

OpenClaw

Cập nhật thông tin và kiến thức trong quá trình cài đặt và sử dụng OpenClaw như một trợ lý cá nhân trong cuộc sống hàng ngày

- [Cài đặt OpenClaw bằng Portainer](#)
- [Cài đặt "bộ não" cho OpenClaw](#)
 - [Cài đặt Codex làm backend cho OpenClaw](#)
- [Cài đặt Chat Channels cho OpenClaw](#)
 - [Cài Discord Bot cho OpenClaw](#)
 - [Cài Telegram Bot cho OpenClaw](#)

Cài đặt OpenClaw bằng Portainer

Mục tiêu

Tài liệu này hướng dẫn cài OpenClaw bằng **Portainer** với Stack trên Web editor.

Hướng dẫn phù hợp cho 2 kiểu triển khai:

Kiểu 1: Docker host + NAS

- Docker chạy trên một máy Ubuntu
- Config/workspace lưu trên NAS mount vào Docker host

Kiểu 2: Một máy duy nhất

- Docker, config, workspace đều lưu trên cùng một máy

Sau khi hoàn tất

- OpenClaw container: openclaw-gateway
- Port: 18789
- Domain ví dụ: <https://oclaw.example.com>
- WebSocket URL: <wss://oclaw.example.com>
- Quản lý stack: Portainer Full Control

Mô hình truy cập qua domain

Client Browser

↓ HTTPS / WSS

Nginx Proxy Manager

↓ HTTP / WebSocket

Docker Host

↓

OpenClaw Gateway Container

Tóm tắt nhanh

Các bước chính:

1. Chọn kiểu lưu dữ liệu: NAS hoặc một máy
2. Tạo thư mục dữ liệu
3. Tạo Gateway Token
4. Tạo Portainer Stack bằng Web editor
5. Nhập biến môi trường cho đúng đường dẫn lưu trữ
6. Deploy OpenClaw Gateway
7. Compose tự tạo config ban đầu trước khi start gateway
8. Cấu hình NPM reverse proxy + WebSocket
9. Truy cập domain HTTPS
10. Approve device pairing nếu cần
11. Quản lý stack bằng Portainer để giữ Full Control

Chuẩn bị thông tin môi trường

Trước khi làm, cần xác định các giá trị sau.

Ví dụ trong bài:

- Docker Host IP: 192.168.1.103
- Nginx Proxy Manager IP: 192.168.1.102
- Domain OpenClaw: oclaw.example.com
- OpenClaw port: 18789

Thay lại theo môi trường thực tế của bạn.

Chọn kiểu lưu dữ liệu

Kiểu A — Docker host + NAS

Dùng khi Docker chạy trên một máy Ubuntu, còn dữ liệu lớn hoặc workspace lưu trên NAS.

Ví dụ:

```
Config:      /mnt/dsm/docker/OpenClaw/config ? Lưu file cấu hình OpenClaw
Workspace:   /mnt/dsm/docker/OpenClaw/workspace ? Lưu workspace, file làm việc, memory, context
Auth:        /opt/appdata/openclaw/auth ? Lưu auth secret local trên Docker host
```

Kiểu B — Cài toàn bộ trên một máy

Dùng khi không có NAS, mọi thứ lưu trên chính Docker host. Kiểu này đơn giản hơn, phù hợp với VPS, mini PC, hoặc homelab một máy.

Ví dụ:

```
Config:    /opt/appdata/openclaw/config ? L?u file c?u hình OpenClaw
Workspace: /opt/appdata/openclaw/workspace ? L?u workspace, file làm vi?c, memory, context
Auth:      /opt/appdata/openclaw/auth ? L?u auth secret local trên Docker host
```

Tạo thư mục dữ liệu

Nếu dùng NAS

```
sudo mkdir -p /mnt/dsm/docker/OpenClaw/config
sudo mkdir -p /mnt/dsm/docker/OpenClaw/workspace
sudo mkdir -p /opt/appdata/openclaw/auth

sudo chown -R 1000:1000 /mnt/dsm/docker/OpenClaw
sudo chown -R 1000:1000 /opt/appdata/openclaw
```

Các đường dẫn dùng trong Portainer:

```
OPENCLAW_CONFIG_HOST_DIR=/mnt/dsm/docker/OpenClaw/config
OPENCLAW_WORKSPACE_HOST_DIR=/mnt/dsm/docker/OpenClaw/workspace
OPENCLAW_AUTH_HOST_DIR=/opt/appdata/openclaw/auth
```

