OpenClaw

The Complete Mastery Guide

המדריך המקיף והשלם

Edition 2026 | +300 עמודים | English | עברית

מהתקנה ראשונה ועד בניית מערכות AI מורכבות —
כל מה שצריך לדעת על פלטפורמת העוזר האישי החזקה בעולם

From first install to advanced agent swarms —
everything you need to master the most powerful open AI assistant platform

אודות המחבר והמדריך

About the Author & This Guide

אריאל איזנשטט

Ariel Eisenstadt

🎬 מייסד ובעלים של **PixMind Studio** — סטודיו להפקת פרסומות וסרטונים

🤖 מפתח ובעלים של **PixiBot** — סטארטאפ AI בתהליכי פיתוח

🎓 סטודנט להנדסת תוכנה | בן 18

PixMind Studio

עברית 🇮🇱

מדריך זה נכתב עבור כל מי שרוצה להבין ולשלוט בפלטפורמת **OpenClaw** — אחת ממערכות העוזר האישי המבוסס AI המתקדמות ביותר הקיימות כיום.

מהתקנה פשוטה ועד בניית **נחילי סוכנים (Agent Swarms)** מורכבים, המדריך מכסה כל היבט של הפלטפורמה — בשפה ברורה, עם דוגמאות מעשיות, ותמיכה מלאה בעברית ובאנגלית.

בין אם אתם מפתחים, מנהלי מוצר, או סתם סקרנים — הספר הזה הוא נקודת ההתחלה שלכם למהפכת ה-AI הבאה.

English 🇬🇧

This guide was written for everyone who wants to understand and master **OpenClaw** — one of the most advanced AI-powered personal assistant platforms available today.

From simple installation to building complex **Agent Swarms**, this guide covers every aspect of the platform — in clear language, with practical examples, and full Hebrew and English support.

Whether you're a developer, product manager, or simply curious — this book is your starting point for the next AI revolution.

+500K	53	+70	∞
שורות קוד Lines of Code	קבצי קונפיג Config Files	תלויות Dependencies	אפשרויות Possibilities

⭐ איך להשתמש במדריך הזה / How to Use This Guide

כל פרק עומד בפני עצמו — תוכלו לקרוא ברצף או לדלג ישירות לנושא שמעניין אתכם. הדוגמאות הן עם קוד אמיתי שתוכלו להשתמש בו מיד.

Each chapter stands on its own — read sequentially or jump directly to topics that interest you. All code examples are real and ready to use.

תוכן עניינים Table of Contents

5 00 מבוא ל-OpenClaw
Introduction to OpenClaw

18 01 התקנה וקונפיגורציה ראשונית
Installation & Initial Configuration

20 דרישות מערכת • System Requirements

24 Docker vs Apple Container

28 התחברות ראשונה • First Login

35 02 ארכיטקטורה עמוקה
Deep Architecture Dive

37 תהליך יחיד • Single Process

42 מסד נתונים SQLite

46 תקשורת IPC

55 03 ערוצי תקשורת
Communication Channels

57 WhatsApp Channel

65 Telegram Channel

72 Discord Channel

79 Slack Channel

86 Gmail Channel

95 04 בידוד קונטיינרים ואבטחה
Container Isolation & Security

115 05 זיכרון וקונטקסט
Memory & Context System

135 06 מערכת הסקילס
Skills System

07 משימות מתוזמנות
Scheduled Tasks 160

08 נחילי סוכנים
Agent Swarms 180

09 התאמה אישית מתקדמת
Advanced Customization 205

10 אינטגרציות ו-MCP
Integrations & MCP 225

11 תרומה לפרויקט
Contributing to OpenClaw 248

12 פריסה לייצור
Production Deployment 265

13 פתרון בעיות נפוצות
Troubleshooting 282

A נספח: כל הפקודות
Appendix: All Commands 298

CHAPTER ZERO • פרק אפס

מבוא ל-OpenClaw

00

Introduction to OpenClaw

מה זה OpenClaw, למה הוא קיים, ולמה הוא שונה מכל עוזר AI אחר שראיתם

What is OpenClaw, why it exists, and why it's different from every other AI assistant you've seen

CHAPTER 00

מבוא ל-OpenClaw

Introduction to OpenClaw

⏱ זמן קריאה: 45 דקות

🎯 מטרות למידה / Learning Objectives

- להבין מהי פלטפורמת OpenClaw ומה היא פותרת
- לזהות את ההבדלים בין OpenClaw לעוזרי AI אחרים
- להכיר את ארכיטקטורת הבסיס של המערכת
- להבין את עקרון הבטיחות שמוטמע בלב הפלטפורמה
- לדעת מה ניתן לבנות עם OpenClaw

Understand the OpenClaw platform • Identify differences from other AI assistants • Learn the base architecture • Understand the built-in security principle • Know what can be built with OpenClaw

מה זה OpenClaw בכלל?

ב-2023, כאשר עוזרי AI כמו ChatGPT ו-Claude הפכו לכלים שגרתיים, עלתה שאלה חשובה: **איך גורמים לעוזר ה-AI לעבוד בשבילי — באמת, לא רק לענות לשאלות?**

OpenClaw נולד מתוך הצורך הזה. זו לא אפליקציה פשוטה של "שאל וענה" — זוהי **פלטפורמה שלמה** שמחברת את מודלי ה-AI החכמים ביותר לחיי היומיום שלכם: WhatsApp, Telegram, Gmail, Slack, ועוד.

עברית 🇮🇱

OpenClaw הוא **עוזר AI אישי ורב-ערוצי** שרץ על השרת שלכם. הוא מאזין להודעות מ-WhatsApp,

English 🇬🇧

OpenClaw is a **multi-channel personal AI assistant** that runs on your server. It listens to

messages from WhatsApp, Telegram, Discord, Slack, and Gmail — and responds using the Claude Agent SDK, the most capable AI agent available today.

What sets OpenClaw apart: every AI agent runs **inside an isolated Linux container** — not just behind permission checks, but in true OS-level isolation.

Telegram, Discord, Slack ו־Gmail — ומשיב עליהן בעזרת Claude Agent SDK, הסוכן החכם ביותר הקיים כיום.

מה שמייחד את OpenClaw: כל סוכן AI רץ **בתוך קונטיינר Linux מבודד** — לא רק מאחורי בדיקות הרשאות, אלא בבידוד אמיתי של מערכת ההפעלה.

📖 מושג מפתח: Agent (סוכן)

Agent = AI that can take actions, not just generate text

בניגוד לצ'אטבוט רגיל שפשוט עונה על שאלות, **סוכן AI** יכול לבצע פעולות אמיתיות: לפתוח קבצים, להריץ קוד, לגשת לאינטרנט, לשלוח מיילים, ועוד. OpenClaw בנוי סביב סוכנים כאלה.

ההיסטוריה: מאיפה בא OpenClaw?

OpenClaw לא נוצר בחלל ריק. הוא מגיע מניסיון מצטבר של שנים של ניסיון עם מערכות AI enterprise ועם הצורך הצומח של אנשים להיות ב"שליטה" על העוזר שלהם.

🕐 2022 — הרקע

מודלי שפה גדולים (LLMs) מתחילים להיות שמישים. ChatGPT משתחרר. אנשים מבינים שה-AI "עובד" — אבל הוא עדיין מוגבל לממשק טקסט.

🕒 2023 — הצורך

הצוות שמאחורי OpenClaw מחליט: אנחנו רוצים עוזר שמגיב ב-WhatsApp, שזוכר את ההקשר, שיכול להריץ קוד — ושניתן לסמוך עליו מבחינת אבטחה.

🕓 2024 — הפיתוח

OpenClaw מתפתח עם כמעט חצי מיליון שורות קוד, 53 קבצי קונפיגורציה ו-70+ תלויות. הפלטפורמה הופכת לאחת המורכבות בתחום.

🕔 2025-2026 — הבשלה

תמיכה ב-Agent Swarms, אינטגרציה עם Claude Agent SDK, ותמיכה ב-MCP (Model Context Protocol). OpenClaw הופך לסטנדרט לעוזרי AI אישיים.

OpenClaw מול שאר העולם

תכונה	OpenClaw	ChatGPT API	Botpress	Rasa
בידוד קונטיינרים	✓ **אמיתי**	✗	✗	✗
ריבוי ערוצים	✓ **מובנה**	✗ ידני	✓	✓
Agent Swarms	✓ **ראשון בתחום**	✗	✗	✗

קוד פתוח מלא	✓	✗	Partial	✓
Claude Agent SDK	✓ **מלא**	✗	✗	✗
זיכרון לפי קבוצה	✓ **מבודד**	Limited	✓	Limited

הארכיטקטורה בגובה עיניים

לפני שמתקינים, חשוב להבין את "הגוף" של OpenClaw. זה ייתן לכם הקשר לכל ההחלטות הטכניות שנראו מוזרות בהתחלה.

איור 0.1: זרימת הנתונים ב-OpenClaw — מהודעה נכנסת ועד תשובה יוצאת | Data flow in OpenClaw — from incoming message to outgoing response

⭐ נקודת מפתח / Key Point

כל המערכת רצה כ**תהליך Node.js יחיד**. זה נשמע פשוט — וזה בדיוק הנקודה. פשטות = בטיחות = יכולת הבנה מלאה.

*The entire system runs as a **single Node.js process**. This sounds simple — and that's exactly the point. Simplicity = security = full comprehensibility.*

עקרונות הבסיס של OpenClaw

כדי להשתמש ב-OpenClaw נכון, חשוב להבין את עקרונות הבסיס שעיצבו אותו. אלה לא סתם עקרונות שיווקיים — הם מתורגמים ישירות לקוד.

🔒

בטיחות דרך בידוד

כל סוכן AI רץ בקונטיינר Linux נפרד. הוא יכול לגשת רק לקבצים שמותקנים (mounted) אליו במפורש. Bash בטוח — כי הפקודות רצות בקונטיינר, לא על המחשב שלכם.

Security through isolation — real OS-level, not just permission checks.

📦

קטן מספיק להבין

OpenClaw תוכנן לאפשר לכם לקרוא ולהבין את כל קוד הבסיס. אין microservices מסובכים, אין שכבות אבסטרקציה מיותרות. תהליך אחד, קומץ קבצים.

Small enough to understand — one process, a handful of files.

🎯

AI-Native

אין אשף התקנה — Claude Code מדריך את ההגדרות. אין לוח בקרה — שאלו את Claude מה קורה. אין כלי דיבוג — תארו את הבעיה ו-Claude יתקן.

