ENTERPRISE DNA

A FIELD GUIDE FROM ENTERPRISE DNA

Working With Claude

How to put AI agents to work across your organization.
What Claude is, what it does in every part of a business,
and how to adopt it well.

FRONT MATTER

A note from Sam

This guide exists because most people have only been shown Claude as a chat box. That badly undersells what it has become.

The shift is simple to state. You no longer ask software a question and click through the answer. You hand an agent a job. It reads what it needs, does the work, checks its own result, and reports back. The person moves up a level, from operating the software to directing the work and approving what matters.

I have spent ten years at Enterprise DNA helping businesses do more with their data. This is the most useful version of that mission we have ever been able to deliver. Not a demo. Real agents doing real work inside a business, every day.

We do not write this from the outside. We run our own company on exactly what is in these pages. Our inbox, our CRM, our projects, our reporting, our proposals, and our content all run on a command center of Claude agents, with a person holding the gate. Every pattern in this booklet is one we use ourselves.

Read it as a field guide. What Claude is, what it can do in each part of your business, and how to put it to work without getting it wrong. If it lands, the last page tells you where to start.

Sam McKay

Founder, Enterprise DNA

CONTENTS

What is inside

PART 1 · THE SHIFT

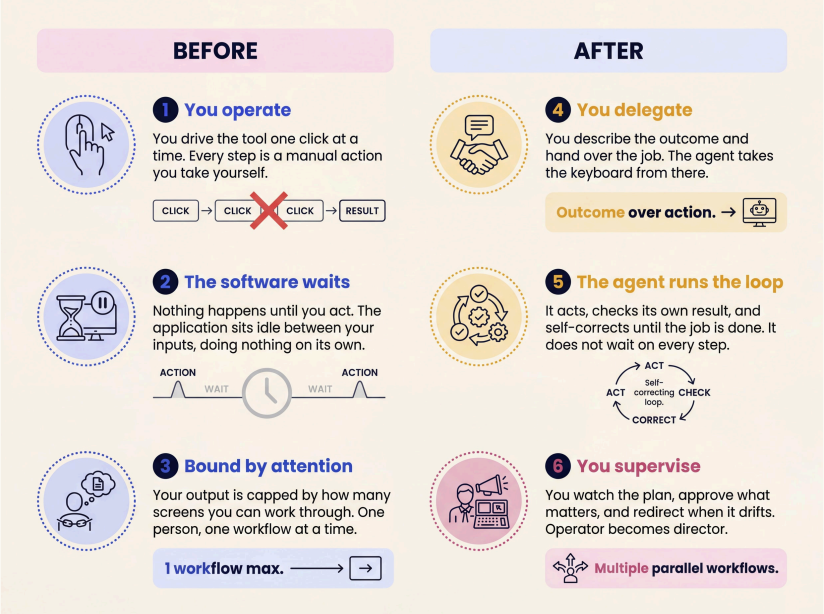
Why this matters now

> *The unit of work just changed.*

For thirty years, putting software to work meant a person asking it a question and clicking through the answer. That step is being removed. You now hand the work to an agent, and it runs.

From question to job. The old pattern was a query. You asked, the software returned a result, and you decided the next click. The new pattern is a job. You describe an outcome, the agent works a loop toward it, fixing its own mistakes as it goes, and reports back when the job is done.

From operator to director. This moves the human up a level. You stop driving every step and start directing the work. You set the goal, watch the plan, approve what matters, and redirect when the agent drifts. The agent holds the keyboard. You hold the judgment.



This is not a chatbot bolted on. A chatbot answers and waits. An agent acts, checks the result, and keeps going. Anthropic measured the difference in its own coding agent. Over six months, the longest unbroken run of autonomous steps roughly doubled, from about 9.8 to 21.2 consecutive tool calls. In the same window, the times a human had to step in per session fell by about a third. The work got longer and more complex, and needed fewer interruptions to get there.

The signal, honestly

> *Real and early. Both are true.*

A guide that only showed you the upside would not be worth your time. So here is the case on both sides, plainly.

The traction is real

Anthropic's revenue run-rate has climbed past roughly 30 billion dollars. That figure is an annualized run-rate, the current monthly pace projected out, not trailing revenue, so treat it as momentum rather than money in the bank. The customer base is broad. More than 300,000 businesses now use the platform. Over 1,000 of them spend more than a million dollars a year. Eight of the Fortune 10 are customers.

