
● STARTING	photo-album-tc	1	61.44 MB	2 min	https://photo-album-tc-i... ↗

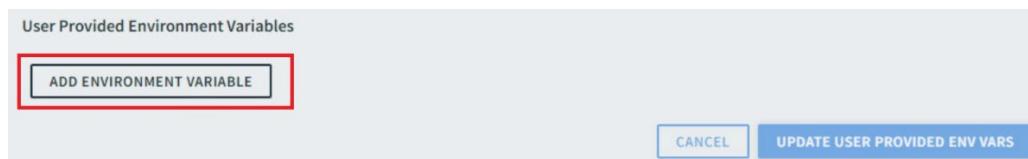
- Now, on the left-hand menu, click 'settings'

Processes and Instances		AUTOSCALING DISABLED	SCALE
Instances	1	Memory Allocated	64 MB
#	CPU	Memory	Disk
0	CRASHED		Uptime

- Scroll down to the section "User Provided Environment Variables" and click on "REVEAL USER PROVIDED ENV VARS"

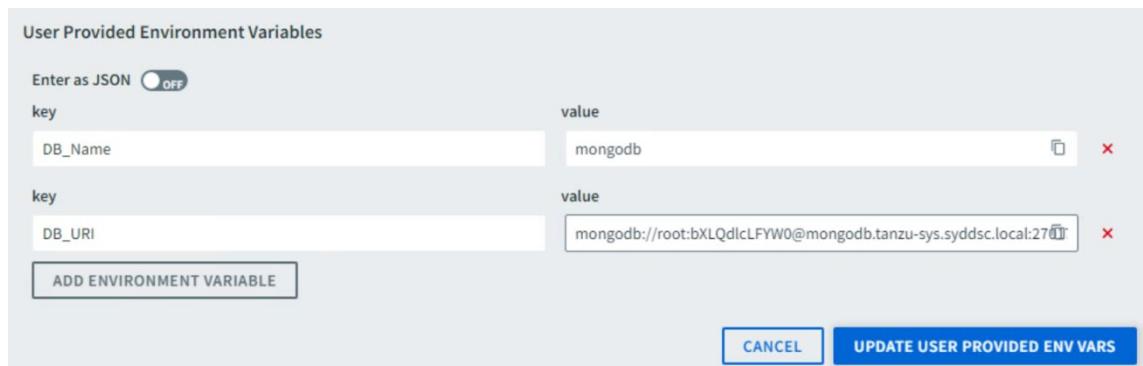


6. Then click on “ADD ENVIRONMENT VARIABLES”



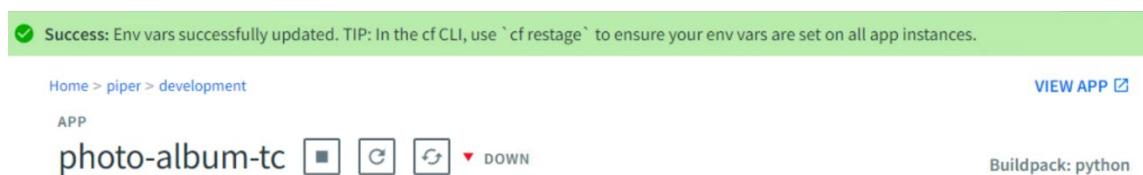
7. Add these two users provided environment variables-

- 1st key value pair as:
 - Key- DB_Name
 - Value- mongodb
- Then, click on ADD ENVIRONMENT VARIABLE and 2nd key value pair as below
 - key- DB_URI
 - value- mongodb://root:bXLQdIcLFYW0@mongodb.sys.tan-zu.com:27017/?authSource=admin

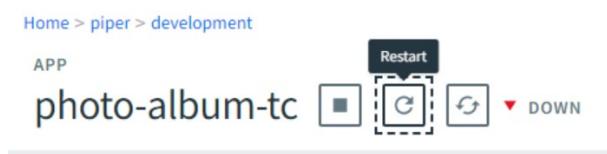


Then, click on ‘update user provided env vars’ to save the changes

8. Upon successful update, you will receive the following message as successfully updated.

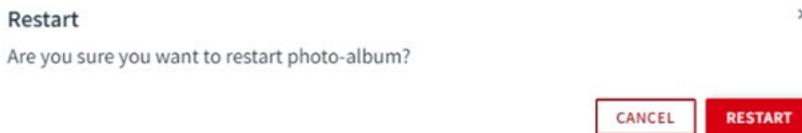


9. Click on restart button on the App- photo-album.

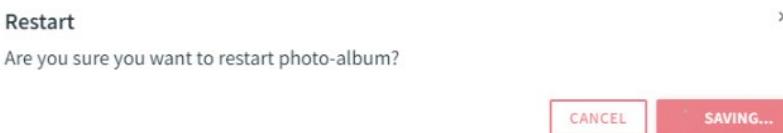




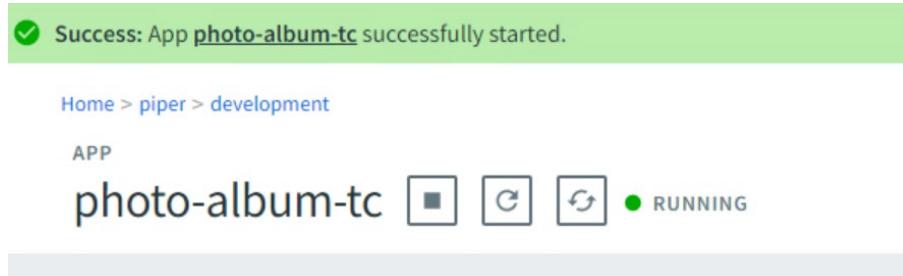
10. Confirm Restart when message pops up for confirmation to restart the app.



