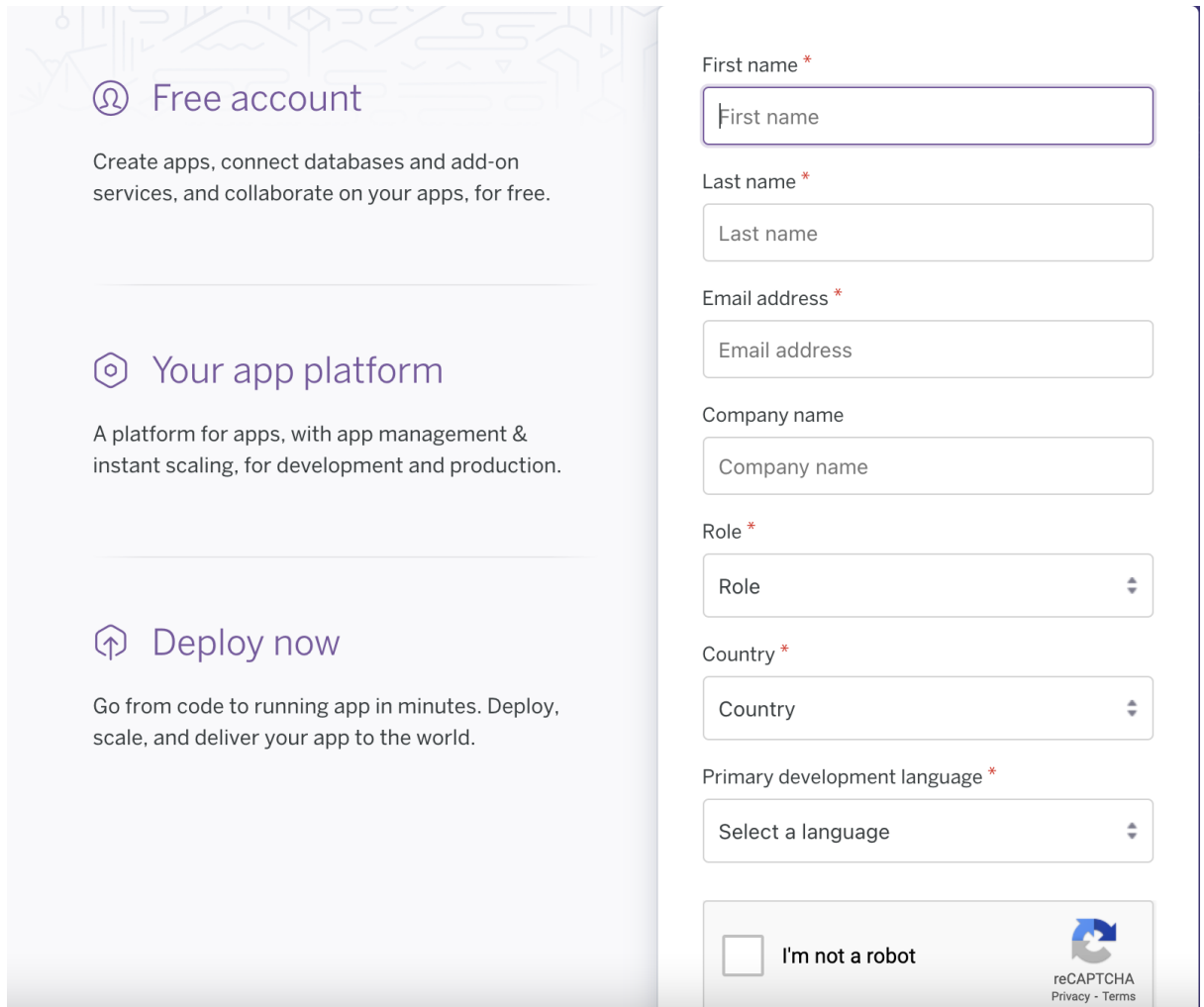


## Sesión # 17 Componente Práctico

### Administrar un servidor en la nube para albergar una aplicación WEB

En el presente componente práctico, vamos a realizar el despliegue de nuestra aplicación (sesión 14) pero esta vez, utilizando Heroku como PaaS y WSGI Gunicorn.

1. Accede a [Heroku](#) y crea una cuenta



**Free account**

Create apps, connect databases and add-on services, and collaborate on your apps, for free.

**Your app platform**

A platform for apps, with app management & instant scaling, for development and production.

**Deploy now**

Go from code to running app in minutes. Deploy, scale, and deliver your app to the world.

