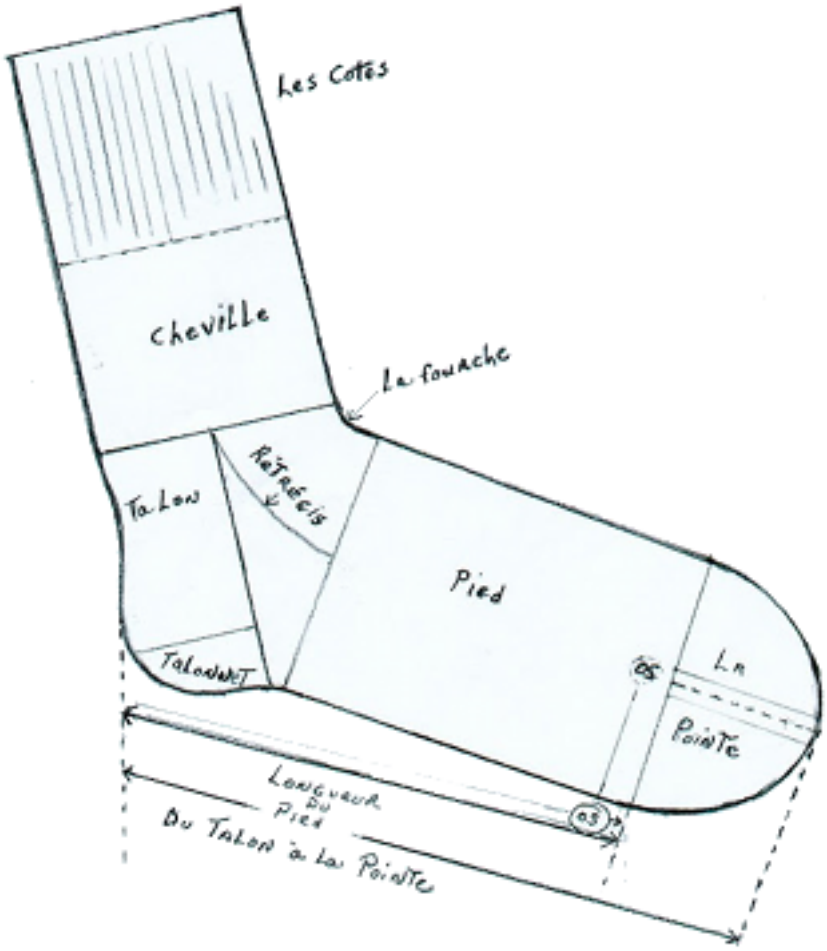
Chaussette Documentation

Release 1.1

Tarek Ziade

August 14, 2014

Contents



Chaussette is a WSGI server you can use to run your Python WSGI applications.

The particularity of **Chaussette** is that it can either bind a socket on a port like any other server does **or** run against **already opened sockets**.

That makes **Chaussette** the best companion to run a WSGI or Django stack under a process and socket manager, such as Circus or Supervisor.

CHAPTER 1

Usage

You can run a plain WSGI application, a Django application, or a Paste application. To get all options, just run *chaussette –help*.

1.1 Running a plain WSGI application

Chaussette provides a console script you can launch against a WSGI application, like any WSGI server out there:

```
$ chaussette mypackage.myapp
Application is <function myapp at 0x104d97668>
Serving on localhost:8080
Using <class chaussette.backend._wsgiref.ChaussetteServer at 0x104e58d50> as a backend
```

1.2 Running a Django application

Chaussette allows you to run a Django project. You just need to provide the Python import path of the WSGI application, commonly located in the Django project's `wsgi.py` file. For further information about how the `wsgi.py` file should look like see the Django documentation.

Here's an example:

```
$ chaussette --backend gevent mysite.wsgi.application
Application is <django.core.handlers.wsgi.WSGIHandler object at 0x10ec3f350>
Serving on localhost:8080
Using <class 'chaussette.backend._gevent.Server'> as a backend
```

1.3 Running a Python Paste application

Chaussette will let you run a project based on a Python Paste configuration file.

You just need to use to provide the path to the configuration file in the **application**, prefixed with **paste:**

Here's an example:

```
$ chaussette paste:path/to/configuration.ini
$ Application is <mozsvc.middlewares.CatchErrorMiddleware object at 0x10d4fdad0>
$ Serving on localhost:8080
$ Using <class chaussette.backend._wsgiref.ChaussetteServer at 0x10cc7e668> as a backend
```

CHAPTER 2

Using Chaussette in Circus

The typical use case is to run Chaussette processes under a process and socket manager. Chaussette was developed to run under Circus, which takes care of binding the socket and spawning Chaussette processes.