No installation wizard, no dashboard, no debug tools — just ask Claude.

🛠️

סקילס > פיצ'רים

במקום להוסיף פיצ'רים לקוד הבסיס, תורמים שולחים סקילים (skill files) שמלמדים את Claude Code איך לשנות את ה-Fork שלכם. אתם מקבלים קוד נקי שעושה בדיוק מה שצריך.

Skills over features — clean customization without bloat.

⭐ כוח ה-Claude Agent SDK

OpenClaw הוא **הראשון** בתחום העוזרים האישיים שתומך ב-Agent Swarms — טכנולוגיה שמאפשרת לצוות של סוכני AI לשתף פעולה על משימות מורכבות. זה לא פיצ'ר נוסף — זה שינוי פרדיגמה.

*OpenClaw is the **first** personal assistant to support Agent Swarms — a technology that enables a team of AI agents to collaborate on complex tasks. This isn't just another feature — it's a paradigm shift.*

❓ שאלות נפוצות / FAQ

שאלה: האם OpenClaw מחליף ChatGPT?

לא. OpenClaw הוא **תשתית** שמשתמשת במודלים כמו Claude. ChatGPT הוא מוצר מוגמר. OpenClaw הוא הפלטפורמה שמאחורי הסצנות.

No. OpenClaw is infrastructure that uses models like Claude. ChatGPT is a finished product. OpenClaw is the platform behind the scenes.

שאלה: האם צריך לדעת לתכנת?

בסיס — כן. אבל Claude Code עצמו יכול לעזור לכם לשנות את OpenClaw מבלי שתצטרכו לדעת TypeScript מעמוק.

Basics — yes. But Claude Code itself can help you modify OpenClaw without deep TypeScript knowledge.

מה אפשר לבנות עם OpenClaw?

OpenClaw הוא כמו לגו חכם — אתם מחליטים מה לבנות. הנה כמה דוגמאות מהחיים האמיתיים:

🤖 עוזר אישי

- שלחו הודעת WhatsApp: *"@Andy, צור סיכום של ישיבת הצוות מה-Git history של השבוע"*
- OpenClaw מריץ סוכן, שולף את הלוג, מנתח ומחזיר סיכום מסודר

🤖 Personal Assistant

- Send a WhatsApp message: *"@Andy, summarize the team meeting from this week's Git history"*
- OpenClaw runs an agent, pulls the log, analyzes and returns a structured summary

📊 עוזר עסקי

- *"@Andy, שלח לי כל בוקר בשעה 9 סקירת צינור המכירות מ-CRM"*
- משימה מתוזמנת שרצה כל יום ומעדכנת אותכם ב-WhatsApp

📊 Business Assistant

- *"@Andy, send me a sales pipeline overview from CRM every morning at 9"*
- Scheduled task that runs daily and updates you on WhatsApp

🔬 מחקר אוטומטי

- *"@Andy, כל יום שני שלח לי עדכוני AI מ-Hacker News ו-TechCrunch"*
- סוכן גולש, מסנן ומסכם את החדשות הרלוונטיות ביותר

🔬 Automated Research

- *"@Andy, send me AI updates from Hacker News and TechCrunch every Monday"*
- Agent browses, filters and summarizes the most relevant news

👥 ניהול קבוצות

- כל קבוצת WhatsApp/Telegram מקבלת סוכן AI מבודד משלה
- זיכרון נפרד לכל קבוצה — אין דליפת מידע בין קבוצות

👥 Group Management

- Each WhatsApp/Telegram group gets its own isolated AI agent
- Separate memory per group — no data leakage between groups

🎉 המסקנה

OpenClaw הוא לא רק כלי — הוא **פלטפורמת AI** שמאפשרת לכם לבנות כל עוזר שאפשר לדמיין. בפרקים הבאים נלמד כיצד לעשות זאת בצורה נכונה, בטוחה ומהירה.

OpenClaw is not just a tool — it's an **AI platform** that lets you build any assistant you can imagine. In the next chapters we'll learn how to do this correctly, safely, and quickly.

📌 נקודות מפתח לפרק זה / Key Points

- OpenClaw הוא עוזר AI אישי ורב-ערוצי שרץ על השרת שלכם
- כל סוכן AI מבודד בקונטיינר Linux — בטיחות אמיתית
- תהליך Node.js יחיד — פשוט מספיק להבין
- ראשון בתחום לתמוך ב-Agent Swarms
- מבוסס Claude Agent SDK — המודל החכם ביותר

• Multi-channel AI assistant on your server • Container-isolated agents • Single Node.js process • First to support Agent Swarms • Built on Claude Agent SDK

CHAPTER ONE • פרק ראשון

התקנה וקונפיגורציה

Installation & Configuration

01

מדרישות המערכת ועד ההתחברות הראשונה — כל מה שצריך כדי להפעיל את OpenClaw בפעם הראשונה

From system requirements to first connection — everything needed to get OpenClaw running for the first time

CHAPTER 01

התקנה וקונפיגורציה

Installation & Configuration

⏱ זמן קריאה: 60 דקות

🎯 מטרות למידה / Learning Objectives

- להכין את סביבת הפיתוח המתאימה (macOS / Linux)
- להתקין ולהגדיר Docker או Apple Container
- לשכפל ולהגדיר את ה-OpenClaw repository
- לבצע את ה-`setup/` הראשוני דרך Claude Code
- לאמת שהמערכת פועלת ומחוברת

דרישות מערכת

רכיב	macOS (מומלץ)	Linux	Windows (WSL2)
מערכת הפעלה	macOS 14+ (Sonoma)	Ubuntu 22.04+ / Debian 12+	WSL2 + Ubuntu 22.04
Node.js	v20+ (LTS)	v20+ (LTS)	v20+ (LTS)
Claude Code	עדכני ביותר	עדכני ביותר	עדכני ביותר
קונטיינר	Apple Container *או* Docker	Docker Desktop / Engine	Docker Desktop (WSL2)
RAM מינימלי	8GB (16GB מומלץ)	8GB	16GB
דיסק	20GB פנוי	20GB פנוי	30GB פנוי

⭐ macOS הוא הבחירה המועדפת / macOS is the Preferred Choice

על macOS תוכלו להשתמש ב-**Apple Container** — כלי קל ומהיר שפותח על ידי Apple עצמה. הוא מהיר יותר מ-Docker ומשתלב טוב יותר עם מחשבי Mac.

*On macOS you can use **Apple Container** — a lightweight, fast tool developed by Apple itself. It's faster than Docker and integrates better with Mac machines.*

שלב 1: התקנת Node.js

ההדרך המומלצת להתקין Node.js היא דרך **nvm** (Node Version Manager) — שמאפשר ניהול גרסאות פשוט:

The recommended way to install Node.js is via **nvm** (Node Version Manager) — enabling simple version management:

```bash
# Install nvm (Node Version Manager)
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash

# Restart terminal, then install Node.js LTS
nvm install --lts
nvm use --lts

# Verify installation
node --version   # Should show v20.x.x or higher
npm --version    # Should show 10.x.x or higher
```

🛑 עצרו ובדקו / Stop and Verify

לפני שממשיכים — וודאו שפקודת `node --version` מחזירה **v20 ומעלה**. גרסאות ישנות יותר גורמות לבעיות בלתי צפויות.

*Before continuing — make sure `node --version` returns **v20 or higher**. Older versions cause unexpected issues.*

שלב 2: התקנת Claude Code

OpenClaw מנוהל דרך **Claude Code** — ה-CLI של Anthropic שמריץ סוכני AI.

```bash
# Install Claude Code globally
npm install -g @anthropic-ai/claude-code

# Verify
claude --version
# Expected: Claude Code CLI 2.x.x (darwin-arm64)
```

```bash - first login
# Start Claude Code and authenticate with Anthropic
claude

# Follow the authentication prompts
# You'll be redirected to claude.ai to authorize the CLI
# After auth, you'll see the Claude Code prompt: >
```

שלב 3: הגדרת קונטיינר

אפשרות א׳: Apple Container (macOS בלבד)

Apple Container הוא הפתרון הקל והמהיר ביותר למשתמשי Mac. הוא מנצל את ה-Virtualization Framework של Apple.

Option A: Apple Container (macOS only)

Apple Container is the lightest and fastest solution for Mac users. It leverages Apple's Virtualization Framework.

```bash - apple container
# Install Apple Container via Homebrew
brew install apple/container/container

# Or download from GitHub releases:
# https://github.com/apple/container/releases
```

```
# Verify
container --version
```

אפשרות ב׳: Docker (כל פלטפורמה)

Docker עובד על macOS, Linux ו-Windows (דרך WSL2). זהו הפתרון הנפוץ ביותר.

Option B: Docker (all platforms)

Docker works on macOS, Linux and Windows (via WSL2). This is the most common solution.

BASH – DOCKER

```
# macOS - install via Homebrew
brew install --cask docker

# Linux (Ubuntu/Debian)
curl -fsSL https://get.docker.com | sh
sudo usermod -aG docker $USER
newgrp docker

# Verify Docker works
docker run hello-world
```

⚙️ מאחורי הקלעים: למה בכלל קונטיינרים?

כש-OpenClaw מריץ סוכן AI, הסוכן יכול לבצע פקודות Bash — לגשת לקבצים, להריץ קוד, לגלוש באינטרנט. בלי קונטיינר, זה אומר שהסוכן יש לו גישה **מלאה** למחשב שלכם.

עם קונטיינר, הסוכן "חי" בסביבה מבודדת — כמו בית קופה שמוגבל לתא שלו. הוא יכול לגשת רק לתיקיות שהרכבתם (mounted) אליו במפורש. שאר המחשב — בלתי נגיש.

Without a container, the AI agent has full access to your machine. With a container, the agent "lives" in an isolated environment — only able to access explicitly mounted directories. The rest of your machine is inaccessible.

שלב 4: שכפול ה-Repository

```bash
# Clone OpenClaw repository
git clone https://github.com/openclaw/openclaw.git
cd openclaw

# Open Claude Code in the project directory
claude
```

שלב 5: הפעלת /setup

זה הרגע שבו ה-**AI מנהל את ההתקנה שלו עצמו** — עקרון ה-AI-Native בפעולה.

