HACKATHON 0: BRONZE TIER

Building Your First Personal AI Employee

Complete Beginner-Friendly Step-by-Step Roadmap

Your life and business on autopilot.

Local-first. Agent-driven. Human-in-the-loop.

Estimated Time: 8–12 hours | Skill Level: Absolute Beginner Welcome

Based on Panaversity Personal AI Employee Hackathon 0

Phase 1: Understand the Big Picture (30 minutes)

Before we touch anything, let’s understand WHAT we are building in the simplest way possible.

1.1 What Are We Building?

We are building a “Digital Employee” – an AI assistant that lives on your computer and works 24/7. It has three parts:

Part	What It Is	Tool
The Brain (Reasoning)	The thinker. Reads tasks, makes plans, writes reports.	Claude Code
The Memory/GUI (Dashboard)	The filing cabinet + dashboard. Stores everything as files.	Obsidian
The Eyes (Watcher)	The watcher. Detects new files dropped into a folder.	Python Script

1.2 The Simple Flow

Here is EVERYTHING that happens, in plain English:

1. You drop a file into the /Inbox folder (like putting a letter on your desk)
2. The Watcher Script detects it and moves it to /Needs_Action folder
3. Claude Code reads it, thinks, and creates a plan in /Plans folder
4. Claude Code does the work and moves the file to /Done folder
5. Claude Code updates the Dashboard.md file so you can see what happened

Think of it like this

Obsidian = Your office (desk, filing cabinets, whiteboard). Claude Code = Your smart employee who sits at that desk and works. The Watcher = A mail carrier who puts new mail on the desk. CLAUDE.md = The instruction manual you give your employee on day one.

1.3 The Key Files

Your AI Employee's entire life runs through these files:

File	Purpose	Who Creates It
Dashboard.md	Shows status of everything (like a whiteboard)	Claude updates it automatically
Company_Handbook.md	Your rules for the AI (how to behave, what to flag)	You write it once
CLAUDE.md	Instructions Claude reads on every run (what to do, where to look)	You write it once
SKILL.md files	Reusable recipes for specific tasks (Agent Skills)	You + Claude create them

1.4 What Are Agent Skills?

Agent Skills are reusable instruction files (SKILL.md) that teach Claude how to do a specific task. Think of them like recipes in a cookbook:

- Without skills: You tell Claude "make me a report" and hope it figures it out each time
- With skills: Claude reads the "report-writing skill" file and follows the exact steps every time

Skills make Claude consistent and reliable. Instead of re-explaining things, you write the instructions once as a SKILL.md file, and Claude follows them every time.

Agent Skills = Employee Training Manual

Just like you train a new employee with step-by-step instructions for each task, Agent Skills train Claude. Each skill lives in a /Skills folder and has a SKILL.md file with clear instructions.

Phase 2: Install All Required Software (2–3 hours)

We need to install 5 tools. Do each one, one at a time. Don't skip any.

If You Get Stuck

Copy-paste any error message into Claude.ai (the website) and ask for help. Or bring your question to the Wednesday Zoom meeting.

STEP 2.1 Install Obsidian (Your AI's Memory & Dashboard)

Obsidian is a free note-taking app. It stores everything as simple text files on YOUR computer. Your data stays private.

1. Go to: https://obsidian.md
2. Click "Get Obsidian for free" and download for your OS (Windows/Mac/Linux)
3. Run the installer (just click Next/Continue through everything)
4. Open Obsidian – you should see a welcome screen. Done!

STEP 2.2 Install Python (For the Watcher Script)

Python is the language our Watcher script is written in. You don't need to learn Python – Claude will write the code.

Windows: Go to python.org/downloads → Download Python 3.13+ → **CHECK "Add Python to PATH"** → Click Install Now

Mac: Open Terminal → type: brew install python (if you have Homebrew) or download from python.org

Linux: Open Terminal → type: sudo apt update && sudo apt install python3 python3-pip python3-venv

MOST COMMON MISTAKE

On Windows, forgetting to check “Add Python to PATH”. If you miss it, uninstall and reinstall with the box checked. Without this, nothing will work later.

Verify it works:

```
python --version      # Should show Python 3.13+
# If that fails, try:
python3 --version
```

What is a Terminal?

