Peer Review Assignment

Estimated time needed: **60** minutes

Objectives

In this assignment you will:

- Use Watson APIs to create functions.
- Create a function that translates English to French.
- Create a function that translates French to English.
- Run coding standards check against the functions above.
- Write unit tests to test the above functions.
- Run unit tests and interpret the results.
- Package the above functions and tests as a standard python package.

Important Notice - Please keep in mind that sessions for this lab environment are not persisted. If you will not be completing the entire lab in one sitting, please save your code outisde of this environment, so you can resume your work without loss.

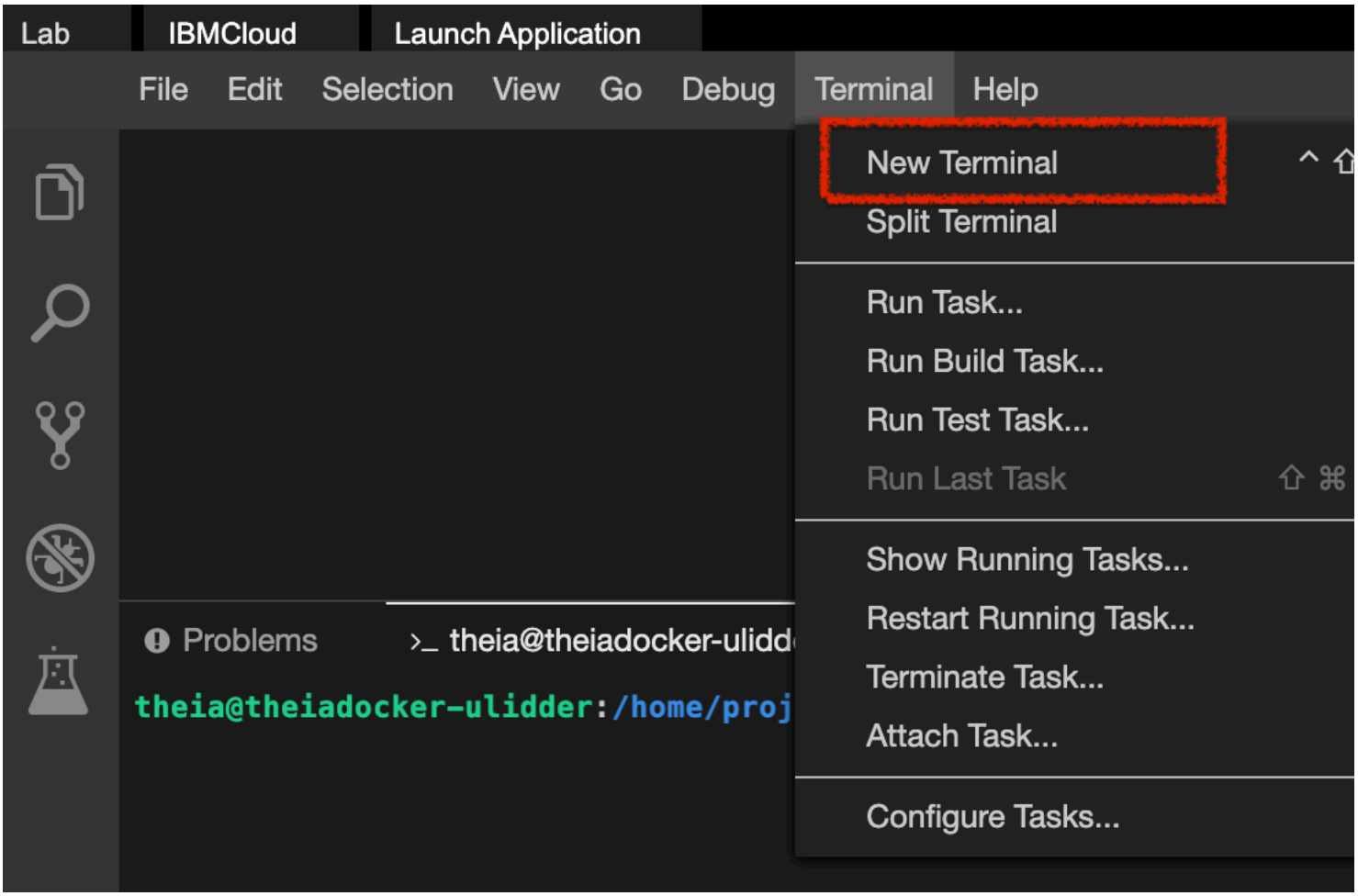
▶ 👉 *Click here for instructions to save your code on github*

Provision Watson Translation Service

- Before you start, ensure you have provisioned an instance of the Watson Language Translator service and have API information available.

Task1: Write a function that translates English text to French in translator.py

1. Open a terminal window by using the menu in the editor: Terminal > New Terminal.



2. Go to the project home directory.

```
1. 1
1. cd /home/project
```

Copied!

3. Run the following command to Git clone the project directory from the clone URL you had copied in the prework lab.

```
1. 1
1. [ ! -d 'xzceb-flask_eng_fr' ] && git clone <paste_your_repo_name>
```

Copied!

4. Change to the `final_project` folder.

```
1. 1
1. cd /home/project/xzceb-flask_eng_fr/final_project
```

Copied!

5. Create folder named `machinetranslation` and change to that directory.

```
1. 1
2. 2
1. mkdir machinetranslation
2. cd machinetranslation
```

Copied!

6. Run the following command to make sure that `python3` is using version 3.8.

```
1. 1
1. sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.8 10
```

Copied!

7. Run the following command to check the python version.

```
1. 1
```

```
1. python3 --version
```

Copied!

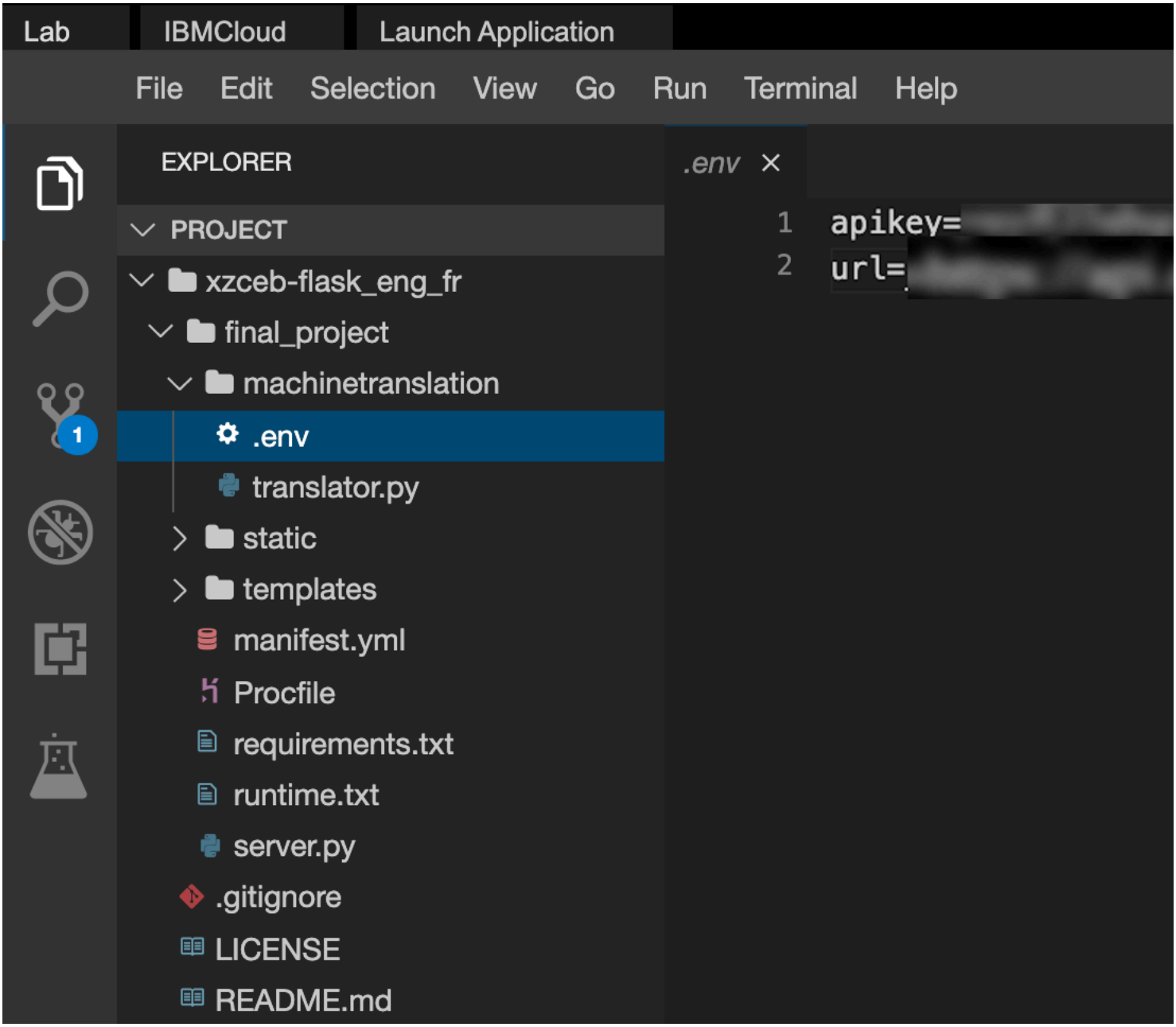
Note: It should be `Python 3.8.0`.

