Deploying Django in Azure

Reference:

<https://www.youtube.com/watch?v=QjEVmQ4rcWA&list=LL&index=14>

<https://www.youtube.com/watch?v=X2eUFKrY00Y&list=LL&index=10>

<https://www.youtube.com/watch?v=_Gx30E3r1xk&list=LL&index=9>

<https://www.youtube.com/watch?v=L5oCPVr7l4c&list=LL&index=8>

Table of Contents

[1. Installing additional packaged for deployment 2](#_Toc200380067)

[2. Creating settings files for deployment (deployment.py) and wsgi.py 2](#_Toc200380068)

[2.1 Create a file “deployment.py”. (See each line and read the purpose of each) 2](#_Toc200380069)

[2.2 Modify “wsgi.py” to call settings as per the environment variable. 4](#_Toc200380070)

[2.3 Create “.production.py” file in main project directory(where manage.py” is placed) and add below lines: 5](#_Toc200380071)

## Installing additional packaged for deployment

The packages python-dotenv and whitenoise serve specific purposes in Python web development, particularly for managing environment variables and serving static files

* Install “pip intall python-dotenv”
* Install “pip install whitenoise”

Dotenv: This package allows you to load environment variables from a .env file into your Python application.

Whitenoise: This package simplifies serving static files (e.g., CSS, JavaScript, images) in Python web applications, especially for frameworks like Django.

Tip: Running python manage.py collectstatic in a Django project does not make the code search for static files in the templates directory. Instead, it collects static files (e.g., CSS, JavaScript, images) from each app’s static directories and places them into the STATIC\_ROOT directory (e.g., staticfiles). When you deploy with whitenoise, it serves these files from STATIC\_ROOT, not templates. The templates directory is for HTML templates, which Django processes separately using the template engine. Ensure STATICFILES\_DIRS and STATIC\_ROOT are correctly set in settings.py to avoid confusion.

## Creating settings files for deployment (deployment.py) and wsgi.py

* 1. 2.1 **Create a file “deployment.py”. (See each line and read the purpose of each)**

import os

from .settings import \*

from .settings import BASE\_DIR

# Allow all hosts: WEBSITE\_HOSTNAME is set in Azure App Service

ALLOWED\_HOSTS = [os.environ['WEBSITE\_HOSTNAME']]

# CSRF\_TRUSTED\_ORIGINS is required for Azure App Service to allow CSRF protection

# CSRF stands for Cross-Site Request Forgery, a security vulnerability that allows

# an attacker to trick a user into submitting a request that they did not intend to make.

CSRF\_TRUSTED\_ORIGINS = ['https://' + os.environ['WEBSITE\_HOSTNAME']]

# DEBUG disabled for production. Because this is a production environment,

# it is important to disable DEBUG mode to prevent sensitive information from being exposed. Like below:

# Using the URLconf defined in RecentCSCSTS.urls, Django tried these URL patterns, in this order:

# admin/

# accounts/ [name='home\_page']

# accounts/ home/ [name='home']

# accounts/ signup/ [name='signup']

# accounts/ register/ [name='signup']

DEBUG = False

# whitenoise.middleware.WhiteNoiseMiddleware : This middleware is used for serving static files in production.

# why these middlewares are used:

# 1. SecurityMiddleware: Provides security enhancements.

# When a client requests the homepage in a Django application using whitenoise,

# the static files (e.g., CSS, JavaScript, images) are not copied to the client on every request.

# Instead, whitenoise serves the static files from the STATIC\_ROOT directory (e.g., staticfiles),

# where they were previously collected by running python manage.py collectstatic.

# These files are sent to the client’s browser only when referenced in the

# HTML (e.g., via <link> or <script> tags), and whitenoise optimizes delivery with compression and caching.

# Subsequent requests may use cached files in the browser, reducing server load.

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

'whitenoise.middleware.WhiteNoiseMiddleware',

]

#whitenoise.storage.CompressedManifestStaticFilesStorage ?

