Azure’s deployment instruction

1. Create an Azure account with a student email.

* Azure requires you to have an account to use the services.
* Signing up for an Azure account with a student’s identity will give you $100 in balance.
* With the student account, we still can use many different services from Azure, but they will be limited in some features.

1. Create an Azure web app service.

* After signing up for an account, you can create an app service.

A logo of a company

Description automatically generated

* Click Create and Web App on the left corner.

A screenshot of a computer

Description automatically generated

(Creating Web App + Database simultaneously will cause complex issues later, so I recommend creating the Web App first).

* From the tab Basics:

+ Subscription: Azure for Students.

+ Resource group: Create a new one.

+ Name: Any name.

+ Publish: Code.

+ Runtime Stack: Python 3.8 (recommended). The higher Python version may lead to failure of deployment when the virtual environment is running and building packages.

+ OS: Linux.

+ Region: East US.

+ Linux plan: F1 (Free).

* Skip to Networking tab, make sure the “Enable network injection” is on.
* Monitoring tab: Enable Application Insights – No.
* Next to Review and Create.
* We will wait for a few minutes for it to be done.

1. Create An Azure Database for PostgreSQL flexible server.

* After we see the web app is deployed green in the notification, you can create a Database server.

A blue cylinder with an elephant head

Description automatically generated

* Click create and Choose Flexible server.
* We will go through similar configurations to the Web App (the Resource group should be the same name as the Web App).
* Choose your admin user and password (keep this in mind, we will use this information later).
* From the Networking tab:

+ Connectivity method: Public access.

+ Firewall Rule: check the “Allow public access from any Azure service within Azure to this server” and click to add the current client IP address.

A close up of a logo

Description automatically generated

* Skip to Review and Create.
* We will wait for a few minutes for it to be done.

1. Connect Azure’s Database server to Azure’s Web App.

* Go to your Web App.
* Go to Configuration.

A grey rectangular object with white text

Description automatically generated

* Click on “New application setting”.

A screenshot of a computer

Description automatically generated

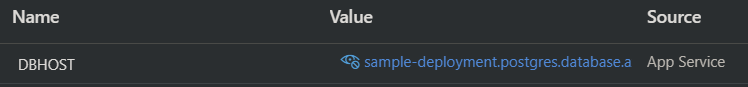
We will add 5 pairs of Names and Values. The Name will be the same as the picture shows. The Value will depend on what you created.

* To find the Value for DBHOST:

+ Go back to your database server, in the Overview setting, you will see it as the Server name. In my case:

A black background with white text

Description automatically generated



* To find the Value for DBNAME:

+ In the database server, go to Databases:



And you will see your database’s name in here, in my case:   


A screenshot of a black and white screen

Description automatically generated

+ Or you can keep the default name “postgres”



* DBPASS, DBUSER: Your password and administrator name when you create the Database server.
* SECRET\_KEY: A replacement of a password, you can choose anything.
* Save all the settings and wait for it to update.

1. Django’s setting file configuration.

* This step we will work locally with our Django’s file.
* Go to your Django’s settings.py.
* Change DEBUG to false (it's required when you deploy a project as production).
* Change the ALLOWED\_HOSTS to your Azure’s web app domain:

+ Go back to your Azure web app, in the Overview section, you will see the Default domain. In my case:  
A close-up of a computer screen

Description automatically generated

A black background with white text

Description automatically generated

* Add this code after that: CSRF\_TRUSTED\_ORIGINS = ["https://<your default domain>"]. In my case:  
  

This code gives us access as administrator and manages the database once the project is deployed.

* In the MIDDLEWARE, add this code at the end (Important): 'whitenoise.middleware.WhiteNoiseMiddleware',

(this one allows your browser to load your front end if your project has complex CSS, JavaScript,… Without this, once it’s deployed, your project will be missing the CSS and JavaScript.)

* Change your DATABASE code in the settings.py, and use the information from your Azure’s Web app and Azure Database Server, now your DATABASE will look like this:

A computer screen with text

Description automatically generated

Make sure you change the ENGINE, add “PORT”: “5432” and “OPTIONS”: {“sslmode”:”require”}.

+ There are other ways to make your information confidential like PASSWORD and SECRET\_KEY without typing straight in by creating an env file (recommend this one when you are working on a real project): <https://dev.to/jakewitcher/using-env-files-for-environment-variables-in-python-applications-55a1>

* Make sure you have “STATIC\_ROOT” in your settings.py:



* If you don’t, just add it.
* If you have “STATICFILES\_DIRS”, just get rid of them or comment them out.

1. Add requirements.txt.

* You have to add a requirements.txt file in the folder that manage.py presents.

A black background with white text

Description automatically generated

* Add all the packages that you need when you finish your project. The required packages of each project may be different, in my case these are all my packages:

A screen shot of a computer

Description automatically generated

1. Deploy and migrate.

* In this step, you have to install the Azure extension in your VScode.

A black and white logo

Description automatically generated

* Once you log in, right-click on your Azure Web App and choose “Deploy to Web App…”, we will wait for a few minutes until it is deployed.

A screenshot of a computer

Description automatically generated

* Once you see no errors present, you have successfully deployed your project on Azure, or you can follow up the log to see what errors are.
* Browse the website to check if your website is now published.
* Go back to the Azure extension in your VScode, right-click on your Azure Web App, and choose “SSH into Web App”.
* In the VScode terminal, you will see we are now in the Azure’s server, type in:

+ “python manage.py makemigrations”

+ “python manage.py migrate”

* If you see no issue, go: “python manage.py createsuperuser”
* Provide your admin user and password, we can use this to manage the project’s database from Azure’s server.
* Now your Django’s website and its database are all migrated on Azure.

\*Bonus: If you want to use Azure’s extension in VScode at the beginning, you can refer to this: <https://learn.microsoft.com/en-us/training/modules/django-deployment/1-introduction>. But make sure you still configure the settings.py correctly and follow this instruction.