💻 תרגיל מעשי: ההתקנה הראשונה

1. פתחו טרמינל ונווטו לתיקיית ה-OpenClaw
2. הפעילו `claude` :Claude Code
3. בפרומפט של Claude, הקלידו: `setup/`
4. Claude יתחיל את תהליך ההגדרה האינטראקטיבי — ענו על שאלותיו
5. Claude יבנה את הקונטיינר, יגדיר את מסד הנתונים, ויחבר את הערוצים
6. בסיום, תראו: `OpenClaw is running and connected ✓`

⭐ מה /setup עושה בפועל?

- מתקין את כל ה-(`npm install`) dependencies
- בונה את קונטיינר ה-(`./container/build.sh`) Linux
- יוצר קובץ `env.` עם המפתחות הנדרשים
- מגדיר את מסד הנתונים SQLite
- מגדיר את שירות ה-launchd (macOS) / systemd (Linux)
- מנחה אתכם לחבר ערוץ ראשון (WhatsApp, Telegram, וכו')

Installs dependencies • Builds Linux container • Creates .env file • Sets up SQLite database • Configures launchd/systemd service • Guides you to connect first channel

❌ שגיאה נפוצה: "Docker daemon is not running"

אם קיבלתם שגיאה זו, פתחו את Docker Desktop וחכו עד שה-daemon מוכן (הסמל בסרגל יהיה ירוק), ואז נסו שוב.

If you get this error, open Docker Desktop and wait until the daemon is ready (icon turns green), then try again.

❌ שגיאה נפוצה: "Permission denied" בעת בניית הקונטיינר

ב-Linux, ודאו שהמשתמש שלכם מוסף לקבוצת `docker`: `sudo usermod -aG docker $USER` ואז הפעילו מחדש את הטרמינל.

On Linux, ensure your user is added to the `docker` group: `sudo usermod -aG docker $USER` then restart terminal.

שלב 6: הפעלת שירות OpenClaw

macOS (launchd) **macOS (launchd)**

BASH - MACOS

```
# Start OpenClaw as a background service
launchctl load ~/Library/LaunchAgents/com.openclaw.plist

# Restart
launchctl kickstart -k gui/$(id -u)/com.openclaw

# Stop
launchctl unload ~/Library/LaunchAgents/com.openclaw.plist

# Check status
launchctl list | grep openclaw
```

BASH - LINUX (SYSTEMD)

```
# Start OpenClaw as a user service
systemctl --user start openclaw

# Enable on boot
systemctl --user enable openclaw

# Check status
systemctl --user status openclaw

# View logs
journalctl --user -u openclaw -f
```

BASH - DEVELOPMENT MODE

```
# Run in development mode with hot reload
npm run dev

# Or build and run
```

```
npm run build
node dist/index.js
```

אימות ההתקנה

אחרי שה-setup הושלם, תוכלו לאמת שהכל עובד:

```
BASH - VERIFY
# Check OpenClaw logs
tail -f ~/openclaw/logs/openclaw.log

# Expected output:
# [INFO] OpenClaw v2.x.x starting...
# [INFO] SQLite database initialized
# [INFO] Channel: WhatsApp connected
# [INFO] Container runtime: Apple Container / Docker ready
# [INFO] Polling loop started (interval: 3s)
# [INFO] OpenClaw is running ✓
```

❓ שאלות נפוצות / FAQ

שאלה: כמה זמן לוקח /setup?

בין 5 ל-15 דקות, תלוי במהירות האינטרנט ובחומרה שלכם. בניית קונטיינר Docker היא השלב הארוך ביותר.

Between 5-15 minutes depending on internet speed and hardware. Container build is the longest step.

שאלה: האם /setup ניתן להרצה מחדש אם שגיאה?

כן! /setup בטוח להרצה מרובה. Claude מזהה מה כבר מוגדר ומדלג על אותם שלבים.

Yes! /setup is safe to run multiple times. Claude detects what's already configured and skips those steps.

שאלה: מה אם אין לי Anthropic API key?

תצטרכו ליצור חשבון ב-console.anthropic.com ולהוציא API key. קיים tier חינמי עם מגבלות שימוש.

Create an account at console.anthropic.com and generate an API key. A free tier exists with usage limits.

📌 נקודות מפתח לפרק זה

- Node.js 20+ ו-Claude Code הם הדרישות המינימליות
- Apple Container מהיר יותר מ-Docker על macOS
- `setup/` הוא הסקיל שמנהל את כל ההתקנה — הפעילו אותו ב-Claude Code
- OpenClaw רץ כשירות רקע (launchd / systemd)

- לוגים זמינים ב-`~/openclaw/logs/openclaw.log`

CHAPTER TWO • פרק שני

ארכיטקטורה עמוקה

Deep Architecture

02

צלילה מעמיקה לתוך הקוד של OpenClaw — כיצד כל חלק עובד ומתקשר עם האחרים

A deep dive into OpenClaw's code — how each part works and communicates with the others

CHAPTER 02

ארכיטקטורה עמוקה

Deep Architecture Dive

⏱ זמן קריאה: 75 דקות

מפת קבצי הליבה

OpenClaw הוא כלי כה פשוט שניתן לקרוא את כל קוד הליבה שלו בשעה אחת. הנה מפה של הקבצים החשובים ביותר:

קובץ	מיקום	תפקיד
`index.ts`	`/src`	האורקסטרטור המרכזי — לולאת הפולינג וניהול המצב
`registry.ts`	`/src/channels`	רישום ערוצים אוטומטי בהפעלה
`ipc.ts`	`/src`	תקשורת IPC בין הקונטיינר לתהליך הראשי
`router.ts`	`/src`	ניתוב תגובות לערוצים הנכונים
`container-runner.ts`	`/src`	הפעלת קונטיינרים עם Claude Agent SDK
`task-scheduler.ts`	`/src`	הפעלת משימות מתוזמנות
`db.ts`	`/src`	פעולות SQLite — הודעות, קבוצות, sessions
`group-queue.ts`	`/src`	תור הודעות לפי קבוצה עם בקרת תאימות
`CLAUDE.md`	`/*/groups`	זיכרון ה-AI לכל קבוצה — מבודד לחלוטין
`config.ts`	`/src`	מחרוזת הטריגר, נתיבים, מרווחי זמן

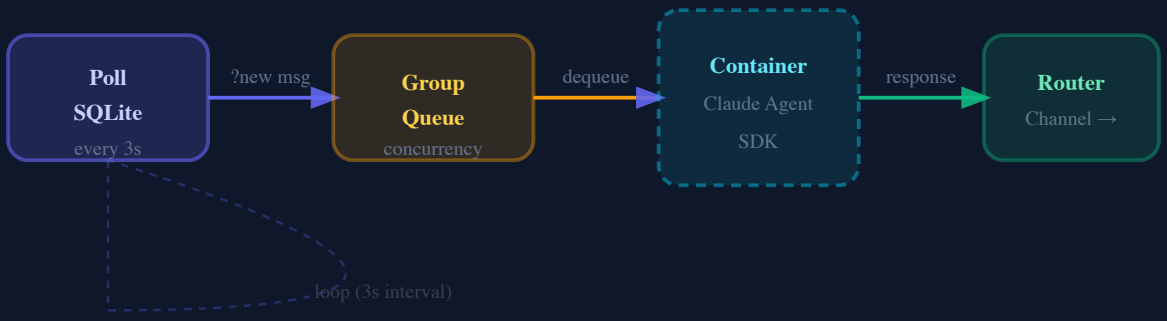
⭐ קוראים את הקוד? / Reading the Code?

אם אתם רוצים לקרוא ולהבין את כל הקוד, התחילו מ-`src/index.ts`. זה הקובץ שמחבר את כל החלקים.
מרגע שמבינים אותו — הכל נופל למקומו.

If you want to read and understand all the code, start with `src/index.ts`. It's the file that connects all the pieces. Once you understand it — everything falls into place.

לולאת הפולינג (Polling Loop)

הלב הפועם של OpenClaw הוא **לולאת פולינג** — לולאה שרצה כל 3 שניות ובודקת אם יש הודעות חדשות ב-SQLite.



איור 2.1: לולאת הפולינג של OpenClaw | The OpenClaw polling loop

TYPESCRIPT — SRC/INDEX.TS (SIMPLIFIED)

```
async function pollLoop() {
  while (running) {
    // 1. Get pending messages from SQLite
    const messages = await db.getPendingMessages();

    // 2. Route each message through group queue
    for (const msg of messages) {
      groupQueue.enqueue(msg.groupId, async () => {
        // 3. Run agent in container
        const response = await containerRunner.run(msg);
        // 4. Send response back to channel
        await router.send(msg.channel, response);
      });
    }

    // Wait 3 seconds before next poll
    await sleep(3000);
```

```
    }
}
```

📖 Group Queue — תור הקבוצות

Group Queue = per-group message queue with concurrency control

כל קבוצה מקבלת תור הודעות עצמאי. אם הסוכן עדיין מטפל בהודעה קודמת — ההודעה הבאה ממתינה בתור.
זה מונע עומס יתר ומבטיח תגובות מסודרות.

מסד הנתונים: SQLite

OpenClaw משתמש ב-**SQLite** — מסד נתונים קל ומהיר שרץ בתהליך אחד, ללא שרת נפרד. זו בחירה מכוונת: אין צורך ב-SQLite — PostgreSQL / Redis מספיק לעומס של שימוש אישי.

הטבלאות הראשיות:

- **messages** — כל ההודעות הנכנסות
- **groups** — מטא-דאטה על קבוצות
- **sessions** — sessions פעילים של Claude
- **state** — מצב הסוכן הנוכחי
- **scheduled_tasks** — משימות מתוזמנות

Main Tables:

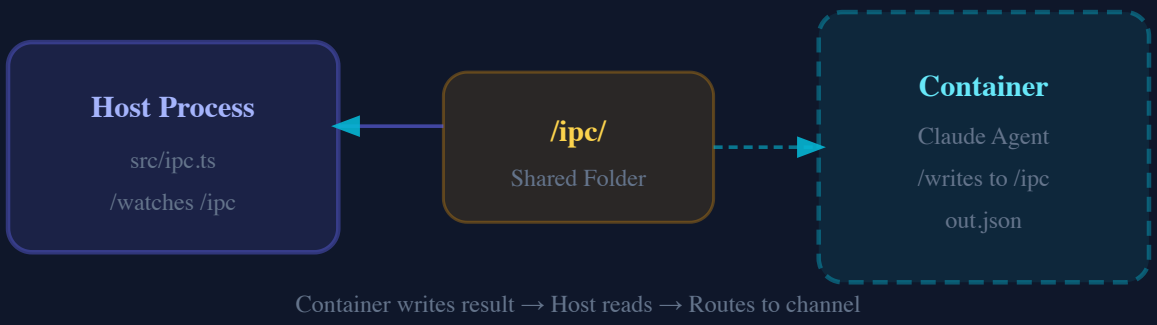
- **messages** — all incoming messages
- **groups** — group metadata
- **sessions** — active Claude sessions
- **state** — current agent state
- **scheduled_tasks** — scheduled tasks

TYPESCRIPT — SRC/DB.TS (KEY OPERATIONS)

```typescript
// Store incoming message
async function saveMessage(
  groupId: string,
  channel: string,
  content: string,
  sender: string
): Promise<void>

// Get messages awaiting processing
async function getPendingMessages(): Promise<Message[]>

// Mark message as processed
async function markProcessed(messageId: string): Promise<void>

// Get or create group record
async function getOrCreateGroup(groupId: string): Promise<Group>
```

מערכת ה-IPC

כשהסוכן רץ בתוך קונטיינר מבודד, הוא צריך דרך לתקשר עם התהליך הראשי. OpenClaw פותר זאת עם **IPC דרך filesystem** — פתרון פשוט ואמין.



