Django auf HostEurope cPanel

Deployment-Handbuch für Django-Webanwendungen
auf Shared Hosting mit Phusion Passenger WSGI

Kein SSH nötig

cPanel Web-UI

Schritt-für-Schritt

Vollständiger Quellcode

F.R.G Publishers · Granaria Foundation · Februar 2026

Dr.Jo's dive in::

Django auf HostEurope cPanel

Deployment-Handbuch für Django-Webanwendungen auf HostEurope Shared Hosting mit cPanel und Phusion Passenger WSGI.

Reihe	Dr.Jo's dive in::
Titel	Django auf HostEurope cPanel
Autor	Dr.Jo
Verlag	F.R.G Publishers
Hrsg.	Granaria Foundation
Ausgabe	1.0 — Februar 2026
Toolchain	LaTeX / Texifier
Lizenz	Intern / Persönlich

1

Kontext, Voraussetzungen & Architektur

1.1 Was dieses Handbuch abdeckt

Dieses Handbuch beschreibt den kompletten Deployment-Prozess einer Django-Webanwendung auf einem HostEurope Shared Hosting Paket mit cPanel. Der Fokus liegt auf einem Szenario, in dem **kein SSH-Zugang** verfügbar ist und alles über die cPanel-Weboberfläche sowie FTP-Upload erledigt wird.

Es enthält ein vollständiges Test-Projekt mit allen Dateien, eine Schritt-für-Schritt Anleitung für das Deployment, sowie ein umfassendes Troubleshooting-Kapitel für die häufigsten Fehler.

1.2 Voraussetzungen

Komponente	Details
Hosting	HostEurope WebHosting mit cPanel und Python-Support
cPanel Feature	„Setup Python App“ muss verfügbar sein
FTP-Client	ForkLift, FileZilla oder cPanel File Manager
Python	$\geq$ 3.9 (auf dem Server, durch cPanel bereitgestellt)
Django	$\geq$ 4.2, $<$ 5.1
Domain	Konfiguriert und auf das Hosting zeigend

1.3 Architektur-Überblick

HostEurope nutzt **Phusion Passenger** als Application Server für Python-Apps. Passenger wird über cPanels „Setup Python App“ Feature konfiguriert und sucht beim Start nach einer Datei namens `passenger_wsgi.py` im Application Root. Diese Datei muss ein WSGI-kompatibles `application`-Objekt exportieren.

```
Browser -> Apache/Nginx -> Passenger -> passenger_wsgi.py -> Django WSGI
    -> views.py -> Template -> Response
```

Tipp

Passenger erstellt automatisch ein Virtual Environment. Packages können über die cPanel-Oberfläche installiert werden — kein SSH nötig!

2

Verzeichnisbaum & Datei-Referenz

2.1 Verzeichnisbaum

```
hosteurope-django/
+-- passenger_wsgi.py      <- Einstiegspunkt fuer Passenger
+-- requirements.txt       <- Python Dependencies
+-- deploy.sh              <- Setup-Script (per SSH)
+-- .htaccess.example      <- Referenz (cPanel setzt das automatisch)
+-- static/                <- Gesammelte Static Files
+-- testproject/
    +-- manage.py
    +-- db.sqlite3          (wird bei migrate erstellt)
    +-- testproject/
    |   +-- __init__.py
    |   +-- settings.py
    |   +-- urls.py
    |   +-- wsgi.py
    +-- testapp/
        +-- __init__.py
        +-- views.py
        +-- urls.py
        +-- templates/testapp/
            +-- home.html
```

2.2 Datei-Referenz

Datei	Funktion	Kritisch?
`passenger_wsgi.py`	WSGI Entry Point für Passenger	Ja
`settings.py`	Django Konfiguration (DB, Hosts, Apps)	Ja
`requirements.txt`	pip Dependencies	Ja
`views.py`	Request Handler / Business Logic	Ja
`home.html`	Django Template	Nein
`deploy.sh`	Automatisiertes Setup (SSH)	Optional
`.htaccess.example`	Apache/Passenger Referenz	Nein

3

Vollständiger Quellcode mit Erklärungen

3.1 passenger_wsgi.py

Dies ist die wichtigste Datei für das Deployment. Passenger sucht beim Start nach dem `application`-Objekt in dieser Datei. Sie muss im Application Root liegen (dort wo cPanel die App anlegt).

```
"""
passenger_wsgi.py - Einstiegspunkt fuer Phusion Passenger (cPanel Python
    App).

Diese Datei muss im App-Root liegen (dort wo cPanel die App anlegt).
Passenger sucht nach dem Objekt 'application'.
"""
import os
import sys

# Projektverzeichnis zum Python-Path hinzufuegen
app_dir = os.path.dirname(os.path.abspath(__file__))
sys.path.insert(0, app_dir)
sys.path.insert(0, os.path.join(app_dir, "testproject"))

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "testproject.settings")

from django.core.wsgi import get_wsgi_application
application = get_wsgi_application()
```

Wichtig

Die Variable muss `application` heißen — nicht `app` oder `wsgi`. Passenger erwartet exakt diesen Namen!

