MIMM 2.0 – Hobby Deployment (Lite)

Minimalistický postup pro jeden VPS (Hetzner Ubuntu 24.04), Docker + Nginx reverse proxy, bez zbytečné zátěže. Redis je volitelný – přidej ho jen pokud ho appka reálně používá.

---

0. Parametry a předpoklady

- VPS: Ubuntu 24.04 (1–2 vCPU, 2–4 GB RAM, 40+ GB disk)
- Domény: `your-domain.com`, `api.your-domain.com`
- Stack: Docker, Docker Compose plugin, Nginx na hostu, ASP.NET Core backend v kontejneru, PostgreSQL v kontejneru, Redis **volitelně**
- .env **jen na serveru** (nenahrávat do gitu), min. 32 znaků pro JWT klíč

---

1. Rychlý průlet 10 kroky

1) SSH hardening: uživatel `mimm`, zákaz root loginu, port 2222, pouze klíče. UFW: povolit 2222, 80, 443. Fail2Ban pro SSH a Nginx.
2) Docker + Compose plugin: instalace, přidat `mimm` do `docker` group.
3) Nginx instalace: vypnout default site, připravit HTTP→HTTPS redirect a proxy na backend 127.0.0.1:5001.
4) Certbot: získat certy pro `your-domain.com` a `api.your-domain.com`, povolit auto-renew.
5) .env: vyplnit DB, JWT, e-mail, URL; chmod 600.
6) docker-compose.prod.yml: postgres + backend; Redis jen pokud potřebuješ.
7) Build & run: `docker compose -f docker-compose.prod.yml up -d`.
8) Migrace DB: `docker compose -f docker-compose.prod.yml run --rm backend dotnet ef database update --no-build`.
9) Smoke test: `curl https://api.your-domain.com/health`, otevřít frontend, otestovat login/registraci.
10) Backup: denní pg_dump do `~/backups`; občas stáhni na vlastní notebook/NAS.

---

2. Konfigurační šablony

2.1 docker-compose.prod.yml (bez Redis)

```yaml
version: '3.8'

services:
  postgres:
    image: postgres:16-alpine
    restart: unless-stopped
    environment:
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_DB: ${POSTGRES_DB}
    volumes:
      - postgres_data:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U ${POSTGRES_USER}"]
      interval: 10s
      timeout: 5s
```

```
      retries: 5
    networks:
      - backend

  backend:
    build:
      context: .
      dockerfile: Dockerfile
    restart: unless-stopped
    environment:
      ASPNETCORE_ENVIRONMENT: Production
      ASPNETCORE_URLS: http://+:5001
      ConnectionStrings__DefaultConnection: "Host=postgres;Port=5432;Database=${POSTGRES_DB};U
      Jwt__Key: ${JWT_SECRET_KEY}
      Jwt__Issuer: ${JWT_ISSUER}
      Jwt__Audience: ${JWT_AUDIENCE}
      Cors__AllowedOrigins__0: ${FRONTEND_URL}
    depends_on:
      postgres:
        condition: service_healthy
    networks:
      - frontend
      - backend
    security_opt:
      - no-new-privileges:true
    cap_drop:
      - ALL
    read_only: true
    tmpfs:
      - /tmp
      - /app/logs
    volumes:
      - ./logs:/app/logs

  # Redis (volitelné, přidej jen pokud ho appka používá)
  # redis:
  #   image: redis:7-alpine
  #   restart: unless-stopped
  #   command: redis-server --requirepass ${REDIS_PASSWORD}
  #   networks:
  #     - backend

  frontend:
    image: nginx:alpine
    restart: unless-stopped
    volumes:
      - ./frontend-build:/usr/share/nginx/html:ro
      - ./nginx-frontend.conf:/etc/nginx/conf.d/default.conf:ro
    networks:
      - frontend

networks:
  frontend:
```

```
    driver: bridge
  backend:
    driver: bridge
    internal: true

volumes:
  postgres_data:
    driver: local
```

