interweb

Generated on 22/10/2026 05:11

KNOWLEDGE BASE · PUBLIC DOCUMENT

Getting a Django application running

Node.js, Python & Docker

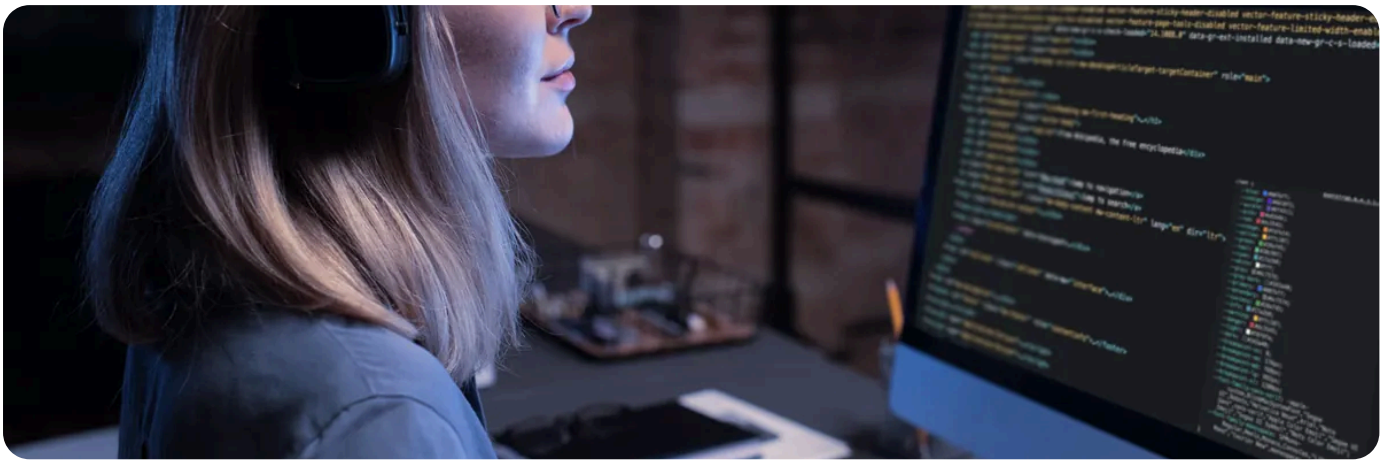
CODE	VERSION	COUNTRY · LANGUAGE
RLX-MZ-KB-504	v2026.10.21	Mozambique · English

Interweb Mozambique
RLX Holdings, S.A. · Group RLX Holdings
Tax ID (NUIT) 401 191 607 · Legal entity no. (NUEL) 105 045 974 · Economic activity code (CAE) 62022

https://www.meu.interweb.co.mz/knowledgebase/504

Getting a Django application running

Node.js, Python & Docker Created on **September 27, 2026** Updated on **October 21, 2026** 4 min read



Django is not in the cPanel apps installer, and nothing is missing: that installer is for PHP applications. Django runs on Python, and Python has a tool of its own in cPanel, under **Software**: **Setup Python App**.

That page builds an isolated Python environment for your application, picks the interpreter version, and attaches the application to an address on your domain. On the day this article was written the versions on offer ran from **2.7** up to **3.13**, with every step of the 3 series in between.

> **Pick a recent Python.**
> The old ones stay in the list to keep applications alive that were never migrated, not to start anything new. A new Django belongs near the top of that list, because that is where the libraries it depends on are still being published.

Creating the application, step by step

1. Sign in to My Interweb at **https://www.meu.interweb.co.mz** and open the cPanel for the plan.
2. Under **Software**, open **Setup Python App** and click to create an application.
3. **Pick the Python version.** Once the application is built, changing version means reinstalling everything, so think twice now.
4. **Application root:** the directory the code will live in. It belongs **outside** the public web directory.
5. **Application URL:** the domain, or the domain plus a path, where the application will answer.

6. **Application startup file** and **Entry point:** the file that starts the application and the object inside it. In a Django project, that is the web application object Django generates for you.

7. Create it. The page then shows you, at the top, **the command that activates this environment** on the command line. Copy it: you will need it.

Getting the code and the libraries in

1. **Upload the project** to the directory you named, by FTP or File Manager: **FTP accounts and FileZilla**. Do not upload the virtual environment from your own machine: it is no use here.

2. **Put a requirements file at the project root**, listing libraries and versions. The selector page has a button that installs them for you, with no command line at all.

3. **Or install from the command line**, if you have SSH: run the activation command the page gave you first, and only then install. Install without activating and the libraries land in the wrong place while the application keeps insisting it cannot find them.

4. **Set the environment variables** on the same page. That is where the Django secret key and the database credentials belong, not written into the code.

Restart the application after every code change.
There is a button for it on the selector page. Unlike PHP, a Python application stays loaded in memory: until you restart, it keeps serving the old code, and you will swear the file never saved.

Database, static files and the rest

Topic	What to do
Database	Create it in cPanel like any other application: **creating databases and using phpMyAdmin**. Then point the Django settings at it through the environment variables.
Static files	In production, Django does not serve static files. Collect them with its own command into a directory inside the public area, and serve them from there.
Allowed hosts	Put your domain in the Django allowed hosts list. If you do not, it rejects every request and you get an error that explains nothing.
Debug mode	**Turn it off before going live.** Left on, any error shows the visitor your paths, your settings and sometimes your credentials.

Topic	What to do
Periodic jobs	Use cPanel scheduled tasks, calling the command inside the environment you created: **cron jobs**.

Processes that stay resident do not fit a shared plan.
The web application starts on demand and that works well; a permanent queue worker, an asynchronous task process or a websocket server hit the account process ceiling and get cut off. If your application needs that, its home is a server of its own: **VPS, Cloud or Dedicated**. On the ceilings, see **the limits of your account**.

For the wider picture, see **which technologies run here**. If your application is in another language, see **Ruby on Rails** or **Laravel**.

NEED A HAND?

Application created and the address errors out? Send us the domain and the directory and we will read the log with you.

Open a support ticket →

SEE ALSO

- Hosting plans and what each one includes →
- VPS Server →
- Support Policy →

interweb

Interweb Mozambique

RLX Holdings, S.A. · Group RLX Holdings

TAX ID (NUIT)	401 191 607
LEGAL ENTITY NO. (NUEL)	105 045 974
ECONOMIC ACTIVITY CODE (CAE)	62022
REGISTERED OFFICE	Bairro 7 de Setembro Vilankulo 1304 · CEP 0411-02, Inhambane, Mozambique
CONTACT	info@interweb.co.mz · +258 84 635 1111 · +258 86 635 1111 · www.interweb.co.mz
OPENING HOURS	Monday to Friday - 08 AM to 04 PM Saturday & Sunday - Closed

Online version of this document

RLX-MZ-KB-504 · v2026.10.21 · Generated on 22/10/2026 05:11