3.2 settings.py

Die zentrale Konfigurationsdatei von Django. Für den Test sind `DEBUG` und `ALLOWED_HOSTS` bewusst offen gesetzt. Vor dem Produktivbetrieb müssen diese Werte geändert werden.

```
"""
Django settings for testproject - konfiguriert fuer HostEurope cPanel.
```

```
"""
import os
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent

# SECURITY WARNING: Fuer Produktion unbedingt aendern!
SECRET_KEY = "django-insecure-CHANGE-ME-vor-produktion-abc123xyz"

# WICHTIG: Fuer Produktion auf False setzen
DEBUG = True

# Domain(s) hier eintragen
ALLOWED_HOSTS = [
    "*",  # Fuer den Test - spaeter einschraenken
]

INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "testapp",
]

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",
]

ROOT_URLCONF = "testproject.urls"

TEMPLATES = [{
    "BACKEND": "django.template.backends.django.DjangoTemplates",
    "DIRS": [],
    "APP_DIRS": True,
    "OPTIONS": {
        "context_processors": [
            "django.template.context_processors.debug",
            "django.template.context_processors.request",
            "django.contrib.auth.context_processors.auth",
            "django.contrib.messages.context_processors.messages",
        ],
    },
}]

WSGI_APPLICATION = "testproject.wsgi.application"

# SQLite fuer den Test
DATABASES = {
    "default": {
```

```
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": BASE_DIR / "db.sqlite3",
    }
}

LANGUAGE_CODE = "de-de"
TIME_ZONE = "Europe/Berlin"
USE_I18N = True
USE_TZ = True

STATIC_URL = "/static/"
STATIC_ROOT = BASE_DIR.parent / "static"
DEFAULT_AUTO_FIELD = "django.db.models.BigAutoField"
```

3.3 urls.py (Projekt)

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path("admin/", admin.site.urls),
    path("", include("testapp.urls")),
]
```

3.4 views.py

Die Test-App zeigt eine Systeminfo-Seite zur Verifizierung des Deployments und bietet einen Health-Check Endpoint als JSON.

```
import sys
import os
import platform
from datetime import datetime

import django
from django.shortcuts import render


def home(request):
    """Startseite mit System-Info zur Deployment-Verifizierung."""
    context = {
        "python_version": sys.version,
        "django_version": django.get_version(),
        "platform": platform.platform(),
        "server_time": datetime.now().strftime("%d.%m.%Y %H:%M:%S"),
        "hostname": platform.node(),
        "working_dir": os.getcwd(),
        "debug_mode": os.environ.get("DJANGO_DEBUG", "nicht gesetzt"),
    }
    return render(request, "testapp/home.html", context)


def health(request):
    """Health-Check Endpoint (JSON)."""
```

```
    from django.http import JsonResponse
    return JsonResponse({
        "status": "ok",
        "timestamp": datetime.now().isoformat(),
        "django": django.get_version(),
        "python": sys.version.split()[0],
    })
```

3.5 urls.py (App)

```
from django.urls import path
from . import views

urlpatterns = [
    path("", views.home, name="home"),
    path("health/", views.health, name="health"),
]
```

3.6 home.html (Template)

Das HTML-Template zeigt eine moderne, dunkle Oberfläche mit einem Glassmorphism-Design. Es zeigt Python-Version, Django-Version, Server-Zeit, Plattform, Hostname und Working Directory an. Ein Link zum Health-Check Endpoint ist im Footer.

Das Template nutzt CSS Grid, `backdrop-filter`, ein lineares Gradient-Hintergrundbild und Monospace-Schrift für die Werte. Es ist vollständig responsiv und benötigt keine externen CSS-Frameworks.

3.7 wsgi.py & manage.py

wsgi.py

```
import os
from django.core.wsgi import get_wsgi_application

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "testproject.settings")
application = get_wsgi_application()
```

manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "testproject.
        settings")
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
```

```
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == "__main__":
    main()
```

