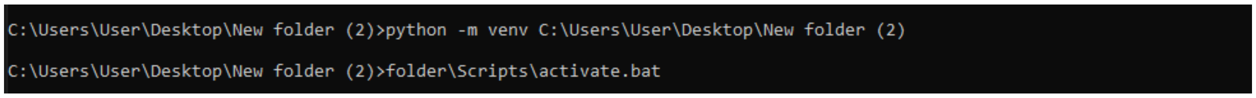
Testing Model on PC and Deploying Using Flask:

1. Create virtual environment and activate it

It's better to create a virtual environment and install the required libraries in it so that the new libraries don't conflict with the existing ones. Create your project's folder, open cmd and type the following commands for environment creation and activation

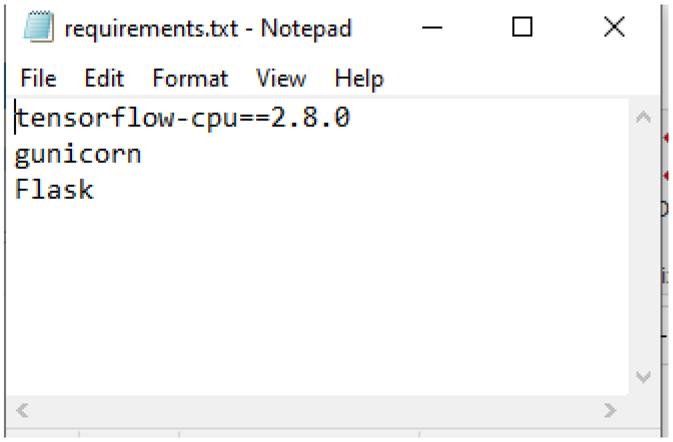


```
C:\Users\User\Desktop\New folder (2)>python -m venv C:\Users\User\Desktop\New folder (2)

C:\Users\User\Desktop\New folder (2)>folder\Scripts\activate.bat
```

Figure 1: Environment creation and activation

2. Set python packages

Once the environment is activated, we have to install the required libraries as a list of python packages and their versions in a separate file called requirements.txt for example.



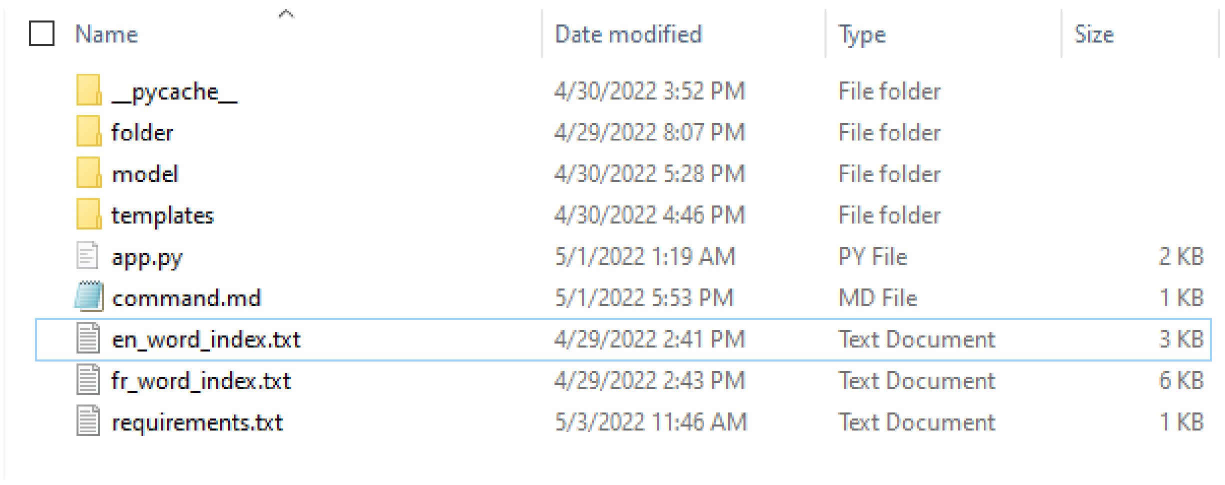
```
tensorflow-cpu==2.8.0
gunicorn
Flask
```

Figure 2: Prerequisite libraries list

Tensorflow library contains many other libraries like numpy, scipy etc.. so they will get installed along the way.
Then we can use the below command to install them.