To run your WSGI application using Circus, define a *socket* section in your configuration file, then add a Chaussette watcher.

Minimal example:

```
[circus]
endpoint = tcp://127.0.0.1:5555
pubsub_endpoint = tcp://127.0.0.1:5556
stats_endpoint = tcp://127.0.0.1:5557

[watcher:web]
cmd = chaussette --fd $(circus.sockets.web) --backend meinheld server.app
use_sockets = True
numprocesses = 5

[socket:web]
host = 0.0.0.0
port = 8000
```

When Circus runs, it binds a socket on the *8000* port and passes the file descriptor value to the Chaussette process, by replacing *${socket:web}* by the file number value.

CHAPTER 3

Using Chaussette in Supervisor

Supervisor includes a socket manager since version 3.0a7, released in 2009. It was originally developed to support FastCGI processes and thus the configuration section is called *fcgi-program*. Despite the name, it is not tied to the FastCGI protocol. Supervisor can bind the socket and then spawn Chaussette processes.

To run your WSGI application using Supervisor, define an *fcgi-program* section in your configuration file.

Minimal example:

```
[supervisord]
logfile = /tmp/supervisord.log

[inet_http_server]
port = 127.0.0.1:9001

[supervisorctl]
serverurl = http://127.0.0.1:9001

[rpcinterface:supervisor]
supervisor.rpcinterface_factory = supervisor.rpcinterface:make_main_rpcinterface

[fcgi-program:web]
command = chaussette --fd 0 --backend meinheld server.app
process_name = %(program_name)s_%(process_num)s
numprocs = 5
socket = tcp://0.0.0.0:8000
```

Notice the `--fd 0` argument to `chaussette`. Each *fcgi-program* section defines its own socket and the file descriptor is always `0`. See the Supervisor manual for detailed information.

Supervisor will create the socket before spawning the first Chaussette child process. When the last child exits, Supervisor will close the socket.

CHAPTER 4

Backends

Chaussette is just a bit of glue code on the top of existing WSGI servers, and is organized around **back ends**.

By default Chaussette uses a pure Python implementation based on **wsgiref**, but it also provides more efficient back ends. Most of them are for Python 2 only, but Chaussette can be used under Python 3 with a few of them - marked in the list below:

- **gevent** – based on Gevent's *pywsgi* server
- **fastgevent** – based on Gevent's *wsgi* server – faster but does not support streaming.
- **meinheld** – based on Meinheld's fast C server
- **waitress** – based on Pyramid's waitress pure Python web server (py3)
- **eventlet** – based on Eventlet's wsgi server
- **geventwebsocket** – Gevent's **pywsgi** server coupled with **geventwebsocket** handler.
- **geventws4py** – Gevent's **pywsgi** server coupled with **ws4py** handler.
- **socketio** – based on gevent-socketio, which is a custom Gevent server & handler that manages the socketio protocol.

You can select your backend by using the **–backend** option and providing its name.

For some backends, you need to make sure the corresponding libraries are installed:

- **gevent** and **fastgevent**: *pip install gevent*
- **meinheld** : *pip install meinheld*
- **waitress** : *pip install waitress*
- **eventlet** : *pip install eventlet*
- **geventwebsocket**: *pip install gevent-websocket*
- **geventws4py**: *pip install ws4py*
- **socketio**: *pip install gevent-socketio*

If you want to add your favorite WSGI Server as a backend to Chaussette, or if you think you can make one of the backend Python 3 compatible, send me an e-mail !

If you curious about how each on of those backends performs, you can read:

- http://blog.ziade.org/2012/06/28/wgsi-web-servers-bench/
- http://blog.ziade.org/2012/07/03/wsgi-web-servers-bench-part-2/

CHAPTER 5

Rationale and Design

Most WGSI servers out there provide advanced features to scale your web applications, like multi-threading or multi-processing. Depending on the project, the *process management* features, like respawning processes that die, or adding new ones on the fly, are not always very advanced.

On the other hand, tools like Circus and Supervisor have more advanced features to manage your processes, and are able to manage sockets as well.

The goal of *Chaussette* is to delegate process and socket management to its parent process and just focus on serving requests.

Using a pre-fork model, the process manager binds a socket. It then forks Chaussette child processes that accept connections on that socket.

For more information about this design, read :

- http://blog.ziade.org/2012/06/12/shared-sockets-in-circus.
- http://circus.readthedocs.org/en/latest/sockets/

CHAPTER 6

Useful links

- Repository : https://github.com/mozilla-services/chaussette