3.8 requirements.txt

```
django>=4.2,<5.1
```

3.9 .htaccess.example

Referenz-Datei — cPanel setzt die `.htaccess` normalerweise automatisch:

```
# Passenger aktivieren (wird normalerweise von cPanel gesetzt)
PassengerEnabled On
PassengerAppRoot /home/DEINUSER/DEINAPP
PassengerPython /home/DEINUSER/virtualenv/DEINAPP/3.11/bin/python3
```

4

Schritt für Schritt zum Live-System

4.1 Schritt 1 — Python App in cPanel anlegen

Logge dich ins cPanel ein und suche nach **„Setup Python App“**:

#	Aktion	Wert
1	**Python Version**	3.11 (oder höchste verfügbare)
2	**Application Root**	`django-test`
3	**Application URL**	Deine Domain/Subdomain
4	**Startup File**	`passenger_wsgi.py`
5	**Entry Point**	`application`
6	**Environment Variables**	Leer lassen (für Test)
7	**Klick**	„Create“ / „Erstellen“

Wichtig

Notiere dir den angezeigten Pfad zum Virtual Environment! Er sieht so aus:
`/home/DEINUSER/virtualenv/django-test/3.11/bin/activate`

4.2 Schritt 2 — Dateien hochladen (FTP)

Verbinde dich per FTP-Client (ForkLift, FileZilla) mit dem Server. Die FTP-Zugangsdaten findest du im cPanel unter „FTP-Konten“ oder in deiner HostEurope-Verwaltung.

Lade alle Projektdateien in das Verzeichnis `django-test/` hoch. Achte darauf, dass `passenger_wsgi.py` direkt im Root des App-Verzeichnisses liegt — nicht in einem Unterordner.

Tipp

ForkLift-Tipp: Nutze den Sync-Modus für schnellere Uploads. Drag & Drop vom lokalen Ordner direkt in das Server-Verzeichnis.

4.3 Schritt 3 — Django installieren

In der cPanel Python App Verwaltung (nach dem Erstellen der App) findest du einen Abschnitt für **Packages/Module**:

- Klicke auf **„Add“** und trage `django` ein

- Oder: Lade `requirements.txt` hoch über **„Run pip install“**

4.4 Schritt 4 — App starten & testen

Klicke in der Python App Übersicht auf **„Restart“**. Dann öffne im Browser:

Endpoint	URL	Erwartung
Startseite	`https://deinedomain.de/`	Blaue Seite mit Systeminfos
Health-Check	`https://deinedomain.de/health/`	JSON: `{ßtatus: ök", ...}`

5

Alles über die Weboberfläche erledigen

5.1 Setup Python App Feature

Das „Setup Python App“ Feature in cPanel ist der zentrale Dreh- und Angelpunkt für Python-Deployments ohne SSH. Es übernimmt automatisch:

- Erstellung eines Virtual Environments
- Konfiguration von Phusion Passenger
- pip Package-Installation via Weboberfläche
- `.htaccess` Konfiguration
- App Start/Stop/Restart

5.2 Package-Installation via Web-UI

Nach dem Erstellen der Python App zeigt cPanel einen Bereich „Enter to the virtual environment“ mit einem Befehl zum Kopieren. Darunter findet sich der Abschnitt für Module/Packages:

- **Einzelnes Package:** Auf „Add“ klicken, Name eingeben (z. B. `django`)
- **Aus requirements.txt:** Datei hochladen und „Run pip install“ klicken

5.3 Environment Variables

Im cPanel Python App Interface können Umgebungsvariablen gesetzt werden. Für den Test-Betrieb können diese leer bleiben, da alle Werte in `settings.py` hardcodiert sind.

Variable	Test	Produktion
`DJANGO_SECRET_KEY`	(in settings.py)	Sicherer Schlüssel
`DJANGO_DEBUG`	(nicht gesetzt)	`False`
`DATABASE_URL`	(nicht nötig)	`mysql://user:pass@host/db`
`ALLOWED_HOSTS`	(* in settings.py)	`deinedomain.de`

6

Häufige Fehler und ihre Lösungen

6.1 500 Internal Server Error

Der häufigste Fehler nach dem ersten Upload. **Ursache:** Die Python App wurde noch nicht in cPanel konfiguriert, das Virtual Environment existiert nicht, oder Django ist nicht installiert.

Lösung:

- Prüfe, ob die Python App in cPanel angelegt ist
- Prüfe, ob Django als Package installiert wurde
- Klicke „Restart“ in der Python App Verwaltung
- Prüfe die Error-Logs: cPanel → „Errors“ / „Fehlerprotokoll“

Falls SSH verfügbar:

```
cat ~/django-test/tmp/passenger.log
```

6.2 Module not found

Django oder ein anderes Modul wird nicht gefunden. Das Virtual Environment ist nicht korrekt aktiviert oder das Package fehlt.