First name \*

Last name \*

Email address \*

Company name

Role \*

Country \*

Primary development language \*

I'm not a robot

reCAPTCHA  
Privacy - Terms

2. Ingresa al dashboard de Heroku y selecciona la opción Create a new app  
<https://dashboard.heroku.com/apps>
3. Define un nombre para tu proyecto ejemplo: *app-sesion16-mintic* y elige United State region.
4. Ve a la sesión de Deploy y selecciona la opción de 'Use heroku CLI'
5. Instala la consola de Heroku en: [Getting Started](#)

6. Inicia tu proyecto como lo venimos haciendo en las sesiones anteriores e instala las dependencias necesarias
7. Instala desde tu consola la biblioteca gunicorn. Para ello ejecuta el siguiente comando:


```
pip install gunicorn
```

8. Para trabajar con variables de entorno, instala decouple.

```
pip install python-decouple
```

9. Una vez con las dependencias instaladas el siguiente paso será generar nuestro archivo requirements.txt el cual guardará todas las librerías necesarias para correr el proyecto

```
pip freeze > requirements.txt
```



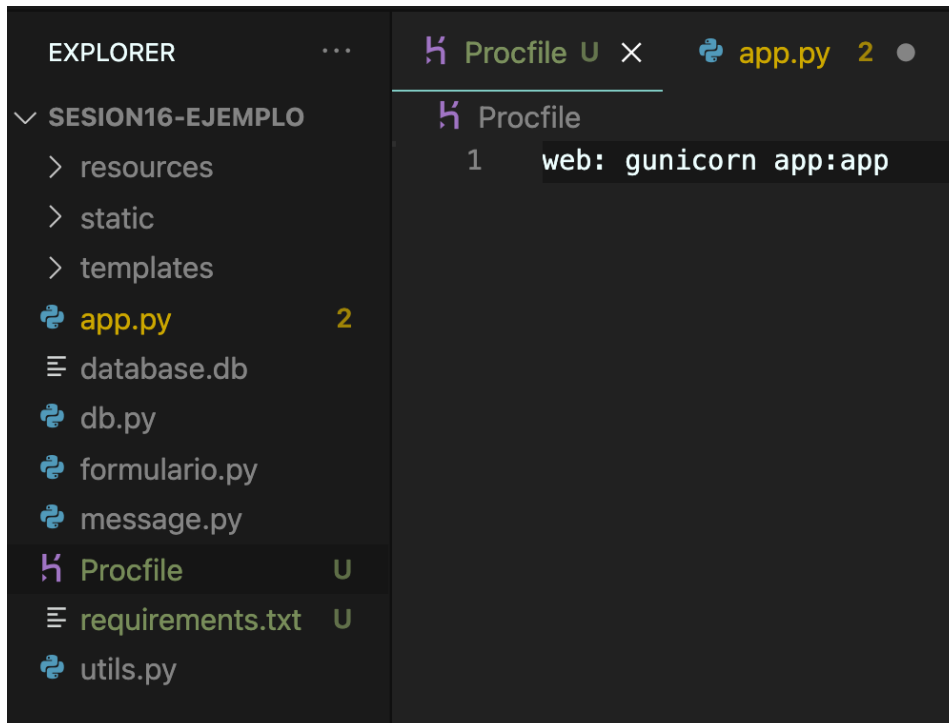
The screenshot shows a code editor with a dark theme. The top bar has tabs for 'Procfile' and 'requirements.txt'. The 'requirements.txt' tab is active, showing a list of dependencies. The text is as follows:

```
requirements.txt
You, seconds ago | 1 author (You)
1  cachetools==4.2.4
2  certifi==2021.10.8
3  charset-normalizer==2.0.7
4  click==8.0.3
5  cssselect==1.1.0
6  cssutils==2.3.0
7  Flask==2.0.2
8  Flask-WTF==0.15.1
9  gunicorn==20.1.0
10 idna==3.3
11 itsdangerous==2.0.1
12 Jinja2==3.0.2
13 lxml==4.6.3
14 MarkupSafe==2.0.1
15 premailer==3.10.0
16 python-decouple==3.5
17 requests==2.26.0
18 urllib3==1.26.7
19 validate-email==1.3
20 Werkzeug==2.0.2
21 WTForms==2.3.3
22 yagmail==0.14.260
23
```

10. Crea un Procfile en tu proyecto. En este fichero se define el proceso que va a ejecutar el dyno de Heroku. Procfile es siempre un archivo de texto simple que se

nombrar Procfile sin una extensión de archivo. Una vez exista Procfile, se debe añadir la siguiente línea:

```
web: gunicorn nombre_archivo_ejecutar:app
```