8. Install the packages that you will be using in this code, namely `ibm_watson` , `python-dotenv`and `Flask`.

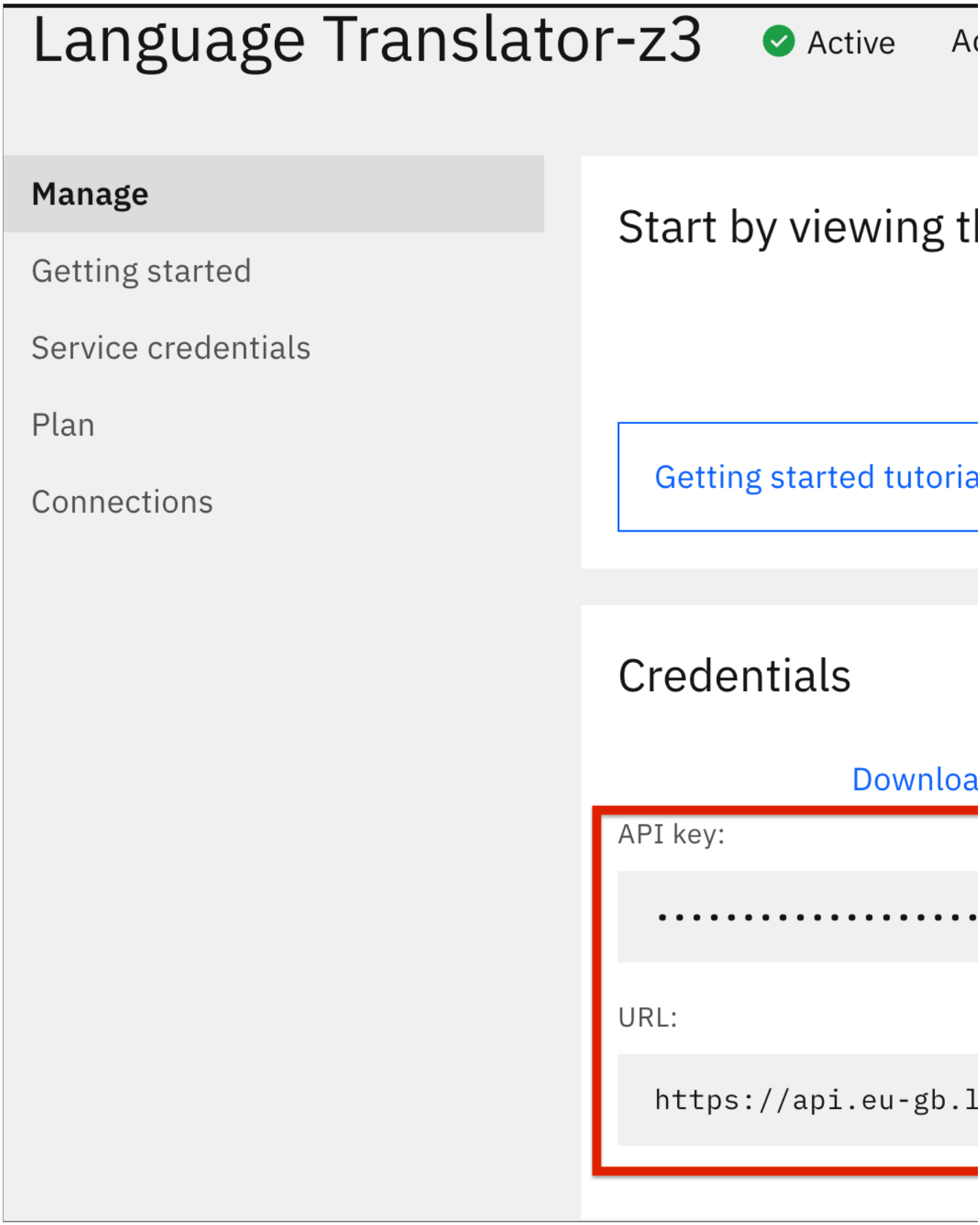
```
1. 1
2. 2
3. 3
1. python3 -m pip install python-dotenv
2. python3 -m pip install ibm_watson
3. python3 -m pip install Flask
```

Copied!

9. Create a file `.env` under the `machinetranslation` directory, which maps the apikey and url of the Language Translator Service that you created in the IBM Cloud.



▼ 👉 Click here to see where the credentials can be copied from



10. In the explorer, go to the `machinetranslation` directory and create a new file called `translator.py`. Enter the following lines of code.

```
1
2
3
4
5
6
7
8
9
10
11
12
```

```
```py
import json
from ibm_watson import LanguageTranslatorV3
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
import os
from dotenv import load_dotenv

load_dotenv()

apikey = os.environ['apikey']
url = os.environ['url']
```
```

Copied!

11. Add code to create an instance of the IBM Watson Language translator in `translator.py`.

    Hint : You may refer to the IBM Watson Language Translation API documents here.

    📷 Take a screenshot of your functions and save it as a .jpg or .png with the filename `translator_instance`. You will be prompted to upload the screenshot in the Peer Assignement that follows.

12. Add function **englishToFrench** which takes in the `englishText` as a string argument, in `translator.py`. Use the instance of the Language Translator you created previously, to translate the text input in English to French and return the French text.

```
1
2
3
def englishToFrench(englishText):
    #write the code here
    return frenchText
```

Copied!

📷 Take a screenshot of your functions and save it as a .jpg or .png with the filename `e2f_translator_function`. You will be prompted to upload the screenshot in the Peer Assignement that follows.

13. Add function **frenchToEnglish** which takes in the `frenchText` as a string argument, in `translator.py`. Use the instance of the Language Translator you created previously, to translate the text input in French to English and return the English text.

```
1
2
3
def frenchToEnglish(frenchText):
    #write the code here
    return englishText
```

Copied!

    📷 Take a screenshot of your functions and save it as a .jpg or .png with the filename `f2e_translator_function`. You will be prompted to upload the screenshot in the Peer Assignement that follows.

Task 2: Write the unit tests for English to French translator and French to English translator function in tests.py

1. Create a new file called `tests.py` in the `machinetranslation` directory.
2. Write at least 2 tests in `tests.py` for each method
3. Test for null input for `frenchToEnglish`
4. Test for null input for `englishToFrench`.
5. Test for the translation of the world ‘Hello’ and ‘Bonjour’.
6. Test for the translation of the world ‘Bonjour’ and ‘Hello’.
7. Take a screenshot of your unit tests and save it as a .jpg or .png with the filename `translation_unittests`.

Task 3: Check your code against python coding standards

1. At the terminal run the following command to install pylint.

```
1. 1
1. python3 -m pip install pylint
```
Copied!

2. Run `pylint translator.py` to check the coding standard compliance in your code. Refer to this exercise you did earlier, if needed.
3. Make sure your rating is at least 7.
4. 📷 Take a screenshot of the output of the pylint analysis report showing your score and save it as a .jpg or .png with the filename `pylint_score`.

Task 4: Run tests

1. At the terminal run the command

```
1. 1
1. python3 tests.py
```
Copied!

2. 📷 Take a screenshot of test results and save it as a .jpg or .png with the filename **unit_test_results**.

Task 5: Package the above functions and tests as a standard python package.