Nếu cài toàn bộ trên một máy

```
sudo mkdir -p /opt/appdata/openclaw/config
sudo mkdir -p /opt/appdata/openclaw/workspace
sudo mkdir -p /opt/appdata/openclaw/auth

sudo chown -R 1000:1000 /opt/appdata/openclaw
```

Các đường dẫn dùng trong Portainer:

```
OPENCLAW_CONFIG_HOST_DIR=/opt/appdata/openclaw/config
OPENCLAW_WORKSPACE_HOST_DIR=/opt/appdata/openclaw/workspace
OPENCLAW_AUTH_HOST_DIR=/opt/appdata/openclaw/auth
```

Tạo Gateway Token

Chạy trên Docker host:

```
openssl rand -hex 32
```

Ví dụ output:

```
aabbccddeeff00112233445566778899aabbccddeeff00112233445566778899
```

Copy token này lại. Lát nữa sẽ nhập vào Portainer với biến:

```
OPENCLAW_GATEWAY_TOKEN
```

Token này dùng để đăng nhập OpenClaw Control UI.

Tạo OpenClaw Stack trong Portainer

Vào Portainer > Stacks > Add stack

Thiết lập:

- Name: openclaw
- Build method: Web editor

Dán compose sau vào Web editor:

Dán compose sau vào Web editor:

```
services:
  openclaw-gateway:
    image: ghcr.io/openclaw/openclaw:latest
    container_name: openclaw-gateway
    restart: unless-stopped
    init: true

  environment:
    TZ: Asia/Bangkok
    HOME: /home/node

    OPENCLAW_STATE_DIR: /home/node/.openclaw
    OPENCLAW_CONFIG_DIR: /home/node/.openclaw
    OPENCLAW_CONFIG_PATH: /home/node/.openclaw/openclaw.json
    OPENCLAW_WORKSPACE_DIR: /home/node/.openclaw/workspace
    OPENCLAW_AUTH_PROFILE_SECRET_DIR: /home/node/.config/openclaw

    OPENCLAW_GATEWAY_TOKEN: ${OPENCLAW_GATEWAY_TOKEN}
    OPENCLAW_DISABLE_BONJOUR: "1"

    OPENCLAW_DOMAIN: ${OPENCLAW_DOMAIN}
    OPENCLAW_TRUSTED_PROXY_IP: ${OPENCLAW_TRUSTED_PROXY_IP}

  volumes:
    - ${OPENCLAW_CONFIG_HOST_DIR}:/home/node/.openclaw
    - ${OPENCLAW_WORKSPACE_HOST_DIR}:/home/node/.openclaw/workspace
    - ${OPENCLAW_AUTH_HOST_DIR}:/home/node/.config/openclaw

  ports:
    - "18789:18789"

  extra_hosts:
    - "host.docker.internal:host-gateway"

  cap_drop:
    - NET_RAW
    - NET_ADMIN

  security_opt:
    - no-new-privileges:true

  command:
    - sh
```

```

- -lc
- |
node dist/index.js config set --batch-json "[
  {\"path\": \"gateway.mode\", \"value\": \"local\"},
  {\"path\": \"gateway.bind\", \"value\": \"lan\"},
  {\"path\": \"gateway.controlUi.allowInsecureAuth\", \"value\": false},
  {\"path\": \"gateway.controlUi.allowedOrigins\", \"value\": [
    \"https://${OPENCLAW_DOMAIN}\",
    \"http://127.0.0.1:18789\",
    \"http://localhost:18789\"
  ]},
  {\"path\": \"gateway.trustedProxies\", \"value\": [\"${OPENCLAW_TRUSTED_PROXY_IP}\"]}
]"

exec node dist/index.js gateway --bind lan --port 18789

healthcheck:
  test:
    [
      "CMD",
      "node",
      "-e",
      "fetch('http://127.0.0.1:18789/healthz').then(r=>process.exit(r.ok?0:1)).catch(()=>process.exit(1))"
    ]
  interval: 30s
  timeout: 5s
  retries: 5
  start_period: 30s

```

Điểm quan trọng của compose này:

- Trước khi start gateway, container tự chạy: `node dist/index.js config set ...`
- Sau đó mới chạy: `node dist/index.js gateway --bind lan --port 18789`

Nhờ vậy, cài mới sẽ không bị lỗi Missing config do thư mục config trống.