Falls SSH verfügbar:

```
# Pruefe ob das venv aktiv ist:
which python
# Sollte /home/DEINUSER/virtualenv/... zeigen

# Django nochmal installieren:
pip install django
```

Ohne SSH: Im cPanel Python App Interface prüfen, ob Django unter „Packages“ gelistet ist.

6.3 Static Files fehlen

CSS/JS werden nicht geladen. `collectstatic` muss ausgeführt werden.

Falls SSH verfügbar:

```
cd ~/django-test/testproject
python manage.py collectstatic --noinput
```

Ohne SSH: Für die Test-App sind alle Styles inline im Template — Static Files werden erst bei Produktions-Apps relevant.

6.4 passenger_wsgi.py Pfad-Probleme

Falls Django nicht gefunden wird, obwohl es installiert ist, stimmen die `sys.path` Einträge in `passenger_wsgi.py` nicht:

```
# Pfade anpassen auf die tatsaechlichen Server-Pfade:
sys.path.insert(0, "/home/DEINUSER/django-test")
sys.path.insert(0, "/home/DEINUSER/django-test/testproject")
```

Tipp

App nach jeder Änderung neu starten: cPanel → Python App → „Restart“ oder `touch ~/django-test/tmp/restart.txt`

7

Vom Test zum Live-Betrieb

7.1 Security Hardening

Vor dem Produktivbetrieb unbedingt anpassen:

Einstellung	Test-Wert	Produktions-Wert
DEBUG	True	False
SECRET_KEY	django-insecure-...	Kryptographisch sicherer Key
ALLOWED_HOSTS	[*"]	["deinedomain.de", "www.deinedomain.de"]

SECRET_KEY generieren (Python):

```
from django.core.management.utils import get_random_secret_key
print(get_random_secret_key())
```

7.2 Datenbank-Migration

SQLite ist für Tests geeignet, für Produktion empfiehlt sich MySQL (in cPanel verfügbar). Datenbank in cPanel anlegen, dann `settings.py` anpassen:

```
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.mysql",
        "NAME": "deinuser_dbname",
        "USER": "deinuser_dbuser",
        "PASSWORD": "sicheres-passwort",
        "HOST": "localhost",
        "PORT": "3306",
    }
}
```

Dazu `mysqlclient` zu `requirements.txt` hinzufügen.

7.3 Admin-Interface

Django Admin aktivieren (SSH erforderlich):

```
cd ~/django-test/testproject
python manage.py createsuperuser
```

Danach erreichbar unter `https://deinedomain.de/admin/`

A

Automatisiertes Setup-Script (SSH)

Dieses Script kann per SSH auf dem Server ausgeführt werden, **nachdem** die Python App im cPanel angelegt wurde. Es automatisiert pip install, Django Migrations, collectstatic und Passenger Restart.

```
#!/bin/bash
# ============================================================
# deploy.sh - Django Setup auf HostEurope cPanel
# ============================================================
# Dieses Script per SSH auf dem Server ausfuehren, NACHDEM
# die Python App im cPanel angelegt wurde.
# Usage:  bash deploy.sh
# ============================================================

set -e

echo "=========================================="
echo "  Django Deployment - HostEurope cPanel"
echo "=========================================="

# --- 1. Virtual Environment aktivieren ---
VENV_PATH="$HOME/virtualenv/$(basename $PWD)/3.11/bin/activate"

if [ -f "$VENV_PATH" ]; then
    echo "[1/5] Virtual Environment gefunden: $VENV_PATH"
    source "$VENV_PATH"
else
    echo "[!] Konnte venv nicht finden unter: $VENV_PATH"
    exit 1
fi

# --- 2. Dependencies installieren ---
echo "[2/5] Installiere Dependencies..."
pip install --upgrade pip
pip install -r requirements.txt

# --- 3. Django Setup ---
echo "[3/5] Django Migrations..."
cd testproject
python manage.py migrate --run-syncdb

echo "[4/5] Static Files sammeln..."
python manage.py collectstatic --noinput
```

```
cd ..

# --- 4. Passenger neu starten ---
echo "[5/5] Passenger Restart..."
mkdir -p tmp
touch tmp/restart.txt

echo ""
echo "  Deployment abgeschlossen!"
echo "  Teste: https://deinedomain.de/"
echo "         https://deinedomain.de/health/"
```