STATICFILES\_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'

STATIC\_ROOT = os.path.join(BASE\_DIR, 'staticfiles')

# Why is this connection string used?

# The connection string is used to connect to the Azure PostgreSQL database.

# It contains the necessary parameters such as database name, user, password, host, and port.

# This allows the Django application to interact with the database for data storage and retrieval.

# The connection string is typically set as an environment variable in Azure App Service for security reasons.

# The connection string is expected to be in the format:

# "dbname=your\_db\_name user=your\_user password=your\_password host=your\_host port=your\_port"

connection\_string = os.environ.get('AZURE\_POSTGRESSQL\_CONNECTIONSTRING', None)

parameters = {param.split('=')[0]: param.split('=')[1] for param in connection\_string.split(" ")}

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql',

'NAME': parameters['dbname'],

'USER': parameters['user'],

'PASSWORD': parameters['password'],

'HOST': parameters['host'],

'PORT': parameters['port'],

}

}

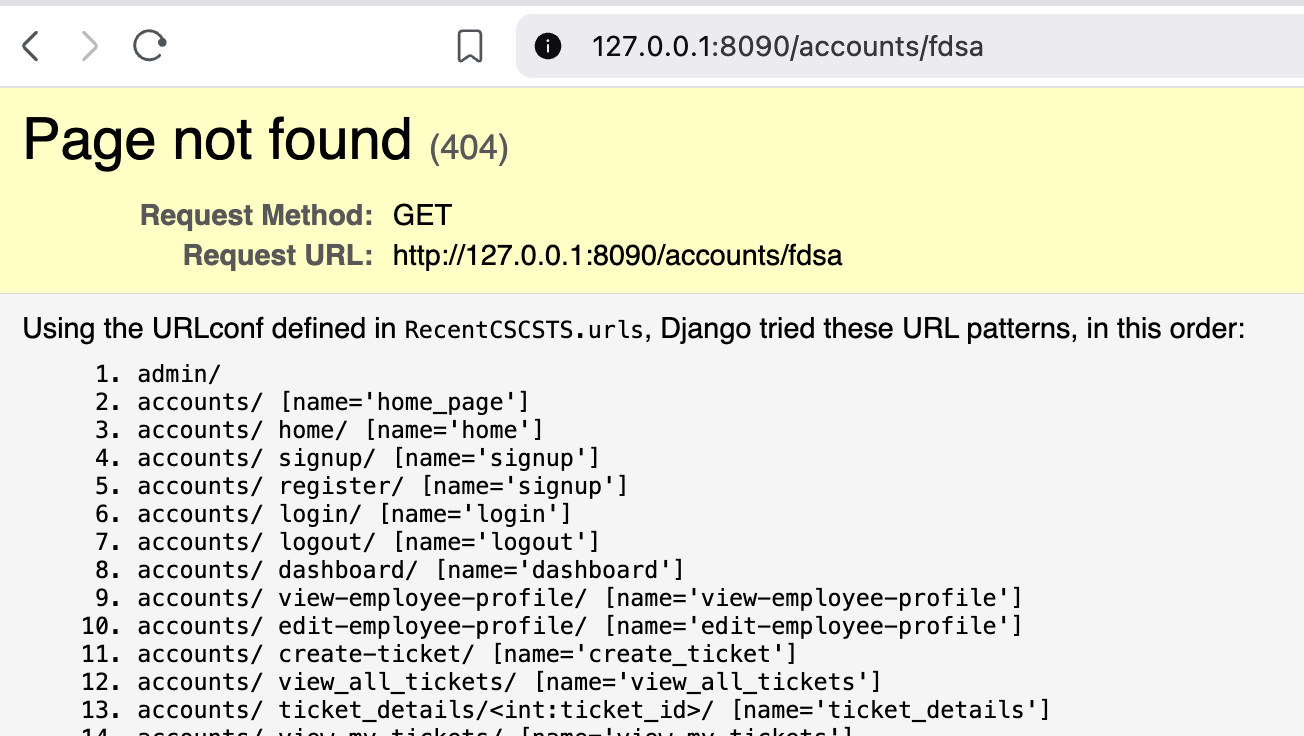


Figure 1 (If DEBUG flag is not set to “False”, the above info will be displayed on browser which is security issue in case of deployment)Figure 1

(If DEBUG flag is not set to “False”, the above info will be displayed on browser which is security issue in case of deployment)

**Note**: “wsgi.py” tells when to use settings.py and when to use deployment.py

## 

**2.2 Modify “wsgi.py” to call settings as per the environment variable.**

"""