Thêm biến môi trường trong Portainer

Trong phần **Environment variables** của Portainer, thêm các biến dưới đây rồi nhấn **Deploy the stack**

Nếu dùng NAS

```

OPENCLAW_GATEWAY_TOKEN=token_v?a_t?o ? Token dùng ?? ??ng nh?p Control UI
OPENCLAW_DOMAIN=oclaw.example.com ? Domain dùng ?? truy c?p OpenClaw, không có https:// phía tr??c
OPENCLAW_TRUSTED_PROXY_IP=192.168.1.102 ? IP c?a Nginx Proxy Manager nhìn t? OpenClaw Gateway

OPENCLAW_CONFIG_HOST_DIR=/mnt/dsm/docker/OpenClaw/config
OPENCLAW_WORKSPACE_HOST_DIR=/mnt/dsm/docker/OpenClaw/workspace
OPENCLAW_AUTH_HOST_DIR=/opt/appdata/openclaw/auth

```

Nếu cài toàn bộ trên máy

```
OPENCLAW_GATEWAY_TOKEN=token_v?a_t?o
OPENCLAW_DOMAIN=oclaw.example.com
OPENCLAW_TRUSTED_PROXY_IP=192.168.1.102
```

```
OPENCLAW_CONFIG_HOST_DIR=/opt/appdata/openclaw/config
OPENCLAW_WORKSPACE_HOST_DIR=/opt/appdata/openclaw/workspace
OPENCLAW_AUTH_HOST_DIR=/opt/appdata/openclaw/auth
```

Kiểm tra trạng thái Gateway

Chạy trên Docker host:

```
docker logs --tail 80 openclaw-gateway
```

Nếu thấy:

```
[gateway] ready
```

là OpenClaw Gateway đã chạy thành công.

Kiểm tra health:

```
curl -fsS http://127.0.0.1:18789/healthz
curl -fsS http://127.0.0.1:18789/readyz
```

Nếu lệnh trả về OK hoặc JSON hợp lệ thì gateway đã sẵn sàng.

Xem log realtime:

```
docker logs -f openclaw-gateway
```

Hoặc các bạn cũng có thể kiểm tra log và trạng thái trên Portainer nếu đã quen sử dụng.

Cấu hình Nginx Proxy Manager

Trong Nginx Proxy Manager, tạo hoặc sửa Proxy Host:

```
Domain Names: oclaw.example.com
Scheme: http
Forward Hostname / IP: 192.168.1.103 ? IP của Docker host chứa OpenClaw
Forward Port: 18789
Websockets Support: ON
Block Common Exploits: ON
```

Tab **SSL**:

Request a new SSL Certificate hoặc chọn certificate đã có
Force SSL: ON
HTTP/2 Support: ON

Truy cập OpenClaw qua domain

Mở trình duyệt `https://oclaw.example.com`

Điền:

- WebSocket URL: `wss://oclaw.example.com`
- Gateway Token: token đã tạo ở bước 6
- Password: bỏ trống

Sau đó bấm **Connect**

Nếu thiết bị hoặc trình duyệt này chưa từng được approve, OpenClaw có thể yêu cầu device pairing.

Approve thiết bị mới

Bạn có thể truy cập Portainer > Containers > openclaw (*container mà bạn đã cài*) > Console

Tại đây bạn chạy các lệnh

```
// Liệt kê danh sách thiết bị đã Paired và thiết bị Pending
openclaw devices list

// Thay 0883e383-3e7d-4abb-af40-65a52296a04c bằng request thực tế mà bạn thấy
openclaw devices approve 0883e383-3e7d-4abb-af40-65a52296a04c
```

Sau đó quay lại browser và bấm **Connect** lại.

Phụ lục

Ghi chú bổ sung

OpenClaw là agent gateway, có thể kết nối tool, browser, file, memory và automation. Không nên public rộng rãi nếu chưa có lớp bảo vệ phù hợp.

Khuyến nghị tối thiểu:

- Luôn dùng HTTPS
- Không bật `allowInsecureAuth` khi dùng domain
- Giữ Gateway Token

- Giữ Device Pairing
- Bật WebSocket qua NPM
- Dùng NPM Access List nếu chỉ muốn truy cập nội bộ

Với domain HTTPS: `https://oclaw.example.com`

WebSocket URL phải là: `wss://oclaw.example.com`

Không dùng `ws://oclaw.example.com` vì domain đang chạy qua HTTPS.

Lỗi Missing config

Hiện tượng

Container OpenClaw restart liên tục. Log có dạng:

```
[gateway] loading configuration...
[gateway] resolving authentication...
Missing config. Run `openclaw setup` or set gateway.mode=local (or pass --allow-unconfigured).
```

Nguyên nhân

Lỗi này xảy ra khi thư mục config của OpenClaw đang trống, chưa có `openclaw.json` hợp lệ.

