PM2 Node process manager

pm2 CLI your commands → talks to → **PM2 daemon** keeps apps alive → spawns → **Your apps** fork or cluster → write to → **Log files** ~/.pm2/logs

CONCEPTS

PM2 runs a background daemon that owns your apps: closing the terminal does not stop them.

The process list lives in the daemon's memory; pm2 save plus pm2 startup make it survive a reboot.

In a container use pm2-runtime instead of pm2: it stays in the foreground as PID 1.

START AND RUN

pm2 start *<file>*
Start an app as a daemonized process

pm2 start *<file>* **--name** *<name>*
Start under a custom name

pm2 start *<file>* **--** *<args>*
Pass arguments to the script

pm2 start *<file>* **--watch**
Restart when files change

pm2 start *<file>* **--watch --ignore-watch "node_modules"**
Watch but skip a path

START AND RUN (cont.)

pm2 start *<file>* **--max-memory-restart** *<size>*
Restart above a memory limit

pm2 serve *<path>* *<port>*
Serve a static directory

INSPECT AND LOGS

pm2 list
Table of all processes

pm2 logs
Stream logs of all apps

pm2 logs *<name>*
Stream logs of one app

pm2 monit
Live CPU, memory and logs

pm2 logs --lines *<n>*
Show the last n lines first

pm2 logs --err
Only error output

pm2 logs --nostream
Print recent logs and exit

pm2 describe *<name>*
All details of one process

pm2 flush
Empty all log files

CONTROL

pm2 restart *<name>*
Kill and start again

CONTROL (cont.)

pm2 stop *<name>*
Stop but keep in the list

pm2 delete *<name>*
Stop and remove from the list

pm2 reload *<name>*
Zero-downtime restart

pm2 restart all
Restart every app

pm2 stop all
Stop every app

pm2 delete all
Remove every app

pm2 restart *<name>* **--update-env**
Restart with the current shell env

CLUSTER MODE

pm2 start *<file>* **-i max**
One instance per CPU core

pm2 start *<file>* **-i** *<n>*
Run n load-balanced instances

pm2 scale *<name>* *<n>*
Set the instance count

restart or reload

Command	Behaviour
`pm2 restart`	hard stop, then start
`pm2 reload`	one instance at a time, cluster only

ECOSYSTEM FILE

pm2 ecosystem
Generate ecosystem.config.js

pm2 start ecosystem.config.js
Start all apps in the file

pm2 start ecosystem.config.js --env *<name>*
Use the env_<name> block

pm2 start ecosystem.config.js --only *<app>*
Act on one app of the file

pm2 startOrRestart ecosystem.config.js
Start new apps, restart running ones

pm2 startOrReload ecosystem.config.js
Same, but with a graceful reload

ecosystem.config.js

```
module.exports = {
  apps: [{
    name: "api",
    script: "./server.js",
    instances: "max",
    max_memory_restart: "300M",
    env: { NODE_ENV: "development" },
    env_production: { NODE_ENV: "production" }
  }]
};
```

BOOT AND UPKEEP

pm2 startup
Print the boot-hook command to run

pm2 save
Save the list for resurrect

pm2 resurrect
Restore the saved list

pm2 update
Reload the daemon after upgrading

pm2 install pm2-logrotate
Rotate log files

pm2 unstartup
Remove the boot hook

pm2 kill
Stop the daemon and all apps

RECIPES

Deploy and survive reboots

```
pm2 start ecosystem.config.js --env production
pm2 save
pm2 startup
# run the sudo command it prints
```

Release with no downtime

```
git pull && npm ci
pm2 reload <name> --update-env
pm2 logs <name> --lines 50
```