1. Create `__init__.py` file in the directory `machinetranslation`.
2. Create a folder called `tests` under the newly created folder
3. Copy the unit `tests` into the tests folder

📷 Take a screenshot of the folder structure of the package (From the menu go to View -> Explorer to set the explorer view) and save it as a .jpg or .png with the filename `package_folder_structure`.

Task 6: Import the package into server.py and create flask end points

1. Import the package `machinetranslation` in server.py.
2. In the server.py, for end-point `/`, implement a method that renders the `index.html`.
3. In the space provided in server.py for end-point `/englishToFrench` implement a method that uses the appropriate translation function through the package you created in the previous part. The function should take the English text as input through the request parameter and return a string.
4. In the space provided in server.py for end-point `/frenchToEnglish` implement a method that uses the appropriate translation function through the package you created in the previous part. The function should take the French text as input through the request parameter and return a string.
5. Push the code to GitHub.

▼ 👉 *Click here for instructions to push your code to github*
Change to the project directory

```
1. 1

1. cd /home/projects/xzceb-flask_eng_fr
```
Copied!

- Run these following commands replacing your github username and the email you use for loggin into github as appropriate.

```
1. 1
2. 2

1. git config --global user.email youremail@domain.com
2. git config --global user.name your_github_username
```
Copied!

```
1. 1

1. git add .
```
Copied!

```
1. 1

1. git commit -m"Final changes"
```
Copied!

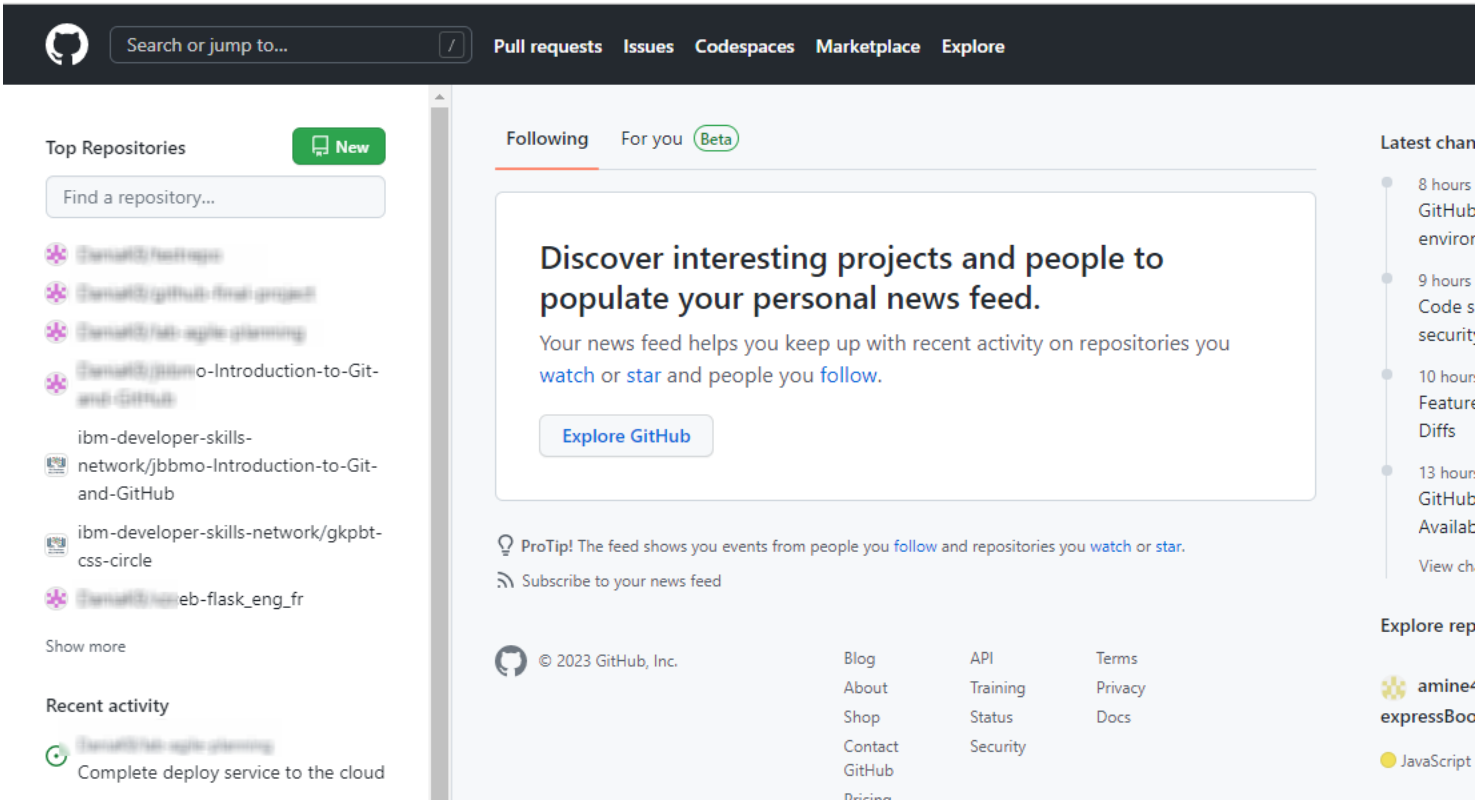
Task - Generate Personal Access Token

- 1. 1
  1. Verify your email address if it hasn't been verified on Github.

  Copied!

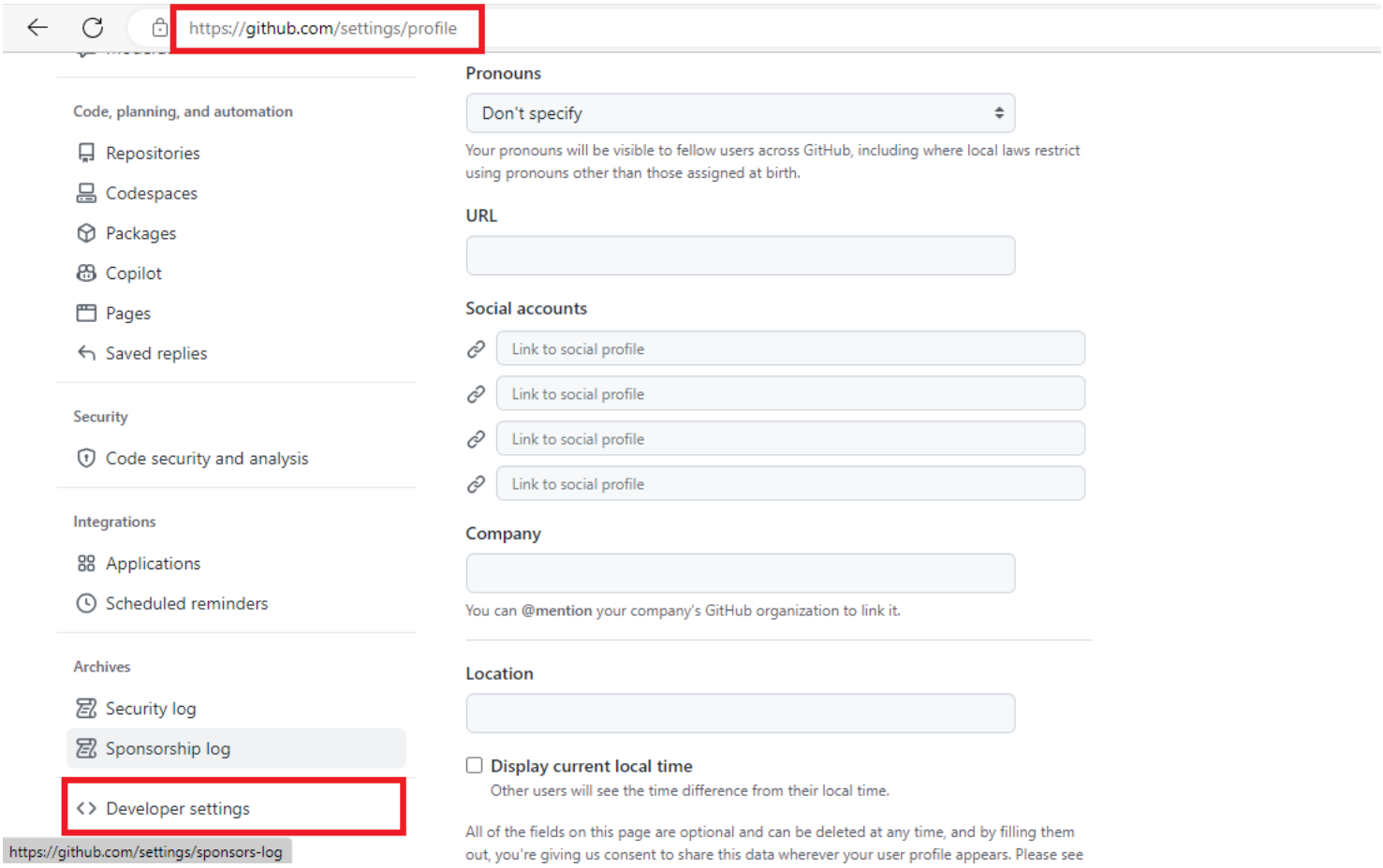
- 1. 1
  1. In the upper-right corner of any page, click your profile photo, then click Settings.