Thường gặp khi:

- Xóa thư mục config cũ
- Mount sai thư mục config
- Container start gateway trước khi tạo config

Cách tránh lỗi

Dùng compose trong tài liệu này.

Compose đã cấu hình để container tự chạy:

```
node dist/index.js config set --batch-json ...
```

trước khi chạy gateway.

Cách sửa nếu vẫn gặp lỗi

Kiểm tra mount config:

Nếu dùng NAS:

```
ls -lah /mnt/dsm/docker/OpenClaw/config
```

Nếu cài toàn bộ trên một máy:

```
ls -lah /opt/appdata/openclaw/config
```

Kiểm tra log:

```
docker logs --tail 80 openclaw-gateway
```

Nếu vẫn thiếu config, vào Portainer kiểm tra lại các biến:

```
OPENCLAW_CONFIG_HOST_DIR
OPENCLAW_WORKSPACE_HOST_DIR
OPENCLAW_AUTH_HOST_DIR
OPENCLAW_DOMAIN
OPENCLAW_TRUSTED_PROXY_IP
OPENCLAW_GATEWAY_TOKEN
```

Sau khi sửa biến, re-deploy stack trong Portainer.

Xóa OpenClaw c? n?u ?ã t?ng cài

Xóa docker container

```
# Dùng stack c? n?u tr??c ?ó cài b?ng Docker Compose CLI
cd /opt/stacks/openclaw 2>/dev/null && docker compose down -v --remove-orphans || true

# Xóa m?i container có tên openclaw
docker ps -aq --filter "name=openclaw" | xargs -r docker rm -f

# Xóa network openclaw c? n?u còn
docker network ls --format '{{.Name}}' | grep -E '^openclaw' | xargs -r docker network rm || true

# Xóa image c? ?? l?n sau pull l?i m?i
docker image rm ghcr.io/openclaw/openclaw:latest 2>/dev/null || true
```

Xóa d? li?u c?

Với kiểu NAS:

```
sudo rm -rf /opt/stacks/openclaw
sudo rm -rf /opt/appdata/openclaw
sudo rm -rf /mnt/dsm/docker/OpenClaw
```

Với kiểu một máy:

```
sudo rm -rf /opt/stacks/openclaw
sudo rm -rf /opt/appdata/openclaw
```

Kiểm tra lại:

```
docker ps -a | grep -i openclaw || echo "OK: no openclaw containers"
docker network ls | grep -i openclaw || echo "OK: no openclaw networks"
```


Cài đặt "bộ não" cho OpenClaw

Chapter này bao gồm các bài viết hướng dẫn cài đặt để kết nối tới các server AI (có thể là cloud hoặc local) làm backend suy luận cho OpenClaw.

Cài đặt "bộ não" cho OpenClaw

Cài đặt Codex làm backend cho OpenClaw

Mục tiêu

Cấu hình OpenClaw dùng tài khoản ChatGPT/Codex làm model provider chính.

Sau khi hoàn tất:

- Discord / Telegram / Web UI
- → OpenClaw Gateway
- → OpenAI/Codex OAuth
- → openai/gpt-5.5
- → OpenClaw trả lời lại người dùng

Lưu ý:

Cách này dùng tài khoản ChatGPT/Codex của bạn → Usage/quota dùng chung với Codex

Tóm tắt nhanh

Các bước chính:

1. Đăng nhập OpenAI/Codex OAuth
2. Kiểm tra auth profile
3. Set model chính: openai/gpt-5.5
4. Xóa fallback cũ nếu có
5. Tùy chọn tắt thinking
6. Restart OpenClaw Gateway
7. Test qua Web UI / Discord / Telegram

Điều kiện trước khi làm

Có tài khoản ChatGPT có đăng ký subscription và có thể sử dụng Codex.

Có container OpenClaw Gateway đã chạy và có thể truy cập.

Đăng nhập OpenAI/Codex OAuth

Chạy trên Docker host:

```
docker exec -it openclaw-gateway \
  node dist/index.js models auth login --provider openai --device-code
```

OpenClaw sẽ hiện hướng dẫn đăng nhập dạng device code.

Làm theo hướng dẫn:

- Mở link đăng nhập được in ra
- Nhập code nếu được yêu cầu
- Đăng nhập bằng tài khoản ChatGPT/Codex
- Cho phép kết nối

Khi thành công, output sẽ gần giống:

```
Auth profile: openai:your-email@example.com (openai/oauth)
Default model available: openai/gpt-5.5
```

Kiểm tra auth profile

Chạy:

```
docker exec -it openclaw-gateway \
  node dist/index.js models auth list --provider openai
```

Kết quả cần thấy profile kiểu:

```
openai:your-email@example.com
```

Nếu chưa thấy profile, chạy lại bước đăng nhập OAuth.