A terminal is a text-based way to talk to your computer. Windows: search for “Command Prompt” or “PowerShell” in Start menu. Mac: search for “Terminal” with Cmd+Space. Linux: Ctrl+Alt+T.

STEP 2.3 Install Node.js (For Claude Code)

Node.js lets Claude Code run on your computer.

1. Go to: https://nodejs.org
2. Download the LTS version (v24+)
3. Run the installer, accept all defaults

Verify:

```
node --version     # Should show v24.x.x
npm --version      # Should show 10.x.x
```

STEP 2.4 Install Claude Code (Your AI’s Brain)

Claude Code is the AI that runs in your terminal. Unlike claude.ai (which runs in a browser), Claude Code runs on your computer and can read/write files directly. This is what makes it powerful.

You need an active Claude subscription (Pro plan or use free Gemini API with Claude Code Router).

```
npm install -g @anthropic/claude-code
```

Verify:

```
claude --version   # Should show version number
```

If you see “command not found”
Close your terminal completely, open a new one, and try again. If still broken, restart your computer.

STEP 2.5 Install GitHub Desktop (Version Control)

Saves snapshots of your project. If something breaks, you go back to a working version.

1. Go to: https://desktop.github.com
2. Download, install, create a free GitHub account, sign in

Installation Checklist

Run every command below. ALL must work before moving on:

Tool	Command	Expected
Obsidian	Open the app	Welcome screen
Python	python --version	Python 3.13+
Node.js	node --version	v24.x.x+
Claude Code	claude --version	Version number
GitHub Desktop	Open the app	Main screen

STOP if anything failed!
Do NOT continue until every tool works. Ask for help in the Wednesday Zoom or paste your error into Claude.ai.

Phase 3: Create Your Obsidian Vault (1 hour)

Now we build the “office” where your AI Employee works. This is just creating folders and files.

STEP 3.1 Create a New Vault

1. Open Obsidian

2. Click “Create new vault”
3. Name it: AI_Employee_Vault
4. Choose a location you can easily find (e.g., Documents folder)
5. Click “Create”

Write Down Your Vault Path!

You need this path later. Example: C:\Users\YourName\Documents\AI_Employee_Vault (Windows) or /Users/YourName/Documents/AI_Employee_Vault (Mac). Find it in Obsidian Settings > Files & Links.

STEP 3.2 Create the Folder Structure

Right-click in Obsidian’s left sidebar and select “New folder” for each of these:

```
AI_Employee_Vault/
├── Inbox/              ← You drop files here
├── Needs_Action/       ← Watcher moves files here for Claude
├── Plans/              ← Claude writes plans here
├── Done/               ← Completed tasks go here
├── Logs/               ← Activity records
├── Skills/             ← Agent Skill files live here
├── Pending_Approval/   ← Sensitive tasks wait here for your OK
├── Approved/           ← Tasks you approved
└── Rejected/           ← Tasks you rejected
```

What Each Folder Does

/Inbox = Your drop box (put files here). /Needs_Action = To-do tray (Watcher moves files here). /Plans = Claude’s action plans. /Done = Completed work. /Skills = Reusable task recipes. /Pending_Approval = Tasks needing your sign-off.

STEP 3.3 Create Dashboard.md

Click the “New note” button in Obsidian, name it Dashboard, and paste this:

```
# AI Employee Dashboard

Last Updated: [Auto-updated by Claude]

## Quick Status
- Pending Tasks: 0
- In Progress: 0
- Completed Today: 0
- Needs My Approval: 0
```

```
## Recent Activity
- No activity yet. Your AI Employee is ready to start!

## Alerts
- None
```

STEP 3.4 Create Company_Handbook.md

This is YOUR rules for the AI. Create a new note called Company_Handbook and paste:

```
# Company Handbook - AI Employee Rules

## Identity
- Owner: [Your Name]
- Business: [Your Business Name or "Personal"]
- AI Employee Name: Atlas

## Communication Rules
- Always be polite and professional
- Never send any message without human approval

## Task Handling Rules
- Read files from /Needs_Action folder
- Create a plan in /Plans before taking any action
- Move completed tasks to /Done
- Update Dashboard.md after every task
- Log every action in /Logs

## Safety Rules
- NEVER delete files without approval
- NEVER send payments without approval
- If unsure about anything, move to /Pending_Approval

## Priority Levels
- URGENT: Handle immediately
- HIGH: Handle within 4 hours
- NORMAL: Handle within 24 hours
- LOW: Handle within 1 week
```

Make It Yours!
Add any rules you want. Examples: “Flag any task mentioning money” or “Always summarize files in 3 bullet points.” The more specific, the better.

