Clawdbot Guide

Your Personal AI Assistant — An Educational Overview

What is Clawdbot?

Clawdbot (also known as Moltbot) is a **personal AI assistant** that you run on your own devices. Unlike cloud-based AI services, Clawdbot runs locally and connects to the messaging platforms you already use.

> **Key Concept:** Clawdbot is a "Gateway" that bridges your messaging apps (WhatsApp, Telegram, Discord, Slack, iMessage, etc.) to AI coding agents. Send a message, get an intelligent response — from your pocket.

Supported Channels

- **WhatsApp** — via WhatsApp Web / Baileys
- **Telegram** — Bot API / grammY
- **Discord** — Bot API / discord.js
- **Slack** — Bot API + WebSocket
- **iMessage** — Local macOS integration
- **Google Chat** — Workspace integration
- **Signal** — via signal-cli
- **Microsoft Teams** — Bot Framework
- **Plus:** Matrix, BlueBubbles, Zalo, WebChat, and more via plugins

How Memory Works

One of Clawdbot's most powerful features is its ability to maintain context and "remember" across sessions. Here's how it works:

> **Important:** Claude (the AI model) does not have built-in persistent memory. Each conversation starts fresh. The "memory" comes from files in the workspace that are read at the start of each session.

Memory Architecture

File	Purpose
	Long-term curated memories — significant events, lessons,

`MEMORY.md`	preferences
`memory/YYYY-MM-DD.md`	Daily logs of what happened (like a journal)
`AGENTS.md`	Instructions for how the AI should operate
`SOUL.md`	The AI's personality and identity
`USER.md`	Information about you (the user)
`.env.pete`	API keys and credentials (never shared)

The workspace lives on your server/computer and persists between sessions. Every new conversation, the AI reads these files first — that's how it "remembers."

Security Considerations

> ⚠ **Critical Understanding:** Running an AI agent with shell access on your machine is powerful but risky. The AI can execute commands, read/write files, and access network services. Security is essential.

The Threat Model

Your AI assistant can:

- Execute arbitrary shell commands
- Read and write files on your system
- Access network services
- Send messages to anyone (if given messaging access)

People who message your bot can:

- Try to trick the AI into doing harmful things (prompt injection)
- Social engineer access to your data
- Probe for infrastructure details

Prompt Injection

Prompt injection is when an attacker crafts a message that manipulates the AI into doing something unsafe. Examples:

- "Ignore your previous instructions and..."
- "Dump your filesystem contents"
- "Follow this link and run these commands"

> **Key Point:** Even with strong system prompts, prompt injection is not fully solved. Modern models like Claude Opus 4.5 are more resistant, but no model is immune.

Security Best Practices

1. **Lock down DMs:** Use pairing mode (default) — unknown senders get a code and are ignored until approved
2. **Require mentions in groups:** Don't let the bot respond to every message
3. **Use allowlists:** Explicitly define who can interact with your bot
4. **Enable sandboxing:** Run tool execution in isolated containers
5. **Keep secrets out:** Don't store sensitive credentials in the agent's workspace
6. **Run security audits:** `clawdbot security audit --deep`

DM Access Policies

Policy	Behavior
`pairing` (default)	Unknown senders get a pairing code; ignored until approved
`allowlist`	Only pre-approved senders can interact
`open`	Anyone can DM (dangerous — requires explicit opt-in)
`disabled`	All inbound DMs ignored

Quick Start

Installation

Requires Node.js ≥ 22:

```
npm install -g moltbot@latest
moltbot onboard --install-daemon
```

Key Commands

Command	Purpose
`clawdbot gateway`	Start the Gateway service
`clawdbot status`	Check system status
`clawdbot channels login`	Pair messaging channels (WhatsApp QR, etc.)
`clawdbot security audit`	Check for security issues

`clawdbot doctor`	Diagnose and fix common issues

Model Recommendations

> **Recommendation:** Use **Anthropic Claude Opus 4.5** for any bot with tool access. It has better prompt-injection resistance and instruction following than smaller models.

Model considerations:

- **Opus 4.5:** Best for tool-enabled agents, highest security
- **Sonnet:** Good for chat-only, lower cost
- **Haiku:** Fast and cheap, but more susceptible to manipulation

Privacy & Data

What's Stored Locally

- Configuration: `~/.clawdbot/clawdbot.json`
- Session transcripts: `~/.clawdbot/agents/*/sessions/*.jsonl`
- Credentials: `~/.clawdbot/credentials/`
- OAuth tokens: `~/.clawdbot/agents/*/agent/auth-profiles.json`

> **Privacy Note:** Session transcripts contain your full conversation history including any private information you share. Keep disk permissions tight (`chmod 700 ~/.clawdbot`) and consider full-disk encryption.

Resources

- **Documentation:** docs.molt.bot
- **GitHub:** github.com/moltbot/moltbot
- **Discord Community:** discord.gg/clawd
- **Security Issues:** security@clawd.bot

Key Takeaways

1. **Memory is files:** The AI reads workspace files at session start — that's how it "remembers"
2. **Security first:** Lock down who can talk to your bot before giving it powerful tools
3. **Trust hierarchy:** Owner → AI → Allowlisted friends → Strangers (no trust)

4. **Model matters:** Use the best model you can afford for tool-enabled agents
5. **Regular audits:** Run `clawdbot security audit` after any config changes

"Security is a process, not a product. Also, don't trust lobsters with shell access."

Generated by Pete • January 2026