Peer Review Assignment

Skills Network

Estimated time needed: **60** minutes

Objectives

In this assignment you will:

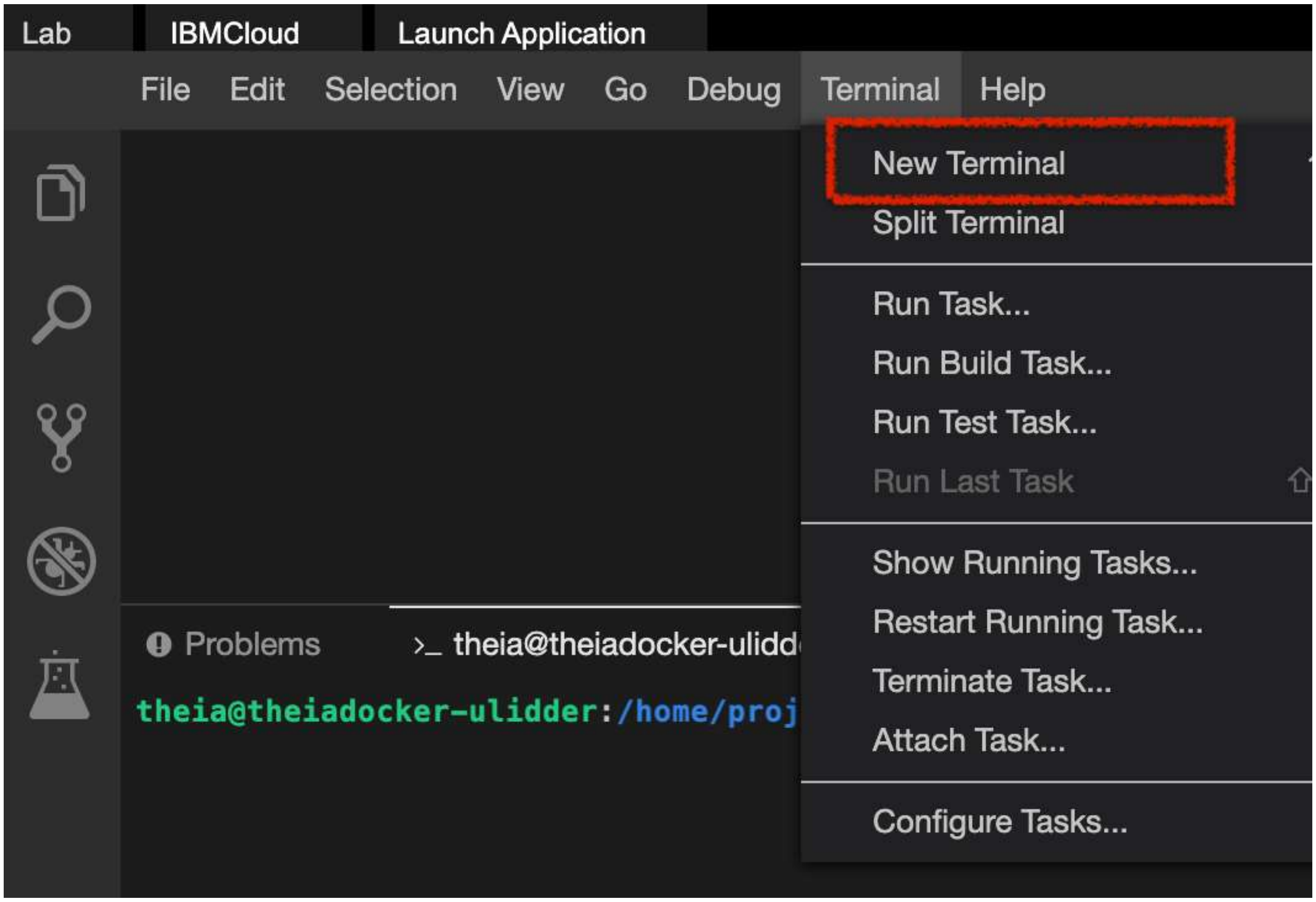
- Use the deep_translator python package for the translation.
- Create a function that translates English to French.
- Create a function that translates French to English.
- Run coding standards check against the functions above.
- Write unit tests to test the above functions.
- Run unit tests and interpret 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*

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.



```
theia@theiadocker-ulidder:/home/proj
```

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.

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

Copied!

4. Change to the `final_project` folder.

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

Copied!

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

```
mkdir machinetranslation
cd machinetranslation
```

Copied!

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

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

Copied!

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

```
python3 --version
```

Copied!

Note: It should be `Python 3.8.0`.

8. Install the packages that you will be using in this code, namely `deep_translator` and `Flask`.

```
python3 -m pip install deep_translator
python3 -m pip install Flask
```

Copied!

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

```
 from deep_translator import MyMemoryTranslator
```

Copied!

�Take a screenshot of your import statement and save it as a .jpg or .png with the filename `import_translator`. You will be prompted to upload the screenshot in the Peer Assignment that follows.

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

```
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 Assignment that follows.

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

```
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 Assignment 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 the translation of the word 'Hello' and 'Bonjour'.
4. Test for the translation of the word 'Bonjour' and 'Hello'.
5. 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.

```
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

```
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.

6. Change to the project directory

```
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.

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

```
git add .
```
Copied!

```
```

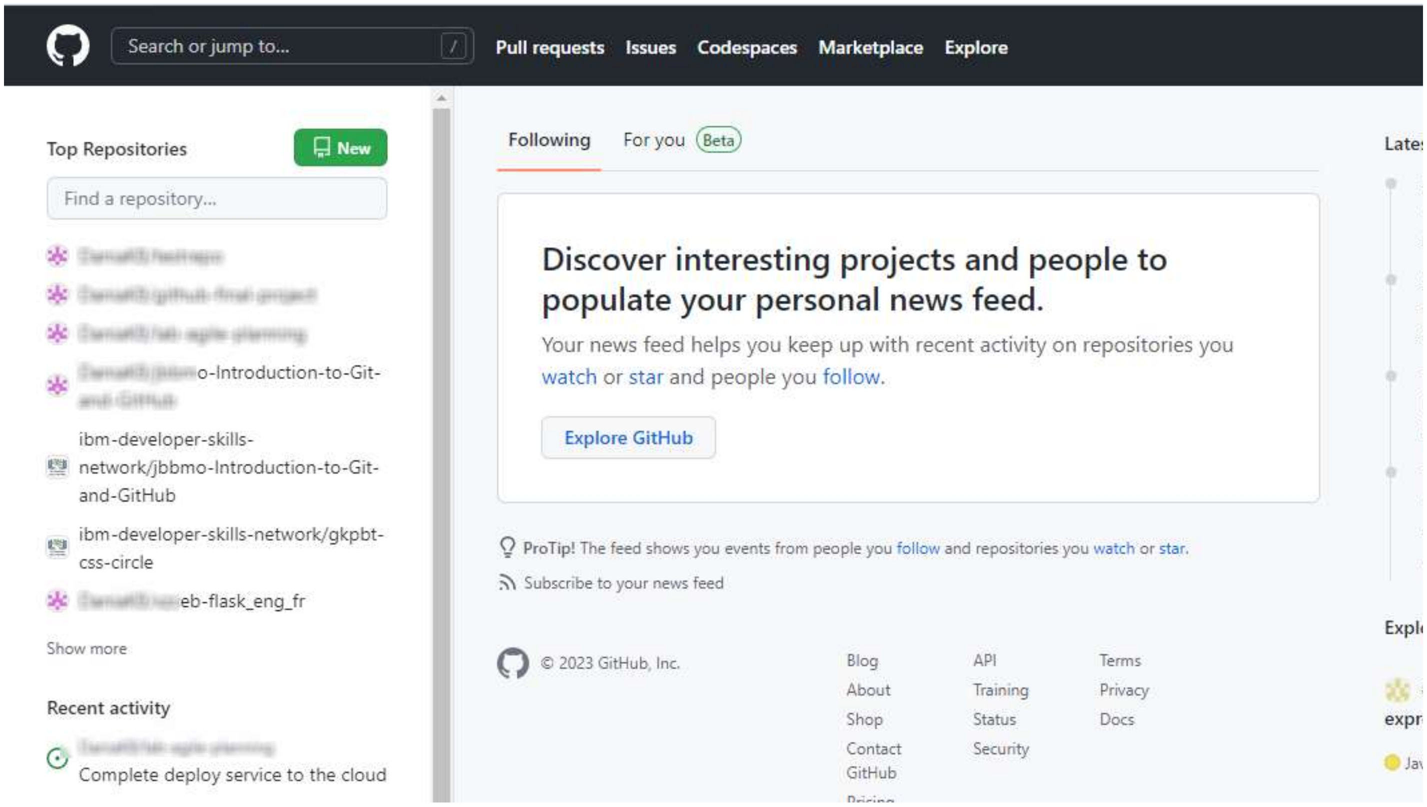
```
1. git commit -m"Final changes"
```

Copied!

Task - Generate Personal Access Token

