

Docker with Python and Caddy

Derafu Docker with Python and Caddy for Fabric

Anonymous · 22/09/2026 · #docker, #python

Derafu Docker with Python and Caddy for Fabric

github [derafu/docker-python3.14-caddy-server](#)

last commit [september](#)

code size [33.1 KiB](#)

open issues [0](#)

A modern Docker setup for hosting Python websites with Caddy web server and SSH access for [Fabric](#) deployments — the Python counterpart of [Docker with PHP and Caddy](#).

Features

- **Python 3.14:** Supported Python version with common extensions.
- **Caddy:** Modern web server with automatic HTTPS.
- **SSH Access:** For automated deployments with Fabric.
- **Automatic Site Discovery:** Just add your site folder and it works.
- **Development Domains:** Test with `.local` domains that map to production folders.
- **Automatic WWW Redirection:** For second-level domains (e.g., `example.com` → `www.example.com`).
- **Auto-HTTPS:** Certificates are automatically generated on-demand.
- **Environment Separation:** Development and production environments managed through Docker Compose override.
- **Optional PHP embed + phpy:** build PHP with `--enable-embed` and install [phpy](#) so Python code running in this container can host and call PHP libraries directly (e.g. [Backbone Bridge Python](#)). Disabled by default — see [PHP Embed + phpy](#) below.

Quick Start

Prerequisites

- Docker and Docker Compose installed on your system.
- SSH key for deployment access.

Setup

1. Clone this repository:

```
git clone https://github.com/derafu/docker-python3.14-caddy-server.git
cd docker-python3.14-caddy-server
```

2. Add your SSH public key to `config/ssh/authorized_keys` for admin, and default deployment, access:

```
cat ~/.ssh/id_rsa.pub > config/ssh/authorized_keys
```

3. Build and start the container:

```
docker-compose up -d
```

Testing Your First Site

1. Create the site directory structure and a Django project:

```
mkdir -p sites/www.example.com/
cd sites/www.example.com/
python3 -m venv venv
source venv/bin/activate
pip install django
django-admin startproject example .
```

2. Run it:

```
/scripts/start_sites.sh www.example.com
```

If no site is specified, the script starts every available site under `/var/www/sites`.

3. Access it at `https://www.example.com` (production, requires DNS) or `https://www.example.com.local:8443` (development, requires a local `/etc/hosts` entry).

PHP Embed + `phpy` (Optional)

This image can optionally build PHP from source with `--enable-embed` and install `phpy`, so Python code can load and call a PHP library directly in the same process — the Python-hosts-PHP direction; the opposite direction, PHP-hosts-Python, is what [Docker with PHP and Caddy](#) provides.

This is **disabled by default**: no regular PHP package (apt, Homebrew, official Docker images) ships

with `--enable-embed`, so getting it requires compiling PHP from source, which adds several minutes to the image build. Enable it explicitly when you actually need it:

```
# In your .env file:
PHPY_ENABLED=true
PHPY_PHP_VERSION=8.5.3    # Optional, defaults to 8.5.3.

docker compose build
docker compose up -d
```

When enabled, the embed-enabled PHP build lives at `/opt/php` (`php`, `php-config`, `phpize` on `PATH`), and `phpy` is installed into the container's Python. Verify it works with:

```
docker compose exec webserver python3 -c "import phpy; print(phpy)"
```

Building with the default `PHPY_ENABLED=false` skips all of this and behaves exactly like a normal Python + Caddy image.

``phpy`` needs a matching `virtualenv`

Since `phpy` is installed into the container's own Python, not into any particular `virtualenv`, create yours with `python3 -m venv --system-site-packages` so it can still see it. See [Backbone Bridge Python](#) for the Python-side package that builds on top of this.

Development vs Production Environment

Docker Compose's override mechanism separates the two:

- **Production** (`docker-compose.yml` alone): minimal configuration, required environment variables, essential ports (HTTP, HTTPS, SSH), no external volumes.
- **Development** (`docker-compose.override.yml`, merged automatically by `docker-compose up -d`): additional development ports and local volume mounts for live editing.

To run production-only:

```
docker-compose -f docker-compose.yml up -d
```

Access and Management

```
# SSH access.
ssh admin@localhost -p 2222

# Direct container shell.
```

```
docker exec -it derafu-sites-server-python-caddy bash

# Restart Caddy.
docker exec -it derafu-sites-server-python-caddy supervisorctl restart caddy

# Stop the container.
docker-compose down
```

Last updated on 22/09/2026