Goal: fast and stable WordPress on SpinupWP (Bricks + many plugins + large uploads)

> I use SpinupWP, and it allows you to configure the following settings. Please explain these settings to me and recommend what I should set: I'll be using BricksBuilder and a whole range of other plugins. I also want to upload large files and edit large pages (a lot of content on a single page). My goal is to achieve excellent performance and stability.
>
> Nginx
>
> - Client Max Body Size
> - Client Body Timeout
> - FastCGI Read Timeout
>
> PHP
>
> - Upload Max File Size
> - Post Max Size
> - Memory Limit
> - Max Execution Time
> - Max File Uploads
> - Max Input Vars
> - Max Input Time
> - Worker Management
>   - Static
>   - Max Workers
>   - Max Requests
>   - Dynamic
>   - Max Workers

    - Max Requests
    - Start Workers
    - Mind Idle Workers
    - Max Idle Workers
  - On-Demand
    - Max Workers
    - Max Requests
    - Idel Worker Timeout

Goal: fast *and* stable WordPress on SpinupWP (Bricks + many plugins + large uploads)

SpinupWP is essentially letting you tune two layers:

1. **Nginx** (the web server): controls request size limits and how long it will wait for data.
2. **PHP-FPM** (PHP workers): controls how much PHP can consume per request and how many requests can run concurrently.

Below is what each setting does and what I'd set as a strong default for a "performance + stability" Bricks/WordPress site that also needs **large uploads** and **large admin edits**. (You can tighten later once you know your real needs.)

Nginx settings

1) **Client Max Body Size**

- **What it is:** Maximum allowed size of an HTTP request body (uploads, big POST requests, some editor saves).
- **If too low:** Uploads fail with `413 Request Entity Too Large`.
- **Trade-off:** Higher values allow bigger uploads, but also increase abuse potential (someone can try to send huge bodies).

Recommendation

- Set to the **largest file you realistically upload**, with buffer.
  - Typical “safe but flexible”: **256M**
  - If you *truly* upload big media/backups via WP: **512M** (or **1G** if you must)
- If you’re using direct-to-object-storage uploads (S3, etc.), you can keep it lower.

---

2) **Client Body Timeout**

- **What it is:** How long Nginx will wait to receive the request body from the client (e.g., slow upload connection).
- **If too low:** Large uploads over slow connections fail mid-way (timeout).
- **Trade-off:** Too high can keep connections open longer (slightly more exposure to slowloris-style behavior, though Nginx is good at handling this).

Recommendation

- **120s–300s**
  - Start at **300s** if you regularly upload large files from typical home/office connections.

---

3) **FastCGI Read Timeout**

- **What it is:** How long Nginx will wait for a response from PHP (via PHP-FPM).
- **If too low:** Long operations (imports, image processing, complex saves) fail with `504 Gateway Timeout`.
- **Trade-off:** Too high can allow truly “stuck” PHP requests to tie up capacity longer.

Recommendation

- **120s–300s**
  - Start at **300s** if you do imports, big edits, or heavy plugins.
  - For normal sites with no long operations, you’d keep it closer to 60–120s.

PHP (runtime) settings

4) **Upload Max File Size (`upload_max_filesize`)**

- **What it is:** Max size for a *single uploaded file* handled by PHP.
- **If too low:** WP media uploads fail.

Recommendation

- Set based on your real uploads:
  - Common: **256M**
  - Heavy media workflows: **512M**
- Must be **≤ Post Max Size** (below).

---

5) **Post Max Size (`post_max_size`)**

- **What it is:** Max size of the entire POST request (can include multiple files + form fields + editor payload).
- **If too low:** Uploads or large editor saves can fail even if `upload_max_filesize` is high enough.