-
  ```
  1. 1
  1. Verify your email address if it hasn鳴been verified on Github.
  ```
  Copied!

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



Search or jump to...

Pull requests Issues Codespaces Marketplace Explore

Top Repositories New

Find a repository...

[redacted]

[redacted]

[redacted]

[redacted]o-Introduction-to-Git-[redacted]

ibm-developer-skills-network/jbbmo-Introduction-to-Git-and-GitHub

ibm-developer-skills-network/gkpbt-css-circle

[redacted]eb-flask_eng_fr

Show more

Recent activity

[redacted]

Complete deploy service to the cloud

Following For you Beta

Discover interesting projects and people to populate your personal news feed.

Your news feed helps you keep up with recent activity on repositories you watch or star and people you follow.

Explore GitHub

ProTip! The feed shows you events from people you follow and repositories you watch or star.

Subscribe to your news feed

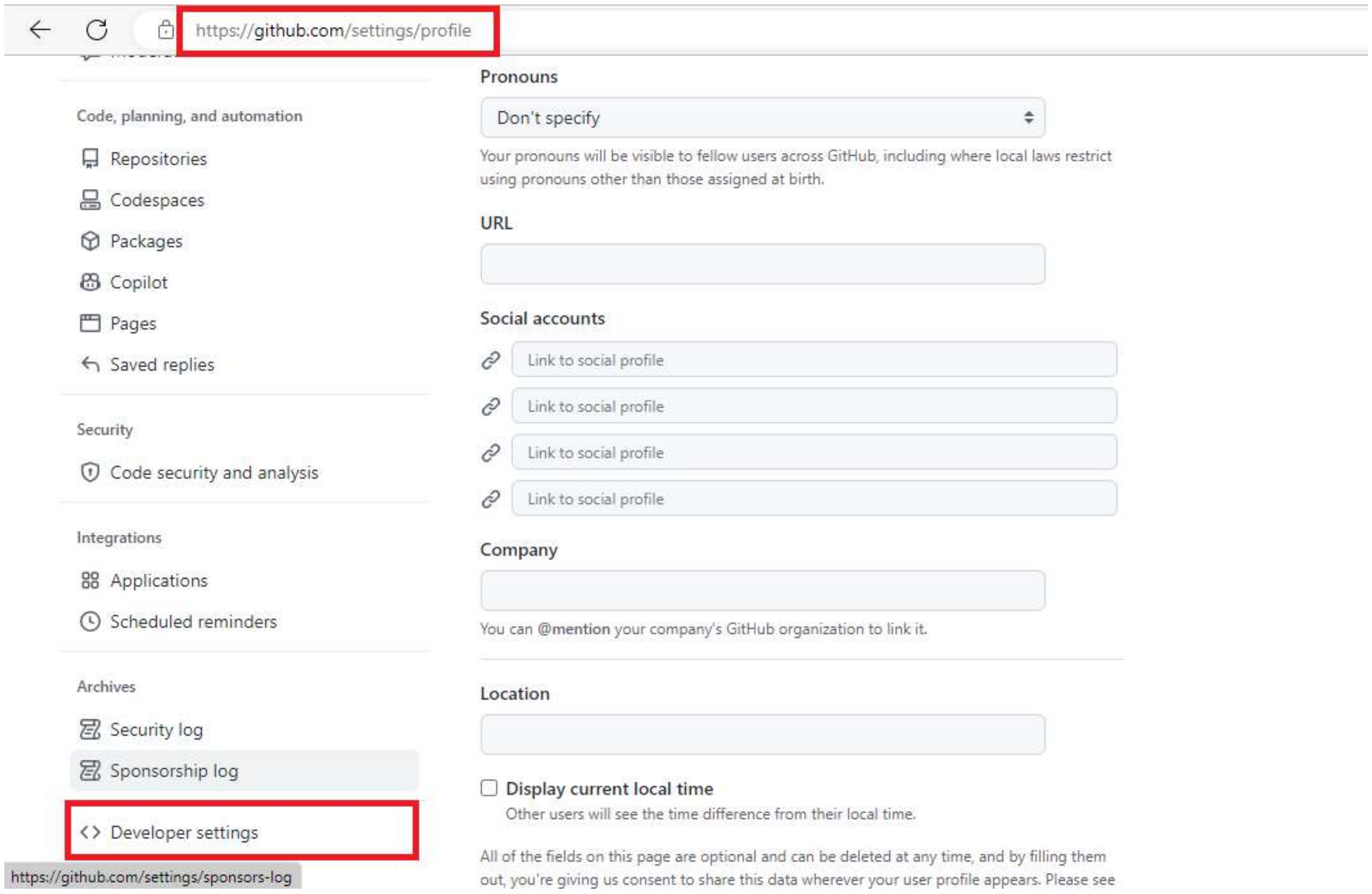
© 2023 GitHub, Inc.

Blog About Shop Contact GitHub Pricing

API Training Status Security

Terms Privacy Docs