  Copied!



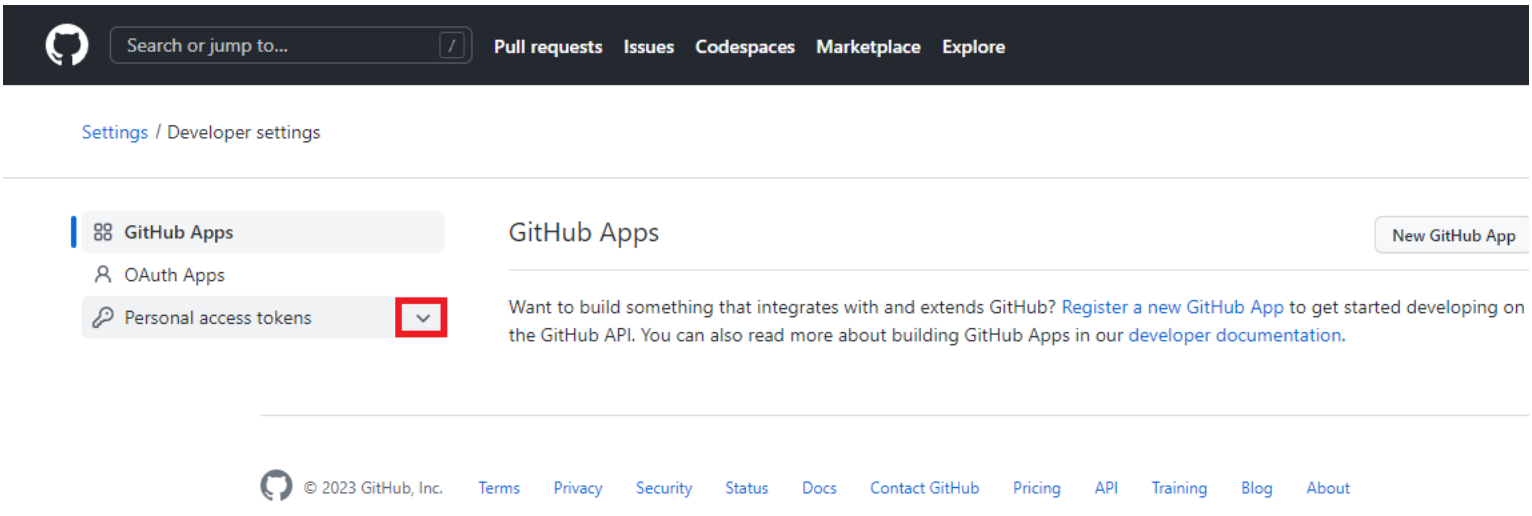
- 1. 1
  1. In the left sidebar, click Developer settings.

  Copied!



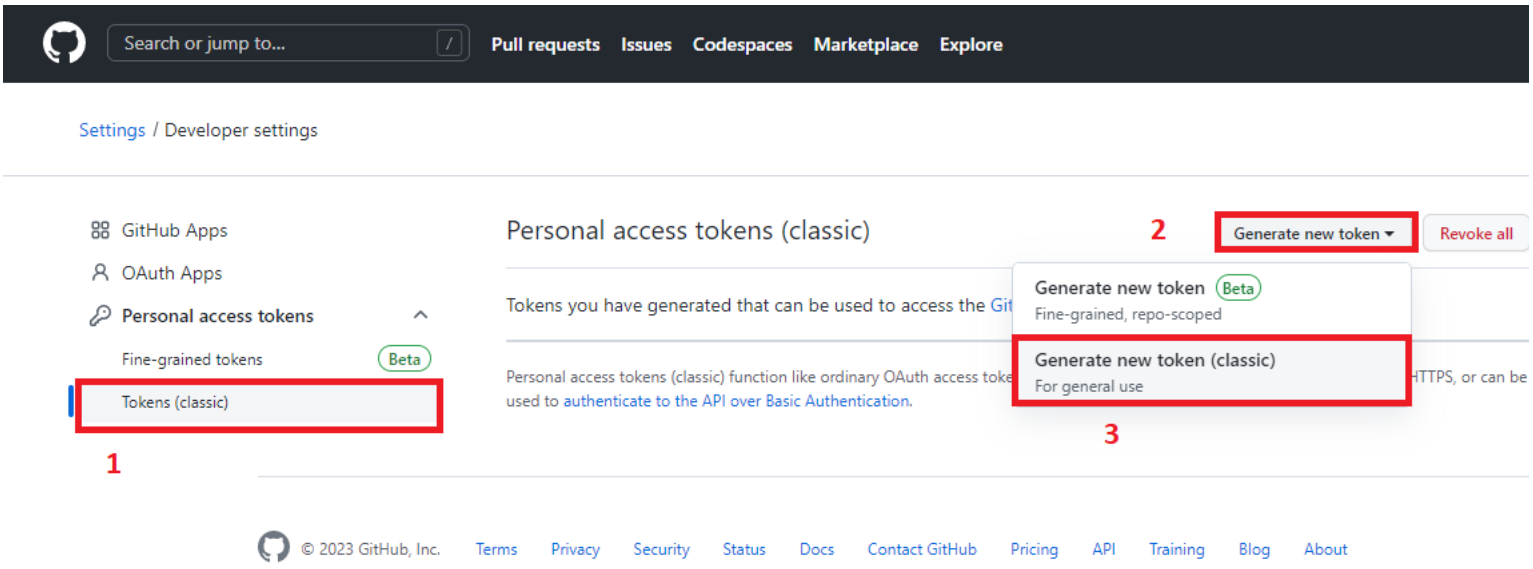
- 1. 1
  1. In the left sidebar, click drop-down menu `Personal access tokens`.

Copied!



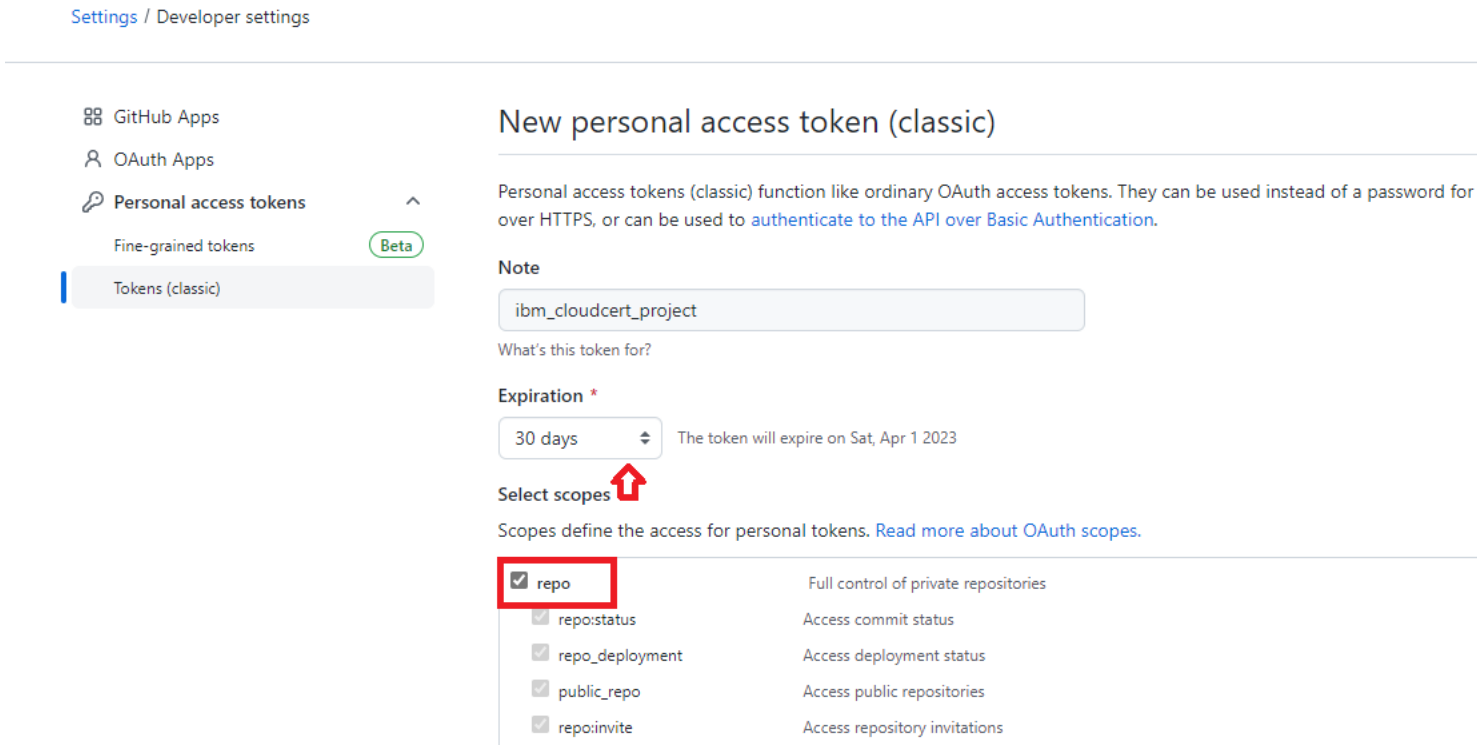
- 1. 1
  1. Click on `Token (classic)` and then click drop-down menu `Generate new token` and click `Generate new token (classic)`.

