Gunicorn Python app server

CONCEPTS

app:app means module:variable. Gunicorn serves one app per process; put nginx in front for TLS and static files.

Client browser
↓ HTTP
nginx TLS, static
↓ proxy_pass
Master gunicorn, no requests
↓ fork
Workers run your app

RUN

gunicorn *<module>*:*<app>*
Serve a WSGI app

gunicorn '*<module>*:*<factory>*()'
Call an app factory

gunicorn -w *<n>* *<module>*:*<app>*
Set worker count

gunicorn -b *<host>*:*<port>* *<module>*:*<app>*
Bind a host and port

gunicorn -b unix:*<path>* *<module>*:*<app>*
Bind a Unix socket

gunicorn *<project>*.**wsgi**
Run a Django project

gunicorn --reload *<module>*:*<app>*
Auto-restart on code change

gunicorn -v
Print the version

WORKERS

-k sync
Default: one request per worker

-k gthread --threads *<n>*
Threaded workers

-k gevent
Async workers via gevent

WORKERS (cont.)

-k asgi
Run an ASGI app

--max-requests *<n>*
Recycle a worker after n requests

-t *<sec>* **/ --timeout** *<sec>*
Kill silent workers after n seconds

--preload
Load app before forking

Pick a worker

Class	Fits
`sync`	simple CPU-bound apps
`gthread`	mixed or blocking I/O
`gevent`	many slow connections
`asgi`	async frameworks

LOGGING

--access-logfile -
Access log to stdout

--error-logfile -
Error log to stderr

--log-level *<level>*
debug, info, warning, error

--capture-output
Send app stdout/stderr to the log

SIGNALS

kill -HUP *<master-pid>*
Reload config, replace workers

kill -TERM *<master-pid>*
Graceful shutdown

CONFIG FILE

gunicorn.conf.py

```
bind = "127.0.0.1:8000"
workers = 4
worker_class = "gthread"
threads = 2
timeout = 30
accesslog = "-"
errorlog = "-"
preload_app = True
max_requests = 1000
```