Đặt Codex/OpenAI làm model chính

Đặt model chính:

```
docker exec -it openclaw-gateway \
  node dist/index.js config set agents.defaults.model.primary openai/gpt-5.5
```

Xóa fallback cũ nếu có:

```
docker exec -it openclaw-gateway \  
node dist/index.js config unset agents.defaults.model.fallbacks
```

Kiểm tra lại:

```
docker exec -it openclaw-gateway \  
node dist/index.js config get agents.defaults.model --json
```

Kết quả mong muốn:

```
{  
  "primary": "openai/gpt-5.5"  
}
```

Restart OpenClaw Gateway

Restart container:

```
docker restart openclaw-gateway
```

Chờ 10-20 giây rồi kiểm tra log:

```
docker logs --tail 80 openclaw-gateway | grep -i "agent model"
```

Kết quả mong muốn:

```
[gateway] agent model: openai/gpt-5.5 (thinking=medium, fast=off)
```

Test qua OpenClaw

Test qua Web UI, Discord, hoặc Telegram:

```
B?n ?ang ho?t ??ng ch?a? Tr? l?i ng?n g?n.
```

Kết quả mong muốn:

```
?ang ho?t ??ng.
```

Nếu bot trả lời được, Codex/OpenAI đã trở thành "*bộ não*" chính của OpenClaw.

Phụ lục

Kiểm tra cấu hình ??y ??

Xem cấu hình agent:

```
docker exec -it openclaw-gateway \
node dist/index.js config get agents.defaults --json
```

Kết quả nên có:

```
{
  "model": {
    "primary": "openai/gpt-5.5"
  }
}
```

Xem trạng thái model:

```
docker exec -it openclaw-gateway \
node dist/index.js models status
```

Hoặc xem danh sách model OpenAI:

```
docker exec -it openclaw-gateway \
node dist/index.js models list --provider openai
```

Lưu ý về quota

Cách này dùng:

- OpenAI/Codex OAuth
- → tài khoản ChatGPT/Codex
- → dùng chung giới hạn với Codex

Không giống với:

- OpenAI API key
- → OpenAI Platform API
- → billing/limit riêng

Vì vậy:

- Dùng Codex OAuth
 - → tiện, không cần API key
 - → phù hợp dùng thử và dùng cá nhân
- Dùng OpenAI API key
 - → tách quota khỏi ChatGPT/Codex
 - → phù hợp nếu muốn kiểm soát billing riêng

Troubleshooting

??ng nh?p xong nh?ng model ch?a ??i

Chạy lại:

```
docker exec -it openclaw-gateway \
  node dist/index.js config set agents.defaults.model.primary openai/gpt-5.5

docker restart openclaw-gateway
```

Không thấy auth profile

Kiểm tra:

```
docker exec -it openclaw-gateway \
  node dist/index.js models auth list --provider openai
```

Nếu không có profile, đăng nhập lại:

```
docker exec -it openclaw-gateway \
  node dist/index.js models auth login --provider openai --device-code
```

Vấn dùng provider c?

Kiểm tra config:

```
docker exec -it openclaw-gateway \
  node dist/index.js config get agents.defaults --json
```

Nếu còn fallback cũ:

```
docker exec -it openclaw-gateway \
  node dist/index.js config unset agents.defaults.model.fallbacks
```

Nếu còn custom provider không dùng nữa, có thể xóa riêng provider đó khỏi config.

Ví dụ với provider `woku`:

```
docker exec -it openclaw-gateway \
  node dist/index.js config unset models.providers.woku
```

Restart:

```
docker restart openclaw-gateway
```

Bot trả lời chậm

Tắt thinking mặc định:

```
docker exec -it openclaw-gateway \
  node dist/index.js config set agents.defaults.thinkingDefault off

docker restart openclaw-gateway
```

Nếu vẫn chậm, nguyên nhân thường nằm ở model/provider hoặc quota backend, không phải Discord/Telegram.

Cài đặt Chat Channels cho OpenClaw

Sau khi cài đặt chúng ta đã có thể chat với OpenClaw qua giao diện web. Nhưng để thuận tiện OpenClaw cung cấp rất nhiều các kênh chat khác nhau như Telegram, Discord, Zalo...