-
  ```
  1. 1
  1. In the left sidebar, click Developer settings.
  ```
  Copied!



https://github.com/settings/profile

Pronouns

Don't specify

Your pronouns will be visible to fellow users across GitHub, including where local laws restrict using pronouns other than those assigned at birth.

URL

Social accounts

Link to social profile

Link to social profile

Link to social profile

Link to social profile

Company

You can @mention your company's GitHub organization to link it.

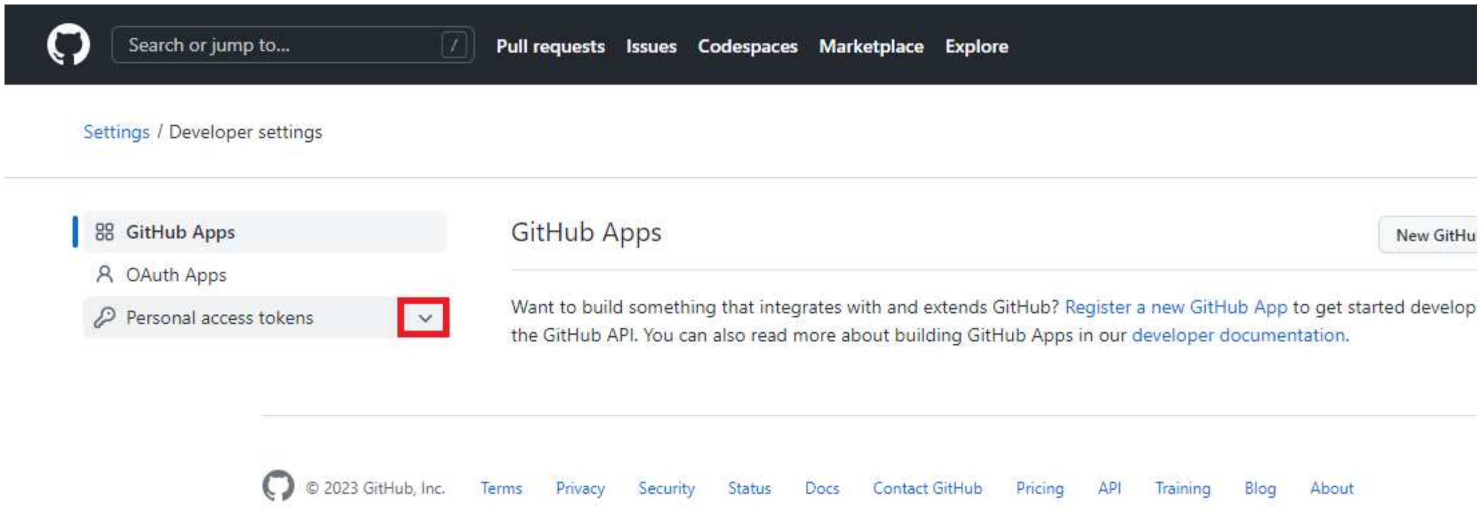
Location

Display current local time

Other users will see the time difference from their local time.

All of the fields on this page are optional and can be deleted at any time, and by filling them out, you're giving us consent to share this data wherever your user profile appears. Please see

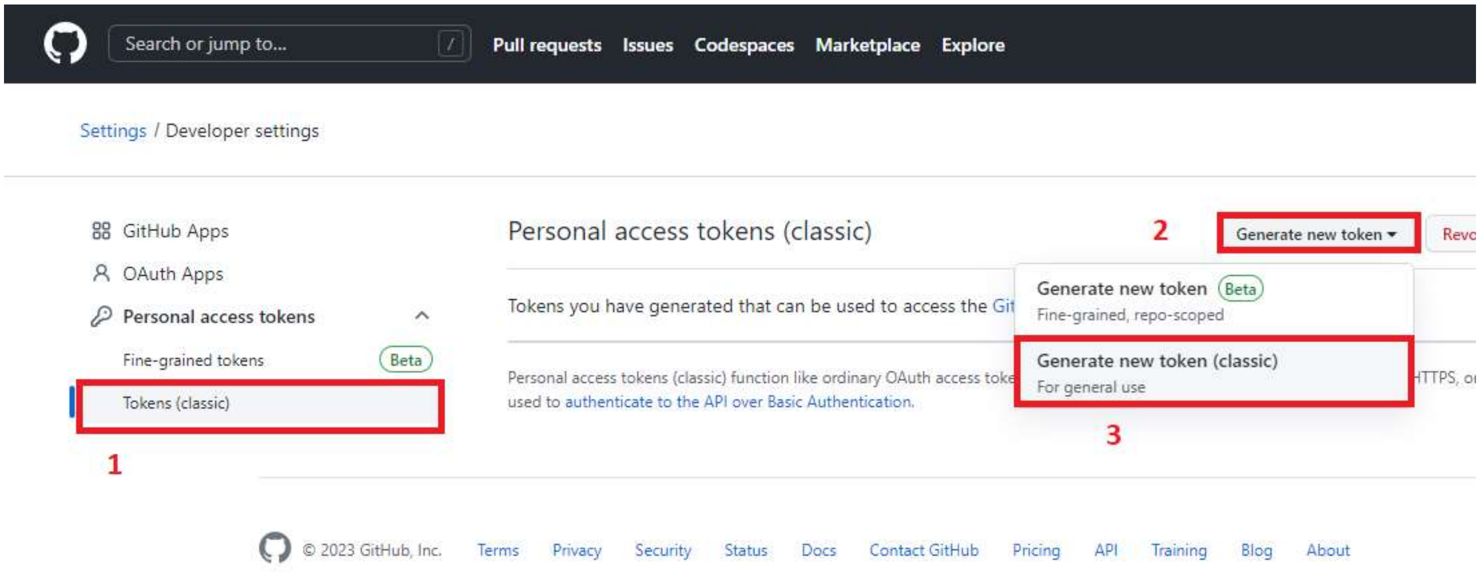
-
  ```
  1. 1
  1. In the left sidebar, click drop-down menu `Personal access tokens`.
  ```
  Copied!



GitHub Apps

Want to build something that integrates with and extends GitHub? Register a new GitHub App to get started develop the GitHub API. You can also read more about building GitHub Apps in our developer documentation.

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


Settings / Developer settings

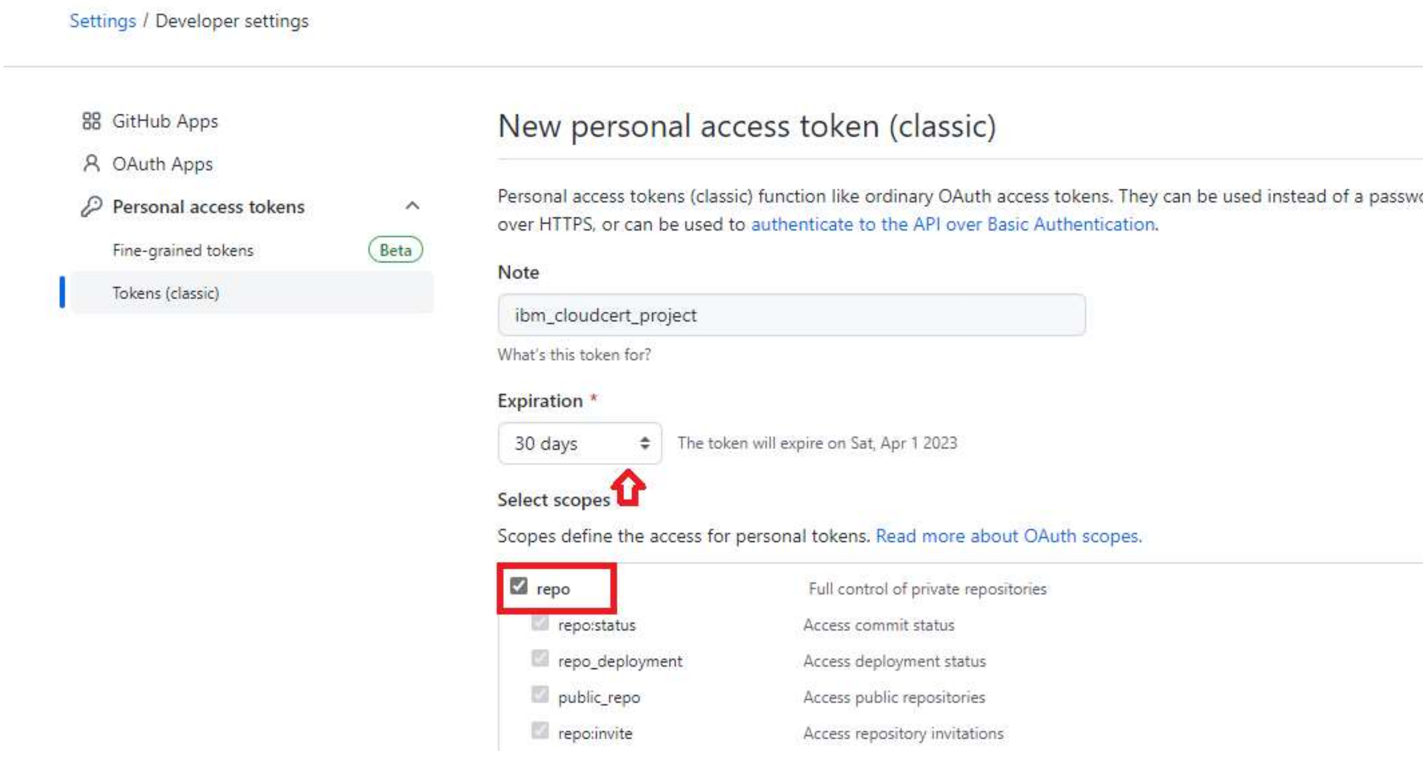
Personal access tokens (classic)

Tokens you have generated that can be used to access the Git

Personal access tokens (classic) function like ordinary OAuth access toke ... HTTPS, or used to authenticate to the API over Basic Authentication.

- 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 the cal

Copied!


Settings / Developer settings

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a passw over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note: ibm_cloudcert_project

What's this token for?

Expiration *: 30 days — The token will expire on Sat, Apr 1 2023

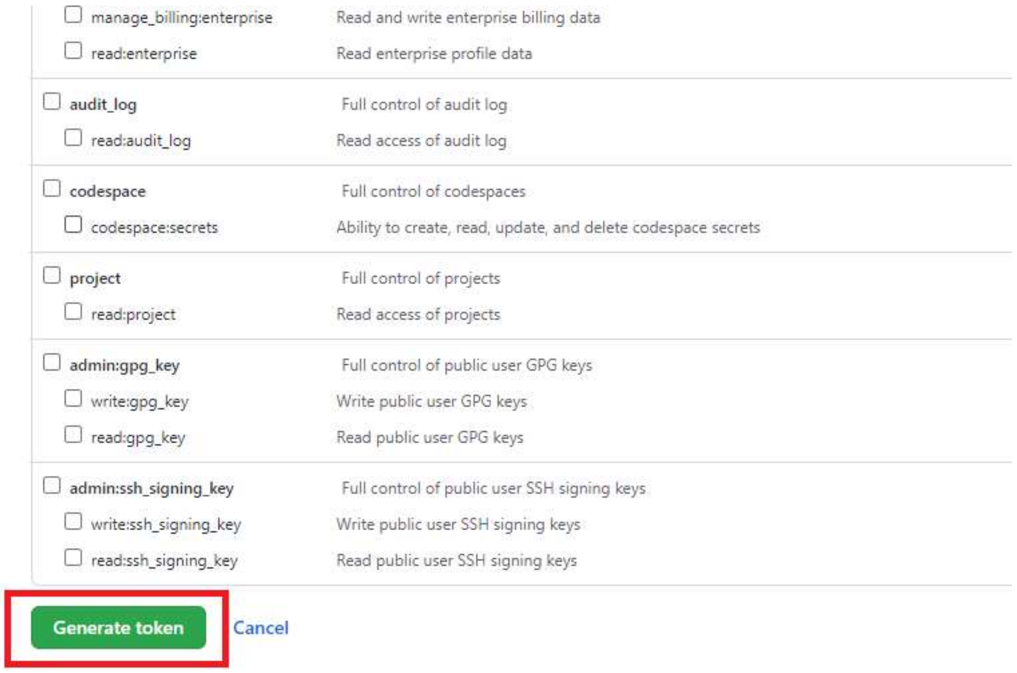
Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

✓ repo	Full control of private repositories
✓ repo:status	Access commit status
✓ repo_deployment	Access deployment status
✓ public_repo	Access public repositories
✓ repo:invite	Access repository invitations

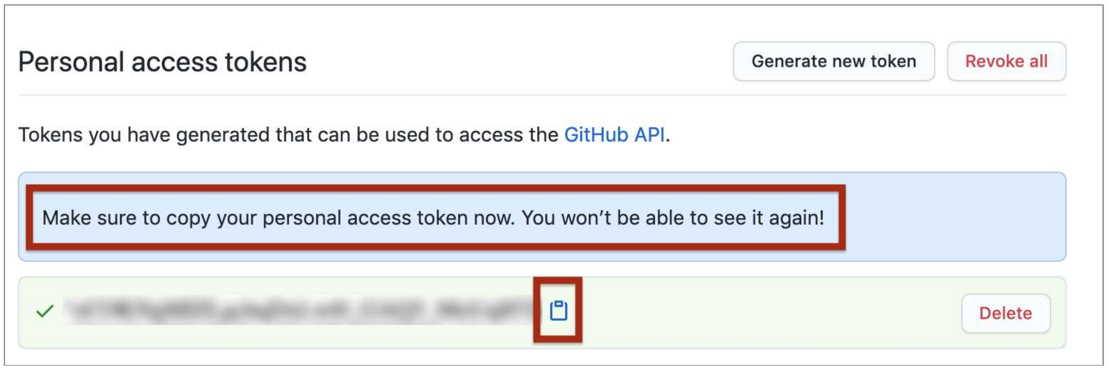
- 1. 1
  1. Click `Generate token` and make a note of it.

Copied!



☐ manage_billing:enterprise	Read and write enterprise billing data
☐ read:enterprise	Read enterprise profile data
☐ audit_log	Full control of audit log
☐ read:audit_log	Read access of audit log
☐ codespace	Full control of codespaces
☐ codespace:secrets	Ability to create, read, update, and delete codespace secrets
☐ project	Full control of projects
☐ read:project	Read access of projects
☐ admin:gpg_key	Full control of public user GPG keys
☐ write:gpg_key	Write public user GPG keys
☐ read:gpg_key	Read public user GPG keys
☐ admin:ssh_signing_key	Full control of public user SSH signing keys
☐ write:ssh_signing_key	Write public user SSH signing keys
☐ read:ssh_signing_key	Read public user SSH signing keys

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



Personal access tokens

Tokens you have generated that can be used to access the GitHub API.

Make sure to copy your personal access token now. You won't be able to see it 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:

```
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

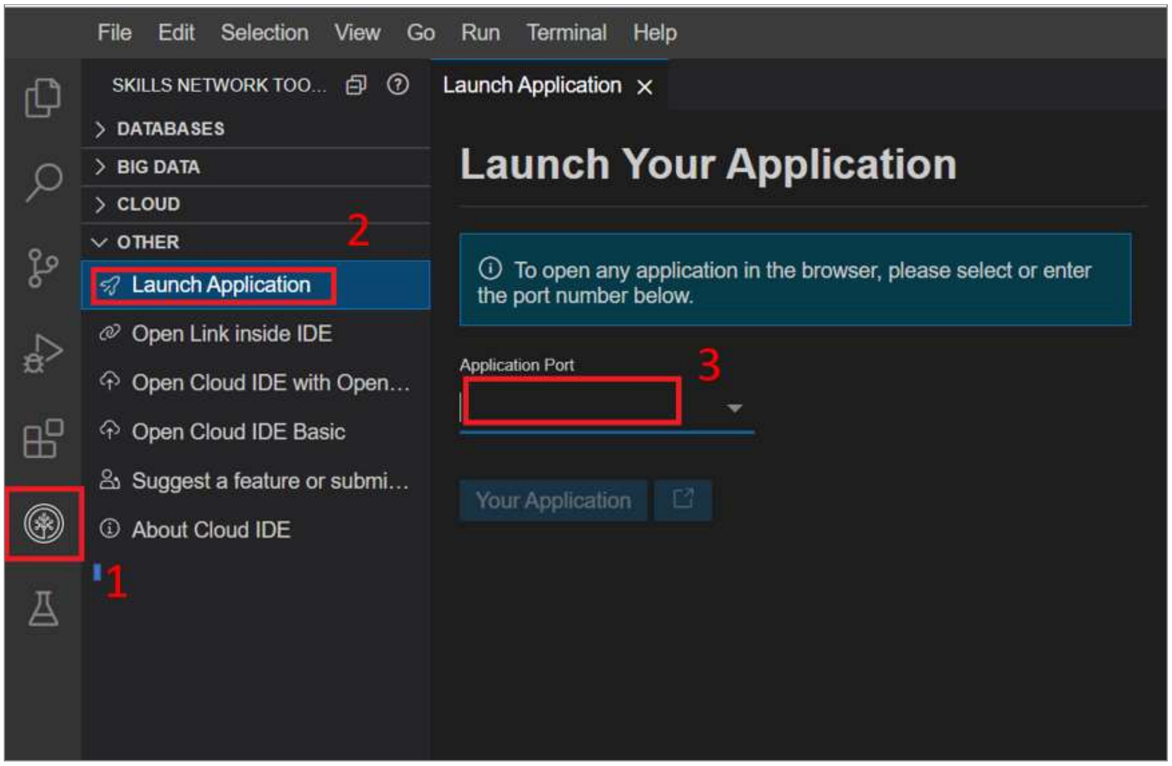
1. Change to the project directory and run the server from your terminal.