The most telling number is quieter. In 2026, 54 percent of new enterprise customers arrived through self-serve. They signed up and started building without a salesperson in the room. That is organic pull, the kind of demand that is hard to manufacture and harder to fake.

The honest counterweight

Adoption is not the same as production. Roughly four in five enterprises have adopted agents in some form. Only about one in nine run them in production, doing real work the business depends on. The gap between trying and trusting is wide. Gartner expects more than 40 percent of agentic-AI projects to be cancelled by 2027, most of them failing on weak governance and unclear returns rather than weak technology.

The thesis. This shift is real and it is early. The winners will not be the companies with the most access to the technology. Access is becoming universal. The winners will be the ones who adopt it well. That is what the rest of this guide is about.

PART 2 · WHAT CLAUDE ACTUALLY IS

Meet the Claude family

Claude is not one thing. It is a small family of surfaces, each pitched at a different kind of work. Here is the plain tour, so a leader knows what is what.

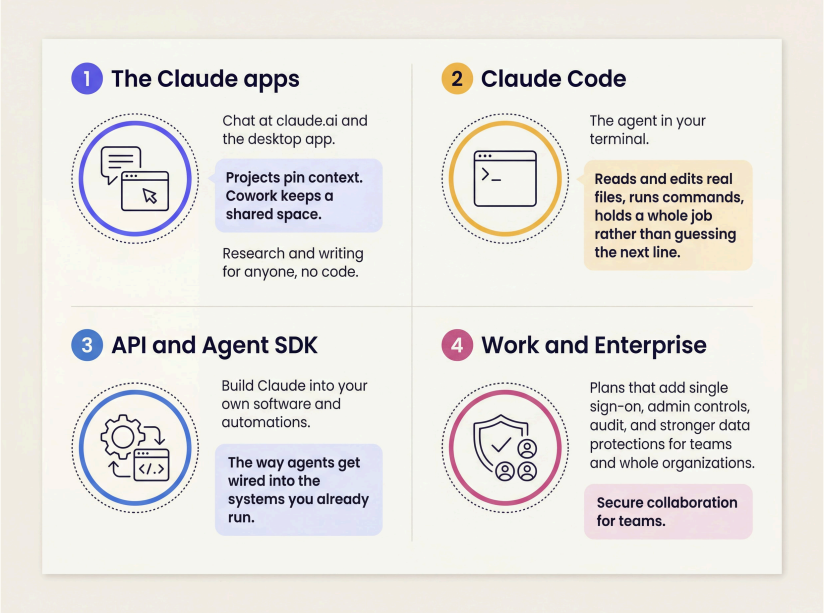
The Claude apps: claude.ai in the browser and the desktop app. Projects pin context to a thread. Cowork keeps a shared space going. Research, writing, and analysis for anyone, no code required.

Claude Code: the agentic worker that runs in a terminal, reads and edits real files, and runs commands. The powerhouse, covered next.

The Claude API and Agent SDK: how you build Claude into your own software and automations.

Claude for Work, Team, and Enterprise: the plans that add single sign-on, admin controls, audit, and stronger data protections.

Most people meet Claude through the apps. Most of the real leverage comes from the others.



Claude Code, up close

This is the centerpiece. A terminal-native agent that runs where the work already is.

It meets the work, not the other way around. Claude Code runs in the command line, in the desktop app on Mac and Windows, in the browser at claude.ai/code, inside VS Code and JetBrains, and headless in a pipeline like GitHub Actions. The same project memory and connections follow you across all of them.

Suggesting a line versus holding a job

An autocomplete tool guesses the next line as you type. An agent holds the job. It gathers context across the whole codebase, makes a plan, edits many files at once, runs the tests, and fixes its own mistakes before it hands back. The difference is not speed. It is responsibility for the outcome.

Not only for engineers. Non-technical people increasingly use Claude Code too. Not to write apps, but as a general agent that can touch files, call APIs, and run multi-step work that used to need a script and a specialist.

