# Diagnose Too many open files on a Linux service

“Too many open files” can prevent a web server from accepting connections or an application from opening its next log, socket or data file. Files here include more than documents on disk. Start with the process that reported the error: changing your SSH shell’s limit may have no effect on an already running service.

# Find the affected process

1. Save the original error and its timestamp. Note whether it names a process, worker or service.
2. For a systemd-managed application, identify its main process:

```
systemctl show myapp.service -p MainPID
systemctl status myapp.service --no-pager
```

Replace `myapp.service` with the correct unit. A service may have several worker processes, and the main PID is not necessarily the one hitting the limit. Use the PID named by the error when possible. If the process has already restarted, its new limits and handle count cannot reconstruct the old failure by themselves.

# Compare usage with the process limit

Replace `1234` with the relevant running PID:

```
cat /proc/1234/limits
sudo find /proc/1234/fd -mindepth 1 -maxdepth 1 -printf "." | wc -c
```

Read the `Max open files` row. The soft limit is the effective boundary the process normally encounters; the hard limit constrains permitted increases. The descriptor-directory count is a

momentary observation, so collect another sample during the problem. Permission errors or a vanished PID mean the sample is incomplete, not that the process has no handles.

# Look for the growth pattern

```
sudo ls -l /proc/1234/fd
```

Inspect whether many entries point to sockets, pipes or repeated paths. Keep this output private. A single busy sample does not establish a leak; compare it with normal traffic and another sample after connections have closed. Check recent changes to connection pools, plugins, worker concurrency and log handling.

Distinguish a per-process limit from a system-wide file-table shortage using the exact error and kernel messages. `ulimit -n` in your current terminal describes that shell, not the service you are diagnosing. Do not raise a server-wide limit merely because an online example uses a large number.

# Make and verify a targeted change

If normal expected concurrency requires a higher service limit, plan a unit-specific change with the application maintainer and account for its restart. If descriptors keep growing without returning, investigate the leak first; a larger limit may only delay the next outage. After any approved change, check the new process’s own limits and observe descriptor use through a normal load cycle.

Send support the PID, service name, limit row, approximate handle counts over time and the exact error if ownership or the cause remains unclear. Avoid killing all workers or repeatedly restarting a database to clear the symptom.

# Related CloudExa guides

- Inspect the service and its logs
- Identify a listening application

# Technical references

- Linux resource limits

- Per-process limit reporting
- Per-process descriptor directory

---

Revision #1
Created 2026-11-10 10:51:41 UTC by CloudExa Hosting Admin
Updated 2026-11-10 10:51:41 UTC by CloudExa Hosting Admin