איור 2.2: תקשורת IPC דרך filesystem | IPC communication via filesystem

⚙️ למה IPC דרך filesystem ולא WebSocket?

WebSocket דורש פתיחת פורטים ונדרש לניהול חיבורים. Filesystem IPC הוא פשוט יותר, ניתן לדיבוג ידני, ולא דורש ניהול state של חיבור. בנוסף, זה עובד גם כש-Claude כותב פלט streams בצורה הדרגתית.

WebSocket requires open ports and connection management. Filesystem IPC is simpler, manually debuggable, and requires no connection state management. It also works when Claude writes streaming output incrementally.

CHAPTER THREE • פרק שלישי

ערוצי תקשורת

Communication Channels

03

WhatsApp, Telegram, Discord, Slack ו-Gmail — כיצד כל ערוץ עובד,
כיצד מחברים אותו, ומה ניתן לעשות איתו

WhatsApp, Telegram, Discord, Slack and Gmail — how each channel
works, how to connect it, and what you can do with it

CHAPTER 03

ערוצי תקשורת

Communication Channels

⏱ זמן קריאה: 80 דקות

🎯 מטרות למידה

- להבין כיצד מערכת הערוצים של OpenClaw עובדת
- לחבר לפחות ערוץ אחד (WhatsApp / Telegram)
- להבין את עקרון ה-self-registration של ערוצים
- לדעת כיצד לנפות שגיאות חיבור לערוצים

איך ערוצים עובדים ב-OpenClaw?

OpenClaw משתמש ב-**מערכת ערוצים מודולרית**. כל ערוץ הוא סקיל (`claude/skills/add-[channel].`) שניתן להוסיף לפי הצורך. בהפעלה, כל ערוץ שיש לו credentials — מרשם את עצמו אוטומטית.

TYPESCRIPT — SRC/CHANNELS/REGISTRY.TS

```
// Channels self-register at startup
const registry = new ChannelRegistry();

// Each channel checks for its own credentials
// If found → registers itself. If not → skips silently.
await registry.loadChannels([
  WhatsAppChannel,
  TelegramChannel,
  DiscordChannel,
  SlackChannel,
  GmailChannel,
]);
```

```
// Result: only channels WITH credentials are active
console.log(registry.getActive());
// → ['whatsapp', 'telegram'] (if only these have credentials)
```

⭐ הכוח של Self-Registration

אתם לא צריכים לערוך קובץ config כדי לפעיל ערוץ. פשוט הוסיפו את ה-credentials הנכונים ב-`env.` — הערוץ יתחבר בפעם הבאה שתפעילו את OpenClaw.

You don't need to edit a config file to enable a channel. Just add the right credentials in `.env` — the channel will connect next time you start OpenClaw.

ערוץ WhatsApp

WhatsApp Channel

via @whiskeysockets/baileys (unofficial API)

ערוץ WhatsApp מאפשר לכם לשוחח עם עוזר ה-AI שלכם ישירות דרך WhatsApp — ממחשב ומסמארטפון. תומך בהודעות טקסט, קול ותמונות.

הוספת /add-whatsapp — WhatsApp

1. ב-Claude Code, הפעילו: `add-whatsapp/`
2. Claude יתקין את ה-Baileys library ויגדיר את הערוץ
3. תתבקשו לסרוק QR Code דרך WhatsApp → Linked Devices
4. אחרי הסריקה, הסשן נשמר — לא צריך לסרוק שוב
5. שלחו הודעה לעצמכם (ה-self-chat): `@Andy שלום!`

ENV — .ENV

```
# WhatsApp credentials are stored automatically after QR scan
# No manual configuration needed in .env
# Session files stored in: ./auth/whatsapp/

# Optional: change the trigger word (default: @Andy)
TRIGGER_WORD=@Andy
```

❌ "WhatsApp not connecting after upgrade"

WhatsApp הוא ערוץ נפרד שאינו מובנה ב-core. אם שדרגתם ו-WhatsApp לא מתחבר — הפעילו `add-/whatsapp` מחדש. ה-credentials הקיימים נשמרים ולא יאבדו.

WhatsApp is a separate channel not bundled in core. If it stopped after upgrade — run `/add-whatsapp` again. Existing auth credentials are preserved.

ערוץ Telegram

✈️ Telegram Channel

via Official Telegram Bot API

Telegram מציע API רשמי ויציב יותר מ-WhatsApp. אידיאלי לקבוצות גדולות ולמשתמשים שמעדיפים API רשמי. תומך בפקודות, keyboards, ופורמט Markdown.

💻 הוספת Telegram — /add-telegram

1. פתחו Telegram ושוחחו עם **@BotFather**
2. שלחו `/newbot` ועקבו אחרי ההוראות
3. קבלו Bot Token בפורמט: `123456789:ABCdef...`
4. ב-Claude Code: `/add-telegram` ועקבו אחרי ההוראות
5. הוסיפו את ה-Bot לקבוצה ושלחו `@YourBot שלום!`

ערוץ Discord

Discord Channel

via Discord.js + Bot API

Discord אידיאלי לקהילות מפתחים, גיימרים ומשתמשים טכנולוגיים. תומך ב-slash commands, embeds, ו-voice channels. כל server מקבל סוכן AI מבודד.

ENV — DISCORD CONFIGURATION

```
# Discord Bot Token (from Discord Developer Portal)
DISCORD_BOT_TOKEN=your_bot_token_here

# Optional: restrict to specific guild IDs
DISCORD_ALLOWED_GUILDS=guild_id_1,guild_id_2
```

ערוץ Slack

Slack Channel

via Slack Bolt SDK

Slack הוא ערוץ אידיאלי לשימוש ארגוני. תומך ב-Blocks, Modals, ו-Interactive components. כל workspace מקבל context מבודד.

ENV — SLACK CONFIGURATION

```
# Slack App credentials (from api.slack.com)
SLACK_BOT_TOKEN=xoxb-your-token
SLACK_SIGNING_SECRET=your-signing-secret
SLACK_APP_TOKEN=xapp-your-app-token  # For Socket Mode
```

ערוץ Gmail

Gmail Channel

via Gmail API + OAuth2

Gmail מאפשר לעוזר AI לקרוא, לענות ולשלוח מיילים. אידיאלי לאוטומציה של כתיבת מיילים, סיכום תיבת דואר, וניהול newsletters.

```
BASH — ADDING GMAIL
# Run the Gmail setup skill
/add-gmail

# Claude will guide you through:
# 1. Creating a Google Cloud project
# 2. Enabling Gmail API
# 3. OAuth2 credentials setup
# 4. First authorization flow
```

❓ איזה ערוץ מומלץ להתחיל איתו?

לשימוש אישי יומיומי:

WhatsApp — הוא כבר פתוח על הטלפון כל הזמן, קל להשתמש, ומרגיש טבעי.

לפרויקטים טכנולוגיים:

Telegram — API רשמי, יציב, ותומך ב-Bot features מתקדמים.

לשימוש ארגוני:

Slack — אינטגרציה מלאה עם workflow הארגוני, תמיכה ב-Blocks מתקדמים.

📌 נקודות מפתח — ערוצים

- כל ערוץ הוא סקיל שניתן להוסיף ולהסיר בנפרד
- ערוצים מרשמים את עצמם אוטומטית אם יש credentials
- כל ערוץ מקבל thread/group עצמאי עם זיכרון מבודד
- ניתן להפעיל מספר ערוצים בו-זמנית

CHAPTER FOUR • פרק רביעי

בידוד קונטיינרים ואבטחה

Container Isolation & Security

04

המודל האבטחתי של OpenClaw — למה בידוד קונטיינרים חזק יותר מ-permission checks, וכיצד מערכת ה-mounting מגנה על הנתונים שלכם

OpenClaw's security model — why container isolation is stronger than permission checks, and how the mounting system protects your data

CHAPTER 04

בידוד קונטיינרים ואבטחה

Container Isolation & Security

⏱ זמן קריאה: 60 דקות

🎯 מטרות למידה

- להבין את ההבדל בין אבטחה ברמת אפליקציה לבין בידוד ברמת מערכת ההפעלה
- להכיר את מודל ה-mounting ואיזה קבצים הסוכן יכול לראות
- לדעת כיצד להרחיב גישת הסוכן לתיקיות נוספות בצורה בטוחה
- להבין את מגבלות האבטחה ומה הן לא מכסות

שני סוגי אבטחה

🔓 אבטחה ברמת אפליקציה

רוב כלי ה-AI מסתמכים על **בדיקות הרשאות** — רשימות לבנות, קודי אישור, חוקים ב-prompt. אבל כל אלה ניתנים לעקיפה: פגם בלוגיקה, prompt injection, באג בקוד — והסוכן פועל מחוץ לגבולות.

🔓 Application-level Security

Most AI tools rely on **permission checks** — allowlists, pairing codes, prompt rules. But all of these can be bypassed: logic flaw, prompt injection, code bug — and the agent operates outside boundaries.