Cài Discord Bot cho OpenClaw

Mục tiêu

Cấu hình Discord Bot để chat với OpenClaw qua DM.

Kết quả sau khi cài:

- Có một Bot Discord online.
- Có thể DM và nhận phản hồi từ Bot.

Điều kiện trước khi làm

OpenClaw Gateway đã chạy bằng Docker/Portainer.

Đã cài đặt backend bất kỳ, có thể chat với OpenClaw thông qua web hoặc nền tảng khác.

Tóm tắt nhanh

Các bước chính:

1. Tạo Discord Application
2. Tạo Bot và copy Bot Token
3. Bật Message Content Intent + Server Members Intent
4. Cấu hình Installation → Guild Install → bot + applications.commands
5. Invite bot vào server riêng
6. Thêm DISCORD_BOT_TOKEN vào Portainer
7. Cài plugin @openclaw/discord
8. Bật channels.discord trong OpenClaw config
9. Restart OpenClaw Gateway
10. Pair Discord DM với OpenClaw
11. Chat với OpenClaw qua DM bot

Tạo Discord Application và Bot

Vào [Discord Developer Portal](#) > New Application > Đặt tên (Ex: OClaw)

Ở menu bên trái chọn **Bot** > Reset Token > Copy Bot Token (*lưu lại token này để dùng sau*)

Bật Gateway Intents cho Bot

Vẫn trong tab **Bot** > Privileged Gateway Intents

- Message Content Intent: ON → Cho phép bot đọc nội dung tin nhắn.
- Server Members Intent: ON → Hữu ích nếu sau này muốn allowlist theo user/role/server.
- Presence Intent: OFF

Save Changes

Cấu hình Installation

Chuyển sang tab **Installation**

Installation Contexts

- Guild Install: ON
- User Install: optional

Install Link

- Chọn Discord Provided Link

Default Install Settings

- Với Guild Install, thêm scopes:
 - bot
 - applications.commands

Khi chọn bot, Discord sẽ hiện phần quyền cho bot. Chọn tối thiểu:

- View Channels
- Send Messages
- Read Message History
- Embed Links
- Attach Files
- Add Reactions

Nếu muốn dùng thread/forum channel, bật thêm:

- `Send Messages in Threads`

Sau đó copy **Install Link**, mở bằng trình duyệt và mời bot vào server riêng của bạn.

Bật Developer Mode và lấy ID

Trong Discord app > User Settings > Advanced > Developer Mode: ON

Lấy các ID cần thiết:

- Right-click server icon → Copy Server ID
- Right-click avatar của bạn → Copy User ID

Các giá trị cần có:

- `DISCORD_BOT_TOKEN=...` // → Không gửi cho người khác.
- `DISCORD_SERVER_ID=...` // → Dùng nếu sau này muốn cấu hình bot trong server/channel.
- `DISCORD_USER_ID=...` // → Dùng để kiểm soát ai được phép chat với bot.

Thêm Discord Bot Token vào Portainer stack

Vào Portainer > Stacks > openclaw > Editor

Trong phần Environment variables, thêm:

- `DISCORD_BOT_TOKEN=token_bot_discord_c?a_b?n`

Trong compose, ở service openclaw-gateway, thêm vào phần environment:

- `DISCORD_BOT_TOKEN: ${DISCORD_BOT_TOKEN}`

Bấm **Update the stack**

Test Discord Bot Token

Chạy:

```
docker exec -it openclaw-gateway sh -lc '
node -e "
fetch("https://discord.com/api/v10/users/@me", {
  headers: { Authorization: \"Bot \" + process.env.DISCORD_BOT_TOKEN }
})
.then(async r => {
  console.log(\"status\", r.status);
  console.log(await r.text());
})
.catch(e => {
  console.error(e);
  process.exit(1);
});
"
```

Kết quả mong muốn:

```
status 200
{"id":"...","username":"...","bot":true,...}
```

Nếu ra:

```
status 401
```

thì token sai hoặc đã bị reset.

Cài plugin Discord cho OpenClaw

Chỉ cấu hình `channels.discord` là chưa đủ. OpenClaw cần plugin Discord.