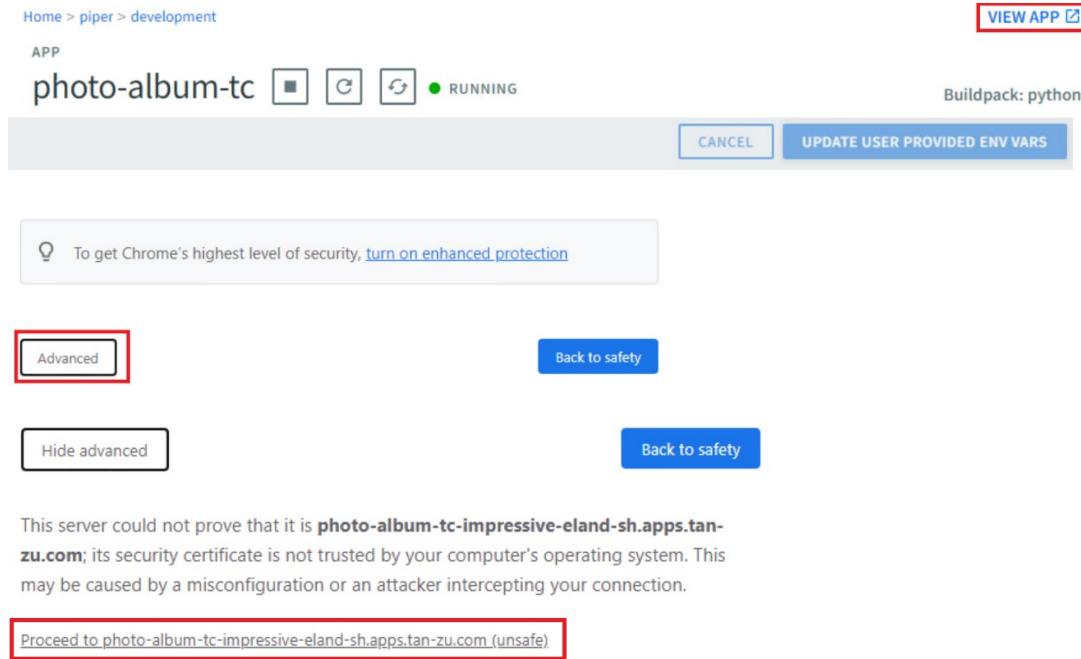
11. Await while its saving... (takes a minute)



12. Upon successfully saving the settings, you should see app successfully started with below message



13. Verify the app by clicking on View App and accept the self-signed certificates again





14. Confirm the functionality of the app by uploading an image.

Album

Photo upload

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

test

Comments

Choose File Dell-Intro.jpg

Upload

Album

New photo added...

Back to home page

Album

Photo upload

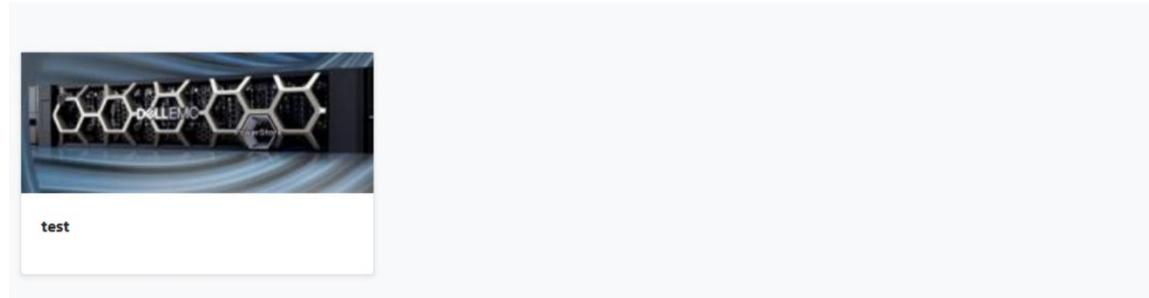
Use this form to upload your photo's to the album.

Title

Comments

Choose File No file chosen

Upload





15. Clean up and delete your app by running the following in the command prompt
 > cf delete photo-album -r

Note: use your unique application name as specified in the manifest file and don't forget the '-r' option at the end to remove the route for your application



Retrospective: Review code and take note of the 'DB_URI' 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 for user defined environment variables for MongoDB that we setup manually in the previous Lab Part B section.

```

19  ### Application configuration settings ####
20  # Check if user defined environment variable exists
21  if "DB_URI" in os.environ:
22      client = MongoClient(os.environ['DB_URI'])
23      DB_NAME = os.environ['DB_Name']

```

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> v Lab 08 - TAS & MongoDB > templates > uploads . gitignore app.py config-example.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> requirements.txt 1 pip==21.0.1 2 Flask==1.1.2 3 boto3==1.17.4 4 Pillow==8.1.0 5 pymongo==3.11.3 6 Werkzeug==1.0.1 7 dnspython==2.1.0 </pre>
	<pre> Procfile 1 web: python3 app.py </pre>	<pre> runtime.txt 1 python-3.9.1 </pre>