Django Deployment Instructions 2024

Setting up a basic Django Project and Deploying to Heroku

This is an updated version of the deprecated ***'Django Blog Cheat Sheet'*** that contained a step by step guide for how to set up a Django project and early Heroku deployment.

This document is for Django v4 projects and contains updated commands for the latest dependencies.

Step 1: Installing Django

Note: It is recommended when you are still learning this content that you type out each line of code, rather than copying and pasting. This will help you learn!

Key:

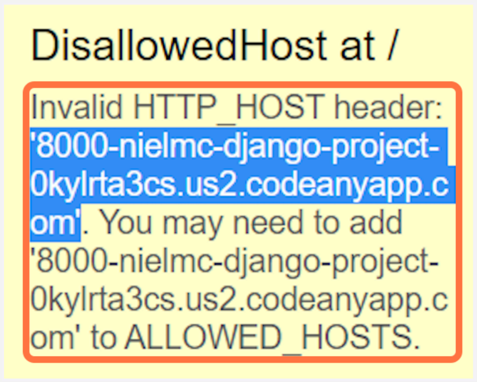
PROJ_NAME = **The name of your project. (Where your settings.py file will be)**

APP_NAME = **App within the larger Django project (Blog, about, comments etc.)**

HEROKU_APP_NAME = This is the name of your live project. This will form part of your deployed project URL

Part 1: Install Django and run the server to test.

In the Terminal:

	Step	Code
	Install Django:	pip3 install Django~=4.2.1
	Create requirements file	pip3 freeze --local > requirements.txt
	Create Project **(PROJ_NAME)**	django-admin startproject **PROJ_NAME .** ***(Don't forget the .)***
	Run Server to Test	python3 manage.py runserver
	You will see a yellow error screen, don't worry! Your server is running properly. This error is telling you that, for security reasons, Django doesn't recognise the hostname - the server name your project is running on. Select and copy the hostname after "Invalid HTTP_HOST header". In this example, that is '8000-nielmc-django-project-0kylrta3cs.us2.codeanyapp.com' - you can include the quotes.	 DisallowedHost at / Invalid HTTP_HOST header: '8000-nielmc-django-project-0kylrta3cs.us2.codeanyapp.com'. You may need to add '8000-nielmc-django-project-0kylrta3cs.us2.codeanyapp.com' to ALLOWED_HOSTS. Paste the hostname between the square brackets of ALLOWED_HOSTS. For the above example, this would look like ALLOWED_HOSTS = ['8000-nielmc-django-project-0kylrta3cs.us2.codeanyapp.com']

Part 2: Creating an app in the Django Project

	Step	Code
	Create App **(APP_NAME)**	python3 manage.py startapp **APP_NAME**

	Add to 'INSTALLED_APPS' **in settings.py**	INSTALLED_APPS = [
		    …
		    'APP_NAME',
		]

	Save file	

Part 3: Setting Up Heroku

In Heroku:

	Step	Code
	Navigate to your Heroku dashboard	Heroku Dashboard
	Create new Heroku app	- Choose a unique app name - Select a region close to you
	Add Config Var in app settings	- Navigate to Settings tab and scroll down to Config Vars - Click "Reveal Config Vars" - Add new key DISABLE_COLLECTSTATIC with value 1

In the Terminal/IDE:

	Step	Code
	Install webserver gunicorn and freeze requirements	pip3 install gunicorn~=20.1 pip3 freeze --local > requirements.txt

	Create a Procfile	Create new file “Procfile” in the root directory **Note:** This file has no file extension and **the P must be capitalised!**
	Declare the process in Procfile	Inside the Procfile, add the following line of code: web: gunicorn **PROJ_NAME**.wsgi
	Add deployed app to ALLOWED_HOSTS	In settings.py add “.herokuapp.com” to the ALLOWED_HOSTS list

In Heroku:

	Connect to repository	- In Heroku app, navigate to Deploy tab - Search for your Github repo
	Check for Add-ons and Dynos	Inside the app’s Resources tabs, ensure you’re using Eco Dynos and delete any Postgres DB Add-ons

---

There are two different methods to create your Postgres Database for your project:

1. Follow the steps below to Use Elephant SQL
2. Use The CI Database Maker

Either of these methods will provide you with a database url that you can utilise both in your project and in Heroku in the same way.

Part 4: Creating a Database

In ElephantSQL:

	Step	**Code**
	Log in to your ElephantSQL account	If you don't have an ElephantSQL.com account yet, the steps to create one are here.
	Click “**Create New Instance**”	
	Set up your plan	● Give your plan a **Name** (this is commonly the name of the project) ● Select the **Tiny Turtle (Free)** plan ● You can leave the Tags field blank
	Click “**Select Region**”	Select a data center near you **Note:** If you receive a message saying "Error: No cluster available in **your-chosen-data-center** yet", choose another region
	Click “**Review**”	Check that your details are correct. Then click “**Create instance**”
	Get Database URL	Navigate back to Dashboard and click on the newly created DB name. Copy your **ElephantSQL database URL** using the Copy icon. It will start with postgres://

Part 5: Connecting to your Database

In the Terminal/IDE:

	Step	Code
	Install Database Packages	pip3 install dj-database-url~=0.5 psycopg **Freeze Requirements!**
	Create env.py file	In the root directory, create a new file “env.py”

In env.py

	Step	Code
	Add env.py to .gitignore	Open the .gitignore file and add “env.py” (This is already added if you’ve used the CI template)
	Import os library	At the top of the env.py file add this line of code: import os
	Set environment variables	In env.py, add the following: os.environ["DATABASE_URL"] = "Paste in ElephantSQL database URL"
	Add in secret key	In **env.py** add the following: os.environ["SECRET_KEY"] = "Make up your own randomSecretKey"