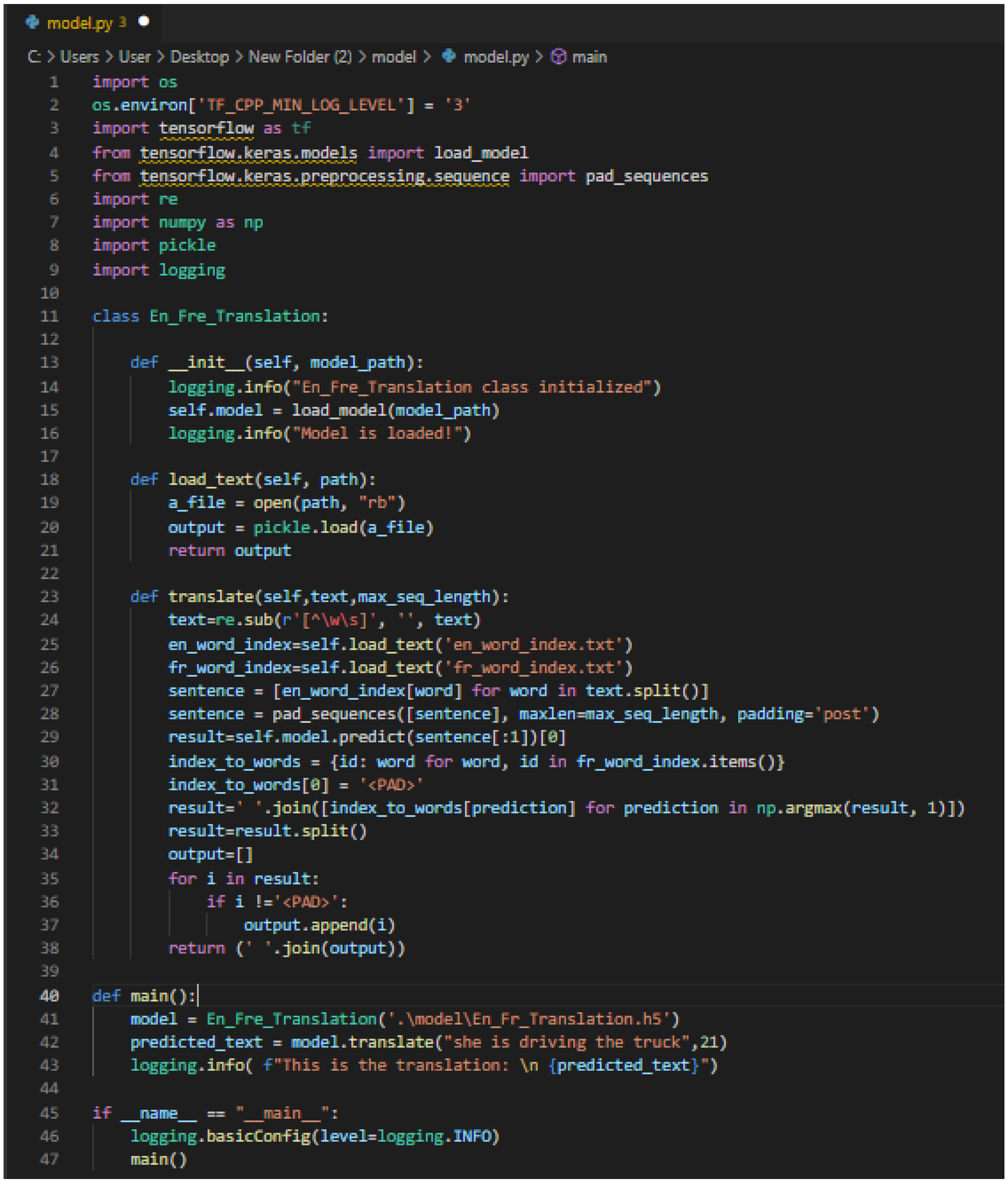
```
pip install -r requirements.txt
```

3. Project Structure



Name	Date modified	Type	Size
__pycache__	4/30/2022 3:52 PM	File folder	
folder	4/29/2022 8:07 PM	File folder	
model	4/30/2022 5:28 PM	File folder	
templates	4/30/2022 4:46 PM	File folder	
app.py	5/1/2022 1:19 AM	PY File	2 KB
command.md	5/1/2022 5:53 PM	MD File	1 KB
en_word_index.txt	4/29/2022 2:41 PM	Text Document	3 KB
fr_word_index.txt	4/29/2022 2:43 PM	Text Document	6 KB
requirements.txt	5/3/2022 11:46 AM	Text Document	1 KB