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

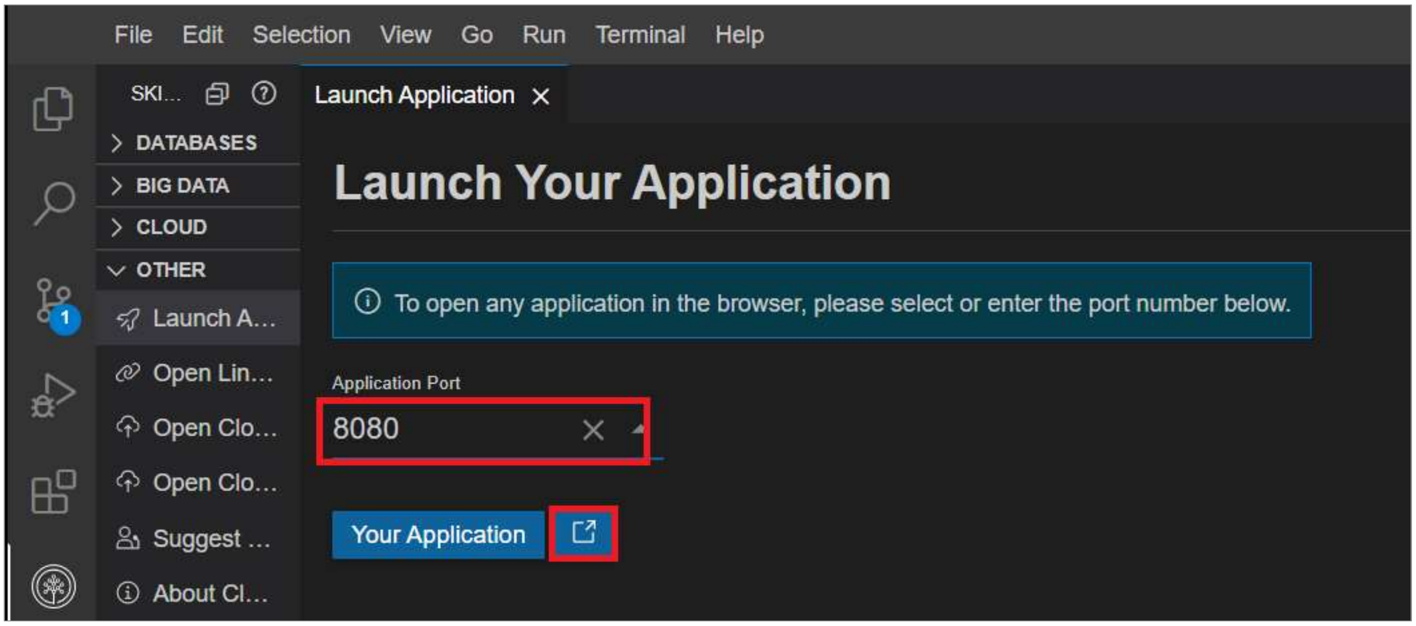
Copied!

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

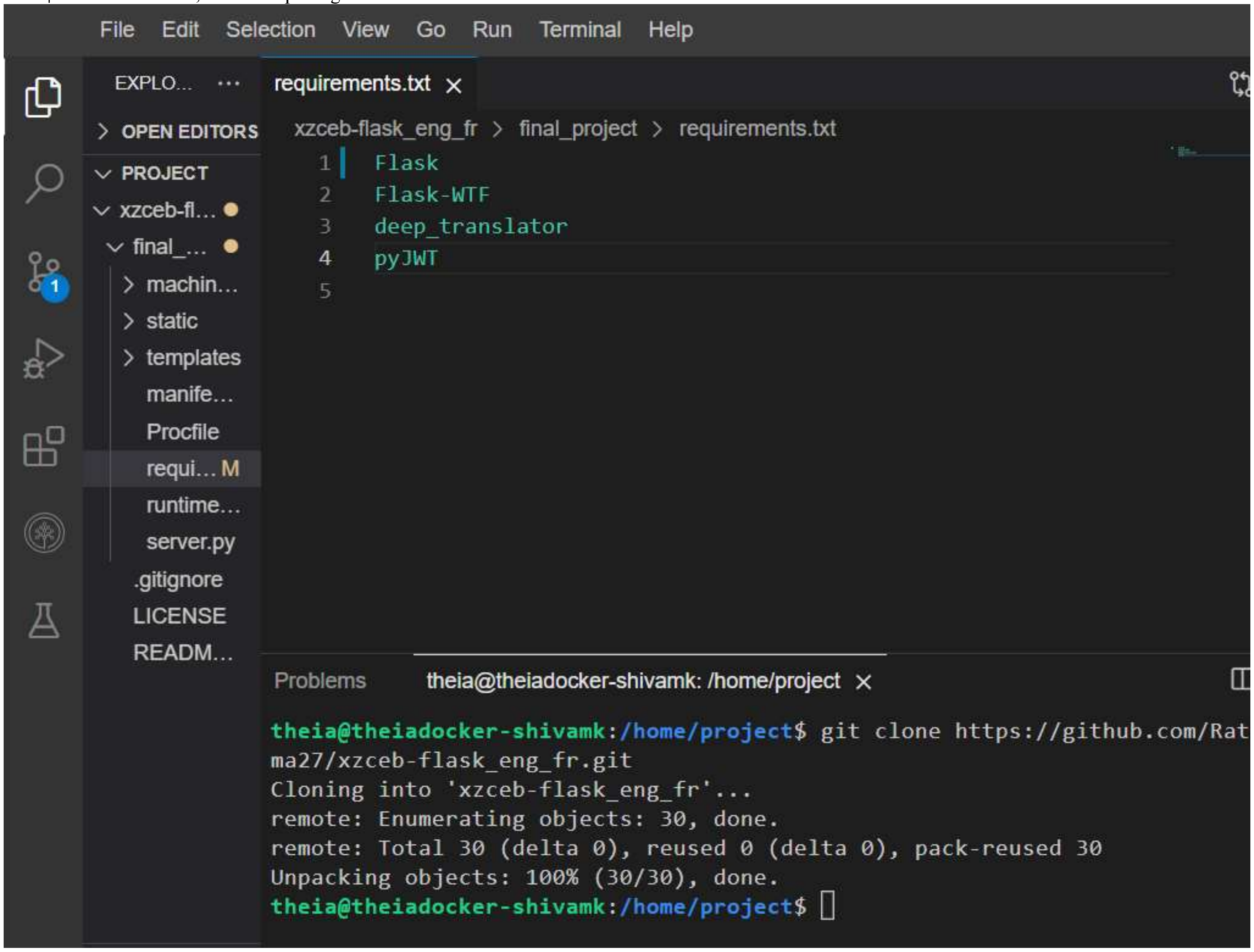
1. Change to the `final_project` directory.

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

Note:

- Make sure `Task 7: Run the servers` runs successfully.