> *In 2024 the models could draft. In 2025 they could use tools. In 2026 they can hold a job.*

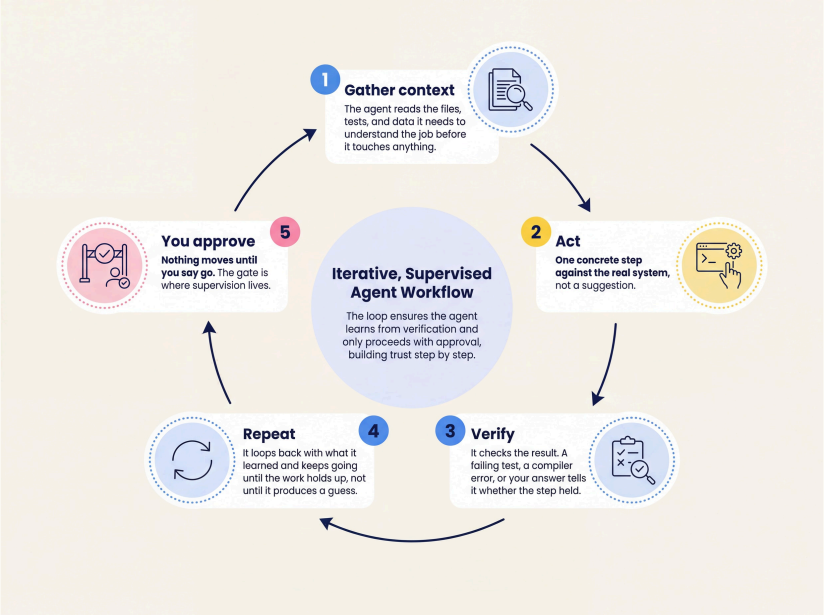
How an agent actually works

Underneath, an agent runs a simple loop. Gather context, act, verify, repeat until the job is done.

The loop. The agent reads what it needs. It makes a change or runs a command. It checks the result, a failing test, a compiler error, your answer. Then it loops back with what it learned. It keeps going until the work holds up, not until it produces a guess.

Where you sit in the loop

Plan mode shows you exactly which files it will touch before it changes anything. Nothing moves until you approve. **Auto-accept** hands trusted, repetitive work to the agent, which edits and tests on its own. You choose the setting per task. The approval gate is the point. It is where supervision lives.



Choosing your model

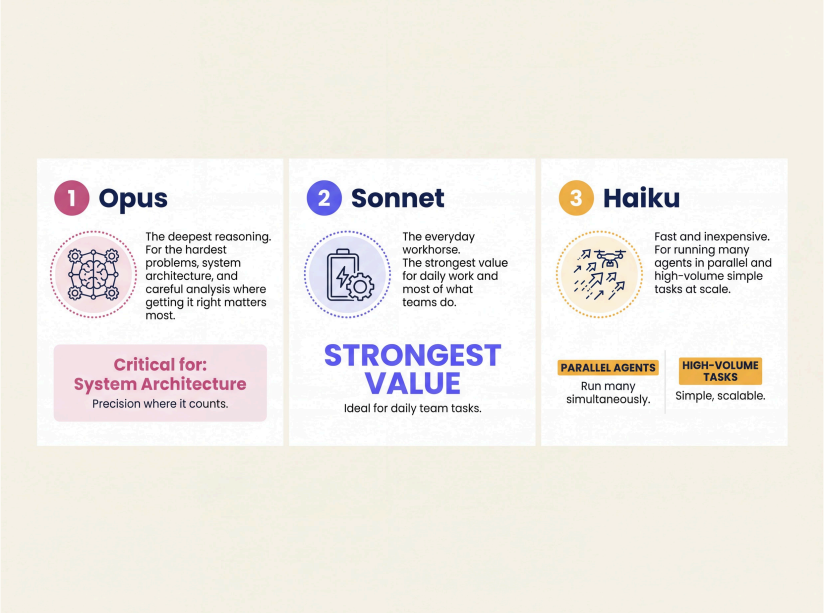
Three models, one decision. Match the model to the difficulty of the work, not the prestige of the name.

Opus
The deepest reasoning. For the hardest problems, architecture, and careful analysis.

Sonnet
The everyday workhorse. The best value for most of the work most teams do.

Haiku
Fast and inexpensive. Ideal for running many agents in parallel and high-volume simple tasks.

Context is generous. Big context windows let the agent hold a lot of your code or documents at once, so it reasons over the whole picture rather than a slice.



Most teams mix. Sonnet for daily work, Haiku for scale, Opus when it is genuinely hard. You can switch mid-task, so a cheap model can do the legwork and a strong one can make the call.

PART 3 · THE BUILDING BLOCKS

What turns a model into your specialist

A general model is capable, but it is generic. It knows a great deal about the world and nothing about your world. You close that gap by layering five things on top of it.

