PERFORMANCE MANAGEMENT SYSTEM · V1.0.0

Deployment Guide

Requirements, configuration, HTTPS/reverse proxy, health checks, logging, backups, monitoring, and a production readiness checklist.

Performance Management System · Deployment Guide · Confidential

Contents

1. Requirements

- .NET 8 runtime (ASP.NET Core runtime, not just the base runtime)
- PostgreSQL 16 (or compatible)
- Outbound SMTP relay access
- ~300 MB disk for the bundled headless-Chromium build (downloaded automatically on first run by PuppeteerSharp, used for PDF report rendering)
- A reverse proxy or load balancer that terminates TLS (nginx, IIS, or a cloud load balancer) — the application itself does not terminate TLS

2. Environment variables & configuration

VARIABLE / KEY	PURPOSE
`PM_CONNECTION` (or `ConnectionStrings:Pm`)	PostgreSQL connection string. **Must** be set in Production — the app refuses to start with the local-dev default.
`PM_ADMIN_USERNAME` / `PM_ADMIN_PASSWORD` (or `AdminAccount:*`)	Initial administrator account. **Must** change the password from the shipped default.
`Security:LoginAsVerificationPassword`	Seeds the "Login As" verification password on first run (can also be changed later via Settings → Authentication). **Must** change from the shipped default.
`Security:DefaultUserPassword`	Default password assigned to newly created employee accounts. **Must** change from the shipped default.
`ASPNETCORE_ENVIRONMENT`	Set to `Production` — enables HTTPS redirection/HSTS, disables the developer exception page, and activates the default-credential boot guard.
SMTP settings	Configured at runtime via Settings → Email (not an environment variable) — see the Administrator Guide.
`General:ApplicationBaseUrl`	The public HTTPS origin used to build links in outgoing email — set this via Settings → General after first deploy.

BOOT GUARD

The application will throw and refuse to start in Production if *any* of the above credentials are still at their shipped default value. This is intentional — treat it as a deployment blocker to fix, not bypass.

3. Database migration

Migrations are applied automatically on every application startup (`Database.MigrateAsync()`), which is safe and convenient for this application's single-node design. If your organization's change-control process requires migrations as a separate, explicit release step instead, run:

```
dotnet ef database update --project src/PerformanceManagement.Core --startup-project src/PerformanceManagement.Web
```

MULTI-INSTANCE WARNING

Do not run more than one application instance against the same database during a deploy unless you are certain only one of them will race to apply migrations — the current design assumes a single instance performs this step.

4. HTTPS & reverse proxy

The application enables HTTPS redirection and HSTS automatically in Production, and trusts `X-Forwarded-For` / `X-Forwarded-Proto` headers from an upstream proxy (so the real client IP is recorded in the audit trail, and secure-cookie logic sees the real scheme). **Lock the trusted-proxy configuration down to your actual reverse proxy's address** in your deployment — the shipped configuration accepts forwarded headers from any upstream by default, which is appropriate only because the real trust boundary is expected to be enforced by network topology (the app should not be directly reachable except through your proxy).

5. Health endpoint

`GET /health` — unauthenticated, returns healthy only when the application can successfully reach PostgreSQL. Point your load balancer's or orchestrator's liveness/readiness probe here.

6. Logging

Standard ASP.NET Core logging configuration (`appsettings.json` / `appsettings.Production.json`). The durable, queryable record of "who did what" for business events is the in-app **Audit Log** (Dashboard Recent Activity, Workflow Administration Details), not console/file logs — treat the two as complementary: application logs for operational troubleshooting, the audit log for accountability.

7. Backups

- **Recommended cadence**: nightly `pg_dump` (or your managed Postgres provider's automated snapshot), retained 30+ days.
- **Scope**: the database is the only stateful store that matters, plus the small `wwwroot/uploads/branding/` folder (the uploaded company logo) — safe to re-upload manually after a restore if not separately backed up.
- **Test the restore**, not just the backup — periodically restore into a scratch environment and confirm the application actually starts and authenticates against it.

Full guidance (including log retention: `EmailLog` is auto-purged after 180 days; `AuditLog` / `ImpersonationLog` are retained indefinitely by design) is in `docs/operations.md`.

8. Monitoring

- `/health` for basic liveness/readiness.
- The Scheduled Jobs page for last-run status/duration/error of every background job.
- The Audit Log (Dashboard) for security-relevant events — repeated `"Login Failed"` / `"Login Blocked: Account Locked"` entries for the same username are the signal to watch for a credential-stuffing attempt.