🔓 בידוד ברמת מערכת הפעלה (OpenClaw)

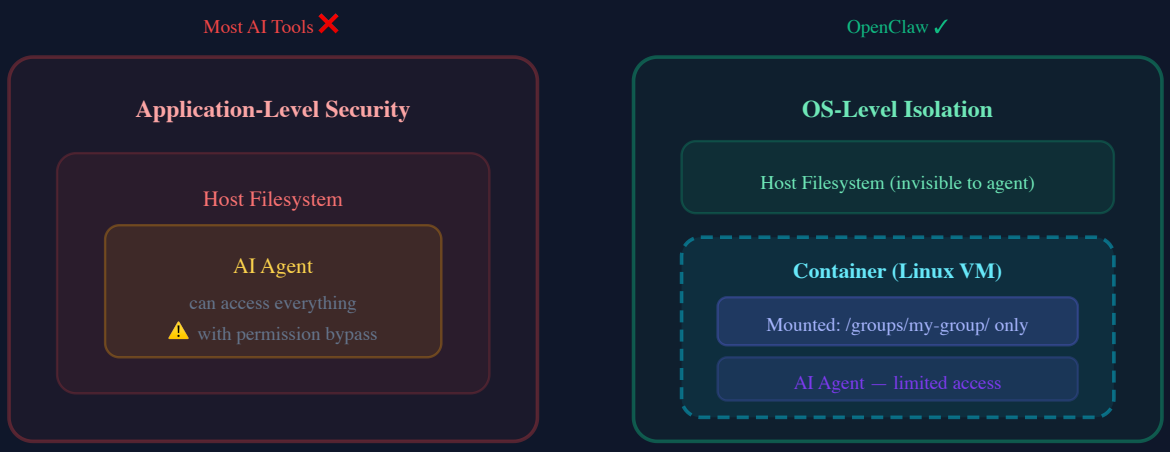
OpenClaw משתמש ב**בידוד אמיתי של מערכת ההפעלה** — קונטיינר Linux. הסוכן לא יכול לגשת

🔓 OS-level Isolation (OpenClaw)

OpenClaw uses **true OS-level isolation** — a Linux container. The agent cannot access files

outside the container, even if it wants to. The boundary is in the kernel, not in logic.

לקבצים שמחוץ לקונטיינר, גם אם רוצה. הגבול הוא ב-kernel, לא בלוגיקה.



איור 4.1: הבדל בין אבטחה ברמת אפליקציה לבין בידוד ברמת מערכת הפעלה

מודל ה-Mounting — מה הסוכן רואה?

כש-OpenClaw מפעיל קונטיינר לטיפול בהודעה מקבוצה `my-group`, הוא עושה **mount** לתיקיות ספציפיות בלבד:

```
CONTAINER MOUNT CONFIGURATION
# What the container sees:
/groups/my-group/          # ← Group context (CLAUDE.md, files)
/tmp/claude-work/          # ← Temporary work area
/usr/local/bin/            # ← Standard tools (git, curl, etc.)

# What the container CANNOT see:
/Users/you/Documents/      # ← Your documents (blocked)
/Users/you/.ssh/           # ← SSH keys (blocked)
/Users/you/other-project/  # ← Other projects (blocked)
~/.env                     # ← Environment secrets (blocked)
```

הרחבת גישה לתיקיות נוספות

אם אתם רוצים שהסוכן יוכל לגשת לתיקייה נוספת (למשל: vault של Obsidian, תיקיית פרויקט), תוכלו להוסיף אותה ל-mount:

```
CLAUDE.MD — GRANTING ADDITIONAL ACCESS
# In your group's CLAUDE.md, tell Claude:
# "I want you to have access to my Obsidian vault"

# Claude will modify container-runner.ts to add:
mounts: [
  // existing mounts...
  {
    host: '/Users/ariel/Obsidian/MyVault',
    container: '/obsidian',
    readonly: true   // read-only for safety
  }
]
```

⭐ readonly=true — ברירת מחדל בטוחה

כשאתם מרכיבים תיקייה חיצונית, תמיד הגדירו `readonly: true` אלא אם יש סיבה ספציפית לכתיבה. כך הסוכן יכול לקרוא את הקבצים מבלי שיוכל לשנות אותם בטעות.

When mounting an external directory, always set `readonly: true` unless there's a specific reason for write access. This lets the agent read files without accidentally modifying them.

מודל האבטחה המלא

סוג גישה	האם מותר?	פירוט
קריאת קבצים בתיקיית הקבוצה	✓ מותר	הסוכן רואה את `/groups/[name]/` במלואה
כתיבה לתיקיית הקבוצה	✓ מותר	הסוכן יכול לשמור קבצים בתיקיית הקבוצה
גישה לתיקיות host שלא mounted	✗ חסום	בלתי אפשרי — kernel block
גלישה באינטרנט	✓ מותר	curl, fetch, npm — זמינים בקונטיינר
הרצת קוד שרירותי	⚠ בקונטיינר בלבד	הקוד רץ בתוך הקונטיינר, לא על ה-host
גישה ל-SSH keys / secrets	✗ חסום	לא mounted לקונטיינר
קריאת CLAUDE.md של קבוצה אחרת	✗ חסום	כל קבוצה מקבלת mount נפרד
גישה לרשת פנימית (LAN)	⚠ תלוי בהגדרה	ניתן להגביל דרך Docker network policy

📌 נקודות מפתח — אבטחה

- OpenClaw משתמש ב**בידוד OS אמיתי**, לא רק permission checks
- הסוכן רואה **רק** את התיקיות שמורכבות (mounted) אליו
- כל קבוצה מבודדת מהאחרות — אין דליפת מידע
- כדאי להגדיר `readonly: true` לתיקיות חיצוניות
- הקוד הוא קטן מספיק כדי שתוכלו לקרוא ולאמת את מודל האבטחה בעצמכם

CHAPTER FIVE • פרק חמישי

זיכרון וקונטקסט

Memory & Context System

05

קובץ CLAUDE.md הוא "מוח" הסוכן שלכם — כיצד לכתוב אותו, מה
לשים בו, וכיצד לנצל את מערכת הזיכרון המבודדת של OpenClaw

CLAUDE.md is your agent's "brain" — how to write it, what to include, and
how to leverage OpenClaw's isolated memory system

CHAPTER 05

זיכרון וקונטקסט

Memory & Context System

⏱ זמן קריאה: 55 דקות

קובץ CLAUDE.md — מה זה בכלל?

כל קבוצה ב-OpenClaw מכילה קובץ `CLAUDE.md`. זהו קובץ Markdown פשוט שנטען לכל conversation עם הסוכן — ומכיל את **כל מה שהסוכן צריך לדעת** על הקונטקסט שלו.

חשבו על CLAUDE.md כמו **תדריך** שנותנים לעובד חדש ביומו הראשון. הוא מכיל:

- מי אתם ומה תפקידו של הסוכן
- כיצד להתנהג ובאיזה טון לדבר
- מידע על הפרויקטים שלכם
- הוראות ספציפיות לקבוצה זו
- זיכרון מצטבר שהסוכן עדכן בעצמו

Think of CLAUDE.md as a **briefing document** given to a new employee on their first day. It contains:

- Who you are and what the agent's role is
- How to behave and what tone to use
- Information about your projects
- Group-specific instructions
- Accumulated memory the agent updated itself

MARKDOWN — GROUPS/WORK-TEAM/CLAUDE.MD (EXAMPLE)

```
# Work Team Assistant

## Identity
You are Andy, the AI assistant for Ariel Eisenstadt's work team.
Ariel is the founder of PixMind Studio and PixiBot startup.
Always respond in Hebrew unless the message is in English.

## Context
- PixMind Studio: video production company, clients include top Israeli brands
```

```
- PixiBot: AI startup in development, focus on video automation
- Team size: 3 people (Ariel + 2 team members)

## Behavior
- Be concise and direct — Ariel prefers short answers
- Use bullet points for lists
- For technical questions, provide code examples
- Always check git history before making assumptions about project state

## Important Files
- /obsidian/Projects/ — project notes (read-only)
- /groups/work-team/tasks.md — current task list

## Memory (auto-updated)
- Last updated: 2026-03-23
- Current priority: PixiBot MVP development
- Preferred model: claude-sonnet-4-6
```

ארכיטקטורת הזיכרון

OpenClaw תומך בשלושה רמות זיכרון — כל אחת עם מטרה שונה:

Global Memory

`groups/global/CLAUDE.md`

מידע שרלוונטי לכל הסוכנים בכל הקבוצות. לדוגמה: שמכם, העדפות כלליות, הוראות בסיסיות.

Group Memory

`groups/[name]/CLAUDE.md`

מידע ספציפי לקבוצה. כל קבוצה מבודדת — הסוכן בקבוצה A לא קורא את הזיכרון של קבוצה B.

File Memory

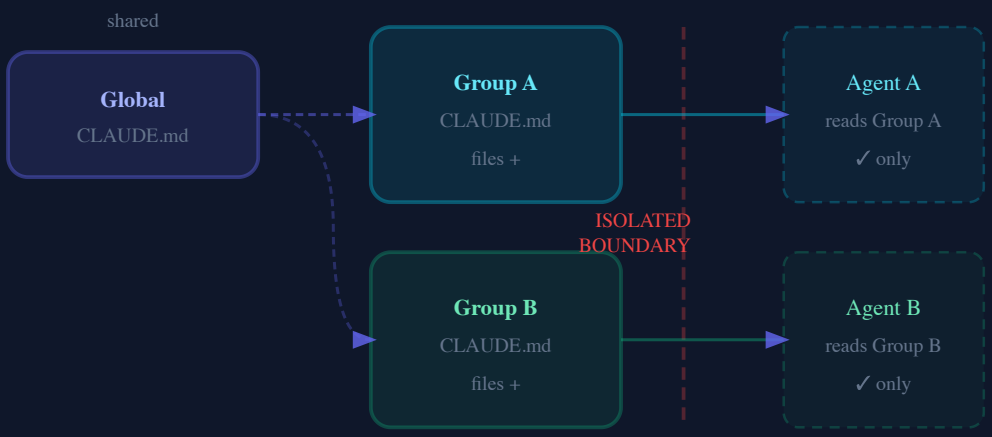
קבצים נוספים ב-`/groups/[name]`

הסוכן יכול ליצור ולעדכן קבצים נוספים — רשימות, הערות, ורשומות. הכל נשמר ב-filesystem.

Session Memory

SQLite sessions table

זיכרון ה-conversation הנוכחי — מה נאמר בשיחה הנוכחית. נמחק כשה-session מסתיים.