Copied!



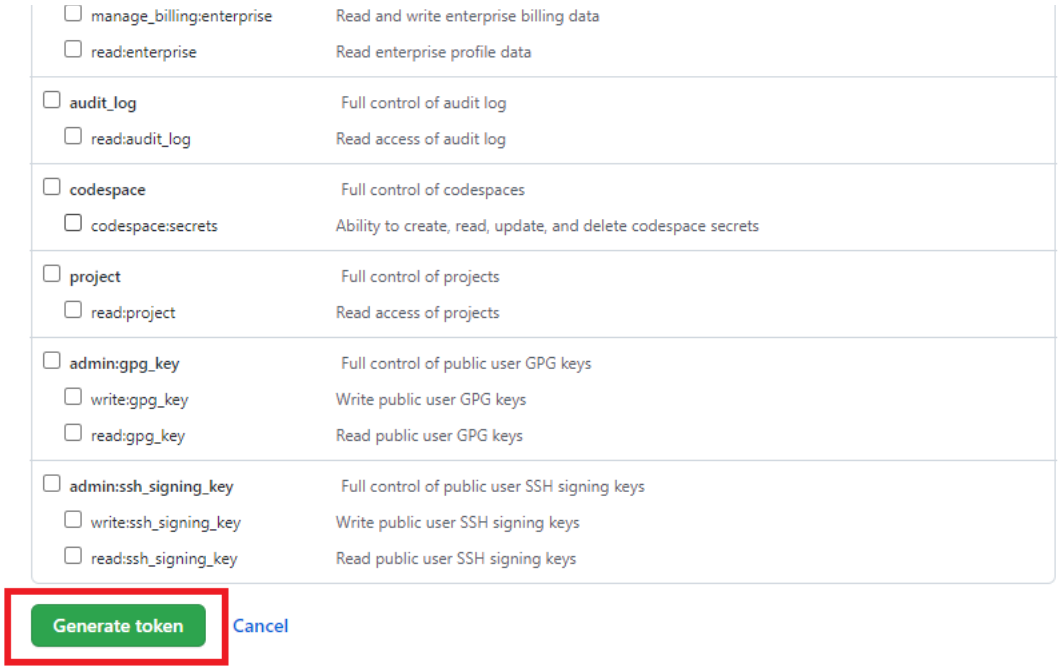
- 1. 1
  1. Give your token a descriptive name. To give your token an expiration, select the Expiration drop-down menu, then click a default or use t

Copied!

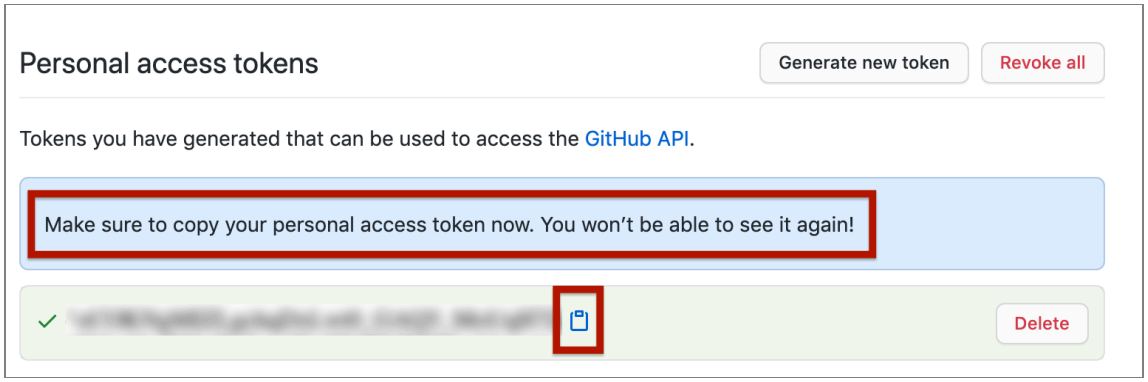


- 1. 1
  1. Click `Generate token` and make a note of it.

Copied!



- Make sure you copy the token and keep it safe. It is not visible to you again.



Treat your tokens like passwords and keep them a secret.

Once you have a token, you can enter the Personal Access Token as password when performing Git operations.

The git `push` command will enable you to sync all the changes made locally to the GitHub web repository.

- Run the following command with your actual HTTPS link:

```
1. 1
1. git push [HTTPS link]
```

Copied!

You will be prompted by git for your username and password.

- Type your GitHub username and for the password, enter the personal access token you generated in the previous task. When you are authenticated, all committed changes are synced with your GitHub repository.

You can now visit the GitHub repository page and check to ensure that the revised and newly added files are in place.

Task 7: Run the server

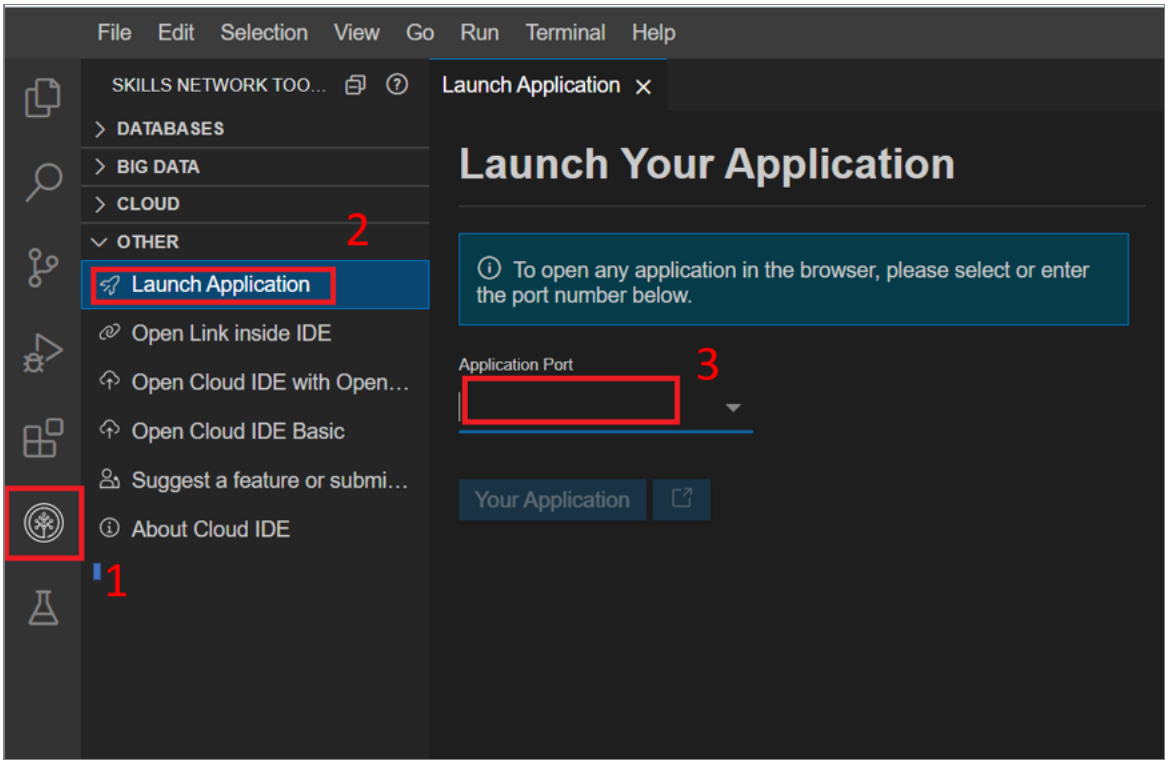
1. Change to the project directory and run the server from your terminal.

```
1. 1
1. cd /home/project/xzceb-flask_eng_fr/final_project && python3 server.py
```