Think of these as the difference between a sharp new hire and a colleague who has been with you for years. The new hire is bright. The colleague knows your standards, your shortcuts, your systems, and the lines they must not cross. These five blocks are how you give a model that same depth, on day one.

Memory
Your standards and the way you work, written down so the model reads them every session.

Commands and Skills
Your workflows packaged by name, so anyone can run them the same way.

Subagents
A team of specialists the main agent can spawn for focused jobs, in parallel.

Hooks
Your guardrails, fired automatically at the moments that matter.

Connections
Your real systems, reached securely through an open standard called MCP.

The rest of Part 3 takes them one at a time. None of them require you to be technical to understand.

Teach it your standards: CLAUDE.md and memory

> *Your standards should live in a file, not in one person's head.*

CLAUDE.md is a plain text file you keep with your project. It loads at the start of every session and tells the agent your rules, your conventions, and the way your team works.

Write it once, everyone inherits it. The same file that guides you guides every teammate, including a new hire on their first day. They do not need to absorb the unwritten rules over months. The rules are written. The agent applies them from the first request.

Standards scale without nagging. An organization can set a managed, company-wide version that no one can override. That is how a leadership team makes sure every project follows the same security practices, the same naming, the same tone, without sitting in every review.

The flip side: memory

CLAUDE.md is what you tell the agent up front. Memory is what the agent records as it works. When it learns a quirk of your setup or a decision you made, it writes that down so it does not relearn the same thing next time. The knowledge compounds instead of resetting. Hard-won context stops walking out the door when people do, because it lives in files that travel with the work.

Codify the way you work: commands and Skills

Slash commands and Skills package a workflow into something anyone can run by name. The hard-won sequence of steps stops being tribal knowledge and becomes a single, repeatable instruction.

A command is a workflow with a name. A /ship command that runs your tests and your lint checks before anything gets committed. A marketing command that checks copy against your brand voice and outputs ready-to-upload ads. The person running it does not need to remember the steps. The steps are baked in.

They spread the moment a colleague pulls the project. Because commands and Skills are plain text checked in alongside your work, a teammate adopts them the instant they pull the repository. No setup call. They simply have your procedures, ready to run.

What this looks like at scale

Finance: Anthropic's own finance team runs more than 150 skills.

Marketing: their marketing team built a tool that turned 30-minute ad creation into 30 seconds.

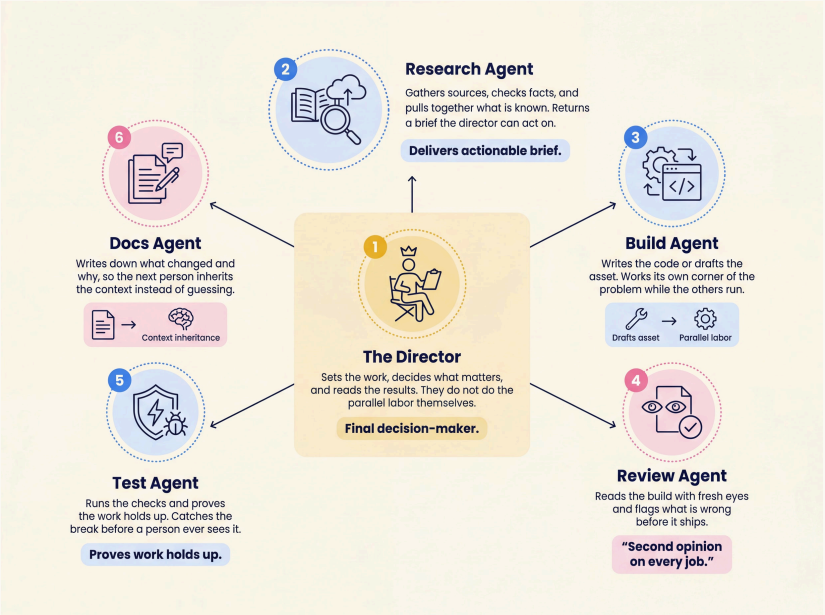
This is the mechanism by which a generic model becomes a specialist for your business. You hand it your procedures once, and it runs them your way every time after.

One human, a fleet of agents: subagents

Subagents are specialized helpers the main agent spawns for focused jobs. Each gets its own clean context, and they often run in parallel.

Fan out, then merge. Send a researcher, a builder, and a reviewer at the same time, each working its own corner of the problem, then bring the results back together. The work that would run in sequence with one worker runs side by side with several.

