# Uvicorn ASGI web server

**START HERE**

**`uvicorn main:app`**
Serve an ASGI app

**`uvicorn main:app --reload`**
Restart on code change

**`uvicorn main:app --host 0.0.0.0`**
Listen on all interfaces

**`uvicorn main:app --port 8080`**
Pick the port

**`uvicorn main:app --workers 4`**
Four worker processes

**`uvicorn --factory main:create_app`**
Call an app factory

**`uvicorn main:app --log-level debug`**
More logging

**`uvicorn main:app --proxy-headers`**
Trust X-Forwarded-*

**Client** browser → HTTPS → **nginx** TLS, static → unix socket → **Uvicorn** HTTP and ASGI → ASGI → **Your app** FastAPI etc.

## CONCEPTS

main:app means module:variable. Uvicorn serves ASGI apps (FastAPI, Starlette, Django ASGI) and listens on 127.0.0.1:8000 by default.

Defaults to 127.0.0.1, so a container needs --host 0.0.0.0 or it is unreachable from outside.

--reload and --workers cannot be combined. Use --reload for development only, workers or Gunicorn for production.

## RUN

**`uvicorn <module>:<app>`**
Serve an ASGI app

**`uvicorn --factory <module>:<func>`**
Treat app as a factory
*The function takes no arguments and returns the app.*

**`uvicorn <module>:<app> --reload`**
Auto-restart on file change
*Development only.*

**`uvicorn <module>:<app> --workers <n>`**
Run n worker processes
*Defaults to $WEB_CONCURRENCY, else 1.*

**`uvicorn --app-dir <dir> <module>:<app>`**
Look for the app in a directory

**`uvicorn --env-file <file> <module>:<app>`**
Load environment from a file

**`uvicorn --interface wsgi <module>:<app>`**
Serve a WSGI app

**`uvicorn --lifespan <auto|on|off>`**
Control startup/shutdown events

**`uvicorn --version`**
Print the version

**`uvicorn --help`**
List all options

**`python -m uvicorn <module>:<app>`**
Run via the interpreter
*Uses the interpreter's environment, handy in venvs.*

## BINDING

**`--host <addr>`**
Bind address (default 127.0.0.1)

**`--port <n>`**
Bind port (default 8000)
*0 picks a free port.*

**`--uds <path>`**
Bind a Unix socket

**`--fd <n>`**
Use an inherited socket descriptor
*For supervisors that hand over the socket.*

**`--backlog <n>`**
Connection backlog size

**`--root-path <prefix>`**
App mounted below a URL path

**Common bind choices**

| Goal | Flags |
|---|---|
| Local only | `(default)` |
| Container / LAN | `--host 0.0.0.0` |
| Behind nginx | `--uds /run/app.sock` |
| Any free port | `--port 0` |

## RELOAD

**`--reload`**
Watch files and restart

**`--reload-dir <dir>`**
Watch this directory only
*Repeatable. Stops it scanning venvs and node_modules.*

**`--reload-include <glob>`**
Also watch matching files
*Only has an effect when watchfiles is installed.*

**`--reload-exclude <glob>`**
Ignore matching files
*Only has an effect when watchfiles is installed.*

Install uvicorn[standard] to get watchfiles for faster, glob-aware reloading.

## WORKERS AND LIMITS

**`--workers <n>`**
Number of worker processes

**`--limit-concurrency <n>`**
Answer 503 above n connections

**`--limit-max-requests <n>`**
Restart a process after n requests

**`--limit-max-requests-jitter <n>`**
Stagger those restarts

## WORKERS AND LIMITS (cont.)

**`--timeout-keep-alive <sec>`**
Idle keep-alive timeout (default 5)
*Keep it above the proxy's upstream keep-alive time.*

**`--timeout-graceful-shutdown <sec>`**
Max wait on shutdown

**`kill -HUP <pid>`**
Gracefully restart workers
*Works with --workers, not on Windows.*

**`kill -TTIN <pid>` / `kill -TTOU <pid>`**
Add / remove one worker

Under --workers, Uvicorn spawns processes rather than forking, so it also works on Windows (signals do not).

## PROTOCOLS

**`--http <impl>`**
h11 or httptools

**`--loop <impl>`**
asyncio or uvloop

**`--ws <lib>`**
websockets, websockets-sansio, wsproto

**`--ws-max-size <bytes>`**
Largest WebSocket message

## LOGGING

**`--log-level <level>`**
critical, error, warning, info, debug, trace

**`--no-access-log`**
Silence request lines

**`--access-log`**
Enable request lines (default)

**`--use-colors / --no-use-colors`**
Force or disable coloured logs

**`--log-config <file>`**
Logging config (.ini, .json, .yaml)

## HTTPS AND PROXIES

**`--ssl-keyfile <key> --ssl-certfile <crt>`**
Serve HTTPS directly

**`--proxy-headers`**
Read X-Forwarded-For and -Proto
*On by default; only trusted IPs are honoured.*

## HTTPS AND PROXIES (cont.)

**`--no-proxy-headers`**
Ignore forwarded headers

**`--forwarded-allow-ips <ips>`**
Trusted proxy IPs or sockets
*Default 127.0.0.1,::1 (or $FORWARDED_ALLOW_IPS). '*' trusts everyone: only if your proxy strips client-sent headers.*

Behind nginx over a Unix socket, set X-Forwarded-For and X-Forwarded-Proto in the proxy and list the proxy in --forwarded-allow-ips.

## GUNICORN AND CODE

**Gunicorn with Uvicorn workers**

```
pip install uvicorn-worker
gunicorn main:app -w 4 -k uvicorn_worker.UvicornWorker
```

uvicorn.workers.UvicornWorker still exists but is deprecated; the docs recommend the separate uvicorn-worker package. Use UvicornH11Worker on PyPy.

**Run from Python**

```
import uvicorn
if __name__ == "__main__":
    uvicorn.run("main:app", host="0.0.0.0", port=8000, reload=True)
```

Pass an import string, not the app object, to uvicorn.run when using reload or workers.

## RECIPES

**Dev server**

```
uvicorn main:app --reload --reload-dir src \
  --log-level debug
```

**Container entrypoint**

```
uvicorn main:app --host 0.0.0.0 --port 8000 \
  --proxy-headers --forwarded-allow-ips '*' \
  --no-access-log
```

**Behind nginx on a socket**

```
uvicorn main:app --uds /run/app.sock --workers 4 \
  --proxy-headers
# nginx: proxy_pass http://unix:/run/app.sock;
```

**Local HTTPS**

```
mkcert localhost
uvicorn main:app --ssl-keyfile localhost-key.pem \
  --ssl-certfile localhost.pem
```