- “folder” folder is my virtual environment
- “model” folder contains “En_Fr_Translation.h5” and “model.py” . h5 file is our model trained, and “model.py” is used to import model and execute the prediction.
- “templates” folder contains an “index.html” file used for flask.
- “app.py” file is the web application using flask
- “en_word_index.txt” and “fr_word_index.txt” are our English and French vocabularies used in the “model.py” script
- “Command.md” includes the commands needed



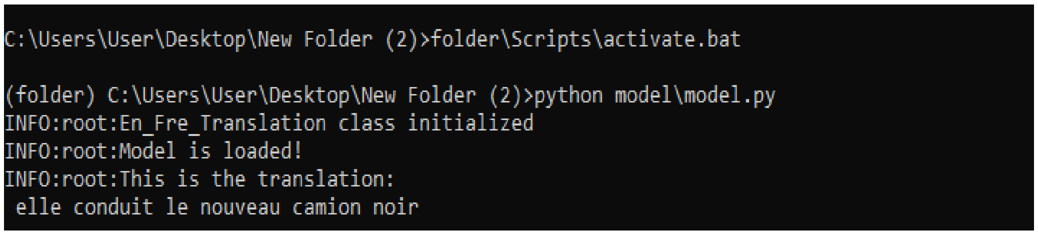
```
model.py 3 ●
C: > Users > User > Desktop > New Folder (2) > model > model.py > main
```

```python
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.sequence import pad_sequences
import re
import numpy as np
import pickle
import logging

class En_Fre_Translation:

    def __init__(self, model_path):
        logging.info("En_Fre_Translation class initialized")
        self.model = load_model(model_path)
        logging.info("Model is loaded!")

    def load_text(self, path):
        a_file = open(path, "rb")
        output = pickle.load(a_file)
        return output

    def translate(self,text,max_seq_length):
        text=re.sub(r'[^\w\s]', '', text)
        en_word_index=self.load_text('en_word_index.txt')
        fr_word_index=self.load_text('fr_word_index.txt')
        sentence = [en_word_index[word] for word in text.split()]
        sentence = pad_sequences([sentence], maxlen=max_seq_length, padding='post')
        result=self.model.predict(sentence[:1])[0]
        index_to_words = {id: word for word, id in fr_word_index.items()}
        index_to_words[0] = '<PAD>'
        result=' '.join([index_to_words[prediction] for prediction in np.argmax(result, 1)])
        result=result.split()
        output=[]
        for i in result:
            if i !='<PAD>':
                output.append(i)
        return (' '.join(output))

def main():
    model = En_Fre_Translation('.\model\En_Fr_Translation.h5')
    predicted_text = model.translate("she is driving the truck",21)
    logging.info( f"This is the translation: \n {predicted_text}")

if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO)
    main()
```

Figure 3: screenshot of model.py script used to load model and give a prediction for a specified example

To test our saved model, we created a model.py script that contains all needed statements to make a prediction for a specified example “she is driving the truck” (Figure 3).

Run the model.py using the below command (Figure 4)