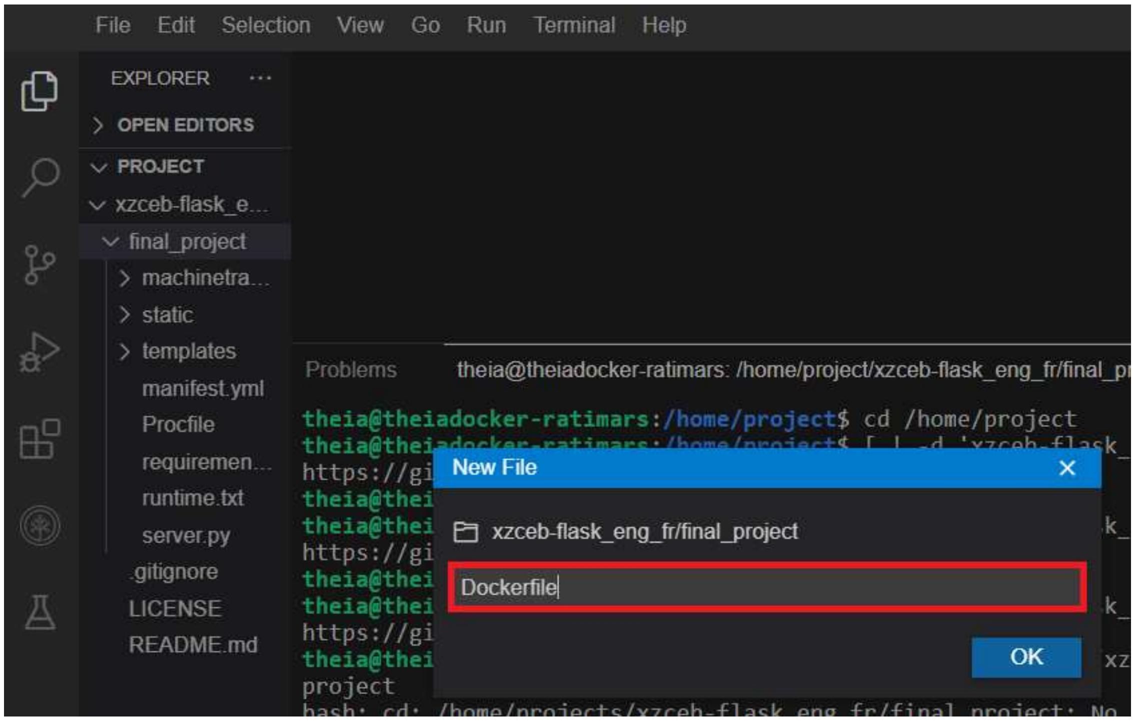
- In `requirements.txt` file, delete the package versions and it should look as shown below:



```
Flask
Flask-WTF
deep_translator
pyJWT
```

```
theia@theiadocker-shivamk:/home/project$ git clone https://github.com/Rat
ma27/xzceb-flask_eng_fr.git
Cloning into 'xzceb-flask_eng_fr'...
remote: Enumerating objects: 30, done.
remote: Total 30 (delta 0), reused 0 (delta 0), pack-reused 30
Unpacking objects: 100% (30/30), done.
theia@theiadocker-shivamk:/home/project$ 
```

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



```
theia@theiadocker-ratimars:/home/project$ cd /home/project
theia@theiadocker-ratimars:/home/project$ [ ! -d 'xzceb-flask_
https://gi
theia@thei
theia@thei                                                       k_
https://gi
theia@thei
theia@thei                                                       k_
https://gi
theia@thei                                                       xz
project
bash: cd: /home/projects/xzceb-flask_eng_fr/final_project: No
```

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

```
1. 1
2. 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" ]
```

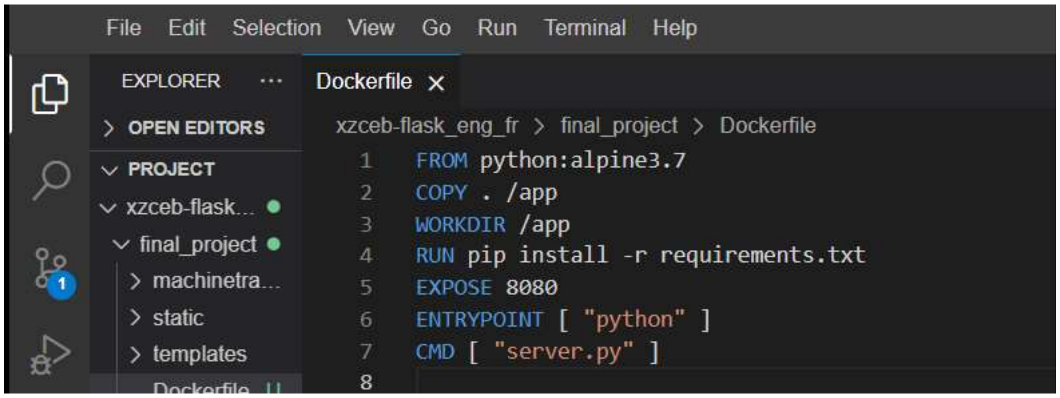
Copied!

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

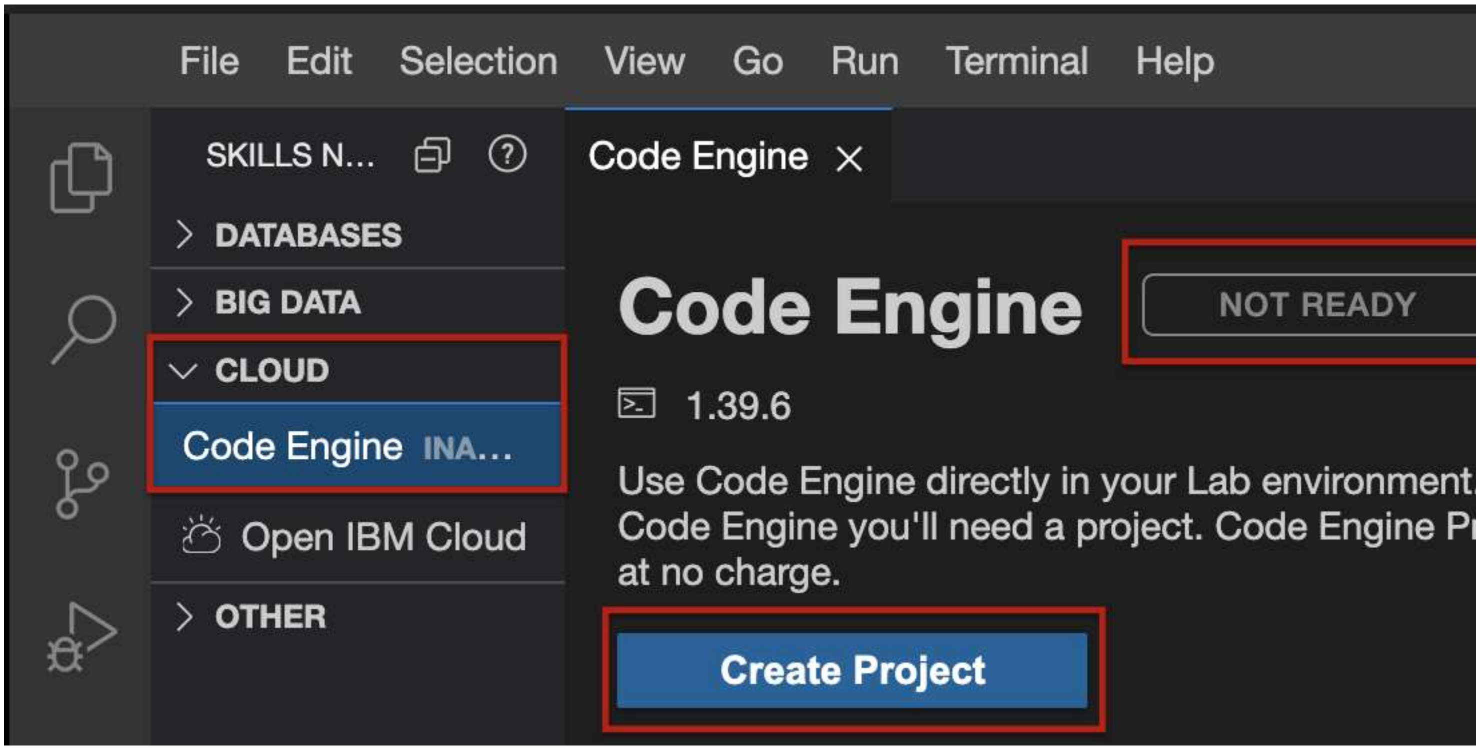
Finally, we are opening port 8080 to usage in the docker container. This will allow us to access our application later once it鑊deployed to the cloud.