STEP 3.5 Create CLAUDE.md (The Master Instructions)

This is the MOST IMPORTANT file. Claude Code reads CLAUDE.md automatically every time it starts. It tells Claude what to do, where to look, and how to behave.

Create a new note in the ROOT of your vault called CLAUDE and paste:

```
# AI Employee - Master Instructions

You are an AI Employee named Atlas. You work inside this
Obsidian vault. Follow these instructions on every run.

## Your Files (Read These First)
1. /Company_Handbook.md - Your rules (READ THIS FIRST)
2. /Dashboard.md - Status board (UPDATE after every task)

## Your Workflow
Every time you run, do this in order:

### Step 1: Check for Work
- Look in /Needs_Action for any files
- If empty, report "No tasks" and update Dashboard

### Step 2: Plan
- For each file in /Needs_Action:
  - Read the file contents
  - Create a plan file in /Plans/PLAN_[taskname].md
  - The plan should list what you will do

### Step 3: Execute
- Follow the plan step by step
- If any step is sensitive (payments, deletes, sends),
  move to /Pending_Approval instead
- Use Agent Skills from /Skills/ when available

### Step 4: Complete
- Move the original task file to /Done
- Move the plan file to /Done
- Update /Dashboard.md with what you did
- Write a log entry in /Logs/[today].md

## Agent Skills
- Check /Skills/ folder for SKILL.md files
- Each skill has step-by-step instructions for a task
- Always use a matching skill if one exists
- If no skill exists, do your best and suggest creating one

## Important Rules
- NEVER delete files, only move them
- ALWAYS update Dashboard.md
- ALWAYS log your actions
- If unsure, move task to /Pending_Approval
```

Why is CLAUDE.md So Important?

Claude Code automatically reads CLAUDE.md every time it starts in your vault folder. Without this file, Claude doesn't know what to do. With it, Claude knows exactly where to look, what rules to follow, and how to process tasks. Think of it as the "day-one orientation" for your new employee.

Phase 4: Create Your First Agent Skills (1 hour)

Agent Skills are reusable instruction files that make Claude consistent. Let's create two simple skills.

STEP 4.1 Understand the Skill Structure

Every skill is a SKILL.md file inside its own folder in /Skills/. The structure is:

```
Skills/
├── summarize-file/
│   └── SKILL.md      ← Instructions for summarizing files
├── process-task/
│   └── SKILL.md      ← Instructions for processing tasks
└── write-report/
    └── SKILL.md      ← Instructions for writing reports
```

STEP 4.2 Create the "Summarize File" Skill

In Obsidian, create the folder: Skills/summarize-file/ then create a file called SKILL.md inside it:

```
# Skill: Summarize File

## When to Use
When a file in /Needs_Action needs to be summarized.

## Steps
1. Read the full file contents
2. Identify the main topic and key points
3. Write a summary with:
   - One sentence overview
```

~~~
   - 3-5 bullet points of key information
   - Any action items found
4. Save the summary inside the original file
   (add a ## Summary section at the bottom)
5. Move file to /Done
6. Update Dashboard.md
7. Log the action in /Logs/[today].md

## Output Format
```
## Summary
**Overview:** [one sentence]
**Key Points:**
- [point 1]
- [point 2]
- [point 3]
**Action Items:**
- [ ] [action if any]
```
~~~

STEP 4.3 Create the “Process Task” Skill

Create the folder: Skills/process-task/ and SKILL.md inside it:

~~~
# Skill: Process Task

## When to Use
When any new task file appears in /Needs_Action.

## Steps
1. Read the task file
2. Read the Company_Handbook.md for rules
3. Determine priority (URGENT/HIGH/NORMAL/LOW)
4. Create a plan in /Plans/PLAN_[taskname].md:
   - What needs to be done
   - Which skills to use
   - Estimated steps
5. Execute the plan
6. If any step requires human approval,
   move to /Pending_Approval and STOP
7. When done, move original file to /Done
8. Move plan to /Done
9. Update Dashboard.md
10. Write log entry

## Plan Template
```
# Plan: [Task Name]
~~~

```
Created: [date/time]
Priority: [level]

## Steps
- [ ] Step 1: ...
- [ ] Step 2: ...

## Status: In Progress
```
```

You Can Create More Skills Later!