Kiểm tra trạng thái channel:

```
docker exec -it openclaw-gateway \
node dist/index.js channels list --all | grep -i discord
```

Nếu thấy:

```
Discord: not installed, configured, disabled, run openclaw plugins install @openclaw/discord
```

thì cài plugin:

```
docker exec -it openclaw-gateway \
node dist/index.js plugins install @openclaw/discord
```

Sau đó kiểm tra lại:

```
docker exec -it openclaw-gateway \
node dist/index.js channels list --all | grep -i discord
```

Bật Discord channel trong OpenClaw config

Ghi config Discord:

```
docker exec -it openclaw-gateway \
node dist/index.js config set --batch-json '[
  {
    "path": "channels.discord.enabled",
    "value": true
  },
  {
    "path": "channels.discord.token",
    "value": {
      "source": "env",
      "provider": "default",
      "id": "DISCORD_BOT_TOKEN"
    }
  },
  {
    "path": "channels.discord.groupPolicy",
    "value": "allowlist"
  }
]'
```

Kiểm tra config:

```
docker exec -it openclaw-gateway \
node dist/index.js config get channels.discord --json
```

Kết quả mong muốn:

```
{
  "enabled": true,
  "token": {
    "source": "env",
    "provider": "default",
    "id": "DISCORD_BOT_TOKEN"
  },
  "groupPolicy": "allowlist"
}
```

Restart gateway:

```
docker restart openclaw-gateway
```

Bot trên Discord nên chuyển sang **Online** là thành công.

Pair Discord DM với OpenClaw

Mở DM với bot Discord, gửi `hello` hoặc `pair` > Bot sẽ trả về pairing code.

Trên Docker host, kiểm tra pairing request:

```
docker exec -it openclaw-gateway \
node dist/index.js pairing list discord
```

Approve pairing code:

```
docker exec -it openclaw-gateway \
node dist/index.js pairing approve discord <CODE>
```

Sau đó quay lại Discord DM và test:

```
B?n ?ang ho?t ??ng ch?a? Tr? l?i ng?n g?n.
```

Nếu bot trả lời được, Discord DM đã hoạt động.

Phụ lục

Bot v?n Offline

Kiểm tra:

```
docker exec -it openclaw-gateway \
node dist/index.js channels list --all | grep -i discord
```

Nếu thấy `not installed`, cài plugin:

```
docker exec -it openclaw-gateway \
node dist/index.js plugins install @openclaw/discord
```

Restart:

```
docker restart openclaw-gateway
```

Token r?ng trong container

Kiểm tra:

```
docker exec -it openclaw-gateway sh -lc '
if [ -n "$DISCORD_BOT_TOKEN" ]; then
  echo "OK"
else
  echo "EMPTY"
fi
'
```

Nếu ra `EMPTY`, kiểm tra lại Portainer Environment variables và compose `environment:`.

Token sai

Test:

```
docker exec -it openclaw-gateway sh -lc '
node -e "
fetch("https://discord.com/api/v10/users/@me", {
  headers: { Authorization: \"Bot \" + process.env.DISCORD_BOT_TOKEN }
})
.then(async r => {
  console.log(\"status\", r.status);
  console.log(await r.text());
})
"
'
```

Nếu ra 401, reset token trong Discord Developer Portal rồi cập nhật lại Portainer.

Bot online nh?ng không ph?n h?i DM

Kiểm tra pairing:

```
docker exec -it openclaw-gateway \
node dist/index.js pairing list discord
```

Approve pairing code nếu có:

```
docker exec -it openclaw-gateway \
node dist/index.js pairing approve discord <CODE>
```

Xem log Discord:

```
docker exec -it openclaw-gateway \
node dist/index.js channels logs --channel discord --lines 200
```

Cài Telegram Bot cho OpenClaw

Mục tiêu

Cấu hình Telegram Bot để chat với OpenClaw qua DM.

Luồng hoạt động:

- Bạn
- → Telegram DM Bot
- → OpenClaw Gateway
- → Model
- → trả lời lại Telegram

Tóm tắt nhanh

Các bước chính:

1. Tạo bot bằng @BotFather
2. Copy Telegram Bot Token
3. Thêm TELEGRAM_BOT_TOKEN vào Portainer
4. Cài plugin @openclaw/telegram nếu chưa có
5. Bật channels.telegram
6. Restart OpenClaw Gateway
7. DM bot để lấy pairing code
8. Approve pairing code
9. Chat với OpenClaw qua Telegram DM

Tạo Telegram Bot

Mở Telegram, tìm: > gửi

Làm theo hướng dẫn:

- Đặt tên hiển thị cho bot
- Đặt username cho bot, bắt buộc kết thúc bằng bot

Sau khi tạo xong, BotFather sẽ trả về token dạng `1234567890:ABCDEF_XXXXXXXXXXXXXXXXXXXX`

Lưu token lại. Không gửi token cho người khác.