You should have a directory structure like this:



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

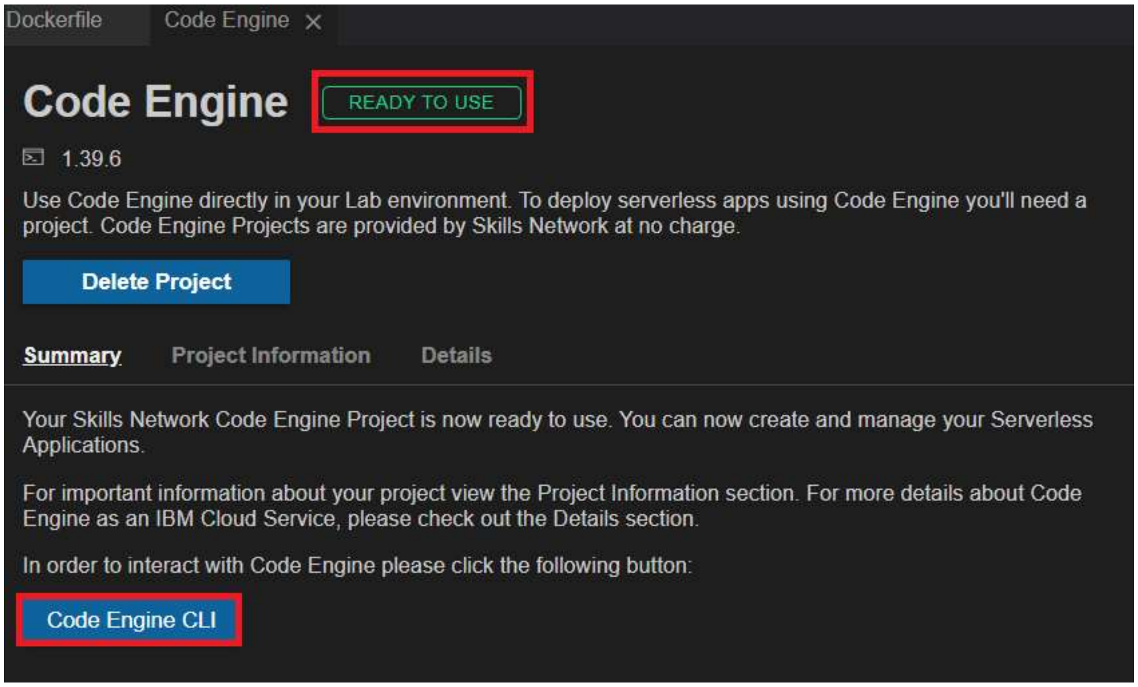
4. On the menu in your lab environment, click `Skills network tools`; Click Cloud dropdown and choose `Code Engine`. The code engine set up panel comes up. Click `Create Project`.



Use Code Engine directly in your Lab environment
Code Engine you'll need a project. Code Engine P
at no charge.

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.



Code Engine

Use Code Engine directly in your Lab environment. To deploy serverless apps using Code Engine you'll need a project. Code Engine Projects are provided by Skills Network at no charge.

Your Skills Network Code Engine Project is now ready to use. You can now create and manage your Serverless Applications.

For important information about your project view the Project Information section. For more details about Code Engine as an IBM Cloud Service, please check out the Details section.

In order to interact with Code Engine please click the following button:

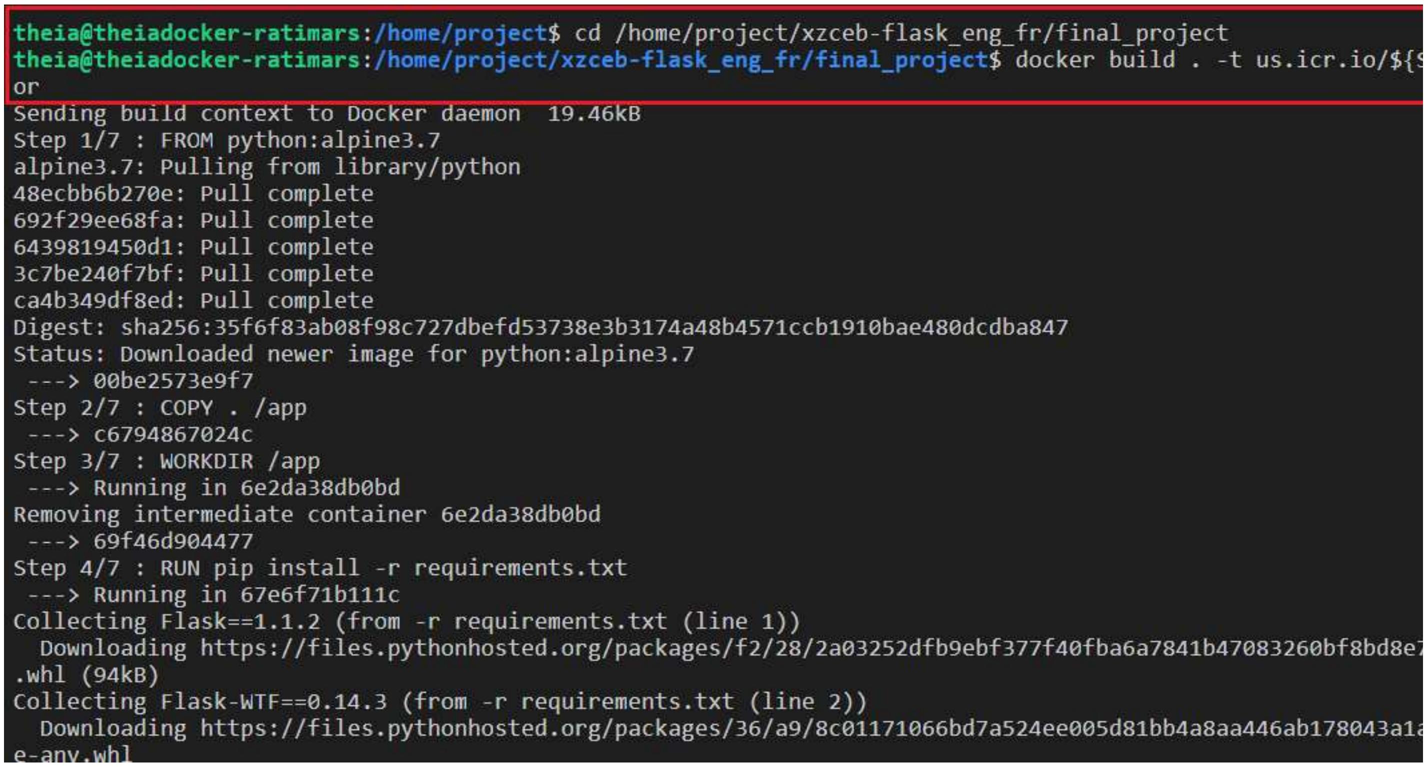
You will now use the CLI to deploy the application.

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`.