איור 5.1: מערכת זיכרון מבודדת לפי קבוצות | Isolated per-group memory system

⭐ הסוכן מעדכן את הזיכרון שלו עצמאית

אתם יכולים להגיד לסוכן: *"זכור לעתיד שאני מעדיף תגובות קצרות"* — והוא יעדכן את `CLAUDE.md` שלו בעצמו. בפעם הבאה, הוא יזכור.

You can tell the agent: "Remember for the future that I prefer short responses" — and it will update its own `CLAUDE.md`. Next time, it will remember.

CHAPTER SIX • פרק שישי

מערכת הסקילס

The Skills System

06

הסקיל הוא יחידת ה-customization של OpenClaw — כיצד לכתוב סקיל,
להפעיל אותו, ולהפוך אותו לתרומה שאחרים יוכלו להשתמש בה

A skill is OpenClaw's customization unit — how to write one, run it, and turn it into a contribution others can use

CHAPTER 06

מערכת הסקילס

The Skills System

⏱ זמן קריאה: 65 דקות

🎯 מטרות למידה

- להבין מהו סקיל ב-OpenClaw ומה ההבדל מ-feature
- לדעת כיצד להפעיל סקיל קיים (`/skill-name`)
- לכתוב סקיל בסיסי משלכם
- להבין מה הופך סקיל לטוב וראוי לתרומה

סקיל vs פיצ'ר — מה ההבדל?

פיצ'ר (Feature)

קוד שנוסף לקוד הבסיס ומתקבל על ידי כולם. אם OpenClaw היה מוסיף Telegram כפיצ'ר — **כולם** היו מקבלים את הקוד של Telegram, גם מי שלא צריך אותו.

זה מוביל ל-bloat — קוד שאתם לא מבינים ולא צריכים.

Feature

Code added to the base codebase, received by everyone. If OpenClaw added Telegram as a feature — **everyone** would get Telegram code, even those who don't need it.

This leads to bloat — code you don't understand and don't need.

סקיל (Skill) ✓

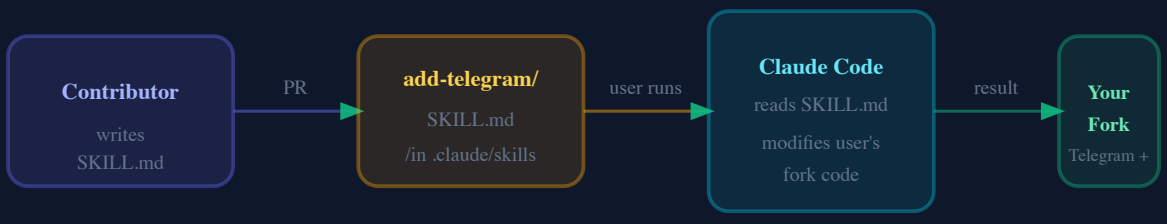
Skill ✓

A `SKILL.md` file containing **instructions for Claude Code** on how to modify your fork. When you run `/add-telegram` — Claude makes the changes in your code.

You get clean code that does exactly what you need, nothing more.

קובץ `SKILL.md` שמכיל **הוראות לסוכן AI** (Claude Code) כיצד לשנות את ה-Fork שלכם. כשאתם מפעילים `/add-telegram` — Claude מבצע את השינויים בקוד שלכם.

אתם מקבלים קוד נקי שעושה בדיוק מה שצריך, ולא יותר.



איור 6.1: מחזור חיים של סקיל — מתרומה ועד לשינוי בקוד שלכם

כתיבת סקיל — מבנה SKILL.md

MARKDOWN — .CLAUDE/SKILLS/ADD-SIGNAL/SKILL.MD

```
# Skill: /add-signal

## Purpose
Add Signal as a communication channel to this OpenClaw installation.

## When to Run
When the user wants to receive and respond to Signal messages.

## Prerequisites
- signal-cli installed on the host
- Phone number registered with Signal
- Java 17+ available

## Steps

### 1. Install signal-cli dependency
Run: `npm install @signal-bot/signal-cli-rest-api`

### 2. Create the channel file
Create `src/channels/signal.ts` with the following structure:
- Implement the Channel interface
- Listen to Signal messages via signal-cli REST API
- Call `registry.register(this)` on startup
- Store auth credentials in `.env` as SIGNAL_PHONE_NUMBER

### 3. Update channel registry
Add SignalChannel to the imports in `src/channels/registry.ts`

### 4. Update .env
Add: SIGNAL_PHONE_NUMBER=+972501234567

### 5. Rebuild
Run: `npm run build`

## Testing
Send a Signal message to the registered number: "@Andy שלום"
Expected: Response from Claude within 10 seconds
```

⭐ **הסקיל הוא הוראות — לא קוד!**

שימו לב: ה-SKILL.md **לא מכיל קוד TypeScript** — הוא מכיל הוראות ל-Claude. Claude הוא שכותב את הקוד בהתאם להוראות. זה אומר שכל הוראה ברורה = קוד טוב.

*Notice: SKILL.md does **not contain TypeScript code** — it contains instructions for Claude Code. Claude is the one who writes the code based on the instructions. This means clear instructions = good code.*

סקילים מובנים ב-OpenClaw

סקיל	מטרה
`setup/`	התקנה ראשונית מלאה
`add-whatsapp/`	הוספת ערוץ WhatsApp
`add-telegram/`	הוספת ערוץ Telegram
`add-discord/`	הוספת ערוץ Discord
`add-slack/`	הוספת ערוץ Slack
`add-gmail/`	הוספת ערוץ Gmail
`customize/`	הנחיה לשינויים מותאמים אישית
`debug/`	איבחון בעיות בקונטיינר ולוגים
`convert-to-apple-container/`	מעבר מ-Docker ל-Apple Container

📌 **נקודות מפתח — סקילס**

- סקיל = קובץ הוראות Markdown, לא קוד
- Claude Code מבצע את הסקיל ושונה את ה-Fork שלכם
- סקיל מוסיף יכולת חד-פעמית — לא מוסיף bloat לקוד הבסיס

- כל אחד יכול לכתוב סקיל ולתרום לפרויקט

CHAPTER SEVEN • פרק שביעי

משימות מתוזמנות

07

Scheduled Tasks

הפכו את העוזר שלכם לאוטומציה שמשרתת אתכם 24/7 — ללא שאתם צריכים לשלוח הודעה

Turn your assistant into automation that serves you 24/7 — without you needing to send a message

CHAPTER 07

משימות מתוזמנות

Scheduled Tasks

⏱ זמן קריאה: 50 דקות

מהי משימה מתוזמנת?

משימה מתוזמנת היא **prompt שרץ אוטומטית** לפי לוח זמנים — ללא שתשלחו הודעה. הסוכן מבצע את המשימה ומחזיר את התוצאות לערוץ שציינתם.

💻 יצירת משימה מתוזמנת — דרך chat

הדרך הקלה ביותר: פשוט תגידו לעוזר שלכם:

```
WHATSAPP MESSAGE
@Andy, every weekday morning at 8:30am,
compile the top 5 AI news stories from
Hacker News and send them to me here.
```

Claude יוצר את המשימה, שומר אותה ב-SQLite, ומתחיל להריץ אותה.

ניהול משימות מתוזמנות

```
CHAT COMMANDS — MANAGING SCHEDULED TASKS
# List all scheduled tasks
@Andy, list all scheduled tasks

# Pause a task
@Andy, pause the morning briefing task
```

```
# Delete a task
@Andy, delete the weekly report task

# Run a task manually right now
@Andy, run the news briefing task now

# View task history
@Andy, show me the last 5 runs of the news task
```

הארכיטקטורה של Task Scheduler

TYPESCRIPT — SRC/TASK-SCHEDULER.TS (SIMPLIFIED)

```typescript
class TaskScheduler {
  async tick() {
    // Check due tasks every minute
    const dueTasks = await db.getDueTasks(now);

    for (const task of dueTasks) {
      // Inject task prompt as if user sent it
      await db.saveMessage({
        groupId: task.groupId,
        channel: task.channel,
        content: task.prompt,
        sender: 'system:scheduler',
        isScheduled: true
      });

      // Update next run time
      await db.updateNextRun(task.id, task.cronExpr);
    }
  }
}
```

⭐ דוגמאות לתזמון — Cron Expressions

תיאור	Cron Expression
כל יום ב-9:00	`* * * 9 0`
ימי חול ב-8:30	`1-5 * * 8 30`
כל יום ראשון ב-18:00	`0 * * 18 0`

כל שעה	`0 * * * *`
כל 15 דקות	`*/15 * * * *`
ה-1 לכל חודש	`0 9 1 * *`

CHAPTER EIGHT • פרק שמיני

נחילי סוכנים

08

Agent Swarms

הפיצ'ר שהפך את OpenClaw לראשון בתחומו — צוותי AI שעובדים יחד
על משימות מורכבות, כל סוכן עם תפקיד ייחודי

The feature that made OpenClaw first in its field — AI teams working together on complex tasks, each agent with a unique role

CHAPTER 08

נחילי סוכנים

Agent Swarms

⏱ זמן קריאה: 70 דקות

🎯 מטרות למידה

- להבין מהו Agent Swarm וכיצד הוא שונה מסוכן יחיד
- לדעת מתי להשתמש ב-Swarm ומתי זה מיותר
- להבין כיצד סוכנים מתקשרים ביניהם ב-OpenClaw
- לבנות את ה-Swarm הראשון שלכם

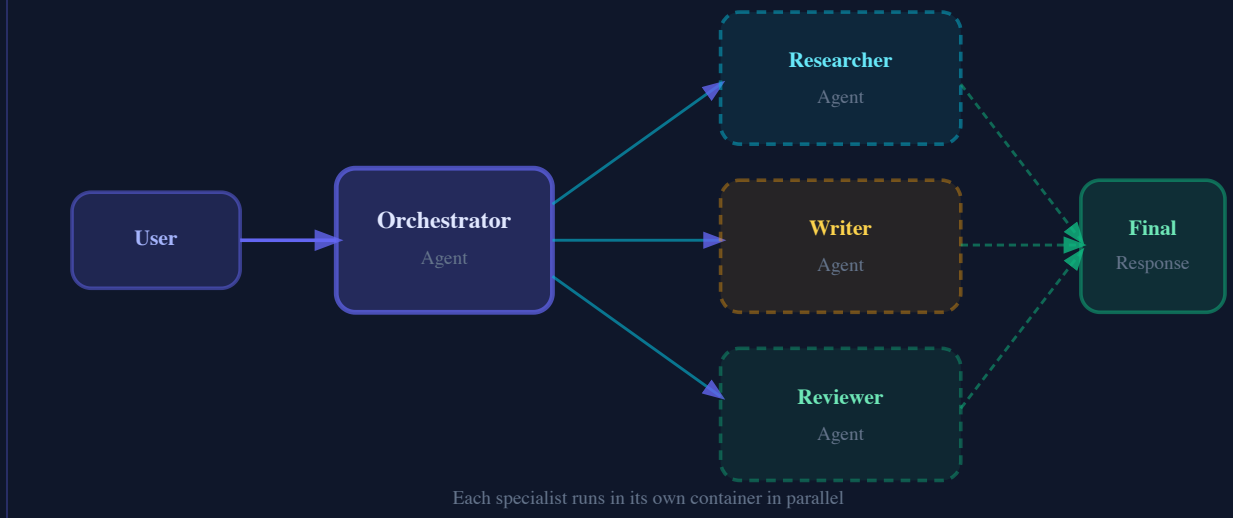
מה זה Agent Swarm?

