

DevOps Health Check Report

Client application: “**BookFlow**” (booking web app) · Agency: “**Northwind Studio**”

This is a sample. The agency, the application and every number below are fictional. They were written to show the format, depth and tone of a real Health Check report. The environment (Laravel on a single VPS managed with Forge) is typical of the small agencies this service is built for.

Scope	1 production environment + 1 staging environment, 1 application
Stack	Laravel 11, PHP-FPM 8.3, NGINX, MySQL 8, Redis, Horizon queues · Laravel Forge · DigitalOcean · GitHub
Access used	Read-only: Forge (viewer), SSH user without sudo write, GitHub (read), DigitalOcean (read-only API token)
Duration	3 working days · 1 kick-off call (30 min) · 1 walkthrough call (45 min)
Changes made	None. Nothing in the client environment was modified during the review.

1. Executive summary

BookFlow runs on a sensible, standard setup and the basics are in place: HTTPS renews automatically, the firewall only exposes SSH and web ports, and staging is separated from production. The weak points are the ones that only show up on a bad day.

- **Backups have never been restored.** Database dumps go to object storage every night, but nobody has tested a restore, and the backups sit in the same account as the server with no versioning. A compromised account, or a mistake, could remove both.
- **Failures are silent.** Uptime is checked only on the homepage. When queue workers stopped for 9 hours last month, booking confirmation emails were not sent and nobody was alerted. The client noticed first.
- **Deploys cause short outages and cannot be rolled back quickly.** Every deploy produces 20–40 seconds of errors, and there is no tested way back if a migration goes wrong.
- **Access has drifted.** Six SSH keys can log in to production; two belong to contractors who left the project.

None of these need a re-platform. **8 of the 10 findings can be fixed within the existing stack in roughly 3–4 days of work**, most of it in the first two weeks (see the 30-day plan in section 4).

2. Scorecard

Each area is scored 0–5 against what is reasonable for a small production web application (not against an enterprise standard). 0 = missing, 3 = adequate with gaps, 5 = solid and tested.

Area	Score	Summary
Deployment & CI/CD	2	Push-to-deploy works; no tests in the pipeline, no zero-downtime, no rollback path
Backups & recovery	1	Nightly dumps exist; never restored; same account; files not backed up
Monitoring & alerting	2	Homepage ping only; queues, disk and errors not alerted
Server security & access	3	Good firewall and key-only SSH; stale keys, root login allowed, patches pending
Application runtime	3	Horizon and scheduler configured; log growth and PHP-FPM limits need tuning
Secrets & configuration	2	Public staging with debug on; shared API keys between staging and production
Cost & capacity	3	Server is oversized for current load; about \$48/month can be saved
Overall	2.3	Healthy foundation, low resilience

What is already good

- Let's Encrypt certificates renew automatically through Forge; no certificate expires within 60 days.
- `ufw` allows only ports 22, 80 and 443; MySQL and Redis listen on localhost only.
- SSH password login is disabled; `unattended-upgrades` installs security updates.
- Staging and production run on separate servers and separate databases.

3. Findings

Ordered by risk. Effort is an estimate for someone familiar with the stack. "Evidence" describes what was checked, so your team can repeat the check.

CRITICAL F1 · Database backups have never been restored

Evidence	Forge backup job runs nightly to a Spaces bucket (last 14 runs successful). No restore test in the project history or runbook. Bucket has no versioning; the same API key that writes backups can delete them.
Risk	A backup that has not been restored is an assumption. If the dumps are incomplete, or the account is compromised, data loss is permanent.
Fix	Run a timed restore to a scratch database and record the steps. Enable bucket versioning and a 30-day retention lock, or copy backups to a second provider with a write-only key. Repeat the restore test monthly.
Effort	0.5 day, then 30 minutes a month

HIGH F2 · Queue failures are not alerted

Evidence	Horizon is running, but nothing alerts when it stops. The <code>failed_jobs</code> table holds 1,240 rows, including a 9-hour gap last month when booking confirmation emails were not sent.
Risk	Customer-facing features fail silently. The client hears about it before the agency does.
Fix	Add a heartbeat check for the scheduler and Horizon (for example with a free cron-monitoring service). Alert on failed-job growth and on queue wait time. Send alerts to the channel the team actually reads.
Effort	0.5 day

HIGH F3 · Deploys cause 20–40 seconds of errors and have no rollback

Evidence	The Forge deploy script runs <code>git pull</code> , <code>composer install</code> and <code>migrate --force</code> in place. Access logs show bursts of 502 responses during each of the last 5 deploys.
Risk	Users see errors on every release. A failed migration leaves the app half-updated, with no quick way back.
Fix	Switch to release-folder (zero-downtime) deployments. Keep the last 3 releases for instant rollback. Write migrations to be backwards-compatible, and take a database snapshot before risky ones.
Effort	0.5–1 day