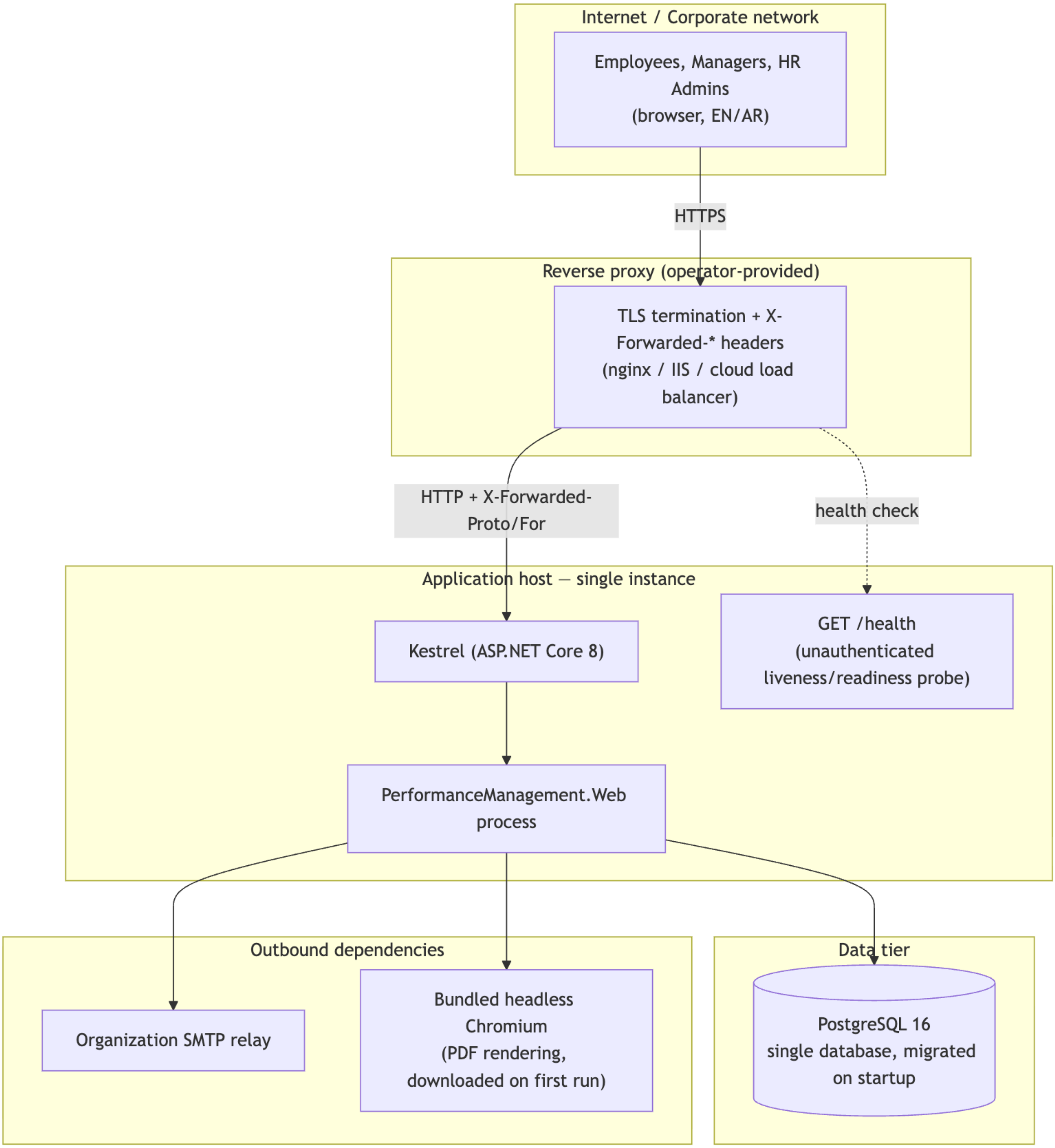
9. Production checklist

ITEM	STATUS
Connection string points to the real production database	Required
Admin password, Login-As verification password, default employee password all changed from shipped defaults	Required (enforced by boot guard)
`ASPNETCORE_ENVIRONMENT=Production`	Required
Reverse proxy terminates TLS; app not directly internet-reachable	Required
SMTP configured and tested (Settings → Email → Send Test Email)	Required
Application Base URL set to the real public HTTPS origin	Required
Company branding/logo configured	Recommended
Database backup schedule configured at the infrastructure level	Required
`/health` wired into load balancer/monitoring	Recommended
Confirmed single application instance (no horizontal scaling without further changes — see §11)	Required

10. Rollback plan

1. Stop the new application version.
2. If the release included a database migration that is safe to leave in place (additive, non-destructive — true of every migration to date), simply redeploy the previous application version; EF Core migrations are forward-compatible with an unchanged schema.
3. If a migration must be reverted, restore the pre-deploy database backup rather than attempting an automatic "down" migration in production — safer and simpler to reason about.
4. Confirm `/health` returns healthy and spot-check a login and one PM Form view before declaring the rollback complete.

11. Deployment topology



SINGLE-NODE ONLY

Session state and the Quartz scheduler are both in-process, in-memory. Running multiple replicas today would mean every replica independently runs every scheduled job (duplicate annual-form generation, duplicate reminder/escalation emails) and races on startup migrations. Do not scale horizontally without first: moving to a distributed session/cache store, switching Quartz to a clustered or persistent job store, and moving `Database.MigrateAsync()` out of application startup into a single, explicit release step.