WSGI config for RecentCSCSTS project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see

https://docs.djangoproject.com/en/5.1/howto/deployment/wsgi/

"""

import os

from django.core.wsgi import get\_wsgi\_application

# Determine the settings module based on the environment variable

# If 'WEB\_HOSTNAME' is set, use the production settings; otherwise, use the default settings.

settings\_module = 'RecentCSCSTS.deployment' if 'WEB\_HOSTNAME' in os.environ else 'RecentCSCSTS.settings'

# Set the DJANGO\_SETTINGS\_MODULE environment variable

os.environ.setdefault("DJANGO\_SETTINGS\_MODULE", settings\_module)

application = get\_wsgi\_application()

**2.3 Create** “.**.production**.py” **file in main project directory**(where manage.py” is placed) **and add below lines:**

[config]

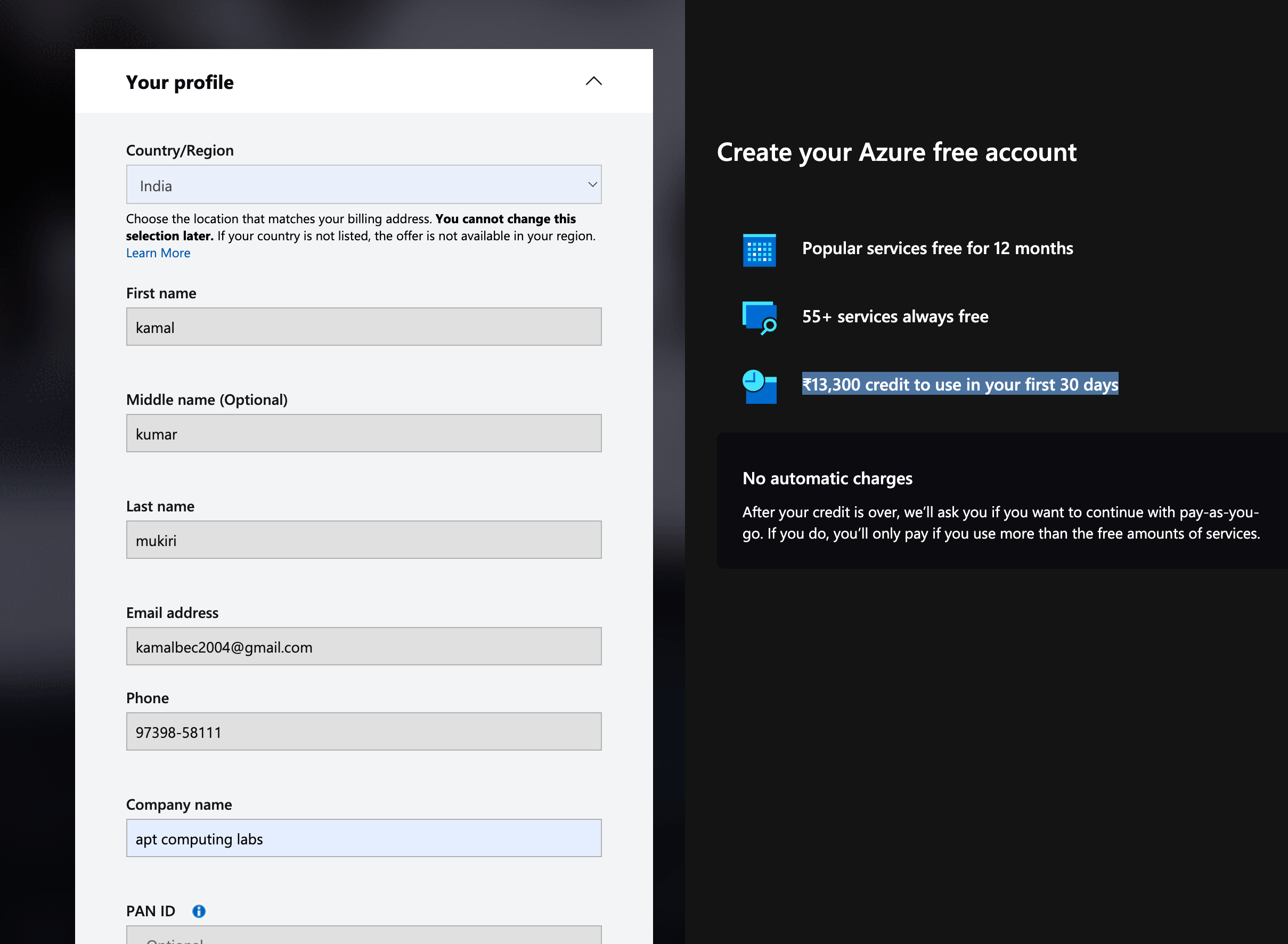
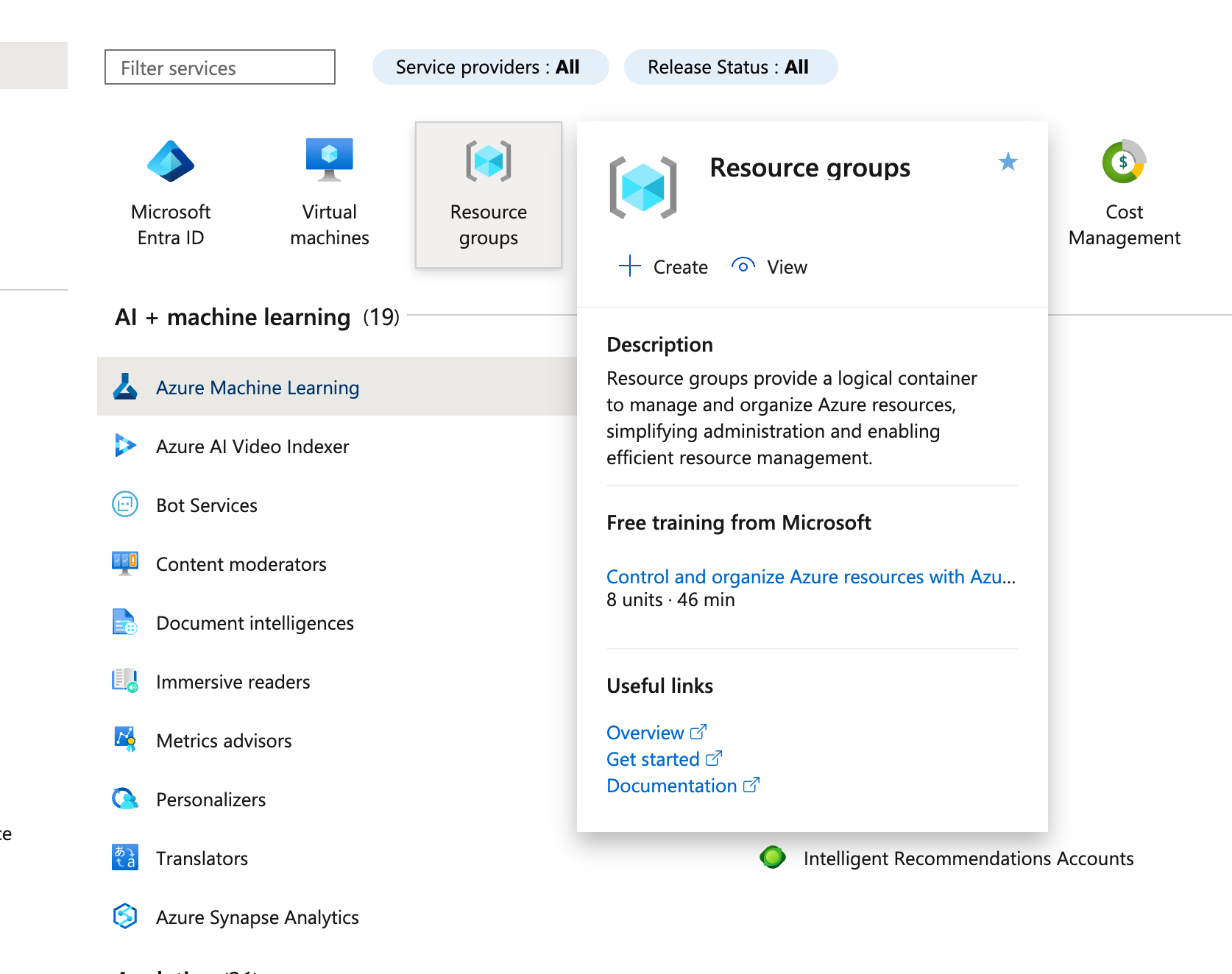
**SCM\_DO\_BUILD\_DURING\_DEPLOYMENT**=true