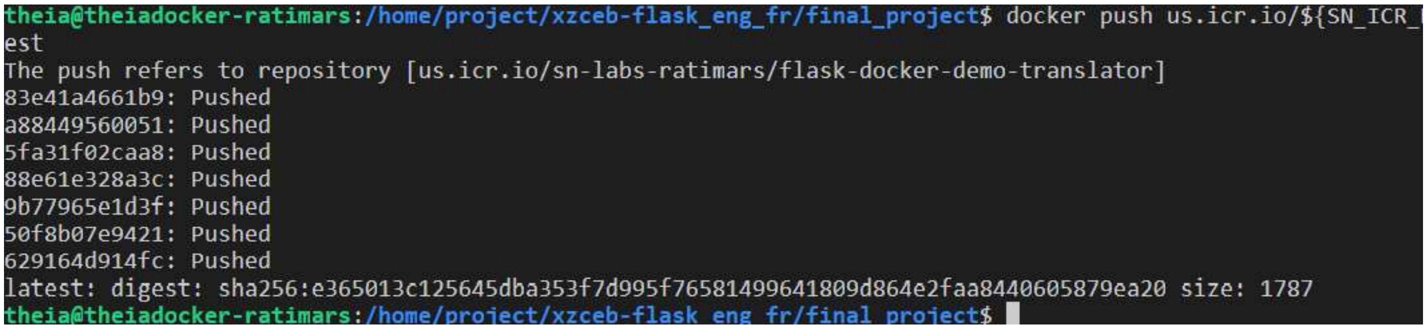
```
1. 1
1. docker build . -t us.icr.io/${SN_ICR_NAMESPACE}/flask-docker-demo-translator
```
Copied!



```
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/${S
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
 ---> c67948670 24c
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/f2/28/2a03252dfb9ebf377f40fba6a7841b47083260bf8bd8e7
.whl (94kB)
Collecting Flask-WTF==0.14.3 (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/36/a9/8c01171066bd7a524ee005d81bb4a8aa446ab178043a1a
e-any.whl
```

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

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

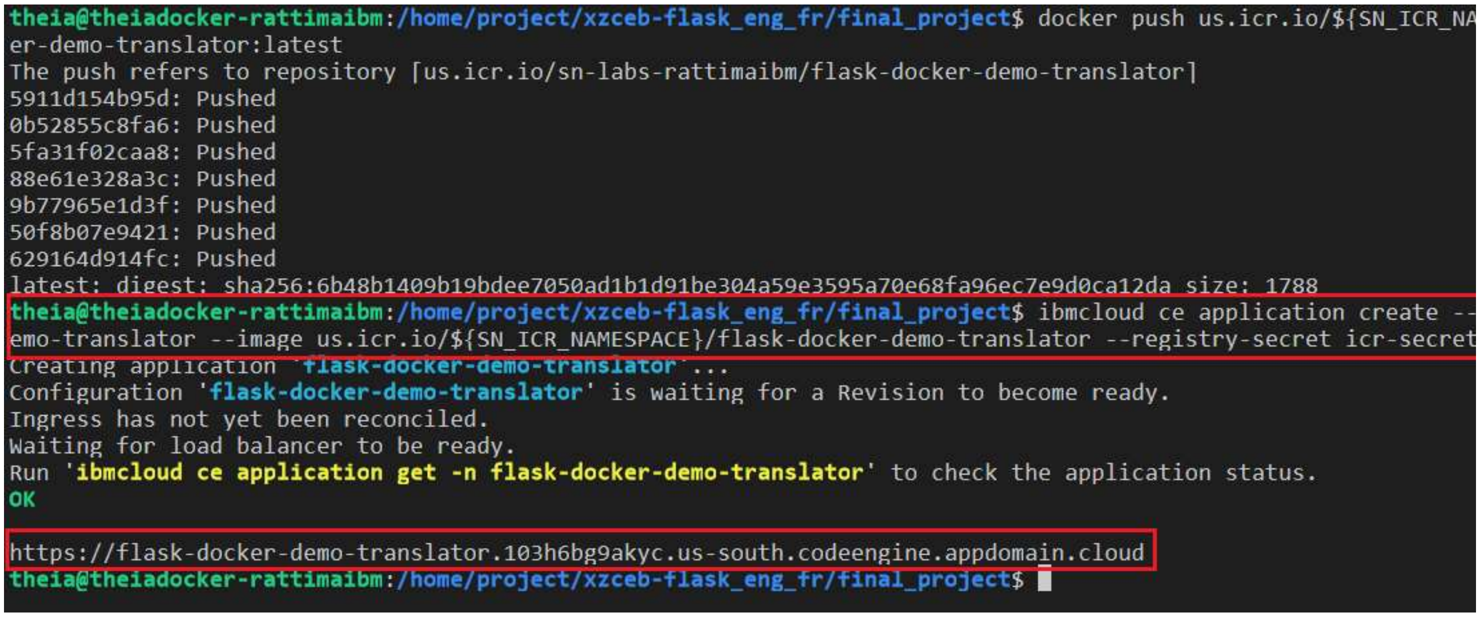


```
theia@theiadocker-ratimars:/home/project/xzceb-flask_eng_fr/final_project$ docker push us.icr.io/${SN_ICR_
est
The push refers to repository [us.icr.io/sn-labs-ratimars/flask-docker-demo-translator]
83e41a4661b9: Pushed
a88449560051: Pushed
5fa31f02caa8: Pushed
88e61e328a3c: Pushed
9b77965e1d3f: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

10. Deploy the application.

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

Copied!

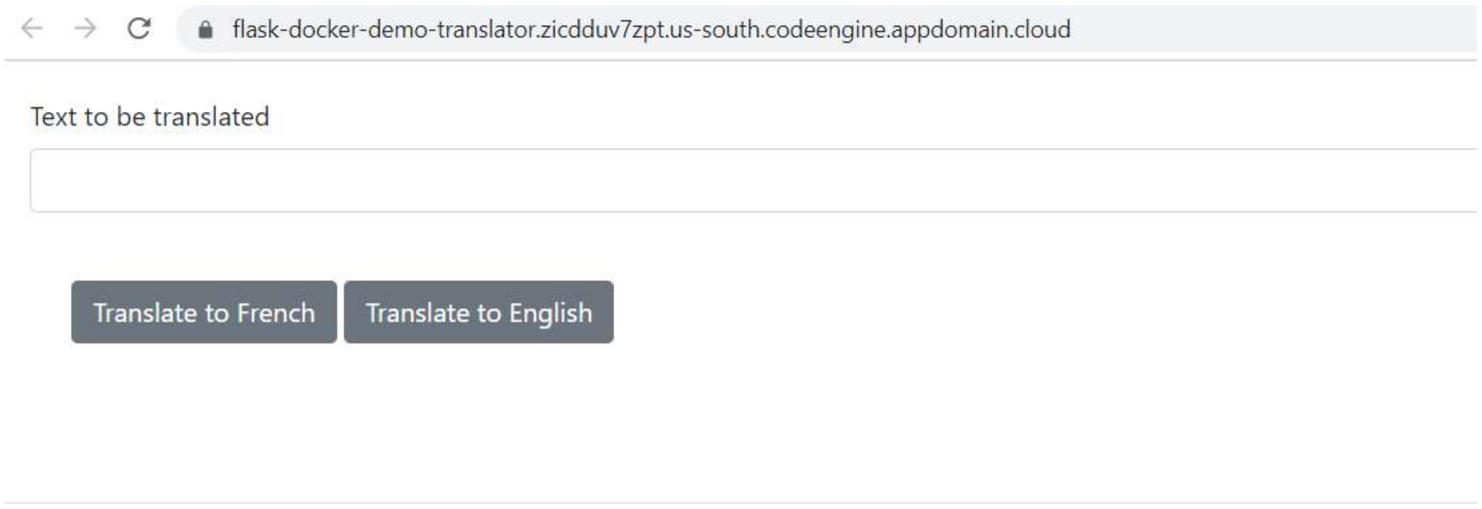


```
theia@theiadocker-rattimaibm:/home/project/xzceb-flask_eng_fr/final_project$ docker push us.icr.io/${SN_ICR_NA
er-demo-translator:latest
The push refers to repository [us.icr.io/sn-labs-rattimaibm/flask-docker-demo-translator]
5911d154b95d: Pushed
0b52855c8fa6: Pushed
5fa31f02caa8: Pushed
88e61e328a3c: Pushed
9b77965e1d3f: Pushed
50f8b07e9421: Pushed
629164d914fc: Pushed
latest: digest: sha256:6b48b1409b19bdee7050ad1b1d91be304a59e3595a70e68fa96ec7e9d0ca12da size: 1788
theia@theiadocker-rattimaibm:/home/project/xzceb-flask_eng_fr/final_project$ ibmcloud ce application create --
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鴠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.


flask-docker-demo-translator.zicdduv7zpt.us-south.codeengine.appdomain.cloud
Text to be translated
Translate to French
Translate to English

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.

Author(s)

Lavanya

Other Contributor(s)

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
2023-06-01	2.5	Shivam	Updated Translator