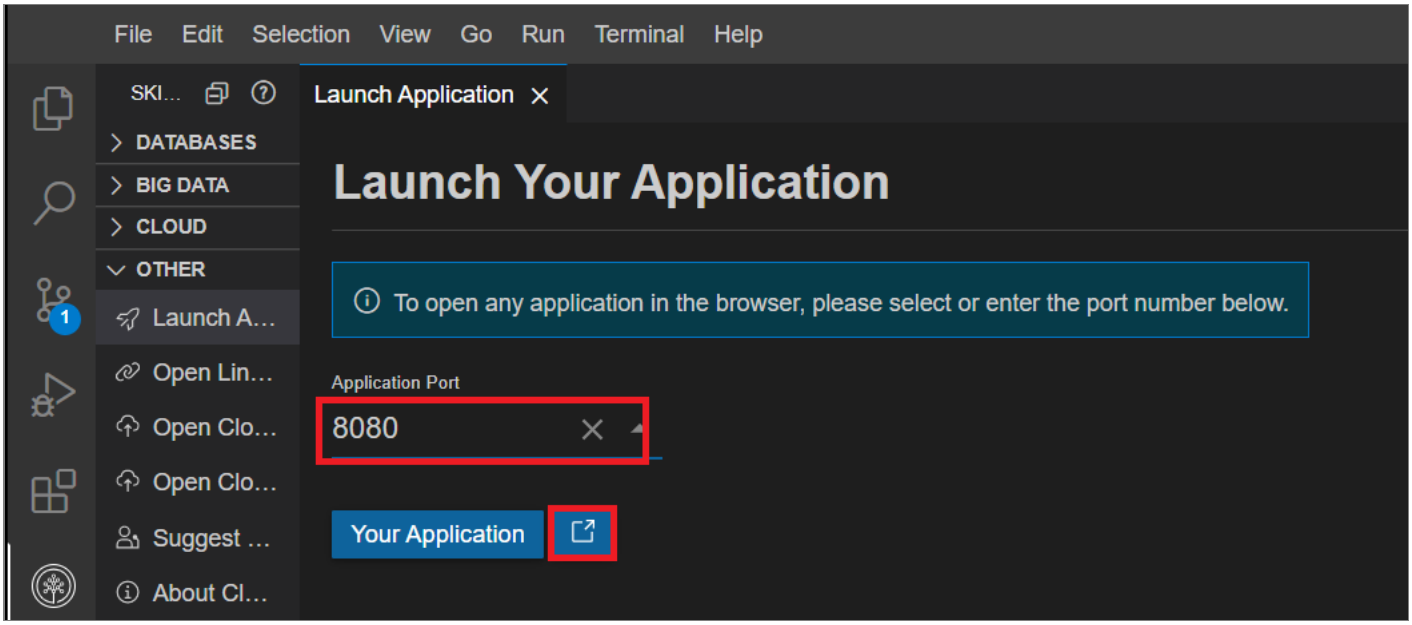
Copied!

2. You will see that the server starts up in port 8080.

3. Click on the `Skills Network button` on the left, it will open the `Skills Network Toolbox`. Then click the `Other` then `Launch Application`. From there you should be able to enter the port.



Connect to port `8080`and click `Launch` button.



4. A new browser window opens up with the index page.
    - 📷 Take a screen shot of the translation from English to French
    - 📷 Take a screen shot of the translation from French to English

Task 8: Deploy the application on Code Engine.

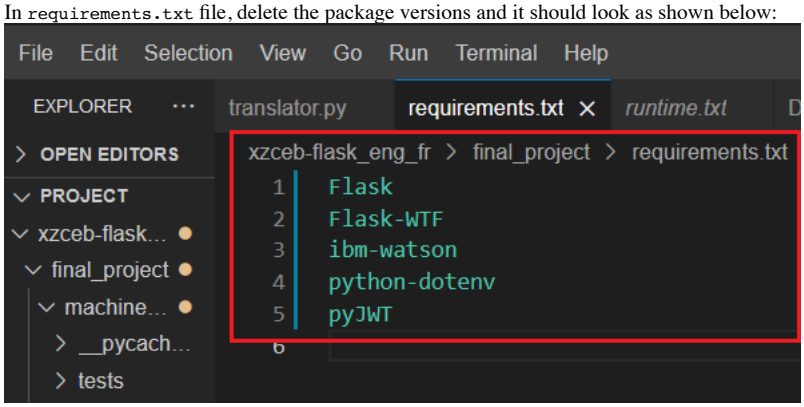
1. Change to the `final_project` directory.

```
1
cd /home/project/xzceb-flask_eng_fr/final_project
```

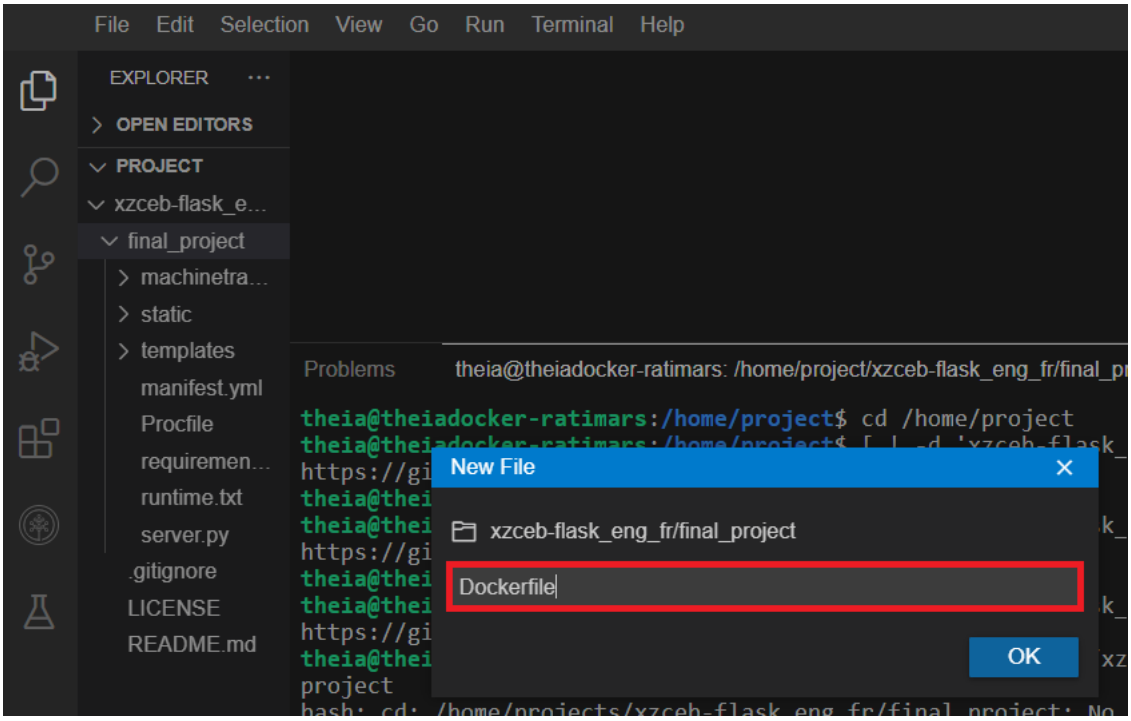
Copied!

