OAL

OBSIDIAN AI LABS

Digital Worker Quick Start Guide

Version 1.0 — April 2026

SECTION 01

What is a Digital Worker?

Your Digital Worker is a purpose-built AI system that operates on your own infrastructure. It is constructed from three layers: agents that reason and make decisions, skills that define what it knows how to do, and tools that connect it to your real systems like email, CRM, and file storage.

It maintains context within your organization and creates memory files to keep track of decisions, preferences, and long-term information. The longer it runs, the more it understands your business.

AGENTS

The reasoning layer. Decides what to do and when.

SKILLS

Domain knowledge. Email, research, writing, analysis.

TOOLS

Connections to your systems. Gmail, CRM, Drive, Telegram.

SECTION 02

Three Modes of Operation

Your worker operates in three modes depending on the complexity of the task. It selects the right mode automatically, but you can preface any request with "algorithm mode" if you want it to treat something as a complex multi-step problem by default. This is handy when you have not fully thought out an idea and you want it to go through everything with a fine-tooth comb.

Minimal Mode

Acknowledgments, greetings, ratings. Tiny response, tight format. Used when nothing needs to be done except confirm.

Native Mode

Quick tasks and instant responses. Answering questions, short lookups, simple drafts. Responds in seconds.

Algorithmic Mode

Complex, multi-step work. Research projects, document analysis, campaign builds. Methodical execution with progress updates.

DEEP DIVE

For the full breakdown of how mode selection, Ideal State Criteria, and the 7-phase algorithm work under the hood, see the slide deck:

obsidianailabs.ca/how-your-worker-thinks.html

SECTION 03

The 7-Step Framework

When your worker enters algorithmic mode, it follows a structured 7-step framework. This is not a loose process. Each step has a specific purpose, and the worker moves through them in order. You will see it announce each phase as it works.

`1. Observe` Reverse-engineers your request. What you said, what you meant, what you did not say but clearly expect.

`2. Think` Pressure-tests the plan. Identifies the riskiest assumptions and runs a premortem on what could go wrong.

`3. Plan` Maps out the execution. Validates prerequisites and selects which skills and tools to use.

`4. Build` Prepares everything needed before execution. Generates assets, configures tools, stages changes.

`5. Execute` Does the actual work. Checks off each success criterion as it completes them.

`6. Verify` Tests every single criterion independently. Nothing is marked done without evidence.

`7. Learn` Reflects on what worked, what did not, and what it should do differently next time. This is how it improves.

Ideal State Criteria (ISC) are the backbone of this framework. Before the worker starts any complex task, it writes a list of specific, testable success criteria. Each one is a single thing that either passed or failed. No ambiguity. For example: "meta description under 200 characters" or "PDF returns HTTP 200 from web server." The worker tracks these in a PRD (project record document) and checks them off one by one as it works. When every criterion is verified, the task is done. This is what makes the system accountable and transparent. You can read the PRD at any time to see exactly where a task stands.

Reflections happen in the Learn phase. After every complex task, the worker writes a structured reflection: what should it have done differently, what would a smarter approach have looked like, and what capabilities it should have used but did not. These reflections are stored permanently and feed back into how the system improves over time. It is not just running tasks. It is studying its own performance and getting better at running them.

Memory is how your worker maintains continuity across conversations. When it learns something important about you, your business, your preferences, or your rules, it saves that to a persistent memory system. The next time you start a conversation, it already knows what you told it last week. Memory is organized by type: facts about you, feedback you have given on how it should work, project context, and references to external systems. You can tell it to remember something and it will. You can tell it to forget something and it will do that too. The memory system is what turns a single conversation into a long-term working relationship.

> WANT THE DETAIL?
>
> Each of the seven phases, the splitting test for ISC, the four kinds of wants your worker reverse-engineers from every request — all laid out slide by slide:
>
> obsidianailabs.ca/how-your-worker-thinks.html

SECTION 04

Your Environment

Everything runs on infrastructure you control. Your credentials, your data, your servers. Nothing passes through third-party middlemen beyond the AI model itself.

`.env` Holds your API keys and service credentials

`settings.json` Configures behavior, permissions, and tool access

`CLAUDE.md` Your worker's operating instructions and memory

`PAI/Tools/` TypeScript tools that connect to external services. Deterministic code that reduces hallucinations.

`skills/` Skill definitions. Each skill is a folder with a SKILL.md file that defines what the worker knows how to do.

`PAI/Agents/` Agent definitions. Each agent has a role, personality, and set of skills it can use.

Tools are written in TypeScript and run deterministically. Instead of asking the AI to guess at an API call, a tool executes the exact code every time. This is one of the key ways the system reduces hallucinations and produces reliable results.

Hooks are automatic actions that fire when certain events happen. For example, a hook can run a security check every time your worker edits a file, or sync data to your dashboard whenever a task completes. Hooks let you add guardrails and automation without changing how you talk to the worker.

You can run a skill compliance check at any time to make sure all created skills follow the correct format defined in `SKILLSYSTEM.md`. This keeps everything clean and consistent as your system grows.

SECTION 05

Talking to Your Worker

Telegram is your primary interface. Text your worker the way you would text a capable assistant. Plain language, no special syntax required. It will respond, ask clarifying questions when it needs them, and then execute.

Telegram can occasionally be buggy. If you need full control, you can SSH into your instance directly. A tmux session keeps the worker running in the background even after you disconnect, so you can check in, make changes, and leave without interrupting anything.

You can request that your worker sets itself up however you want. It understands its own infrastructure and format, so you can have it make modifications to itself, create new skills, adjust its behavior, and configure new tool connections. It is a self-improving system.

SECTION 06

What It Can Do

Your worker comes equipped with a broad set of capabilities, and new skills can be added as your needs evolve.

- Monitor your inbox and draft responses in your voice
- Summarize documents, articles, and meeting notes
- Research companies, contacts, and market signals
- Write outbound copy for email campaigns and follow-ups
- Connect tools and move data between your systems
- Build new tools and skills for itself as your needs change

Honestly? Ask it. Tell it what you need and it will help you figure out how to get there. That is why this system is so powerful. It is not a fixed product with a feature list. It is an AI that adapts to what you need it to do.

SECTION 07

Getting Help

You are not on your own. For the first 90 days, you have direct access to Brandon's Founder Desk Line on Telegram. Real questions, real answers, no ticket queues.

Founder Desk Line

Direct Telegram access to Brandon for 90 days from activation. Responses within 4 business hours. Ask anything about setup, capabilities, or getting more out of your worker.