These two skills are enough for Bronze tier. As you use your AI Employee, you'll notice patterns – that's when you create new skills. Ask Claude Code: "Create a new agent skill for [task]" and it will write the SKILL.md for you.

Phase 5: Connect Claude Code to Your Vault (1 hour)

Now we teach Claude Code where your vault is and verify it can read and write files.

STEP 5.1 Open Terminal in Your Vault Folder

Windows: Open File Explorer → navigate to your vault folder → click the address bar → type cmd → press Enter

Mac/Linux: Open Terminal → type: cd /path/to/AI_Employee_Vault

Verify you're in the right place:

```
ls         # Mac/Linux - should show your folders
dir        # Windows - should show your folders
```

STEP 5.2 Start Claude Code

```
claude
```

Claude Code will start. It will automatically read your CLAUDE.md file and know its instructions!

STEP 5.3 Test: Can Claude Read Files?
```

Type this in Claude Code:

```
Read Dashboard.md and Company_Handbook.md and tell me what they say.
```

Claude should show you the contents of both files. If it does, Claude can see your vault!

STEP 5.4 Test: Can Claude Write Files?

Type this:

```
Create a test task file in /Needs_Action called TEST_001.md with this content:
---
type: test
priority: normal
status: pending
---
## Test Task
Please summarize this: Our company had a great Q1 with $10,000 revenue.
```

Check Obsidian – you should see the file in /Needs_Action.

STEP 5.5 Test: The Full Workflow

Now test the complete cycle. Type:

```
Process all tasks in /Needs_Action. Follow the instructions in CLAUDE.md.
Use the skills in /Skills/ where applicable.
```

Claude should:

- Read the test task from /Needs_Action
- Create a plan in /Plans
- Use the summarize-file skill to summarize it
- Move the task to /Done
- Update Dashboard.md
- Write a log entry in /Logs

Check all your folders in Obsidian to verify everything moved correctly!

If It Worked – Amazing!
You just ran the core loop of your AI Employee! Claude read a task, planned, executed, organized, and logged. This is exactly what happens automatically once we add the Watcher.

Phase 6: Build the File System Watcher (2–3 hours)

The Watcher monitors the /Inbox folder. When you drop a file in, it automatically moves it to /Needs_Action for Claude to process. It's just ONE single Python file.

STEP 6.1 Install the Watchdog Library

Open a NEW terminal window (keep Obsidian open) and run this single command:

```
pip install watchdog

# If that doesn't work, try:
pip3 install watchdog
```

What is "watchdog"?

It's a pre-built Python tool that watches folders for changes. Someone already built it – we just install and use it. One command, done.

STEP 6.2 Create the Watcher Script

We need ONE single Python file. You have two ways to create it:

Way 1 (Recommended): Ask Claude Code to write it:

In Claude Code (running inside your vault folder), type:

```
Create a single Python file called file_watcher.py in the root
of this vault that does this:
1. Watches the /Inbox folder for any new files
2. When a new file appears, moves it to /Needs_Action
   with a timestamp prefix (e.g., 20260115_143000_myfile.txt)
3. Creates a .md metadata file alongside it with
   frontmatter and suggested actions
4. Prints a message each time it detects a file
5. Runs continuously until Ctrl+C
6. Has error handling so it doesn't crash
```

Way 2: Create it manually:

Create a file called file_watcher.py in the ROOT of your vault (same level as Dashboard.md) and paste this code:

```python
import time
import shutil
import logging
from pathlib import Path
from datetime import datetime
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler

# ============================================
# CHANGE THIS TO YOUR VAULT PATH:
VAULT_PATH = Path("/path/to/AI_Employee_Vault")
# ============================================

INBOX = VAULT_PATH / "Inbox"
NEEDS_ACTION = VAULT_PATH / "Needs_Action"

logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(message)s")
logger = logging.getLogger("FileWatcher")


class InboxHandler(FileSystemEventHandler):
    def on_created(self, event):
        if event.is_directory:
            return
        source = Path(event.src_path)
        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        new_name = f"{timestamp}_{source.name}"

