Flask with Gunicorn

Contents

Overview

Gunicorn is a type of Web Server Gateway Interface (WSGI) HTTP server.

WSGI

There are many different frameworks that are available for Python to develop web applications, because of this compatibility issues arise.
WSGI provides a specification (PEP 3333) to adhere to, making any servers that use it compatible with any frameworks that also use it.

Pre-fork Worker Model

Gunicorn uses a pre-fork worker model which is a master process which controls 1 or more worker processes:

- The master process knows nothing about the client connections to the worker processes.
- Process signals are sent between the workers & master to determine whether a new worker needs to be booted.
- If a worker process crashes from a fatal exception, then another worker can be booted to replace it.

Installation

Gunicorn can be installed by using `pip`:

```
pip install gunicorn
```

Usage

You can start an application by running the command below, this would work for an application written in a file called `app.py` with the application set to an `app` variable:

```
# gunicorn [APP_MODULE]:[VARIABLE_NAME]
gunicorn app:app
```

- `APP_MODULE`
  This is will be the module (Python file) that the application is in.
- `VARIABLE_NAME`
  By default, Gunicorn will look for a variable called `application`, if yours is defined as something else then it may be specified here.

Workers

The amount of workers can be specified with the `-w` or `--workers` options.
The Gunicorn documentation recommends that you use 2-4 workers per core the server you are using has:

```
# gunicorn --workers [NUMBER]  app:app
gunicorn --workers 4 app:app
```

Specify Server Socket to Bind to

You may provide a server socket in the form of `[HOST]` or `[HOST]:[PORT]` with the `-b` or `--bind` options.
To allow traffic from anywhere on port `8001` you can use the following:

```
# gunicorn --bind=[HOST]:[PORT] app:app
gunicorn --bind=0.0.0.0:8001 app:app
```

Working Directory

Gunicorn can operate in a specified directory using the `--chdir` option, this would typically be the application root:

```
# gunicorn --chdir=[DIRECTORY] app:app
gunicorn --chdir=/opt/project app:app
```

Tutorial

Run a Simple Flask Application with Gunicorn

Prerequisites

- Linux Operating System (Debian 9/Ubuntu 18)
- Python 3
- `virtualenv`

These prerequisites can be setup using the following commands:

```
sudo apt update
sudo apt install -y python3 virtualenv
```

Create an Application Folder

Create a folder for the test application and change to it

```
mkdir -p gunicorn-test && cd $_
```

Create the Application

This is going to be a very simple application which uses the Flask framework.
Enter the following into a file called `app.py` and save it:

```
#! /usr/bin/env python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello from Flask App\n"

if __name__ == '__main__':
    app.run()
```

Create a Virtual Environment and Install the Dependencies

We will be needing the `flask` and `gunicorn` dependencies for this.
A virtual environment can be created and loaded:

```
virtualenv -p python3 venv
source ./venv/bin/activate
```

And then the required dependencies can be installed in the aforementioned virtual environment:

```
pip install flask gunicorn
```

Run the Application

The application can now be started using a gunicorn command:

```
gunicorn --workers=4 --bind=0.0.0.0:5000 app:app
```

Something similar to this should be outputted:

```
[2019-12-04 15:58:37 +0000] [3006] [INFO] Starting gunicorn 20.0.4
[2019-12-04 15:58:37 +0000] [3006] [INFO] Listening at: http://0.0.0.0:5000
(3006)
[2019-12-04 15:58:37 +0000] [3006] [INFO] Using worker: sync
[2019-12-04 15:58:37 +0000] [3009] [INFO] Booting worker with pid: 3009
[2019-12-04 15:58:37 +0000] [3010] [INFO] Booting worker with pid: 3010
[2019-12-04 15:58:37 +0000] [3011] [INFO] Booting worker with pid: 3011
[2019-12-04 15:58:37 +0000] [3012] [INFO] Booting worker with pid: 3012
```

Exercises

Try this on Another Project

Use another Python Flask application and run it using Gunicorn just like in the tutorial above.
If you don't have a Flask app available to you then feel free to use this provided example.