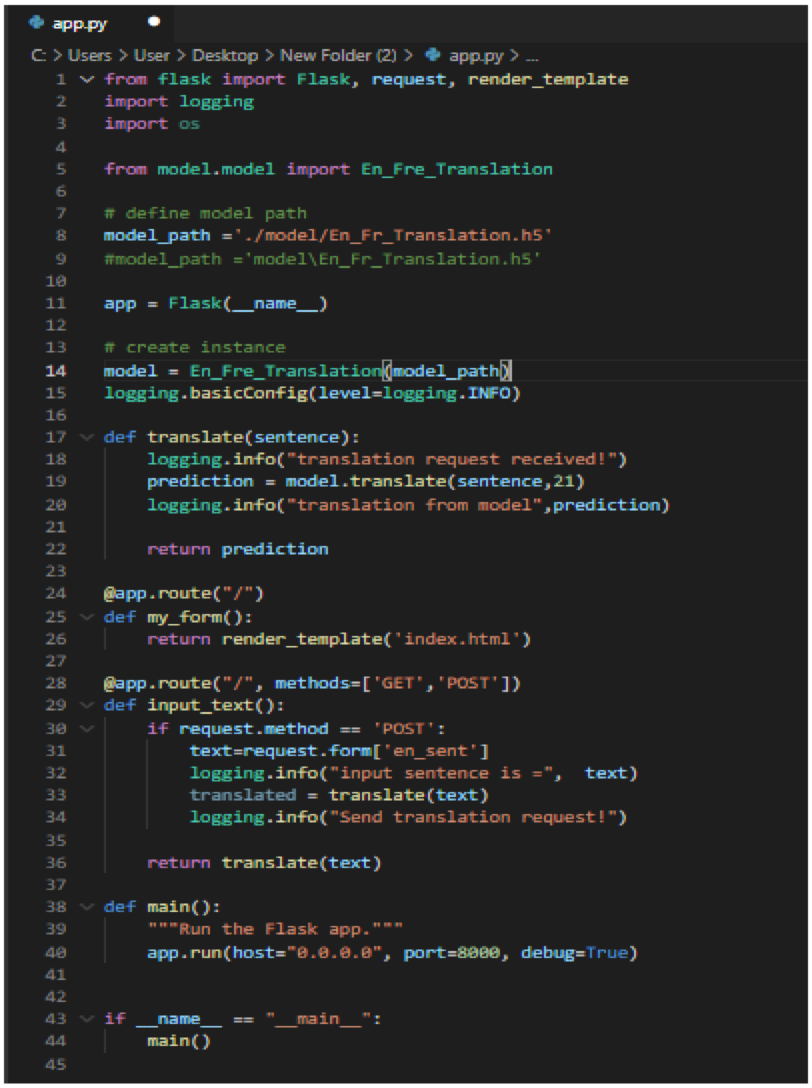


```
C:\Users\User\Desktop\New Folder (2)>folder\Scripts\activate.bat

(folder) C:\Users\User\Desktop\New Folder (2)>python model\model.py
INFO:root:En_Fre_Translation class initialized
INFO:root:Model is loaded!
INFO:root:This is the translation:
 elle conduit le nouveau camion noir
```

Figure 4: screenshot of activating environment and running model.py

4. Creating Web application using flask

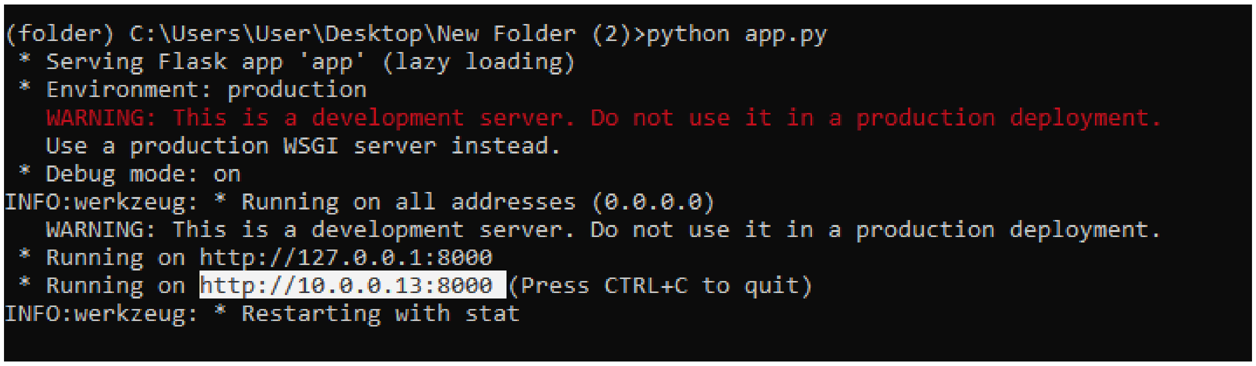
Here we create the routes for the endpoints of our application. We used html files to create an input form so the user can write English sentence and press the submit button to translate it to French using route (”\”). The sentence is translated using translate function that imports the model.py file to make the prediction.



```python
from flask import Flask, request, render_template
import logging
import os

from model.model import En_Fre_Translation

# define model path
model_path ='./model/En_Fr_Translation.h5'
#model_path ='model\En_Fr_Translation.h5'

app = Flask(__name__)

# create instance
model = En_Fre_Translation(model_path)
logging.basicConfig(level=logging.INFO)

def translate(sentence):
    logging.info("translation request received!")
    prediction = model.translate(sentence,21)
    logging.info("translation from model",prediction)

    return prediction

@app.route("/")
def my_form():
    return render_template('index.html')

@app.route("/", methods=['GET','POST'])
def input_text():
    if request.method == 'POST':
        text=request.form['en_sent']
        logging.info("input sentence is =",  text)
        translated = translate(text)
        logging.info("Send translation request!")

    return translate(text)

def main():
    """Run the Flask app."""
    app.run(host="0.0.0.0", port=8000, debug=True)


if __name__ == "__main__":
    main()