2.2 Nginx (backend)

Minimalní blok s HTTPS, reverse proxy a WS:

```
server {
    listen 80;
    server_name api.your-domain.com;
    location /.well-known/acme-challenge/ { root /var/www/certbot; }
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl http2;
    server_name api.your-domain.com;

    ssl_certificate /etc/letsencrypt/live/api.your-domain.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/api.your-domain.com/privkey.pem;

    add_header Strict-Transport-Security "max-age=63072000; includeSubDomains; preload" always
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header Referrer-Policy "strict-origin-when-cross-origin" always;
    server_tokens off;

    client_max_body_size 10M;

    upstream mimm_backend { server 127.0.0.1:5001; keepalive 16; }

    location / {
        proxy_pass http://mimm_backend;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /hubs/ {
        proxy_pass http://mimm_backend;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
```

```
    location /health {
        proxy_pass http://mimm_backend;
        access_log off;
    }
}
```

2.3 Nginx (frontend)

```
server {
    listen 80;
    server_name your-domain.com www.your-domain.com;
    location /.well-known/acme-challenge/ { root /var/www/certbot; }
    return 301 https://your-domain.com$request_uri;
}

server {
    listen 443 ssl http2;
    server_name your-domain.com;

    ssl_certificate /etc/letsencrypt/live/your-domain.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/your-domain.com/privkey.pem;

    add_header Strict-Transport-Security "max-age=63072000; includeSubDomains; preload" always
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
    server_tokens off;

    root /var/www/mimm-frontend;
    index index.html;

    location / {
        try_files $uri $uri/ /index.html =404;
    }
}
```

2.4 .env šablona (jen na serveru)

```
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
REDIS_PASSWORD=STRONG_REDIS_PASS  # jen pokud používeš Redis
SENDGRID_API_KEY=...
SENDGRID_FROM_EMAIL=noreply@your-domain.com
```

---

3. Zkrácený postup (detailněji)

- VPS update: `apt update && apt upgrade -y`
- Uživatel + SSH hardening: port 2222, `PermitRootLogin no`, `PasswordAuthentication no`

- UFW: allow 2222, 80, 443; enable
- Fail2Ban: basic jail pro sshd a nginx
- Docker + Compose: install, přidat uživatele do skupiny, `docker run hello-world`
- Nginx: instalace, zrušit default site, připravit configy výše
- Certbot: standalone nebo nginx plugin, získat certy pro domény, ověřit `certbot renew --dry-run`
- Kód: naklonovat do `/home/mimm/mimm-app`, přidat .env (600)
- Build+run: `docker compose -f docker-compose.prod.yml up -d`
- Migrace: `docker compose -f docker-compose.prod.yml run --rm backend dotnet ef database update --no-build`
- Smoke test: health endpoint, login/registrace ve frontendu
- Backup: cron denně pg_dump do `~/backups`; jednou za čas ručně stáhnout

---

4. Monitoring (light)

- UptimeRobot ping na `https://api.your-domain.com/health`
- Ruční kontrola logů: `docker compose -f docker-compose.prod.yml logs --tail=200` a `/var/log/nginx/*.log`
- `docker stats` pro rychlý přehled

---

5. Performance (light)

- Jednorázově: `ab -n 100 -c 5 https://api.your-domain.com/health`
- Pokud vše OK a latence < 300 ms, dál nic neřeš

---

6. Bezpečnost (minimum)

- SSH hardening + UFW + Fail2Ban
- HTTPS only, HSTS, security headers
- Non-root kontejnery, cap_drop ALL, read-only fs kde to dává smysl
- Žádné exposed DB/Redis porty ven
- Secrets jen v .env na serveru

---

7. Backup (light)

- Denní: `pg_dump` → `~/backups/mimm_db_YYYYMMDD.sql.gz`
- Retence: 30 dní lokálně
- Off-site: ručně 1× za čas stáhnout na vlastní zařízení (nebo rsync/S3 až později)

---

8. Troubleshooting (rychlé)

- Backend logy: `docker compose -f docker-compose.prod.yml logs backend --tail=200`
- DB: `docker exec -it mimm-postgres psql -U mimmuser -d mimm -c "SELECT 1;"`
- Nginx config test: `nginx -t && nginx -s reload`
- SSL expiry: `echo | openssl s_client -connect api.your-domain.com:443 2>/dev/null | openssl x509 -noout -dates`

---

9. Co je volitelné / až později

- Docker Bench Security, Lynis, App Insights, status page
- Redis, pokud zatím nepotřebuješ cache/SignalR scale-out
- Škálování (load balancer, více uzlů)

Hotovo. Tohle je "MVP subset" pro rychlé a bezpečné nasazení bez enterprise režie.