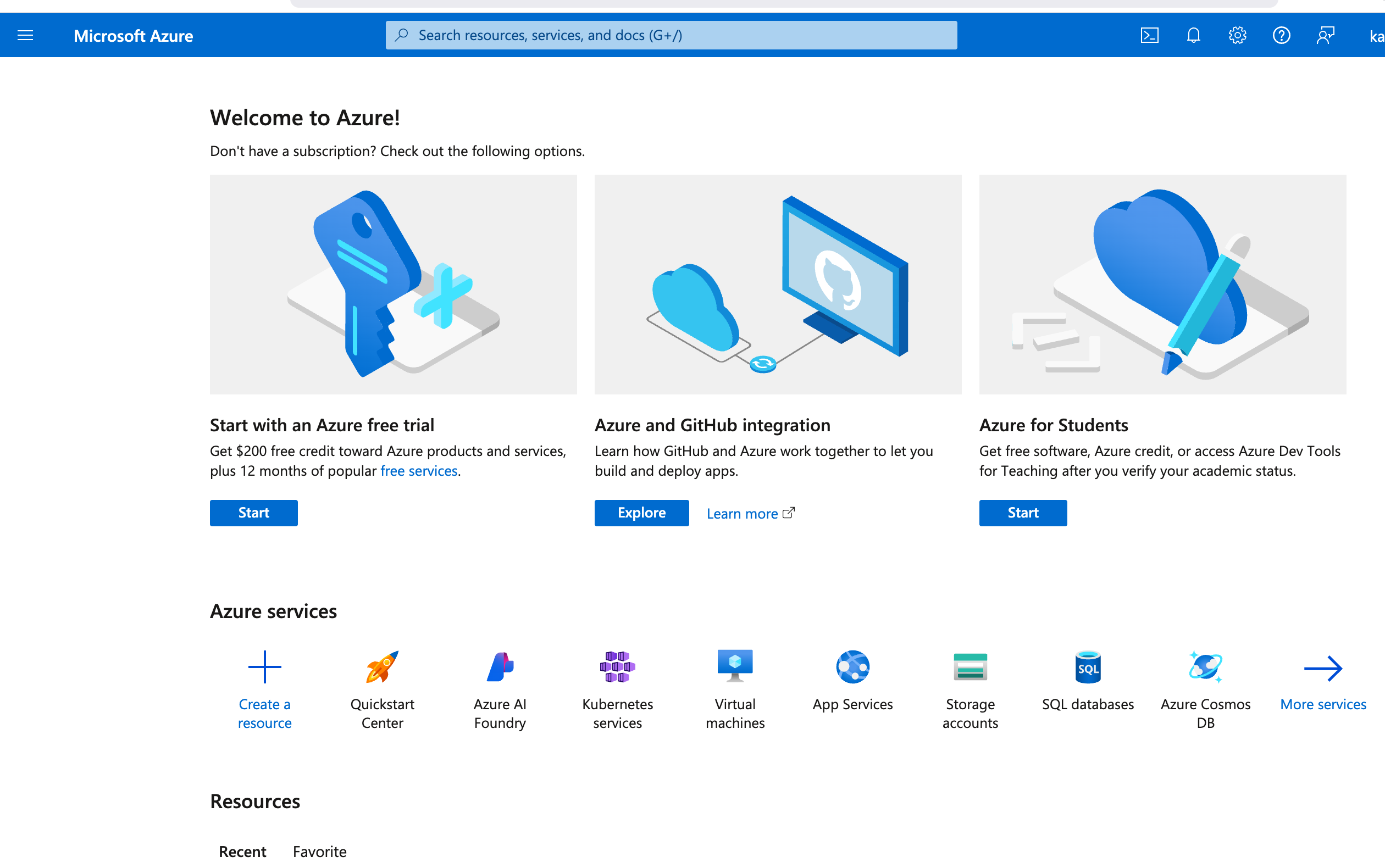
The configuration line [config] **SCM\_DO\_BUILD\_DURING\_DEPLOYMENT**=true is typically used in a platform like Render or another cloud deployment service to control how the deployment process handles builds for your application. Since your previous questions focused on Django, whitenoise, python-dotenv, and CSRF tokens, I’ll assume this is related to deploying a Django application and explain its purpose in that context.