Cover a whole stack at once. A nightly run can put one agent on each service you operate, every one of them working through the night. You wake up to the results instead of the queue.



The shape of the work changes, but your role does not. The human stays the director. You set the task, decide what matters, and read what comes back. The fleet does the parallel labor. You make the calls.

Automation and guardrails: hooks

> *A guardrail that runs every time and never forgets.*

Hooks fire automatically at set moments. Before a tool runs. After a file changes. When a session ends. They let you automate the boring-but-important and enforce the rules without slowing anyone down.

Automate the work nobody should have to remember. A hook can format every file the agent touches, so the output is always clean. It can log every sensitive action to your audit trail, so there is always a record. These are the tasks that matter most when they are skipped, and hooks make sure they never are.

Stop the wrong thing before it happens. A hook can block a dangerous command before it runs, not after. It can post a summary to your team channel the moment a job finishes, so the right people always know. The check is built into the moment, not bolted on later.

Why leaders care

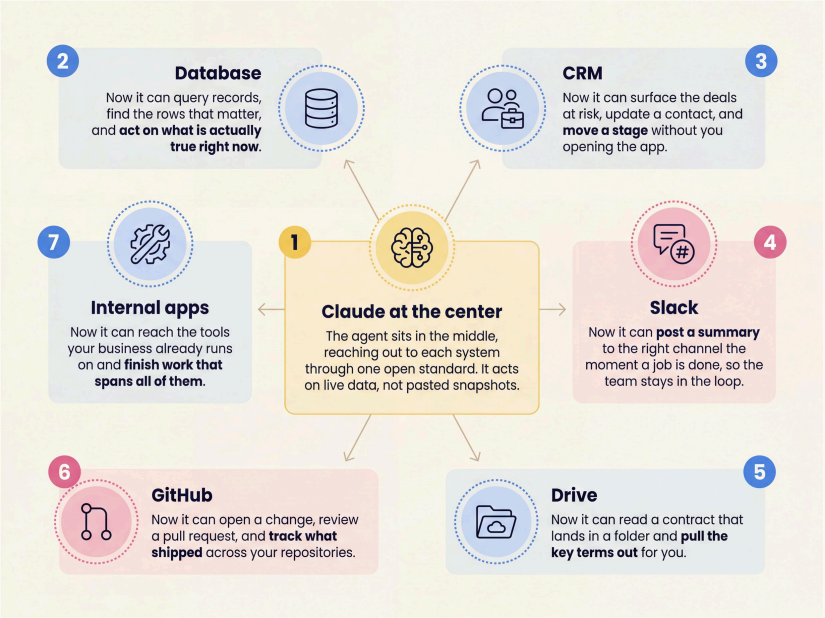
Hooks are a large part of how a company keeps agents safe and consistent at scale. A policy you ask people to remember gets forgotten under pressure. A policy wired into a hook runs every single time, on every agent, without exception.

Plug it into your systems: MCP

MCP is an open standard, now governed by a foundation, that lets Claude securely connect to your real systems. Your database, your CRM, Slack, Drive, GitHub, your internal apps.

Without it, the agent only knows what you paste in. It is sharp, but it is sealed off from the live state of your business. With MCP, it reads and acts on the real thing.

One instruction can now span several systems. "Find the deals at risk in the pipeline" becomes a real query against your CRM. "When a contract lands in the folder, pull the key terms, create the project, and post a summary to the channel" becomes one instruction that reaches across three systems and finishes the job.



This is the connective tissue of the whole operating layer. Memory teaches the model your standards. Commands package your workflows. Subagents give you a team. Hooks hold the line. MCP plugs all of it into the systems where your work actually lives.