- En mi caso coloco app:app ya que la aplicación de Flask (app) se encuentra en mi archivo app.py. en caso fuera main.py debería ir main:app

## Configura Heroku

11. Accede a heroku mediante CLI. Usa el siguiente comando y sigue las indicaciones para generar una clave con SSH

```
heroku login
```

```
innovacion@MacBook-Pro-de-Innovacion-1 ~ % 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/df9da023-f3db-4e74-8dc0-7436b1cd5519?requestor=SFMyNTY.g2gDbQAAAA4x0DEuMjM1Ljg4LjEyMG4GAKh9NYZ8AWIAAVGA.UJGITqbfaIaPp-Sy0tyD52ZV-Nhg_-K8w1bZog7rm1U
Logging in... done
Logged in as vlarsonz_e925z@pebih.com
```

12. Una vez logueado y si tu rama principal del repositorio es diferente a main, desde la ubicación de tu proyecto, cambia de tu rama master a main (dado el caso aplique)  
git checkout -b main  
git branch -D master

y resetea el repositorio que tienes en heroku:

```
heroku repo:reset -a appname
```

### 13. Despliega tu aplicación

`git push heroku rama-principal`

```
Compressing objects: 100% (24/24), done.
Writing objects: 100% (28/28), 78.91 KiB | 39.46 MiB/s, done.
Total 28 (delta 1), reused 23 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
remote: -----> Using buildpack: heroku/python
remote: -----> Python app detected
remote: -----> No Python version was specified. Using the same version as the last build: python-3.9.7
remote: -----> To use a different version, see: https://devcenter.heroku.com/articles/python-runtimes
remote: -----> No change in requirements detected, installing from cache
remote: -----> Using cached install of python-3.9.7
remote: -----> Installing pip 20.2.4, setuptools 47.1.1 and wheel 0.36.2
remote: -----> Installing SQLite3
remote: -----> Installing requirements with pip
remote: -----> Discovering process types
remote: Procfile declares types -> web
remote:
remote: -----> Compressing...
remote: Done: 69.5M
remote: -----> Launching...
remote: Released v4
remote: https://app-prueba-mintic.herokuapp.com/ deployed to Heroku
remote:
```

...

```
remote: -----> No change in requirements detected, installing from cache
remote: -----> Using cached install of python-3.9.7
remote: -----> Installing pip 20.2.4, setuptools 47.1.1 and wheel 0.36.2
remote: -----> Installing SQLite3
remote: -----> Installing requirements with pip
remote: -----> Discovering process types
remote: Procfile declares types -> web
remote:
remote: -----> Compressing...
remote: Done: 69.5M
remote: -----> Launching...
remote: Released v4
remote: https://app-prueba-mintic.herokuapp.com/ deployed to Heroku
remote:
remote: !
remote: ! ## Warning - The same version of this code has already been built: 1950e05313821225d667eb84ba783a9cbc7eecd4
remote: !
remote: ! We have detected that you have triggered a build from source code with version 1950e05313821225d667eb84ba783a9cbc7eecd4
remote: ! at least twice. One common cause of this behavior is attempting to deploy code from a different branch.
remote: !
remote: ! If you are developing on a branch and deploying via git you must run:
remote: !
remote: !     git push heroku <branchname>:main
remote: !
remote: ! This article goes into details on the behavior:
remote: !   https://devcenter.heroku.com/articles/duplicate-build-version
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/app-prueba-mintic.git
* [new branch]    main -> main
```

### 14. Una vez terminen de cargar todas las dependencias, podrás acceder a tu página web. El enlace aparecerá en consola, así:

```

re remote: ! https://devcenter.heroku.com/articles/duplicate-build-
re remote:
re remote: Verifying deploy... done.
re To https://git.heroku.com/app-prueba-mintic.git
re * [new branch]      main -> main
re
re innovacion@MacBook-Pro-de-Innovacion-1 sesion16-ejemplo %

```

### Para tener en cuenta:

- Siempre que hagas cambios a tu aplicación debes hacer push al repositorio y luego `git push heroku rama-principal`.
- En caso obtengas un error así:

```

innovacion@MacBook-Pro-de-Innovacion-1 sesion16-ejemplo % git push heroku main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 399 bytes | 399.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: !       Your account has reached its concurrent builds limit
remote:
To https://git.heroku.com/app-prueba-mintic.git
! [remote rejected] main -> main (pre-receive hook declined)
error: failed to push some refs to 'https://git.heroku.com/app-prueba-mintic.git'

```

Ejecuta el comando `heroku restart` y luego nuevamente `git push heroku rama-principal`.

- Para ver los logs