Note:

- Make sure `Task 7: Run the servers` runs successfully.
- In `requirements.txt` file, delete the package versions and it should look as shown below:



2. Let's create a Dockerfile in your project directory. Dockerfile is the blueprint for building a container image for our app.



3. Create `Dockerfile` and add the following lines to your file:

```
1
2
3
4
5
6
7
FROM python:alpine3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
EXPOSE 8080
ENTRYPOINT [ "python" ]
CMD [ "server.py" ]
```

Copied!

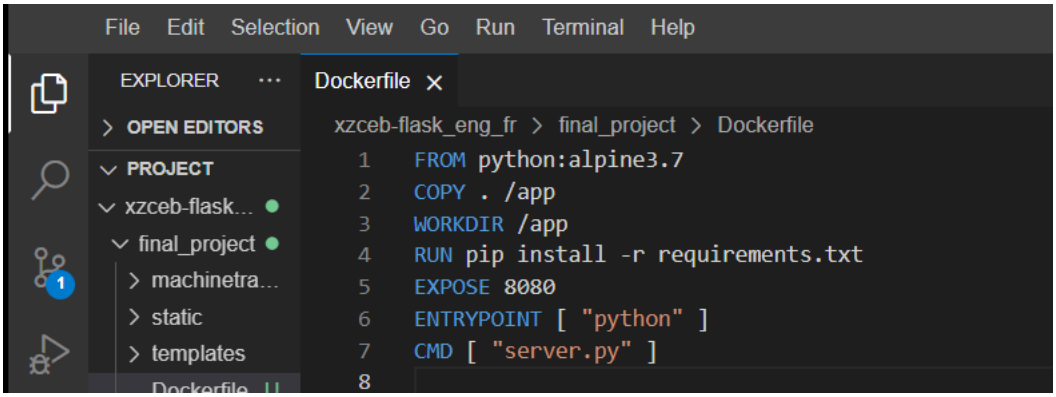
On the first line we are importing the Docker image `python:alpine3.7` which comes with support for Python 3. This image allows us to create Flask web applications in Python

that run in a single container. We are interested in the latest version of this image available, which supports Python 3.

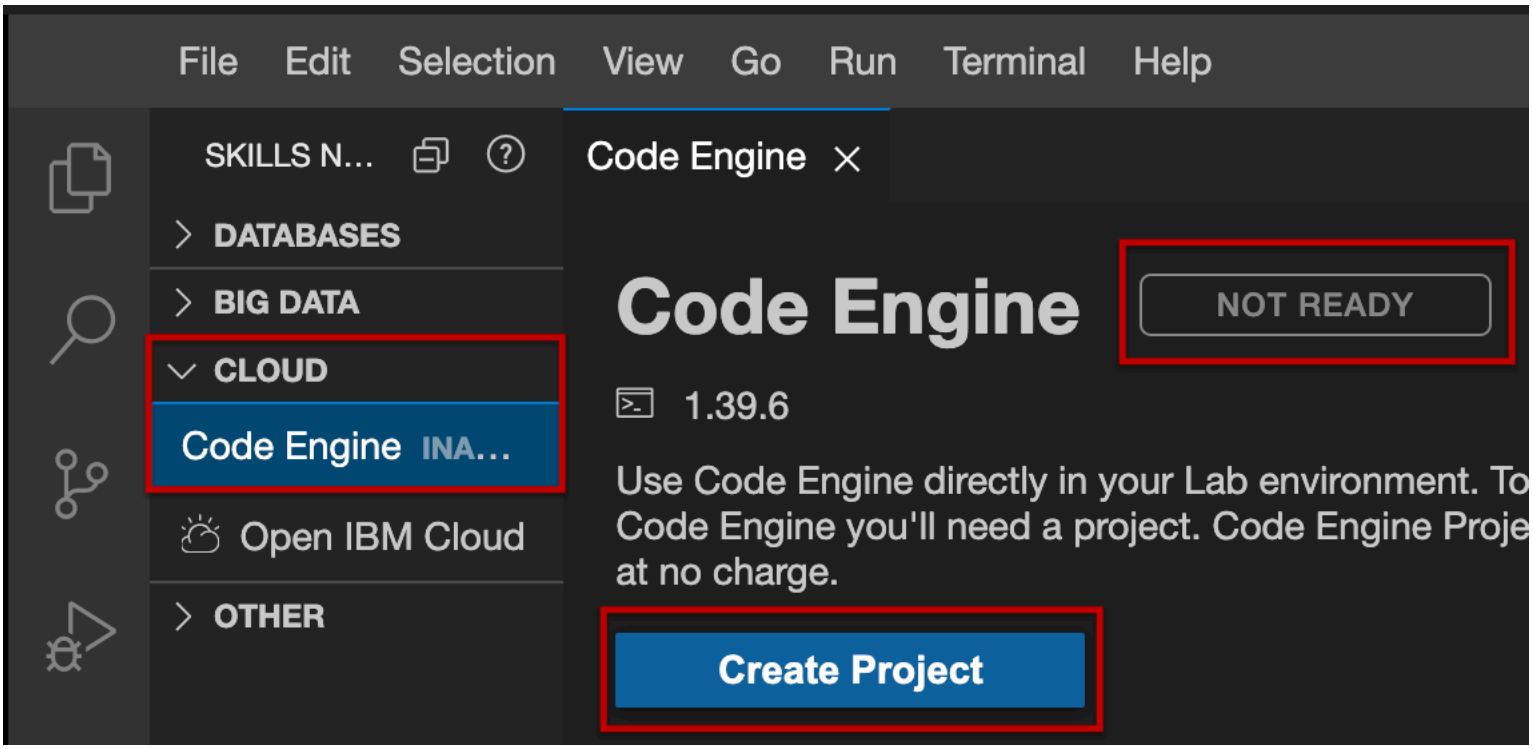
On the next 2 lines, we are opening port 8080 to usage in the docker container. This will allow us to access our application later once it's deployed to the cloud.

Finally, we copy the contents of the final_project directory we just created, into an app directory in the container image. Pretty easy, right!

You should have a directory structure like this:

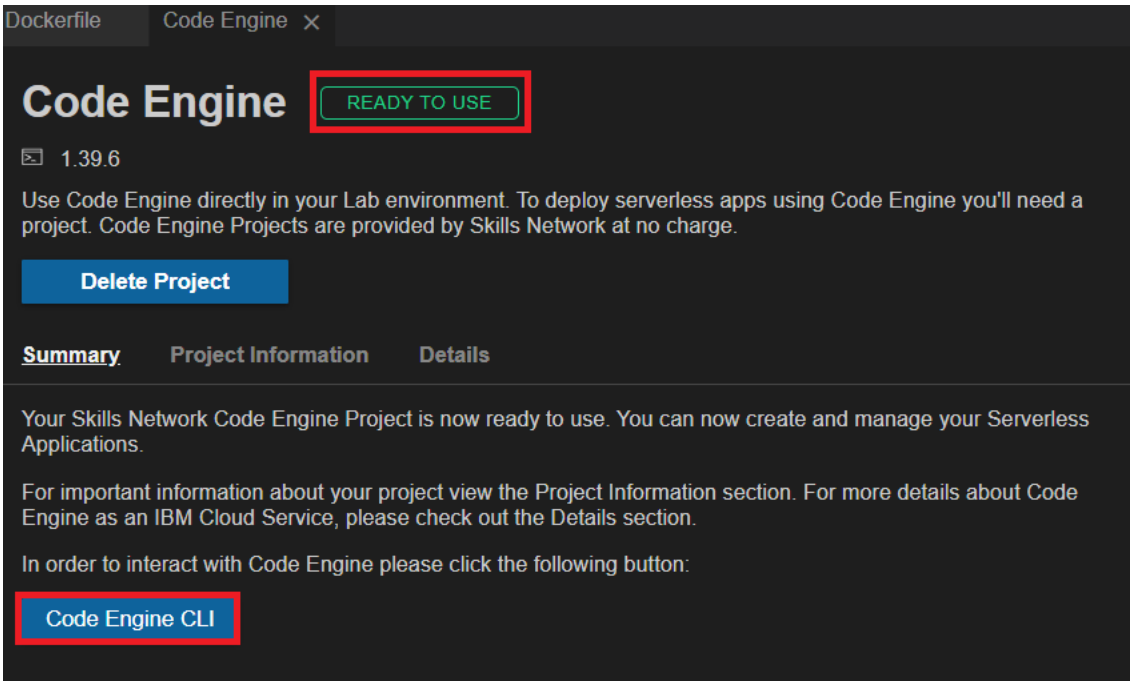


4. On the menu in your lab environment, click `Skills network tools`; Click Cloud dropdown and choose `Code Engine`. The code engine set up panel comes up. Click `Create Project`.