Recommendation

- Set to **a bit higher** than `upload_max_filesize`:
  - If upload is 256M → post max **300M–320M**
  - If upload is 512M → post max **600M–640M**

---

6) **Memory Limit (`memory_limit`)**

- **What it is:** Max memory PHP can use per request (front-end requests, WP admin, WP-Cron, AJAX, builder operations).
- **If too low:** White screens, fatal errors, “Allowed memory size exhausted”, builder instability.
- **Trade-off:** Higher per-request memory means fewer concurrent requests fit in RAM (important when combined with PHP worker count).

Recommendation (Bricks + plugins + big pages)

- Start at **512M**
- If you see memory-related fatals or heavy Woo/ACF/SEO stacks: **768M** (or **1024M** only if you truly need it and have RAM headroom)

> Stability tip: it’s usually better to run *fewer PHP workers with enough memory* than many workers that cause swapping/oom.

7) **Max Execution Time (`max_execution_time`)**

- **What it is:** Max CPU time a PHP script is allowed to run.
- **If too low:** Imports, backups, resizing, and some builder operations can fail.
- **Trade-off:** Too high lets inefficient tasks run longer and hold workers.

Recommendation

- **120s–300s**
  - Start at **300s** if you do imports/large operations.
  - If you don’t, **120s** is often plenty.

---

8) **Max File Uploads (`max_file_uploads`)**

- **What it is:** Max number of files that can be uploaded in one request.
- **If too low:** Multi-file upload UIs may fail or partially upload.

Recommendation

- **50** (safe default)
- If you never bulk upload huge batches: **20** is fine.

---

9) **Max Input Vars (`max_input_vars`)**

- **What it is:** Limits how many input fields PHP will accept (forms, page builder payloads, mega menus, complex admin screens).
- **If too low:** Settings don’t save completely; missing fields; builder content or meta boxes not persisting.
- **Trade-off:** Higher values slightly increase parsing overhead and can be abused (but typically fine behind WP auth).

Recommendation (large builder pages)

- Start at **10,000**
- If you have extremely large pages / many dynamic fields: **20,000**
- Avoid absurdly high numbers unless you need them.

---

10) **Max Input Time (`max_input_time`)**

- **What it is:** Time limit for PHP to parse input data (POST/GET/file uploads).
- **If too low:** Large POST payloads (big editor saves) can fail.

Recommendation

- **120s–300s**
  - Start at **300s** for your use case.

PHP-FPM Worker Management (this is the big performance/stability lever)

PHP-FPM runs a pool of PHP “workers.” Each concurrent uncached request generally needs a worker. More workers = more concurrency, but also more RAM usage. Too many workers = memory pressure → swapping → slow site → crashes.

Choosing the mode: **Static vs Dynamic vs On-Demand**

1. **Static**
   - Always keeps exactly `Max Workers` running.
   - Best for predictable, steady traffic and lowest latency.
   - Can waste RAM on quiet sites.
2. **Dynamic** *(usually best general-purpose choice)* □□
   - Keeps a baseline and scales up/down with demand.
   - Great balance of responsiveness and RAM efficiency.
3. **On-Demand**
   - Starts workers only when needed; kills them after idle timeout.
   - Efficient for very low traffic.
   - Slight latency spikes as workers spawn; can feel worse in WP admin if constantly cold-starting.

Recommendation

- Use **Dynamic** for “excellent performance + stability” on a typical production WP server.

Worker parameters (what they mean + what to set)

A) **Max Workers (`pm.max_children`)**

- **What it is:** Maximum number of PHP requests that can run at the same time.
- **If too low:** Requests queue → slow TTFB under load, admin feels laggy when multiple users.
- **If too high:** RAM exhaustion → swapping/oom → site instability.

How to size it (rule of thumb)

- Estimate average memory per PHP worker under your workload:
  - many WP sites: **80–200MB per worker**
  - heavy builders/admin ops can spike higher
- Then ensure: `max_workers × (avg worker RAM)` fits comfortably in available RAM **after** OS + Nginx + DB + caches.

