MIMM 2.0 – Deployment Checklist (Lite)

Minimalistický checklist pro jeden VPS (Hetzner, Ubuntu 24.04). Osekáno na to nejnutnější. Redis je volitelný.

Datum nasazení: _____
Provedl: _____
VPS IP: _____

Co sakra potřebuješ mít ready PŘED začátkem

- **VPS server** – Hetzner/DigitalOcean/cokoliv, Ubuntu 24.04, aspoň 2 GB RAM
- **Doménu** – registrovanou (např. Cloudflare, GoDaddy, cokoliv) a nastavenou DNS na IP serveru
- **SSH klíče** – vygenerované na svém PC (`ssh-keygen`, pokud nemáš)
- **Silná hesla** – pro DB (min. 16 znaků), Redis, JWT key (min. 32 znaků) – použij password manager
- **Email pro SSL** – na notifikace od Let's Encrypt
- **Git repo** – s tvým kódem, buď přístupný přes HTTPS nebo nastav SSH deploy key
- **Frontend build** – zkompilovaný Blazor WASM (složka `dist` nebo `wwwroot`)
- **Volitelné API klíče:**
  - SendGrid (pokud posíláš emaily)
  - Last.fm (pokud máš integraci)

Tohle všechno si připrav, než začneš. Jinak se budeš v půlce deployment zastavovat a hledat credentials.

Fáze A: Základ serveru (15–20 min)

- ☐ Přihlášení: `ssh root@IP` (Windows: PowerShell/PuTTY; macOS: Terminal)
- ☐ Aktualizace balíčků: `apt update && apt upgrade -y`
- ☐ Vytvoření uživatele: `adduser mimm` → sudo práva: `usermod -aG sudo mimm`
- ☐ SSH hardening: `nano /etc/ssh/sshd_config` → nastav `Port 2222`, `PermitRootLogin no`, `PasswordAuthentication no`; ulož, pak `systemctl restart sshd`
- ☐ Firewall UFW: `ufw allow 2222/tcp && ufw allow 80/tcp && ufw allow 443/tcp && ufw enable` (potvrď `y`)
- ☐ Fail2Ban: `apt install -y fail2ban` → zapnout: `systemctl enable --now fail2ban` (ochrana proti brute force)

Fáze B: Docker + Nginx (15–20 min)

- ☐ Docker + compose plugin: `curl -fsSL https://get.docker.com -o get-docker.sh && sh get-docker.sh && apt install -y docker-compose-plugin` (nainstaluje Docker i Compose)
- ☐ Přidat uživatele do docker skupiny: `usermod -aG docker mimm` (odhlaš/přihlaš, aby se projevilo)
- ☐ Test Dockeru: `docker run hello-world` (musí vytisknout "Hello from Docker!")
- ☐ Nginx: `apt install -y nginx` → vypnout default site: `rm /etc/nginx/sites-enabled/default` → test configu: `nginx -t`

Fáze C: Certy (10 min)

- ☐ Certbot: `apt install -y certbot python3-certbot-nginx` (nástroj na Let's Encrypt)
- ☐ Získat certy: `certbot certonly --nginx -d your-domain.com -d www.your-domain.com -d api.your-domain.com` (HTTPS pro domény)

- ☐ Ověřit obnovu: `certbot renew --dry-run` (zkouška automatické obnovy)
- ☐ Webroot pro ACME: `mkdir -p /var/www/certbot && chown www-data:www-data /var/www/certbot` (pokud bude potřeba http-01)

Fáze D: Aplikace (20–30 min)

- ☐ Kód: `git clone <repo> /home/mimm/mimm-app` (nebo nahrát přes SFTP) a `cd /home/mimm/mimm-app`
- ☐ `.env`: vytvoř a vyplň (DB/JWT/URL), pak `chmod 600 .env` (ať k němu nikdo cizí nečte)
- ☐ Docker compose: ujisti se, že máš `docker-compose.prod.yml` (Redis přidej jen pokud ho appka používá)
- ☐ Nginx configy: ulož do `/etc/nginx/sites-available/`, symlink `ln -s ... sites-enabled/`, test `nginx -t`, reload `nginx -s reload`
- ☐ Build & run: `docker compose -f docker-compose.prod.yml up -d` (spustí kontejnery)
- ☐ Migrace DB: `docker compose -f docker-compose.prod.yml run --rm backend dotnet ef database update --no-build` (vytvoří tabulky)
- ☐ Frontend statiky: build zkopíruj do `/var/www/mimm-frontend` (např. `rsync -av frontend-build/ /var/www/mimm-frontend/`)

Fáze E: Smoke test (10 min)

- ☐ Backend health: `curl -I https://api.your-domain.com/health` (očekávej HTTP 200)
- ☐ Frontend: otevři https://your-domain.com v prohlížeči, stránka se musí načíst bez chyb
- ☐ Registrace/login: založ test účet, ověř přihlášení a odpovědi API
- ☐ Krátké logy po deploy: `docker compose -f docker-compose.prod.yml logs --tail=50` (ověř, že nejsou chyby)

Fáze F: Backup (5 min)

- ☐ Složka: `mkdir -p ~/backups` (kam se budou dávat dumpy)
- ☐ Jednorázový dump: `docker exec mimm-postgres pg_dump -U mimmuser mimm | gzip > ~/backups/mimm_db_$(date +%Y%m%d).sql.gz`
- ☐ Cron (denně 2:00): `crontab -e` → přidej řádek `0 2 * * * docker exec mimm-postgres pg_dump -U mimmuser mimm | gzip > ~/backups/mimm_db_\%Y\%m\%d.sql.gz`
- ☐ Občas stáhni poslední dump na vlastní PC/NAS (ručně s `scp`/SFTP)

Fáze G: Minimum monitoringu (5 min)

- ☐ UptimeRobot: založ HTTP(S) check na `https://api.your-domain.com/health` (hlídá dostupnost)
- ☐ Rychlé logy: `docker compose -f docker-compose.prod.yml logs --tail=200` (ověř chyby po deploy)
- ☐ Základní troubleshoot 502: `docker ps | grep backend`, `docker compose -f docker-compose.prod.yml logs backend --tail=50`, `nginx -t && nginx -s reload`

Příloha: ukázka `.env` (rychlý tahák)

```
POSTGRES_USER=mimmuser
POSTGRES_PASSWORD=STRONG_DB_PASS
POSTGRES_DB=mimm
JWT_SECRET_KEY=AT_LEAST_32_CHARS_SECRET
JWT_ISSUER=https://api.your-domain.com
JWT_AUDIENCE=mimm-frontend
FRONTEND_URL=https://your-domain.com
# Pokud používáš Redis:
REDIS_PASSWORD=STRONG_REDIS_PASS
```

Volitelné (neblokuje go-live)

- ☐ Redis přidán, pokud appka potřebuje cache/SignalR
- ☐ Docker Bench / Lynis (až později)
- ☐ Off-site backup sync (ručně nebo rsync/S3)
- ☐ Výkonnostní test (ab -n 100 -c 5 na /health)

Go/No-Go:
- [ ] Vše zelené → GO
- [ ] Něco chybí → NO-GO, doplnit