RÃ©sumÃ© dÃ©taillÃ© du projet Obohr OS / VAâ€“Pipeline
Date: 2026-04-12 11:30

1) Vue dâ€™ensemble
- Objectif: Orchestrer un systÃ¨me multiâ€“agents (Obohr1Bot, ObohrBot, Obohr2Bot) avec contrÃ´le humain, sÃ©curitÃ©, logs, et extensions (marketplace, mobile, autoâ€“learning).
- Infra actuelle: VABOT1 = orchestrateur/gateway, VABOT2 = exÃ©cution (OpenClaw node). Communication via tunnel SSH + Tailscale.
- Bots Telegram: @Obohr1Bot (entrÃ©e principale), @ObohrBot (monitoring), @Obohr2Bot (exÃ©cution). Webhooks via Tailscale Funnel.

2) Ce qui existe dÃ©jÃ (infrastructure)
- Tunnel SSH VABOT2 -> VABOT1: service systemd user "vabot1-tunnel.service".
- Node OpenClaw sur VABOT2: service systemd user "openclaw-node-vabot2.service".
- Gateway OpenClaw sur VABOT1: service "openclaw-gateway.service".
- Tailscale Funnel actifs (HTTPS):
  * https://vabot1.tail565fc3.ts.net/orchestrator -> 127.0.0.1:8787 (Obohr1Bot)
  * https://vabot1.tail565fc3.ts.net/monitor -> 127.0.0.1:8788 (ObohrBot)
  * https://vabot1.tail565fc3.ts.net/exec -> 127.0.0.1:8789 (Obohr2Bot)
- Webhooks Telegram configurÃ©s vers /telegram/webhook/<SECRET> sur ces URLs.

3) Ce qui existe dÃ©jÃ (codebase VAâ€“Pipeline)
RÃ©pertoire: /home/vaga/VA-DEV/va-pipeline
Structure (principale):
- api/ API FastAPI (REST + WebSocket)
- agents/ Obohr1Bot, ObohrBot, Obohr2Bot + business agents
- core/ config, logging, memory, autoâ€“learning, marketplace
- tasks/ Celery worker (Obohr2Bot)
- memory/ vector store (Chroma/FAISS selon config)
- monitoring/ hooks/monitoring
- security/ modules sÃ©curitÃ©
- voice/ STT/TTS pipeline (squelette)
- ui/ UI React (HUD futuriste)
- mobile/ App React Native (Expo)
- plugins/ marketplace + plugins
- docs/ docs marketplace + business agents
- docker-compose.yml Postgres + Redis
- requirements.txt dÃ©pendances Python

Modules majeurs dÃ©jÃ ajoutÃ©s:
- Autoâ€“learning: enregistre interactions, approvals, feedback; recommandations; mÃ©moire long terme.
- Marketplace plugins: registry, publish/install/uninstall/scan.
- Business agents: templates (sales.ops, marketing.growth, customer.success), KPI tracking.
- SaaS auth (squelette): JWT, OAuth, routes auth.
- UI JARVIS (React + Tailwind) + WebSocket.
- Mobile app (React Native/Expo) pour contrÃ´le distant.

4) Endpoints API principaux (FastAPI)
- /input -> entrÃ©e humaine (Obohr1Bot)
- /approve -> validation humaine obligatoire

- /dispatch -> exÃ©cution Obohr2Bot (Celery)
- /status -> Ã©tat du pipeline
- /ws -> WebSocket live
- /learning/* -> autoâ€‘learning (report, feedback, recommandations)
- /marketplace/* -> marketplace plugins
- /business/* -> business agents + KPI

5) Comment lâ€™utiliser (mode local dev)
PrÃ©â€‘requis:
- Python 3.11+ (venv)
- Docker + Docker Compose
- Node.js LTS (UI)

Ã‰tapes backend:
1) cd /home/vaga/VA-DEV/va-pipeline
2) python3 -m venv .venv && source .venv/bin/activate
3) pip install -r requirements.txt
4) docker compose up -d postgres redis
5) export LOG_DIR=/home/vaga/VA-DEV/va-pipeline/logs
6) export PYTHONPATH=/home/vaga/VA-DEV/va-pipeline
7) python3 scripts/init_db.py
8) uvicorn api.main:app --reload --host 0.0.0.0 --port 8000

Ã‰tapes worker Celery (Obohr2Bot):
- celery -A tasks.celery_app worker -Q obohr2 -n obohr2@%h -l info

Ã‰tapes UI (React):
- cd /home/vaga/VA-DEV/va-pipeline/ui
- npm install
- npm run dev -- --host 0.0.0.0 --port 5173

Ã‰tapes mobile (Expo):
- cd /home/vaga/VA-DEV/va-pipeline/mobile
- npm install
- npm run start

6) Utilisation via Telegram (production actuelle)
- Envoyer un message Ã @Obohr1Bot (commande naturelle).
- Le bot demande validation si action critique.
- Si validÃ©e, Obohr2Bot exÃ©cute via queue.
- Les logs et alertes partent sur @ObohrBot + groupe dâ€™alertes.

7) Logs & fichiers
- Logs VAâ€‘Pipeline: /home/vaga/VA-DEV/va-pipeline/logs
- DonnÃ©es autoâ€‘learning: /home/vaga/VA-DEV/va-pipeline/memory
- Registry marketplace: /home/vaga/VA-DEV/va-pipeline/plugins/marketplace.json

8) Ce qui manque encore (gaps)
- DÃ©ploiement prod â€œcleanâ€•: systemd services (API, worker, UI).
- Reverseâ€‘proxy TLS (Nginx/Caddy) si sortie via domaine public.
- ObservabilitÃ© complÃ¨te (Prometheus/Grafana) + alerting standard.

- Rotation et backup secrets GPG horsâ€‘host (Phase 6 prÃ©vue).
- CI/CD + tests automatiques endâ€‘toâ€‘end.
- Monitoring infrastructure (uptime, queue depth, usage CPU/RAM).
- RBAC Enterprise complet + multiâ€‘tenant (spÃ©cification ok, implÃ©mentation Ã faire).
- App mobile: build prod + notifications push (Expo/EAS).

9) Ã‰tat actuel (fonctionnel)
- Webhooks Telegram actifs (Obohr1/Obohr/Obohr2).
- VABOT2 connectÃ© au gateway OpenClaw (node stable).
- API FastAPI dÃ©marre (aprÃ¨s LOG_DIR + PYTHONPATH + emailâ€‘validator).
- Celery worker dÃ©marre (queue obohr2).
- UI React prÃªte.

10) Recommandations prochaines Ã©tapes
1) Services systemd prod pour API/worker/UI.
2) Mise en place reverseâ€‘proxy + TLS (si exposition publique hors Tailscale).
3) Monitoring standard + alerting.
4) CI/CD + tests automatiques.
5) ImplÃ©mentation enterprise (RBAC, audit, multiâ€‘tenant).

â€” Fin du rÃ©sumÃ© â€”