Thêm token vào Portainer Stack

Portainer → Stacks → openclaw → Editor

Trong phần Environment variables, thêm:

- `TELEGRAM_BOT_TOKEN=token_telegram_bot_c?a_b?n`

Trong compose, thêm vào service openclaw-gateway.environment:

- `TELEGRAM_BOT_TOKEN: ${TELEGRAM_BOT_TOKEN}`

Update the stack.

Kiểm tra token trong container

Chạy:

```
docker exec -it openclaw-gateway sh -lc '
if [ -n "$TELEGRAM_BOT_TOKEN" ]; then
  echo "OK: TELEGRAM_BOT_TOKEN is set"
else
  echo "ERROR: TELEGRAM_BOT_TOKEN is empty"
fi
'
```

Test token với Telegram API:

```
docker exec -it openclaw-gateway sh -lc '
node -e "
fetch("https://api.telegram.org/bot\" + process.env.TELEGRAM_BOT_TOKEN + \"/getMe\")
.then(async r => {
  console.log(\"status\", r.status);
  console.log(await r.text());
})
.catch(e => {
  console.error(e);
  process.exit(1);
});
"
```

Kết quả đúng:

```
status 200
{"ok":true,...}
```

Nếu `ok:false` hoặc lỗi `401`, token sai hoặc đã bị reset.

Cài plugin Telegram

Kiểm tra trạng thái Telegram channel:

```
docker exec -it openclaw-gateway \
  node dist/index.js channels list --all | grep -i telegram
```

Nếu thấy `not installed`, cài plugin:

```
docker exec -it openclaw-gateway \
  node dist/index.js plugins install @openclaw/telegram
```

Kiểm tra lại:

```
docker exec -it openclaw-gateway \
  node dist/index.js channels list --all | grep -i telegram
```

Bật Telegram channel

Ghi config Telegram:

```
docker exec -it openclaw-gateway \
  node dist/index.js config set --batch-json '[
    {
      "path": "channels.telegram.enabled",
      "value": true
    },
    {
      "path": "channels.telegram.dmPolicy",
      "value": "pairing"
    },
    {
      "path": "channels.telegram.groupPolicy",
      "value": "allowlist"
    }
  ]'
```

Ý nghĩa:

```
// Ng??i nh?n DM l?n ??u c?n pairing code.
dmPolicy = pairing

// Không cho bot ho?t ??ng t? do trong group n?u ch?a c?u h?nh.
groupPolicy = allowlist
```

Restart OpenClaw

```
docker restart openclaw-gateway
```

Pair Telegram DM

Mở Telegram, nhấn cho bot `/start` hoặc `hello`

Bot sẽ trả về pairing code.

Trên Docker host, xem pairing request:

```
docker exec -it openclaw-gateway \
node dist/index.js pairing list telegram
```

Approve code:

```
docker exec -it openclaw-gateway \
node dist/index.js pairing approve telegram <CODE>
```

Ví dụ:

```
docker exec -it openclaw-gateway \
node dist/index.js pairing approve telegram ABCD1234
```

Pairing code thường hết hạn sau khoảng 1 giờ.

Test chat

Trong Telegram DM với bot, gửi tin nhắn bất kỳ.

Nếu bot trả lời được, Telegram đã hoạt động.

Troubleshooting

Bot không tr? l?i

Kiểm tra channel:

```
docker exec -it openclaw-gateway \
node dist/index.js channels list --all | grep -i telegram
```

Xem log channel:

```
docker exec -it openclaw-gateway \
node dist/index.js channels logs --channel telegram --lines 200
```

Token r?ng

```
docker exec -it openclaw-gateway sh -lc '
if [ -n "$TELEGRAM_BOT_TOKEN" ]; then echo OK; else echo EMPTY; fi
'
```

Nếu ra `EMPTY`, kiểm tra lại Portainer Environment variables và compose.

Token sai

Test lại Telegram API:

```
docker exec -it openclaw-gateway sh -lc '
node -e "
fetch("https://api.telegram.org/bot\" + process.env.TELEGRAM_BOT_TOKEN + \"/getMe\")
.then(async r => {
  console.log(\"status\", r.status);
  console.log(await r.text());
})
"
'
```

Nếu response có lỗi `401`, tạo/reset token trong BotFather rồi cập nhật lại Portainer.

Bot tr? pairing code nh?ng approve xong v?n không tr? l?i

Kiểm tra pairing:

```
docker exec -it openclaw-gateway \
node dist/index.js pairing list telegram
```

Xem log:

```
docker logs --tail 160 openclaw-gateway | grep -iE "telegram|pairing|error"
```