Agent Swarm ("נחיל סוכנים") הוא קבוצת סוכני AI שעובדים **בו-זמנית ובשיתוף פעולה** על משימה אחת מורכבת. במקום שסוכן אחד יעשה הכל, כל סוכן מתמחה בתפקיד אחד.

OpenClaw הוא **העוזר האישי הראשון** שתמך ב-Agent Swarms בדרך ה-Claude Agent SDK.

An **Agent Swarm** is a group of AI agents working **simultaneously and collaboratively** on one complex task. Instead of one agent doing everything, each agent specializes in one role.

OpenClaw is the **first personal assistant** to support Agent Swarms via the Claude Agent SDK.



איור 8.1: ארכיטקטורת Agent Swarm — סוכן מנצח ומומחים מקבילים

מתי להשתמש ב-Swarm?

המשימה	סוכן יחיד	Swarm
"ענה לי על שאלה"	✓ מתאים	מיותר
"סכם מייל"	✓ מתאים	מיותר
"כתוב דוח מחקר מקיף"	אפשרי אך ארוך	✓ מתאים
"בצע code review מלא"	אפשרי	✓ מתאים
"צור אסטרטגיית שיווק שלמה"	מוגבל	✓ מתאים
"נתח 50 קבצים ובצע refactor"	✗ קשה	✓ נדרש

דוגמאות מעשיות לנחילי סוכנים

Swarm 1: PixiBot Research Swarm

אריאל רוצה מחקר שוק מקיף לפני השקת PixiBot. במקום לשאול סוכן אחד שיתייגע — מפעיל Swarm:

ORCHESTRATOR

Research Coordinator

מנהל את כל הצוות, מחלק משימות, מסכם את הממצאים

AGENT 1

Competitor Analyst

חוקר מתחרים: Synthesia, HeyGen, Runway — מחירים, features, חולשות

AGENT 2

Market Size Analyst

נתונים על שוק ה-AI video: גודל, צמיחה, segments

AGENT 3

Customer Researcher

מחפש ב-Reddit, Product Hunt, Twitter: מה לקוחות אומרים על כלי AI video

WHATSAPP — TRIGGERING THE SWARM

```
@Andy, I need a comprehensive market research report for PixiBot.
Research: competitors, market size, customer pain points.
Use a research swarm and send me the final report here.
```

Swarm 2: PixMind Studio Production Swarm

צוות PixMind צריך לייצר 20 תסריטים לפרסומות בשפות שונות:

ORCHESTRATOR

Creative Director Agent

מקבל briefing, מחלק עבודה, עורך תוצאות סופיות

AGENTS 1-5

Script Writers (parallel)

5 סוכני כתיבה שעובדים במקביל — כל אחד כותב 4 תסריטים

AGENTS 6-10

Translators (parallel)

מתרגמים לאנגלית, ערבית, צרפתית, ספרדית, גרמנית — בו-זמנית

⭐ כוח ה-Parallelism

20 תסריטים + 5 שפות = 100 פיסות תוכן. סוכן יחיד: שעות. Swarm של 15 סוכנים מקבילים: **דקות**.

*20 scripts + 5 languages = 100 pieces of content. Single agent: hours. Swarm of 15 parallel agents: **minutes**.*

❓ שאלות נפוצות — Agent Swarms

כמה סוכנים ניתן להפעיל במקביל?

OpenClaw לא מגביל — אבל הזיכרון ועומס ה-API יגבילו אתכם בפועל. 5-10 סוכנים מקבילים הוא מספר סביר לרוב המשימות.

האם כל סוכן בנחיל מבודד?

כן! כל סוכן בנחיל רץ בקונטיינר משלו. הם מתקשרים דרך filesystem IPC, לא דרך זיכרון משותף.

כמה עולה להפעיל Swarm?

כל סוכן מנצל tokens של Claude API. Swarm של 10 סוכנים עם 10k tokens כל אחד = 100k tokens לכל הריצה. חשבו על מחיר ה-API לפני שמפעילים Swarms גדולים.

CHAPTER NINE • פרק תשיעי

התאמה אישית מתקדמת

Advanced Customization

09

OpenClaw לא מגיע עם configuration sprawl — אתם משנים קוד. כיצד לשנות כל דבר: מילת הטריגר, הטון, ה-model, ופעולות מותאמות

OpenClaw doesn't come with configuration sprawl — you change code. How to change everything: trigger word, tone, model, and custom behaviors

CHAPTER 09

התאמה אישית מתקדמת

Advanced Customization

⏱ זמן קריאה: 55 דקות

עיקרון: קוד במקום קונפיגורציה

OpenClaw בנוי על עיקרון פשוט: **אין configuration sprawl**. אין 53 קבצי קונפיג כמו ב-OpenClaw המקורי. כשאתם רוצים לשנות משהו — אתם **משנים קוד**.

כי הקוד מספיק קטן כדי שיהיה בטוח לשנות אותו. ו-Claude Code מספיק חכם כדי שיעשה את זה בשבילכם.

> OpenClaw is built on a simple principle: **no configuration sprawl**. No 53 config files like the original OpenClaw. When you want to change something — you **change code**.
>
> Because the code is small enough to safely modify. And Claude Code is smart enough to do it for you.

שינויים נפוצים

1. שינוי מילת הטריגר

CHAT — IN CLAUDE CODE

```
Change the trigger word from @Andy to @Pixie
```

TYPESCRIPT — SRC/CONFIG.TS

```typescript
export const config = {
  triggerWord: '@Pixie',  // changed from '@Andy'
  pollingInterval: 3000,
  maxConcurrent: 5,
};
```

2. שינוי המודל

```
ENV — .ENV
# Use different Claude model
ANTHROPIC_MODEL=claude-opus-4-6

# Or use a custom API endpoint (compatible models)
ANTHROPIC_BASE_URL=https://your-endpoint.com
ANTHROPIC_AUTH_TOKEN=your-token
```

3. שינוי מרווח הפולינג

```
CHAT — IN CLAUDE CODE
Change the polling interval to 1 second for faster responses
```

4. הוספת greeting מותאמת

```
CHAT — IN CLAUDE CODE
Add a custom greeting: when someone says "good morning" in Hebrew,
respond with a motivational quote about creativity.
```

> **⭐ customize/ — הסקיל לשינויים מותאמים**
>
> בכל פעם שאתם רוצים שינוי גדול או מורכב, הפעילו `customize/` ב-Claude Code. הסקיל ינחה אתכם בצורה אינטראקטיבית ויוודא שהשינויים נכונים.
>
> *Whenever you want a large or complex change, run `/customize` in Claude Code. The skill guides you interactively and ensures changes are correct.*

שינוי התנהגות לפי קבוצה

כל קבוצה יכולה לקבל **התנהגות שונה לחלוטין** דרך ה-CLAUDE.md שלה:

קבוצת "עבודה"

- טון מקצועי ותמציתי
- גישה ל-Obsidian vault
- ידע על PixMind Studio
- שפה: עברית בעיקר

"Work" Group

- Professional, concise tone
- Access to Obsidian vault
- Knowledge of PixMind Studio
- Language: Hebrew primarily

קבוצת "משפחה"

- טון חברותי ומשחקי
- גישה לתמונות משפחה בלבד
- אין גישה לנתוני עסק
- שפה: עברית מדוברת

"Family" Group

- Friendly, playful tone
- Access to family photos only
- No access to business data
- Language: casual Hebrew

📌 נקודות מפתח — התאמה אישית

- שינויים = שינוי קוד, לא קונפיג
- Claude Code עושה את השינויים — אתם רק מתארים מה אתם רוצים
- כל קבוצה יכולה להתנהג אחרת דרך CLAUDE.md נפרד
- `customize/` הוא הסקיל לשינויים מורכבים

CHAPTER TEN • פרק עשירי

אינטגרציות ו-MCP

10

Integrations & MCP

Model Context Protocol — הדרך להרחיב את יכולות הסוכן עם כלים חיצוניים: Notion, GitHub, Airtable, ועוד אינסוף שירותים

Model Context Protocol — the way to extend agent capabilities with external tools: Notion, GitHub, Airtable, and endless services

CHAPTER 10

אינטגרציות ו-MCP

Integrations & MCP

⏱ זמן קריאה: 50 דקות

מה זה MCP?

Model Context Protocol הוא תקן פתוח שפיתחה Anthropic שמאפשר לסוכני AI להשתמש בכלים חיצוניים בצורה אחידה. במקום לכתוב אינטגרציה מותאמת לכל שירות — MCP server מגדיר ממשק אחיד.

דמיינו MCP כמו **מתאם universal**. בדיוק כמו שמתאם USB-C מחבר כל מכשיר — MCP מחבר כל שירות לסוכן ה-AI שלכם.

Think of MCP as a **universal adapter**. Just as a USB-C adapter connects any device — MCP connects any service to your AI agent.