HIGH F4 · Former contractors can still log in to production

Evidence	6 keys in <code>authorized_keys</code> for the deploy user; 2 belong to people no longer on the project. <code>PermitRootLogin</code> is <code>prohibit-password</code> (root login with a key is allowed).
Risk	Access that nobody remembers is access that nobody revokes when a laptop is lost.
Fix	Remove the stale keys. Set <code>PermitRootLogin no</code> . Keep a short access list (who, why, since when) and review it quarterly.
Effort	1–2 hours

HIGH F5 · Staging is public with debug mode on

Evidence	<code>staging.</code> subdomain is reachable without authentication. <code>APP_DEBUG=true</code> shows full stack traces, including environment details. Staging uses the same mail and payment-sandbox API keys as production.
Risk	Debug pages leak configuration details to anyone who triggers an error. Shared keys mean a staging leak is a production leak.
Fix	Put staging behind HTTP basic auth or an IP allow-list. Turn debug off on anything public. Issue separate keys per environment and rotate the shared ones.
Effort	2–3 hours

MEDIUM F6 · Tests exist but do not gate deploys

Evidence 142 PHPUnit tests in the repository; no GitHub Actions workflow. Forge deploys on every push to `main`.

Fix Add a GitHub Actions workflow that runs tests and static analysis on pull requests. Trigger the Forge deploy only after it passes.

Effort 0.5 day

MEDIUM F7 · Uptime check covers only the homepage

Evidence One HTTP check on `/`. The booking API, login and checkout are not checked. There is no alert on disk usage, which is at 78%.

Fix Add a `/health` endpoint that checks the database, Redis and the queue. Monitor it and two key user paths. Alert on disk usage above 80% and memory above 90%.

Effort 0.5 day

MEDIUM F8 · Log file is 3.8 GB and growing

Evidence `storage/logs/laravel.log` uses the `single` channel with no rotation. It is the main reason disk usage is at 78%.

Fix Switch to the `daily` channel with 14-day retention. Optionally ship errors to an error tracker so they are searchable.

Effort 1 hour

MEDIUM F9 · Kernel update waiting for reboot for 47 days

Evidence `/var/run/reboot-required` is present; the running kernel is two security releases behind.

Fix Schedule a monthly maintenance window and reboot. With F3 fixed, the reboot is the only planned downtime.

Effort 30 minutes a month

LOW F10 · Server is oversized for current traffic

Evidence 30-day average CPU 6%, peak 31%; memory peak 3.1 GB of 16 GB.

Fix After F7 is in place (so the change can be watched), resize to the next tier down. Estimated saving: about \$48/month.

Effort 1 hour

4. 30-day plan

When	Work	Findings	Effort
Week 1	Revoke stale keys and root login; lock down staging and rotate shared keys; first restore test; bucket versioning	F1, F4, F5	~1 day
Week 2	Queue and scheduler heartbeats; health endpoint; disk and memory alerts; log rotation	F2, F7, F8	~1 day
Week 3	Zero-downtime deploys with rollback; CI on pull requests gating deploys	F3, F6	~1–1.5 days
Week 4	Maintenance window and reboot; resize server; write a one-page runbook (restore, rollback, who to call)	F9, F10	~0.5 day

Your team can run this plan using the report alone. If you prefer, the same work can be done as a fixed-price follow-up, or covered by a small monthly care plan (monitoring, patching, monthly restore test and a short monthly report).

5. What was checked

Deployment & CI/CD

- Deploy script, triggers and branch protection
- Tests and static analysis in the pipeline
- Zero-downtime and rollback path
- Migration safety

Backups & recovery

- Database and file backups: schedule, location, retention
- Restore test (timed, on a scratch target)
- Separation from the production account

Monitoring & alerting

- Uptime and key user paths
- Queues, scheduler, disk, memory, certificates
- Where alerts go and who sees them

Cost & capacity

- Resource usage against server size
- Unused resources (volumes, snapshots, IPs)

Server security & access

- Open ports and firewall
- SSH configuration and authorized keys
- OS patch level and pending reboots
- Provider and Forge account access, 2FA

Application runtime

- PHP-FPM pool sizing, OPcache
- Queue workers and scheduler supervision
- Log channels, rotation and error tracking

Secrets & configuration

- `.env` handling and debug flags
- Key separation between environments
- Secrets in the repository history

6. How a Health Check works

Step	What happens
Kick-off (30 min)	We agree on scope (one app, one environment) and you grant read-only access.
Review (3 days)	Hands-off review against the checklist above. No changes are made to your systems.
Report	A report like this one: scorecard, prioritised findings with evidence, and a 30-day plan.
Walkthrough (45 min)	We go through the findings together and answer questions from your team.
Price	\$750 fixed for one application and environment.

Prepared by **Taha Bagatur**, DevOps Engineer (CI/CD, Linux, Docker, NGINX, monitoring, backups) · taha@tahabagatur.com · github.com/mtbagatur

Sample report: the agency, the application and all figures are fictional.