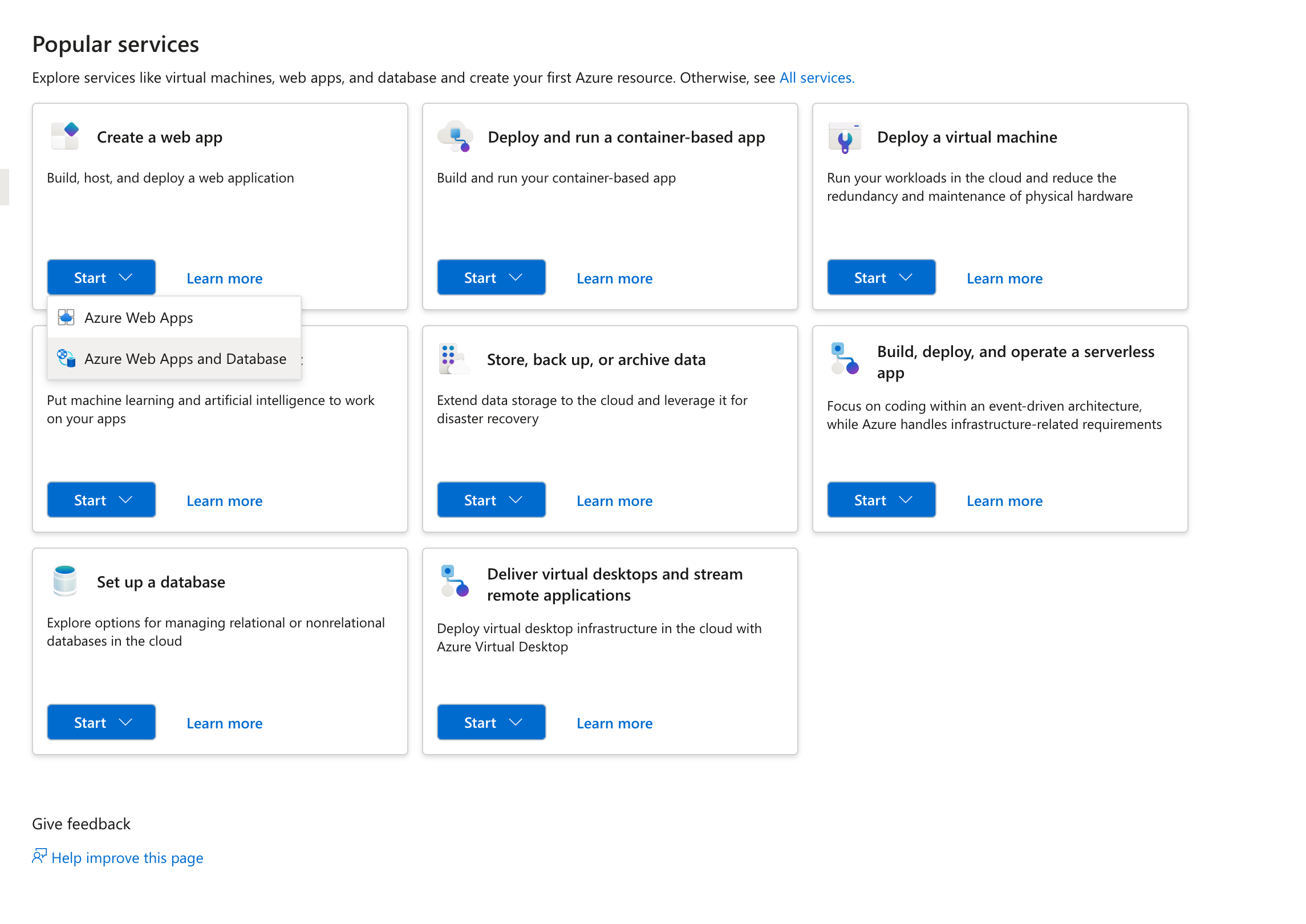
**2.4 Create “requirements.txt” by calling “pip freeze > requirements.txt”**

**2.5 Push the code to github: branch name: azuregit.**

100. SSL cerficates







A screenshot of a computer

AI-generated content may be incorrect.