In Heroku:

	Step	Code	
	Add Secret Key to Config Vars	SECRET_KEY, "randomSecretKey"	
	Add a Config Var called DATABASE_URL	DATABASE_URL, "yourDBUrlgoeshere"	**Note:** The value should be the **ElephantSQL database url** you copied in the previous step

In settings.py

	Step	Code

	Reference env.py
	(Note: font in **bold** is new)
		from pathlib import Path
		import os
		import dj_database_url
		
		if os.path.isfile("env.py"):
		   import env

	Remove the insecure secret key **and replace** - *links to the SECRET_KEY variable on Heroku* (Note: font in **bold** is new)	SECRET_KEY = **os.environ.get('SECRET_KEY')**

	Comment out the old DataBases Section	**# DATABASES = {**
		# 'default': {
		# 'ENGINE': 'django.db.backends.sqlite3',
		# 'NAME': BASE_DIR / 'db.sqlite3',
		# }
		# }

	Add **new** DATABASES Section
	
	- *links to the DATATBASE_URL variable on Heroku*
		DATABASES = {
		   'default':
		dj_database_url.parse(os.environ.get("DATABASE_URL"))
		}

Part 6: Migrating your Database

	Step	Code
	Save all files and Make Migrations	python3 manage.py migrate

---

Creating a Super User

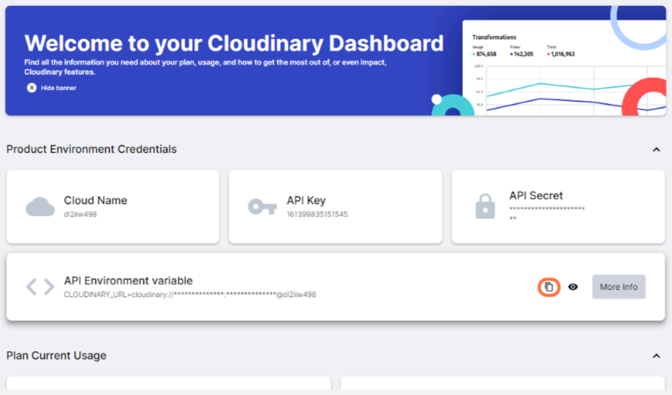
Creating a super user in Django is an important step that creates an admin user. A Django admin user is a special type of user which has access to the backend interface and extra privileges.

	Step	Code
	Create Super User	python3 manage.py createsuperuser

Part 7: Get our static and media files stored on Cloudinary:

	Step	Code
	Install Cloudinary	pip3 install dj3-cloudinary-storage~=0.0.6 pip3 install urllib3~=1.26.15 **Freeze Requirements!**

In Cloudinary.com: (Note: must be logged in)

	Step	Code
	Copy your CLOUDINARY_URL e.g. API Environment Variable.	From Cloudinary Dashboard  Welcome to your Cloudinary Dashboard Find all the information you need about your plan, usage, and how to get the most out of, or even impact, Cloudinary features. Hide banner Transformations Image 874,658 Video 142,305 Total 1,016,963 Product Environment Credentials Cloud Name di2iiw498 API Key 161399835151545 API Secret ********************** API Environment variable CLOUDINARY_URL=cloudinary://**************:**************@di2iiw498 More Info Plan Current Usage

In env.py:

	Step	Code
	Add Cloudinary URL to env.py - *be sure to paste in the correct section of the link*	os.environ["CLOUDINARY_URL"] = "cloudinary://************************"

In Heroku:

	Step	Code
	Add Cloudinary URL to Heroku Config Vars - *be sure to paste in the correct section of the link*	Add to Settings tab in Config Vars e.g. CLOUDINARY_URL, cloudinary://************************

In settings.py:

	Step	**Code**

	Add Cloudinary Libraries to installed apps	INSTALLED_APPS = [
		    …,
		
		    'django.contrib.staticfiles',
		    'cloudinary_storage',
		    'cloudinary',
		    …,
		]
		
		(note: order is important)

	Setup Static Files	STATIC_URL = 'static/'
		STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static'),]
		STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')

	Link file to the templates directory in Heroku *Place under the BASE_DIR line*	TEMPLATES_DIR = os.path.join(BASE_DIR, 'templates')

	Change the templates directory to TEMPLATES_DIR
	Place within the TEMPLATES array
		TEMPLATES = [
		    {
		        …,
		        'DIRS': [**TEMPLATES_DIR**],
		       …,
		            ],
		        },
		    },
		]

In the IDE file explorer or terminal:

	Step	**Code**
	Create 3 new folders on top level directory	media, static, templates

*** Note:** Save all files

	Step	Code
	Install WhiteNoise	pip3 install whitenoise~=5.3.0 **Freeze Requirements**
	Wire up WhiteNoise to Django's **MIDDLEWARE** in the **settings.py** file.	‘whitenoise.middleware.WhiteNoiseMiddleware’, Note: The ‘whitenoise’ middleware must be placed directly after the Django SecurityMiddleware

In order to test if your project is working properly, and your static files are being served, create a basic view to render a template, add a html template to your templates folder and wire up your urls. Create and link a custom stylesheet to your template before deployment.

In the Terminal:

	Step	Code
	Add, Commit and Push	git add . git commit -m “Deployment Commit” git push

In Heroku:

	Step	Code
	Deploy Content manually through heroku/	E.g Github as deployment method, on main branch