Installation

CTFd is a standard WSGI application so most if not all Flask documentation on deploying a Flask based application should apply. This page will focus on the recommended ways to deploy CTFd.

TIP

The easiest way to get started is to use a hosted CTFd instance.

Server Requirements

At a minimum you should have a dual core CPU with at least 1 GB of RAM.

A generally recommended setup would have a quad core CPU with at least 2 GB of RAM.

Docker

CTFd provides automatically generated Docker images and one of the simplest means of deploying a CTFd instance is to use Docker Compose.

CAUTION

While Docker can be a very simple means of deploying CTFd, it can make debugging and receiving support more complicated. Before deploying using Docker, be sure you have a cursory understanding of how Docker works.

1. Install Docker
2. Install Docker Compose

3. Clone the CTFd repository with `git clone https://github.com/CTFd/CTFd.git`
4. Modify the `docker-compose.yml` file from the repository to specify a `SECRET_KEY` environment for the CTFd service. :

   ```
   environment:
     - SECRET_KEY=<SPECIFY_RANDOM_VALUE>
     - UPLOAD_FOLDER=/var/uploads
     - LOG_FOLDER=/var/log/CTFd
     - DATABASE_URL=mysql+pymysql://root:ctfd@db/ctfd
     - REDIS_URL=redis://cache:6379
     - WORKERS=4
   ```

TIP

You can also run `head -c 64 /dev/urandom > .ctfd_secret_key` within the CTFd repo to generate a .ctfd_secret_key file.

5. Run `docker-compose up`
6. You should now be able to access CTFd at http://localhost:8000

CAUTION

The default Docker Compose configuration files do not configure SSL/TLS. This is left as an exercise to the reader by design.

Standard WSGI Deployment

As a web application, CTFd has a few dependencies which require installation.

1. WSGI server
2. Database server
3. Caching server

WSGI Server

While CTFd is a standard WSGI application and most WSGI servers (e.g. gunicorn, UWSGI, waitress) will likely suffice, CTFd is most commonly deployed with gunicorn. This is because of its simplicity and reasonable performance. It is installed by default and used by other deployment methods discussed on this page. By default it is configured to use gevent as a worker class.

Database Server

CTFd makes use of SQLAlchemy and as such supports a number of SQL databases. The recommended database type is MySQL. CTFd is tested with and has been installed against SQLite, Postgres, and MariaDB.

By default CTFd will create a SQLite database if no database server has been configured.

INFO

CTFd makes use of the JSON data type which your database backend must support.

INFO

CTFd is typicaly used with MySQL, MariaDB, or SQLite. Using CTFd with Postgres is uncommon and could be deprecated in any version. Even though Postgres may be part of the CTFd test suite, using Postgres as your database backend is at your own peril.

Any issues raised regarding Postgres support will carry the expectation that the issue author write an accompanying pull request to resolve said issue.

Caching Server

CTFd makes heavy use of caching servers to store configuration values, user sessions, and page content. It is important to deploy CTFd with a caching server. The preferred caching server option is Redis.

By default if no cache server is configured, CTFd will attempt to use the filesystem as a cache and store values in the `.data` folder. This type of caching is not very performant thus it is highly recommended that you configure a Redis server.

Vagrant

CTFd provides a basic Vagrantfile for use with Vagrant. To run using Vagrant run the following commands:

```
vagrant up
```

Visit http://localhost:8000 where CTFd will be running.

To access the internal gunicorn terminal session inside Vagrant run:

```
vagrant ssh
tmux attach ctfd
```

> **CAUTION**
>
> CTFd's Vagrantfile is not commonly used and is only community supported

Debug Server

The absolute simplest way to deploy CTFd merely involves running python serve.py to start Flask's built-in debugging server. This isn't recommended for anything but debugging and should not be used for any kind of load. It is discussed here because the debugging server can make identifying bugs and misconfigurations easier. In addition, development mostly occurs using the debug server.

> **CAUTION**
>
> CTFd makes every effort to be an easy to setup application. However, deploying CTFd for large amounts of users can be difficult.
>
> Fully managed and maintained CTFd deployments are available at https://ctfd.io. If you're interested in a specialized CTFd deployment with custom features please contact us.

Previous
« Overview

Next
Updating »

Was this page helpful? 👍 👎

Share your feedback