```

Figure 5: “app.py” script to create flask application

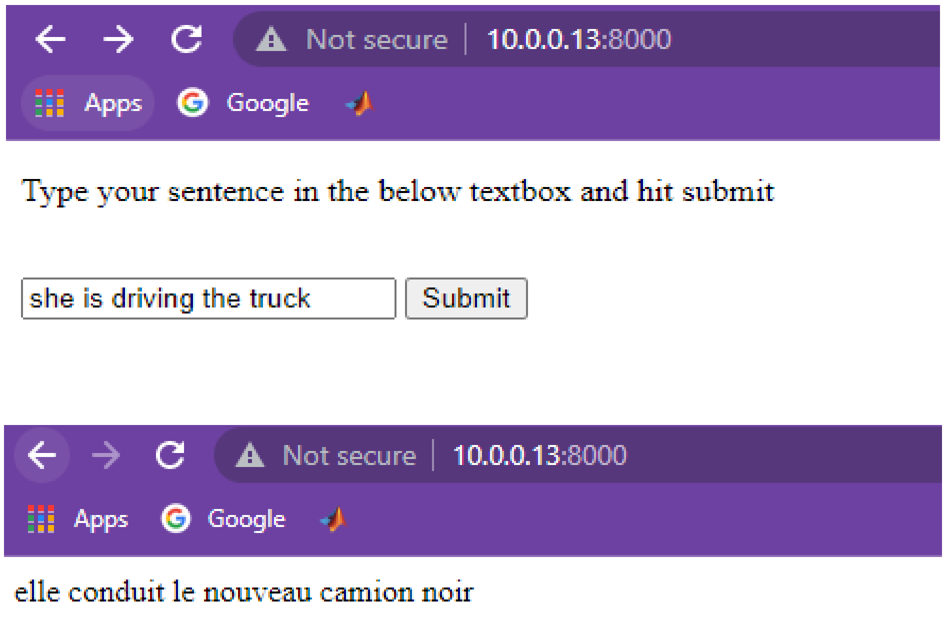
The flask app runs on the host at port 8000. To test the app, go to cmd and write:



```
(folder) C:\Users\User\Desktop\New Folder (2)>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: on
INFO:werkzeug: * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:8000
 * Running on http://10.0.0.13:8000 (Press CTRL+C to quit)
INFO:werkzeug: * Restarting with stat
```

Figure 6: Run application on local host

We used this url "http://10.0.0.13:8000" to run application on the browser. Once the page is opened, you can type your sentence, press submit button to view the translation.



Type your sentence in the below textbox and hit submit

she is driving the truck

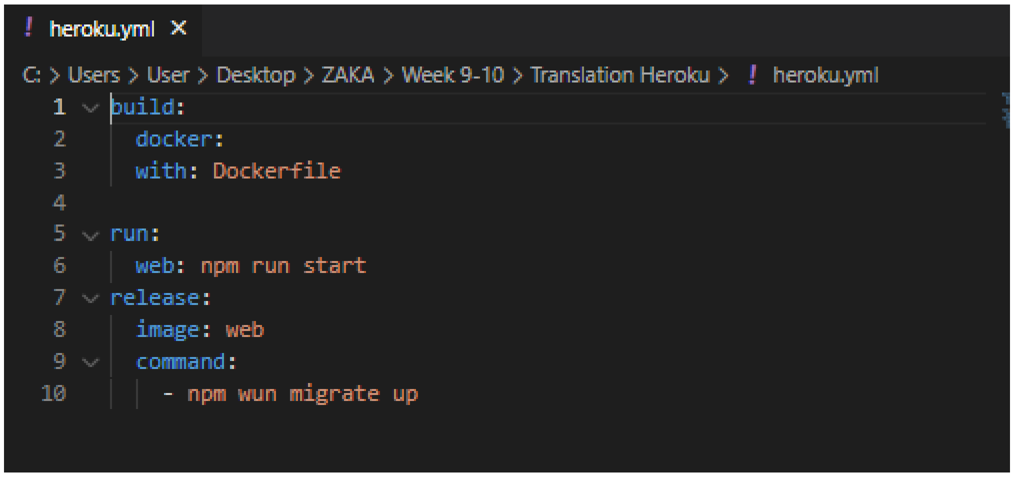
elle conduit le nouveau camion noir

Figure 7: Sentence translation

Deploying containerized model using Heroku Method 2

After creating a new project on Heroku namaed contain-en-fr. We used the file named "heroku.yml" that can use Dockerfile and execute the app.