5. The code engine environment takes a while to prepare. You will see the progress status being indicated in the set up panel.

6. Once the code engine set up is complete, you can see that it is active. Click on `Code Engine CLI` to begin the pre-configured CLI in the terminal below.



You will now use the CLI to deploy the application.

7. Change to the app directory where the Dockerfile was created.

```
1. 1
1. cd /home/project/xzceb-flask_eng_fr/final_project
```
Copied!

8. Now run `docker build` in the app directory and tag the image. Note that in the below command we are naming the app `flask-docker-demo-translator`.

```
1. 1
1. docker build . -t us.icr.io/${SN_ICR_NAMESPACE}/flask-docker-demo-translator
```
Copied!

```
theia@theiadocker-ratimars:/home/project$ cd /home/project/xzceb-flask_eng_fr/final_project
theia@theiadocker-ratimars:/home/project/xzceb-flask_eng_fr/final_project$ docker build . -t us.icr.io/${SN_IC
or
Sending build context to Docker daemon  19.46kB
Step 1/7 : FROM python:alpine3.7
alpine3.7: Pulling from library/python
48ecbb6b270e: Pull complete
692f29ee68fa: Pull complete
6439819450d1: Pull complete
3c7be240f7bf: Pull complete
ca4b349df8ed: Pull complete
Digest: sha256:35f6f83ab08f98c727dbefd53738e3b3174a48b4571ccb1910bae480dcdba847
Status: Downloaded newer image for python:alpine3.7
 ---> 00be2573e9f7
Step 2/7 : COPY . /app
 ---> c6794867024c
Step 3/7 : WORKDIR /app
 ---> Running in 6e2da38db0bd
Removing intermediate container 6e2da38db0bd
 ---> 69f46d904477
Step 4/7 : RUN pip install -r requirements.txt
 ---> Running in 67e6f71b111c
Collecting Flask==1.1.2 (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/f2/28/2a03252dfb9ebf377f40fba6a7841b47083260bf8bd8e737b0
.whl (94kB)
Collecting Flask-WTF==0.14.3 (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/36/a9/8c01171066bd7a524ee005d81bb4a8aa446ab178043a1ad6cb
e-any.whl
```

9. Now push the image to the namespace so that you can run it.

```
1
docker push us.icr.io/${SN_ICR_NAMESPACE}/flask-docker-demo-translator:latest
```
Copied!

```
theia@theiadocker-ratimars:/home/project/xzceb-flask_eng_fr/final_project$ docker push us.icr.io/${SN_ICR_NAMES
est
The push refers to repository [us.icr.io/sn-labs-ratimars/flask-docker-demo-translator]
83e41a4661b9: Pushed
a88449560051: Pushed
5fa31f02caa8: Pushed
88e61e328a3c: Pushed
9b77965e1d3f: Pushed
50f8b07e9421: Pushed
629164d914fc: Pushed
latest: digest: sha256:e365013c125645dba353f7d995f76581499641809d864e2faa8440605879ea20 size: 1787
theia@theiadocker-ratimars:/home/project/xzceb-flask_eng_fr/final_project$
```

10. Deploy the application.

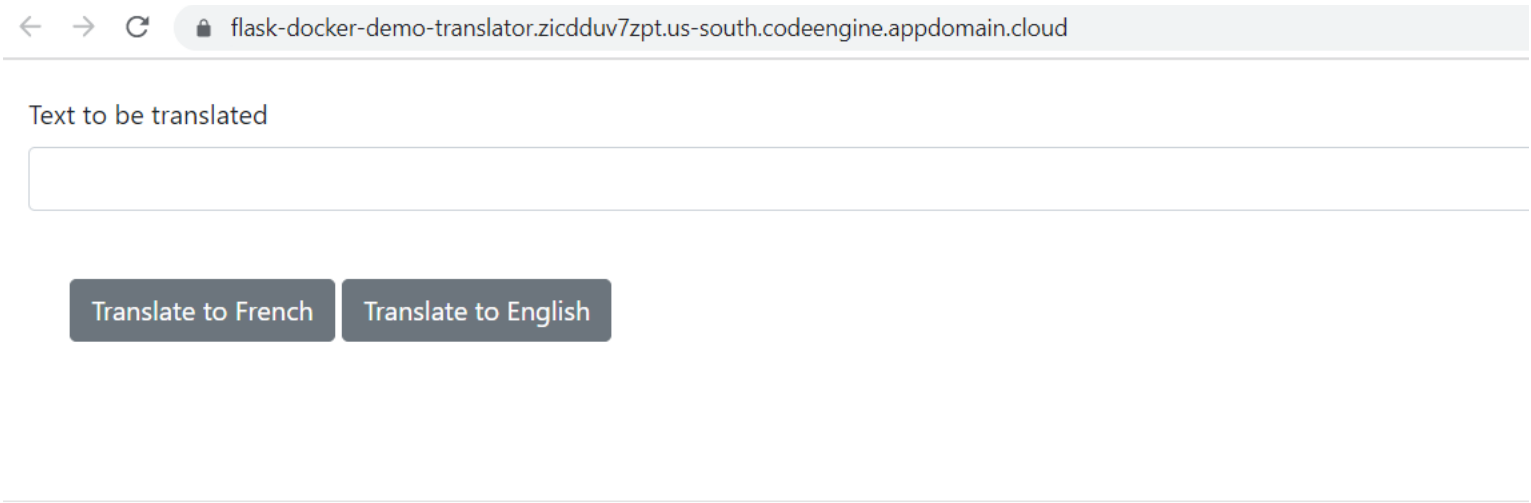
```
1
ibmcloud ce application create --name flask-docker-demo-translator --image us.icr.io/${SN_ICR_NAMESPACE}/flask-docker-demo-translator --r
```
Copied!

```
theia@theiadocker-rattimaibm:/home/project/xzceb-flask_eng_fr/final_project$ docker push us.icr.io/${SN_ICR_NAMESP
er-demo-translator:latest
The push refers to repository [us.icr.io/sn-labs-rattimaibm/flask-docker-demo-translator]
5911d154b95d: Pushed
0b52855c8fa6: Pushed
5fa31f02caa8: Pushed
88e61e328a3c: Pushed
9b77965e1d3f: Pushed
50f8b07e9421: Pushed
629164d914fc: Pushed
latest: digest: sha256:6b48b1409b19bdee7050ad1b1d91be304a59e3595a70e68fa96ec7e9d0ca12da size: 1788
theia@theiadocker-rattimaibm:/home/project/xzceb-flask_eng_fr/final_project$ ibmcloud ce application create --name
emo-translator --image us.icr.io/${SN_ICR_NAMESPACE}/flask-docker-demo-translator --registry-secret icr-secret
Creating application 'flask-docker-demo-translator'...
Configuration 'flask-docker-demo-translator' is waiting for a Revision to become ready.
Ingress has not yet been reconciled.
Waiting for load balancer to be ready.
Run 'ibmcloud ce application get -n flask-docker-demo-translator' to check the application status.
OK

https://flask-docker-demo-translator.103h6bg9akyc.us-south.codeengine.appdomain.cloud
theia@theiadocker-rattimaibm:/home/project/xzceb-flask_eng_fr/final_project$
```

Please note this command will run only in a Code Engine CLI. If you didn't follow the steps 4 to 7 to start the Code Engine CLI, you may get errors.

11. Press ctrl(Windows)/cmd(Mac) and the link that is created. Alternatively copy the link and paste it in a browser page and press enter. The `flask-docker-demo-translator` application page renders as given below.



Congratulations!

You have completed the tasks for this project. In the peer assignement that follows, you will be required to upload the screenshots you saved in this lab.

Authors

Lavanya

Other Contributors

Ramesh Sannareddy

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-05-14	1.0	Lavanya	Created initial version of the lab
2022-09-01	2.0	Ratima	Updated screenshot and step of Lauch Application
2022-09-23	2.1	Ratima	Updated instruction
2022-10-21	2.2	Ratima	Updated Skill Network Logo screenshot
2022-12-16	2.3	Ratima	Updated instruction
2023-03-21	2.4	Ratima	Updated Task 8