Bink Coding Test

Coding Test

Build a basic tool in Python, Ruby or another language of your choice that does *something* you'd normally end up having to do manually. Some ideas:

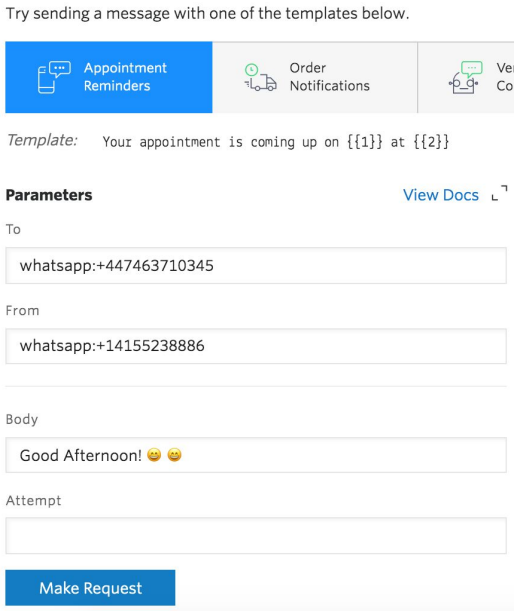
- Backing up a database and sending the output to a cloud storage solution.
- A basic API to provide you with some useful information.
- Getting some information back from an external source and presenting it to the end user.

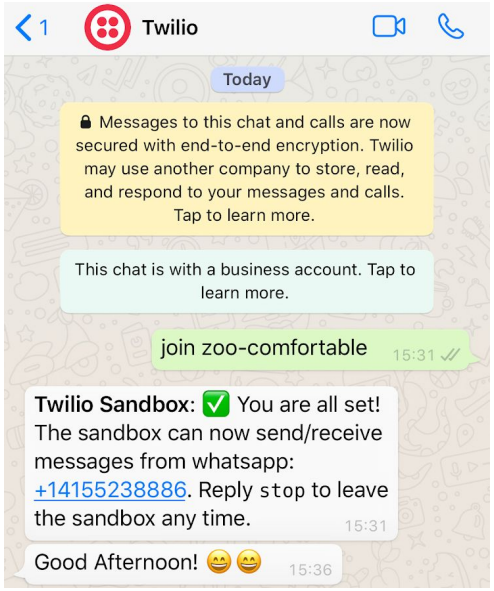
Again, bonus points:

- If you're using third-party packages, make sure to use a packaging solution such as Pipenv or Bundler.

The tool I built uses Twilio to send Whatsapp messages to a specific phone number and will send these messages at regular time intervals. This tool was deployed to Heroku to simulate pushing an application to a production environment.
A use case for this tool could be for sending a reminder message to someone such as “Don’t forget to pick up your sister from school at 3pm” or sending a “Good morning” text to a significant other.

1. Installed Python and pip
2. Installed pipenv
3. Used Twilio as a messaging service





4. Created a **.env** file to store my auth token
    - Make sure to add this file in **.gitignore** file before pushing to GitHub

```
message.py    .env    Pipfile
.env
1   TWILIO_AUTH_TOKEN="abcdefghijklmnop"
```

5. Used the **python-dotenv** package to get environment variables from **.env** file

```
message.py > ...
import os
from dotenv import load_dotenv
from twilio.rest import Client

load_dotenv()

account_sid = 'AC191770330f0d4cae92dd66688b4eadf7'
auth_token = os.getenv('TWILIO_AUTH_TOKEN')
```

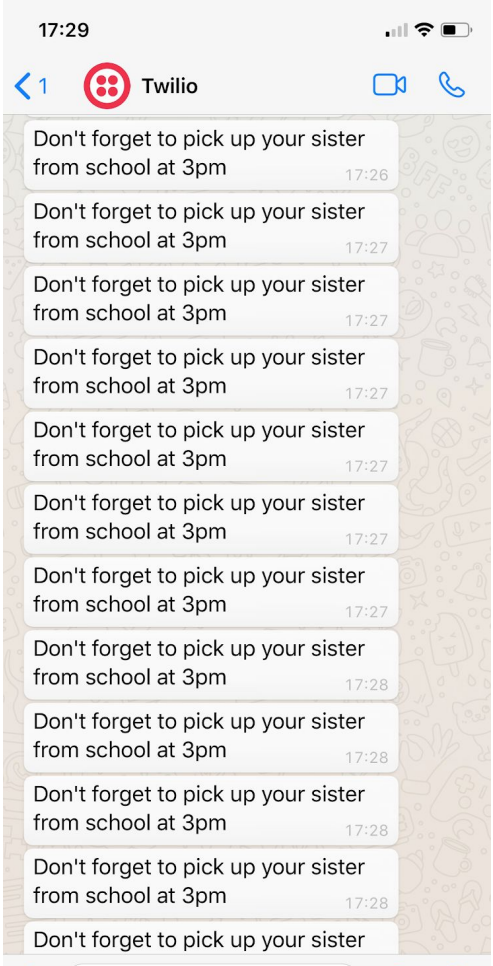
6. Used **Advanced Python Scheduler** to execute code periodically

```
scheduler.py > ...
from apscheduler.schedulers.blocking import BlockingScheduler
from message import send_message

sched = BlockingScheduler()

# Schedule job_function to be called every two hours
sched.add_job(send_message, 'interval', seconds=10)

sched.start()
```