In this method, we don't use the gunicorn library just tensorflow and flask

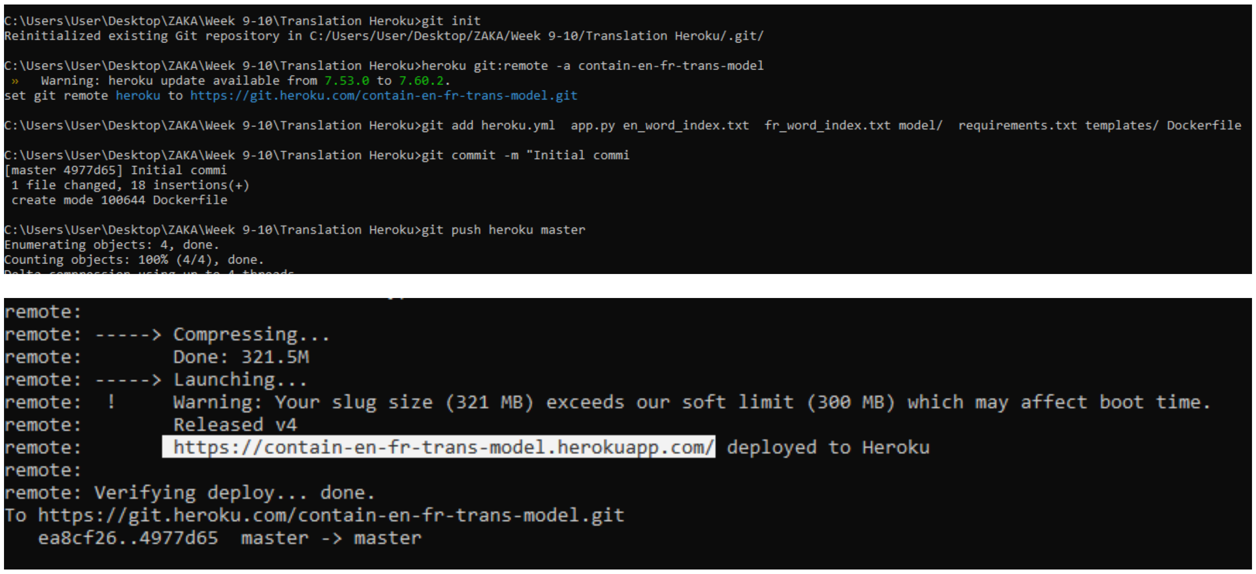


heroku.yml

C: > Users > User > Desktop > ZAKA > Week 9-10 > Translation Heroku > heroku.yml

```yaml
build:
  docker:
  with: Dockerfile

run:
  web: npm run start
release:
  image: web
  command:
    - npm wun migrate up
```

Figure 8: heroku.yml script



```
C:\Users\User\Desktop\ZAKA\Week 9-10\Translation Heroku>git init
Reinitialized existing Git repository in C:/Users/User/Desktop/ZAKA/Week 9-10/Translation Heroku/.git/

C:\Users\User\Desktop\ZAKA\Week 9-10\Translation Heroku>heroku git:remote -a contain-en-fr-trans-model
 »   Warning: heroku update available from 7.53.0 to 7.60.2.
set git remote heroku to https://git.heroku.com/contain-en-fr-trans-model.git

C:\Users\User\Desktop\ZAKA\Week 9-10\Translation Heroku>git add heroku.yml  app.py en_word_index.txt  fr_word_index.txt model/  requirements.txt templates/ Dockerfile

C:\Users\User\Desktop\ZAKA\Week 9-10\Translation Heroku>git commit -m "Initial commi
[master 4977d65] Initial commi
 1 file changed, 18 insertions(+)
 create mode 100644 Dockerfile

C:\Users\User\Desktop\ZAKA\Week 9-10\Translation Heroku>git push heroku master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
```

```
remote:
remote: -----> Compressing...
remote:        Done: 321.5M
remote: -----> Launching...
remote:  !     Warning: Your slug size (321 MB) exceeds our soft limit (300 MB) which may affect boot time.
remote:        Released v4
remote:        https://contain-en-fr-trans-model.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/contain-en-fr-trans-model.git
   ea8cf26..4977d65  master -> master
```

Figure 9: Second heroku deployment using docker container

Type (https://contain-en-fr-trans-model.herokuapp.com/)to try the app on Heroku