Hey Walter

How we build software.

Autonomous development, human judgment, and software you actually own.

Build it. Tend it. Own it.

heywalter.dev

THE OLD WAY

Custom software used to work like this.

Months of calendar time

Discovery, specs, sprints, and a launch date that keeps moving.

A backlog you wait behind

Your small fix sits behind someone else's big feature.

A change order for every tweak

The scope conversation costs more than the change.

A codebase you cannot touch

It works until the agency stops answering the phone.

You paid for software and got a dependency.

THE SHIFT

So we rebuilt how the work happens.

You describe the outcome. A pipeline of specialized AI agents plans it, writes it, reviews it, security checks it, documents it, and ships it to production. You stay in control of scope, budget, and every decision that matters.

Weeks
not months

Every change
reviewed and security checked

Zero
backlog to wait behind

THE UNIT OF WORK

Everything starts with one outcome.

We do not hand the agents a task list. We hand them a result to be responsible for, written the way you would say it out loud.

WHAT WE WRITE

"Users should be able to sign in with Google."

NOT THIS

"Add an OAuth handler to /api/auth/google."

Describe the outcome and the agents handle the cross file work themselves. Pre deciding the implementation is how you get software that technically matches the spec and still misses the point.

This is why your team is in the room. You know the outcome. We know how to say it so it gets built.

THE PIPELINE

Every change makes five passes before it ships.

One outcome runs as a graph of specialized agents, each with a different job, working in an isolated sandbox against a real copy of your codebase.

1

Plan

Reads your codebase and writes the implementation plan. Nothing changes yet.

2

Implement

Writes the actual code, running tests as it goes. The longest pass.

3

Review

A different agent re reads the diff and either approves or sends it back.

4

Security

Audits for vulnerabilities, leaked secrets, and auth gaps.

5

Docs

Updates the documentation, tidies the diff, then merges.

There is also a read only investigate pass for questions like "how does this work?" that answers without touching a line of code.

QUALITY

The work has to prove itself.

Speed only counts if the thing works. So verification is not a phase at the end, it is built into every pass.

A contract before any code

The agents commit to a testable definition of done first, so there is an objective bar to hit, not a vibe.

Tests run during the build

Not after. Problems surface while the context is still fresh.

Checked against a running app

Verification includes live runtime checks against a real instance, not just a passing test file.

Evidence you can read

Every ticket carries its reports: what was tested, what passed, and screenshots from the verifier.

Most tickets pass review on the first attempt. The ones that do not get sent back with specific feedback and fixed before anyone sees them.

SECURITY

Security is its own required pass.

Not a checklist at the end of the project. Not an add on. A dedicated agent audits every single change before it can merge.

Vulnerabilities

The diff is read adversarially, looking for what could be exploited.

Leaked secrets

Keys and credentials never quietly end up in your codebase.

Auth gaps

Permission holes get caught before they reach production.

Data isolation

One customer's data cannot leak into another's view.

Ask your current provider how many of your changes got a security review. The honest answer is usually none of them.

CONTROL

Autonomous does not mean unattended.

The agents do the labour. You keep every decision that involves judgment, money, or your business.

1

It asks when it matters

When something needs a business decision the run pauses and waits for your answer, then picks up where it left off.

2

Budgets never surprise you

If the work will cost more than agreed it stops and asks. It cannot quietly overspend.

3

Stop it any time

Cancel a piece of work mid flight. Nothing reaches your app until the final step.

4

Everything is reversible

Each change lands as a tracked commit you can inspect, or undo.

Human at the center. AI at the edges.

SHIPPING

Shipping is boring, on purpose.

When work is approved it merges, builds, and rolls out to your live app automatically. No release nights, no maintenance windows.

Merge › Build › Roll out › Live

Seconds

Switchover time. Requests already in progress finish on the old version, so nobody sees a blip.

Three tries

Infrastructure hiccups retry automatically before anyone is bothered about it.

Handled

Custom domains and their certificates are issued and renewed for you.

AFTER LAUNCH

It watches itself.

Most software quietly breaks and waits for a customer to complain. Yours does not sit still.

Every 30 minutes

The system scans for errors it has not seen before and opens a fix ticket automatically, so a regression gets caught long before someone reports it.

Every 6 hours

A wider sweep reviews warnings and app behaviour, looking for the things worth a closer look that never actually crashed anything.

Repeat problems are grouped, so you are never buried in duplicate alerts. Each finding flows through the same five passes as planned work.

This is what tending actually looks like.

OWNERSHIP

You own it.
You can walk away.

Radical delivery speed raises a fair question: what is the catch? There is not one. What we build for you is a real codebase in a real repository, and it is yours.

Take the code

Clone the whole repository whenever you want. No permission needed.

Run it yourself

It is a complete, runnable project, documented for whoever picks it up next.

Bring your own developers

Your team or another shop can work on it and ship through the same pipeline.

No hostage situation

If the arrangement stops making sense, you leave with everything.

UNDER THE HOOD

Tuned per project. Never locked to one AI.

Each pass in the pipeline can run on a different model, chosen for that specific job and tuned for your project. When something better arrives, we swap it in.

Plan	Deep reasoning, because the plan sets the ceiling for everything after it.
Implement	Strong coding capability and stamina for the longest stretch of work.
Review and Security	Independent judgment, deliberately not the model that wrote the code.

Swap the engine, keep the app. Your software is not a bet on one AI company.

THE PAYOFF

What this means for your business.

Weeks, not months

Work that used to need a team and a quarter now takes a fraction of both.

Quality that is checked

Reviewed, security audited, and verified on every change, not just at launch.

A fixed price, agreed first

Scope and cost are settled before we build. No change order theatre.

Software you own

A real asset on your side of the table, not a subscription to someone else's platform.

The engine is why it is fast. The people are why it is right.

WORKING TOGETHER

How a build actually runs.

1

Discovery

We get the people who live inside the workflow in one room and find the one thing worth building first.

2

Blueprint

You get a clear scope and a fixed price before a line of code is written.

3

Build with you

The pipeline does the work. You see progress as it happens and feed changes in as you think of them.

4

Adoption

Training and documentation, so your team actually uses what we built.

5

Tend it

We monitor, fix, and extend it as your business changes. That is Walter Operator.

Done with you, not dumped on you.

Build it.
Tend it.
Own it.

Bring us the workflow that is costing you the most. We will tell you what it takes to build.

hello@heywalter.dev

heywalter.dev

Hey Walter Labs · Future Ready Holdings Inc. · Langley, British Columbia