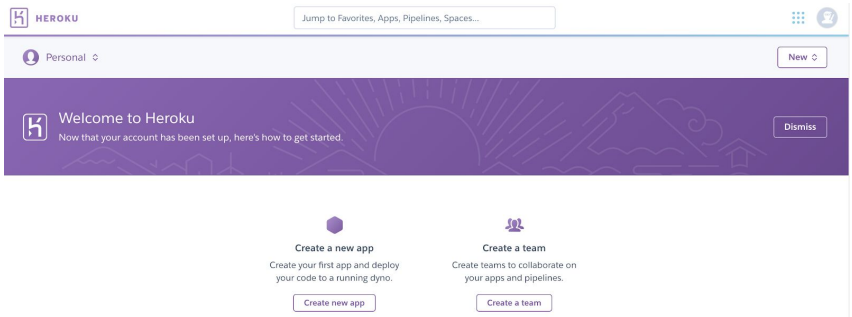
7. Created a function in **message.py** which will send the message. This will get called by the scheduler

```
def send_message():
    message = client.messages.create(
                                    from_='whatsapp:+14155238886',
                                    body="Don't forget to pick up your sister from school at 3pm",
                                    to='whatsapp:+447463710345'
                                )

    print(message.sid)
```

Whatsapp message:



8. Next, I used Heroku to run my code so the application will still keep on sending messages even if I turn off my local computer



9. Followed Heroku's app creation instructions, heroku login etc.

```
→  02_coding_test heroku login
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/48a4b
kbgYAgMoKWnQB.jRd2S5-fSVO7FZWCoW5_qlkxeXUmyPNaHv7q3ZHewmc
Logging in... done
Logged in as regi.azure@gmail.com
```

10. Created a **Procfile**

```
Procfile
1   clock: python clock.py
```

11. Checked if there's any vulnerabilities with the dependencies before pushing to Heroku

```
→  02_coding_test git:(master) ✗ pipenv check
Checking PEP 508 requirements…
Passed!
Checking installed package safety…
All good!
```

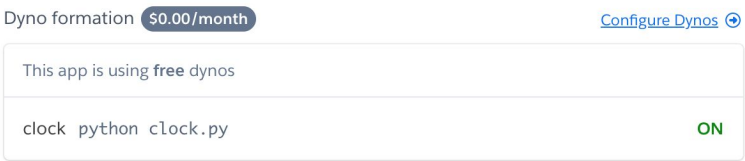
12. Pushed to Heroku

```
→  02_coding_test git:(master) git add .
→  02_coding_test git:(master) ✗ git commit -am "Bink Coding Test"
[master 9076e41] Bink Coding Test
 1 file changed, 1 insertion(+), 1 deletion(-)
→  02_coding_test git:(master) git push heroku master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
```

13. Heroku installing dependencies from **Pipfile.lock**

```
remote: -----> Installing python-3.7.9
remote: -----> Installing pip 9.0.2, setuptools 47.1.1 and wheel 0.34.2
remote: -----> Installing dependencies with Pipenv 2018.5.18…
remote:        Installing dependencies from Pipfile.lock (610361)…
remote: -----> Installing SQLite3
remote: -----> Discovering process types
remote:        Procfile declares types -> clock
remote:
remote: -----> Compressing...
remote:        Done: 65.2M
remote: -----> Launching...
remote:        Released v11
remote:        https://bink-coding-test.herokuapp.com/ deployed to Heroku
```

14. Turned on Dyno resource from Heroku



15. The application has been deployed to Heroku

Notes:

Use **pipenv shell** to activate shell for virtual environment

Display packages and dependencies using **pipenv lock -r** command

pipenv check command to check for security vulnerabilities for any of our installed packages

pipenv graph command to get a visual representation of packages and their dependencies. This is good for debugging conflicting dependencies, check where certain packages are installed from etc.

```
➜  02_coding_test git:(master) ✗ pipenv graph
APScheduler==3.6.3
  - pytz [required: Any, installed: 2020.1]
  - setuptools [required: >=0.7, installed: 46.1.3]
  - six [required: >=1.4.0, installed: 1.15.0]
  - tzlocal [required: >=1.2, installed: 2.1]
    - pytz [required: Any, installed: 2020.1]
python-dotenv==0.14.0
twilio==6.45.1
  - PyJWT [required: >=1.4.2, installed: 1.7.1]
  - pytz [required: Any, installed: 2020.1]
  - requests [required: >=2.0.0, installed: 2.24.0]
    - certifi [required: >=2017.4.17, installed: 2020.6.20]
    - chardet [required: >=3.0.2,<4, installed: 3.0.4]
    - idna [required: >=2.5,<3, installed: 2.10]
    - urllib3 [required: >=1.21.1,<1.26,!=1.25.1,!=1.25.0, installed: 1.25.10]
  - six [required: Any, installed: 1.15.0]
```

pipenv lock command to update **Pipfile.lock** file which will be pushed to production.

Use the command **pipenv install --ignore-pipfile** to install everything from our **Pipfile.lock**.