Recommendation (practical starting points)

- If your server is **2GB RAM**: **4–6 workers**
- **4GB RAM**: **6–12 workers**
- **8GB RAM**: **12–24 workers**
- If you set `memory_limit=512M`, do **not** assume each worker uses 512M—but plan conservatively.

B) **Max Requests (`pm.max_requests`)**

- **What it is:** How many requests a worker handles before it’s recycled.
- **Why it matters:** Mitigates memory leaks/fragmentation from plugins/themes over time.
- **Trade-off:** Too low = more respawns (minor overhead).

Recommendation

- **500–1000**
  - Start at **500** if you run lots of plugins and want stability.

- Use **1000** if everything is stable and you want slightly fewer respawns.

Dynamic mode-specific

C) **Start Workers (`pm.start_servers`)**

- **What it is:** Workers spawned on PHP-FPM start/reload.
- **Recommendation**
- Set to about **25–50% of Max Workers**
  - Example: max 12 → start **4–6**

D) **Min Idle Workers (`pm.min_spare_servers`)**

- **What it is:** Minimum number of idle workers to keep ready.
- **Recommendation**
- About **15–30% of Max Workers**
  - Example: max 12 → min idle **2–4**

E) **Max Idle Workers (`pm.max_spare_servers`)**

- **What it is:** Maximum idle workers to keep around.
- **Recommendation**
- About **50–70% of Max Workers**
  - Example: max 12 → max idle **6–8**

On-Demand mode-specific

F) **Idle Worker Timeout (`pm.process_idle_timeout`)**

- **What it is:** How long an idle worker lives before it’s killed.
- **Recommendation**
- **10s–30s**
  - For WP admin comfort, lean **30s**
  - For ultra-low-traffic efficiency, **10s–15s**

Suggested “strong baseline” config (most Bricks sites)

Use this as a starting point, then adjust based on real usage. □□

Nginx

1. **Client Max Body Size: 512M**
2. **Client Body Timeout: 300s**
3. **FastCGI Read Timeout: 300s**

PHP

1. **Upload Max File Size: 512M**
2. **Post Max Size: 640M**
3. **Memory Limit: 512M** *(go 768M if you hit memory fatals)*
4. **Max Execution Time: 300**
5. **Max File Uploads: 50**
6. **Max Input Vars: 10,000** *(20,000 if large builder pages still don’t save fully)*
7. **Max Input Time: 300**

PHP-FPM (Dynamic)

- **Max Workers:** depends on RAM:
  - **4GB RAM → 10**
  - **8GB RAM → 18**
- **Max Requests: 500**
- **Start Workers:** ~40% of max
- **Min Idle Workers:** ~20% of max
- **Max Idle Workers:** ~60% of max

Concrete examples:

1. **If 4GB RAM**
    1. Max Workers: **10**
    2. Max Requests: **500**
    3. Start Workers: **4**
    4. Min Idle Workers: **2**
    5. Max Idle Workers: **6**
2. **If 8GB RAM**
    1. Max Workers: **18**
    2. Max Requests: **500**
    3. Start Workers: **6**
    4. Min Idle Workers: **3**
    5. Max Idle Workers: **10**

Two quick guardrails (important for “stability”)

1. **Don’t oversubscribe workers**
    - If you see swapping (disk activity high, load average high, site suddenly slow), reduce **Max Workers** first.
2. **Make sure limits agree**
    - `client_max_body_size` (Nginx) must be ≥ `post_max_size` (PHP) must be ≥ `upload_max_filesize` (PHP), otherwise you get confusing failures.

If you tell me these 4 details, I can give a precise worker count ⚙

1. Server RAM (and whether DB is on the same server)
2. CPU cores
3. Traffic level (avg + peak concurrent visitors)
4. Whether you run WooCommerce / membership / heavy search/filter plugins

I’ll then recommend an exact **Dynamic** worker configuration sized to your hardware and workload.

Revision #4
Created 2026-04-08 18:46:12 UTC by art10m

Updated 2026-04-08 19:34:34 UTC by art10m