        # Move file to Needs_Action
        dest = NEEDS_ACTION / new_name
        shutil.move(str(source), str(dest))
        logger.info(f"Moved: {source.name} -> /Needs_Action/{new_name}")

        # Create metadata file
        meta = NEEDS_ACTION / f"{new_name}.md"
        meta.write_text(f"""---
type: file_drop
original_name: {source.name}
size: {dest.stat().st_size} bytes
detected: {datetime.now().isoformat()}
priority: normal
status: pending
---

## New File for Processing
A file was dropped into Inbox and needs processing.
## Suggested Actions
- [ ] Review the file contents
- [ ] Summarize or process as needed
- [ ] Move to Done when complete
""")
        logger.info(f"Metadata created: {new_name}.md")


def main():
    INBOX.mkdir(exist_ok=True)
    NEEDS_ACTION.mkdir(exist_ok=True)

    handler = InboxHandler()
    observer = Observer()
    observer.schedule(handler, str(INBOX), recursive=False)
    observer.start()

    logger.info("=" * 50)
    logger.info("FILE WATCHER IS RUNNING")
    logger.info(f"Watching: {INBOX}")
    logger.info("Drop any file into /Inbox to process it.")
    logger.info("Press Ctrl+C to stop.")
    logger.info("=" * 50)

    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        observer.stop()
        logger.info("Watcher stopped.")
    observer.join()


if __name__ == "__main__":
    main()
```

CHANGE THE VAULT PATH!

Update VAULT_PATH = Path("/path/to/AI_Employee_Vault") to YOUR actual path. Windows example: Path("C:/Users/John/Documents/AI_Employee_Vault"). Mac example: Path("/Users/john/Documents/AI_Employee_Vault").

STEP 6.3 Run the Watcher

In your terminal, navigate to your vault and run:

```
cd /path/to/AI_Employee_Vault
python file_watcher.py
```

You should see “FILE WATCHER IS RUNNING”. Leave this terminal open!

STEP 6.4 Test It!

1. Open your file explorer and go to the /Inbox folder inside your vault
2. Drop any file in there (create a text file, drag a PDF, anything)
3. Watch the terminal – you should see “Moved: ...” messages
4. Open Obsidian → check /Needs_Action → the file and a .md metadata file should be there

Your Watcher Works!

The “eyes” of your AI Employee are now active. Any file dropped in /Inbox automatically appears in /Needs_Action for Claude to process.

Phase 7: Set Up the Ralph Wiggum Loop (1 hour)

This is the final piece that makes your AI Employee truly autonomous. Instead of you typing commands every time, the Ralph Wiggum loop keeps Claude running and processing tasks automatically.

Phase 7: Test Everything and Submit (1–2 hours)

STEP 7.1 Run the Full System

Open TWO terminal windows:

Terminal 1 – Watcher (leave running):

```
cd /path/to/AI_Employee_Vault
python file_watcher.py
```

Terminal 2 – Claude Code:

```
cd /path/to/AI_Employee_Vault
claude

# Then tell Claude:
Process all tasks in /Needs_Action following CLAUDE.md.
Use skills from /Skills/. Update Dashboard.md when done.
```

How It Works Together

Terminal 1 keeps watching /Inbox for new files and moves them to /Needs_Action. In Terminal 2 you tell Claude to process tasks whenever you want. Each time you tell Claude, it reads CLAUDE.md, checks /Needs_Action, processes everything, and updates the Dashboard.

STEP 7.2 Run These 5 Tests

Test	What to Do	What Should Happen
1	Drop a .txt file into /Inbox	File moves to /Needs_Action, tell Claude to process, ends up in /Done
2	Drop an image file into /Inbox	Same as above – metadata .md created, file processed
3	Drop 3 files at once	All 3 get picked up by Watcher, Claude processes all 3
4	Check /Dashboard.md	Shows correct count of completed tasks and recent activity
5	Check /Logs/ folder	Has log entries for every action Claude took

STEP 7.3 Common Problems and Fixes

Problem	Fix
Watcher crashes	Check that VAULT_PATH in the script matches your actual vault path
Claude can't find files	Make sure you ran claude from inside your vault folder
Files don't show in Obsidian	Click another folder and back, or restart Obsidian
Module not found error	Run: pip install [missing module name]
Permission denied	Close any other programs using the files

STEP 7.4 Save to GitHub

1. Open GitHub Desktop
2. Add your vault as a repository
3. FIRST: Create a .gitignore file. In Claude Code, type:

```
Create a .gitignore file that excludes: .env, credentials.json, __pycache__/,
.obsidian/, *.secret, *.token
```

4. Commit with message: “Bronze tier - AI Employee complete”
5. Publish repository

STEP 7.5 Record Demo Video (5–10 minutes)

Screen-record yourself showing:

1. Your vault folder structure in Obsidian
2. Dashboard.md, Company_Handbook.md, and CLAUDE.md
3. The Agent Skills in /Skills/
4. Starting the Watcher (Terminal 1)
5. Starting Claude Code (Terminal 2)
6. Dropping a file in /Inbox
7. Watching it flow: /Inbox → /Needs_Action → /Plans → /Done
8. Dashboard.md being updated

Recording Tools

Windows: Win+G (Xbox Game Bar). Mac: QuickTime (File > New Screen Recording). Or free tools: OBS Studio, Loom.

STEP 7.6 Write README.md and Submit

Ask Claude Code:

```
Create a README.md that includes:
1. Project name and description
2. Tier: Bronze
3. Architecture: Watcher + Claude Code + Obsidian + Agent Skills
4. Setup instructions
5. How to run (2 terminals)
6. File flow: Inbox -> Needs_Action -> Plans -> Done
7. Security: how credentials are handled (.gitignore)
```

Submit at: https://forms.gle/JR9T1SJq5rmQyGkGA

Quick Reference: Everything You Built

Component	What It Is	Status
Obsidian Vault	AI Employee’s memory and dashboard (GUI)	Created
Dashboard.md	Real-time status board	Created
Company_Handbook.md	Your rules for the AI	Created
CLAUDE.md	Master instructions Claude reads automatically	Created
Agent Skills (/Skills/)	Reusable task recipes (SKILL.md files)	Created (2 skills)
file_watcher.py	Watches /Inbox, moves files to /Needs_Action	Running
Claude Code	The brain – reads, plans, executes, logs	Running
Folder Flow	Inbox → Needs_Action → Plans → Done	Working
Dashboard Auto-Update	Claude updates Dashboard.md after each task	Working
Logging	Every action logged in /Logs/	Working

Bonus: Ralph Wiggum Loop (Optional – For Extra Credit)

If you want to take your AI Employee to the next level, you can add the Ralph Wiggum Loop. This is completely optional for Bronze tier but makes your system more autonomous.

What It Does

Right now, you have to tell Claude “process tasks” each time. The Ralph Wiggum Loop automates this – it keeps Claude running in a loop, automatically processing new tasks until there are none left.

The flow becomes fully hands-free:

1. File drops in /Inbox
2. Watcher moves it to /Needs_Action (automatic)
3. Claude picks it up and processes it (automatic – Ralph Wiggum loop)
4. Claude moves to /Done and updates Dashboard (automatic)
5. Loop checks: more tasks? If yes, keep going. If no, stop.

How to Set It Up

In your terminal (inside your vault folder), start Claude Code and type:

```
Help me install the Ralph Wiggum stop hook plugin.
Reference: https://github.com/anthropics/claude-
code/tree/main/.claude/plugins/ralph-wiggum
Set it up so it keeps me working until all files in /Needs_Action are processed.
```

Once installed, start the loop with:

```
/ralph-loop "Process all files in /Needs_Action following CLAUDE.md. \
Use skills from /Skills/. Update Dashboard.md." \
--completion-promise "TASK_COMPLETE" \
--max-iterations 10
```

What each part means:

- The text in quotes = your instructions for each loop iteration
- --completion-promise = Claude says this word when all tasks are done
- --max-iterations 10 = safety limit so it doesn’t loop forever

When to Add This

Try this AFTER you have the basic system working. Get comfortable with the manual flow first (drop file → tell Claude → check results). Once that feels smooth, add the Ralph Wiggum loop to make it fully automatic.

You did it! Your Personal AI Employee is alive and working.

Drop a file. Watch the magic. Happy building!