MCP Servers פופולריים

שירות	MCP Server	יכולות
Notion	`notionhq/notion-mcp@`	קריאה/כתיבה לדפים ו-databases
GitHub	`modelcontextprotocol/github@`	issues, PRs, commits, code search
Airtable	`airtable/mcp@`	records, tables, bases
Slack	`slack/mcp-server@`	messages, channels, reactions
Google Drive	`gdrive/mcp@`	קבצים, Docs, Sheets
Supabase	`supabase/mcp@`	SQL, storage, edge functions
Brave Search	`brave/search-mcp@`	גלישה ואינדקס

text-to-speech, voice cloning	`elevenlabs/mcp@`	ElevenLabs

JSON — .CLAUDE/SETTINGS.JSON (ADDING MCP)

```json
{
  "mcpServers": {
    "notion": {
      "command": "npx",
      "args": ["-y", "@notionhq/notion-mcp"],
      "env": {
        "NOTION_API_KEY": "secret_xxx"
      }
    },
    "github": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-github"],
      "env": {
        "GITHUB_PERSONAL_ACCESS_TOKEN": "ghp_xxx"
      }
    }
  }
}
```

⭐ PixiBot + MCP — שילוב עצמתי

לפרויקט PixiBot של אריאל: MCP של GitHub מאפשר לסוכן לקרוא את ה-codebase, לבצע code reviews, ולפתוח issues — ישירות מ-WhatsApp. MCP של Notion מאפשר לעדכן את ה-project wiki אוטומטית.

For Ariel's PixiBot project: GitHub MCP allows the agent to read the codebase, perform code reviews, and open issues — directly from WhatsApp. Notion MCP allows auto-updating the project wiki.

CHAPTER TWELVE • פרק שנים עשר

פריסה לייצור

Production Deployment

12

כיצד להריץ OpenClaw 24/7 בסביבת production — VPS, backup, monitoring, ו-zero-downtime updates

How to run OpenClaw 24/7 in production — VPS, backup, monitoring, and zero-downtime updates

CHAPTER 12

פריסה לייצור

Production Deployment

⏱ זמן קריאה: 60 דקות

אפשרויות פריסה

אפשרות	עלות	יתרונות	חסרונות
Mac Mini בבית	$0/חודש (חשמל בלבד)	נתונים אצלכם, לא תלוי בענן	IP דינמי, תלוי בחשמל
VPS (DigitalOcean / Hetzner)	$6-24/חודש	זמינות 99.9%, IP סטטי	נתונים בענן, Linux only
Raspberry Pi	$5-15 (חומרה בלבד)	זול, ביתי, Low power	Docker בלבד, ביצועים מוגבלים
AWS / GCP	$20-100/חודש	גמישות מלאה, scaling	יקר, מורכב להגדרה

VPS Production Setup (Ubuntu)

BASH — VPS INITIAL SETUP

```
# 1. Update system
sudo apt update && sudo apt upgrade -y

# 2. Install Node.js 20
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
sudo apt install -y nodejs

# 3. Install Docker
```

```
curl -fsSL https://get.docker.com | sh
sudo usermod -aG docker $USER

# 4. Install Claude Code
npm install -g @anthropic-ai/claude-code

# 5. Clone and setup OpenClaw
git clone https://github.com/openclaw/openclaw.git
cd openclaw
claude  # Then run /setup
```

BASH — SYSTEMD SERVICE FOR 24/7 OPERATION

```
# Create systemd service file
sudo tee /etc/systemd/system/openclaw.service <<EOF
[Unit]
Description=OpenClaw AI Assistant
After=network.target docker.service

[Service]
User=ubuntu
WorkingDirectory=/home/ubuntu/openclaw
ExecStart=/usr/bin/node dist/index.js
Restart=always
RestartSec=10
Environment=NODE_ENV=production

[Install]
WantedBy=multi-user.target
EOF

sudo systemctl enable openclaw
sudo systemctl start openclaw
```

Backup Strategy

BASH — DAILY BACKUP SCRIPT

```
#!/bin/bash
# backup.sh — run daily via cron

BACKUP_DIR="/backups/openclaw"
DATE=$(date +%Y%m%d)

# Backup SQLite database
sqlite3 openclaw.db ".backup $BACKUP_DIR/openclaw-$DATE.db"
```

```
# Backup group CLAUDE.md files
tar -czf "$BACKUP_DIR/groups-$DATE.tar.gz" groups/

# Keep last 30 days
find "$BACKUP_DIR" -mtime +30 -delete

echo "Backup completed: $DATE"
```

⭐ הגדרת cron לגיבוי יומי

```bash
# Run backup every day at 2am
crontab -e
# Add: 0 2 * * * /home/ubuntu/openclaw/backup.sh
```

CHAPTER THIRTEEN • פרק שלושה עשר

פתרון בעיות

Troubleshooting Guide

13

כל בעיה שנתקלתם בה — כבר נתקלנו בה לפניכם. המדריך המקיף לאיבחון ותיקון בעיות ב-OpenClaw

Every problem you encounter — someone encountered it before you. The complete guide to diagnosing and fixing OpenClaw issues

CHAPTER 13

פתרון בעיות נפוצות

Common Troubleshooting

⏱ זמן קריאה: 40 דקות

גישת ה-AI-Native לניפוי שגיאות

בפני כל בעיה, הצעד הראשון הוא **לתאר אותה ל-Claude**:

```
CLAUDE CODE PROMPT
The scheduler isn't running. Messages aren't being responded to.
Last thing I did was add a new channel. Please diagnose the issue.
```

Claude יבחן את הלוגים, הקוד והמצב — ויתקן. אבל אם אתם רוצים להבין את הבעיה בעצמכם, המשיכו לקרוא.

❌ הודעות לא מקבלות תגובה

🔍 סיבה אפשרית: לולאת הפולינג נעצרה, או הקונטיינר לא רץ

✅ תיקון: בדקו `systemctl status openclaw`, ואז `container list` / `docker ps`. אם הקונטיינר לא רץ — `container/build.sh && npm run dev/.`

❌ "Container failed to start"

🔍 סיבה אפשרית: Docker daemon לא פועל, או image לא נבנה

✅ תיקון: `docker ps` לבדיקה. אם Docker לא פועל — פתחו Docker Desktop. לבנייה מחדש: `docker buildx prune -f && ./container/build.sh`

❌ WhatsApp מתנתק כל כמה ימים

🔍 סיבה אפשרית: WhatsApp מנתק linked devices לאחר אי-שימוש, או session files פגומים

✅ תיקון: הפעילו `add-whatsapp/` מחדש לסריקת QR חדש. ה-auth credentials הקיימים נשמרים.

❌ "Context window exceeded"

🔍 סיבה אפשרית: ה-CLAUDE.md גדל מדי, או conversation ארוך מדי

✅ תיקון: קצרו את ה-CLAUDE.md, או הוסיפו ל-CLAUDE.md הוראה לסכם שיחות ארוכות. אפשר גם לבקש: "summarize this conversation and start fresh"

❌ הסוכן מתעלם ממילת הטריגר

🔍 סיבה אפשרית: שינוי במילת הטריגר לא הופעל מחדש, או typo בהגדרה

✅ תיקון: בדקו `src/config.ts` ו-`env.`. הפעילו מחדש: `npm run build && systemctl restart openclaw`

❌ המשימות המתוזמנות לא רצות

🔍 סיבה אפשרית: time zone שגויה, cron expression שגוי, או TaskScheduler לא הופעל

✅ תיקון: בדקו `timedatectl` לאימות timezone. שאלו Claude: "show me all scheduled tasks and their next run time"

כלי Debug שימושיים

BASH — DEBUGGING TOOLKIT

```
# View live logs
tail -f ~/openclaw/logs/openclaw.log

# View SQLite state
sqlite3 openclaw.db "SELECT * FROM messages ORDER BY created_at DESC LIMIT 20;"
sqlite3 openclaw.db "SELECT * FROM scheduled_tasks;"

# Check container health
docker ps -a
docker logs openclaw-agent --tail=50

# Check process status
systemctl --user status openclaw
journalctl --user -u openclaw -f --since "1 hour ago"

# Run /debug skill for guided diagnosis
```

```
# In Claude Code:
# /debug
```

APPENDIX • נספח

כל הפקודות

Complete Command Reference

מדריך עזר מהיר לכל הפקודות, סקילים והגדרות של OpenClaw

Quick reference guide for all OpenClaw commands, skills and settings

נספח א׳: מדריך עזר לפקודות

סקילים מובנים (Claude Code Skills)

`/setup`	Full initial installation — dependencies, container, first channel
`/add-whatsapp`	Add WhatsApp channel via QR code scan
`/add-telegram`	Add Telegram channel via Bot Token
`/add-discord`	Add Discord channel via Bot Token
`/add-slack`	Add Slack channel via Bolt SDK
`/add-gmail`	Add Gmail channel via OAuth2
`/customize`	Guided customization of behavior and settings
`/debug`	Diagnose container issues, logs, and problems
`/convert-to-apple-container`	Switch from Docker to Apple Container (macOS)

פקודות Chat (שלחו לסוכן)

`@Andy list all scheduled tasks`	Show all scheduled tasks and next run times
`@Andy pause the [task name]`	Pause a scheduled task

`@Andy run [task name] now`	Run a scheduled task immediately
`@Andy join the [group name] group`	Add the bot to a new group (from main channel)
`@Andy what's in the logs?`	Show recent system logs
`@Andy summarize this conversation`	Compact long conversation context

פקודות מערכת (Bash)

`npm run dev`	Start with hot reload (development)
`npm run build`	Compile TypeScript
`./container/build.sh`	Rebuild agent container
`systemctl --user restart openclaw`	Restart the service (Linux)
`launchctl kickstart -k gui/$(id -u)/com.openclaw`	Restart the service (macOS)
`sqlite3 openclaw.db ".tables"`	Show all database tables

משתני סביבה (.env)

`ANTHROPIC_API_KEY`	Required — your Anthropic API key
`ANTHROPIC_MODEL`	Model to use (default: claude-sonnet-4-6)
`TRIGGER_WORD`	Trigger word (default: @Andy)
`ANTHROPIC_BASE_URL`	Custom API endpoint (for compatible models)

`ANTHROPIC_AUTH_TOKEN`	Auth token for custom endpoints
`DISCORD_BOT_TOKEN`	Discord bot token
`SLACK_BOT_TOKEN`	Slack bot token (xoxb-...)
`SIGNAL_PHONE_NUMBER`	Signal registered phone number

🎉 כל הכבוד!

Congratulations!

סיימתם את המדריך המקיף ל-OpenClaw.
עכשיו יש לכם את כל הכלים להפוך את OpenClaw
לעוזר ה-AI המושלם בשבילכם.

You've completed the OpenClaw Complete Guide.
You now have all the tools to make OpenClaw
the perfect AI assistant for you.

נכתב עבור

אריאל איזנשטט

Founder, PixMind Studio & PixiBot • Age 18 • Software Engineering Student

OpenClaw Complete Mastery Guide • 2026 Edition • cc-by-4.0
Generated with Claude Code & OpenClaw Platform