PART 4 · CLAUDE ACROSS THE BUSINESS

Claude across the business

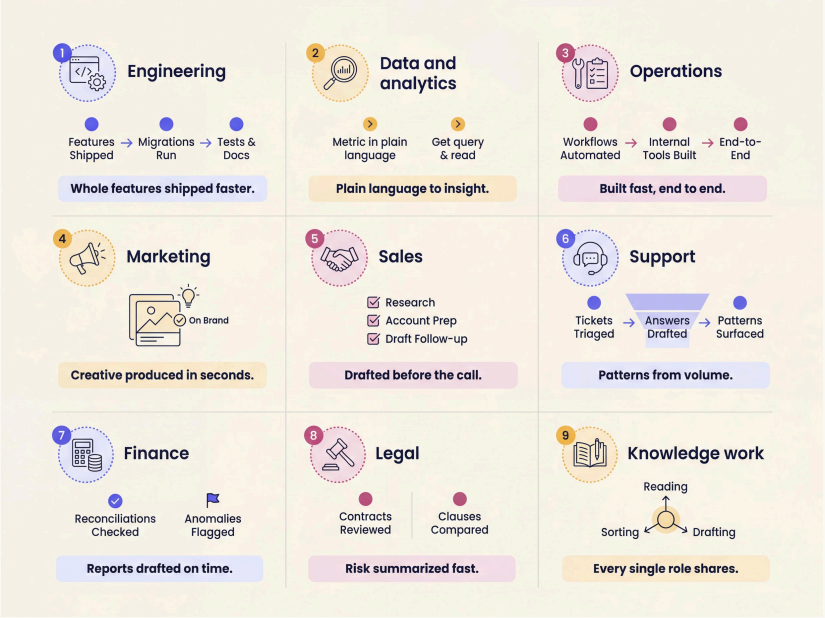
> *The agent does the production work. The human authorizes the consequential step.*

The next nine pages walk function by function. They all follow the same shape, so once you see the pattern in one, you can read the rest fast and port it to a function this guide does not name.

The pattern, every time

The agent produces, the human authorizes. The agent does the bulk of the work. It writes the drafts, matches the records, runs the first pass, preps the analysis. Then it stops at the step that has consequences. A person checks that step and authorizes it. The work that used to fill the day becomes the work you review.

So as you read each function, watch for two things. What the agent now does on its own, and exactly where the human still signs off. That second line is not a limitation. It is the design.



Engineering

> *This is the most mature function. Not autocomplete. Whole features, shipped.*

Ship features end to end. The first wave of coding assistants finished your line. Agents now take a ticket, write the code across the files it touches, run the tests, and open the pull request. The unit of work moved from the keystroke to the feature.

Migrations at a new speed

Big rewrites used to be the work nobody had time for. Stripe moved 10,000 lines of Scala to Java in four days, roughly ten engineer-weeks of work. Wiz migrated a 50,000-line Python library to Go in about 20 hours against a two-to-three-month estimate. The agent does the mechanical translation. The engineer checks the judgment calls.

Smartsheet: engineers shipped about 3x more code and merged 31% more pull requests than peers.

Ramp: cut incident-investigation time by 80% and shipped more than a million lines in 30 days.

And the work that always slips. Agents read unfamiliar legacy code and explain it in plain terms. They write the tests and the docs that get deprioritized every sprint, because for them that work is not a tax.

Where the human stays

The production cutover, the rollback plan, and the final merge stay with an engineer. The agent gets you to a reviewable change. A person decides it goes live.

Data and analytics

> *Describe the metric in plain language. Get the query, the chart, and the read.*

The shift. You used to hand-write the SQL or the DAX. Now you describe the metric in business language and Claude drafts, optimizes, and explains the query against your real schema, once it is connected to the database through MCP. Report build times drop from days to under an hour, and you get the query, the chart, and a written read of the trend in one pass.

A real question, answered

Ask why margin dropped in the north region last quarter. The agent turns that into the measures and visuals you need, then explains the logic so the analyst learns the pattern, not just the answer. The explanation is the part that compounds.

Before the analysis, the cleanup

It handles the unglamorous part too. Point it at a messy dataset and it profiles the columns, flags the broken join keys, and proposes a cleaning plan. And it lets a non-analyst self-serve a query they would otherwise have queued with the data team.

Where the human stays

Metric definitions and join semantics still need someone who knows the business. The agent can write the query. It cannot know that "active customer" means something specific to you. And board numbers never ship unread.

Operations and internal tooling

> *This is where MCP turns Claude into an operator, wired into your systems.*

Automate a workflow end to end. Connect the tools and the agent runs the whole sequence across them, not just one step in isolation. The handoffs that used to need a person passing work between apps close up.

Tools without a developer

Describe the small internal tool you keep wishing you had. A routing tool. An intake processor. A reconciliation checker. The agent builds it from the description, so the team that feels the gap is the team that fills it.

Volume the team could never reach

Process documents at scale. Newfront, an insurance brokerage, cut document-processing costs by 60%. And on project hygiene, the agent reads the week's commits and threads, updates the tracker, drafts the status report, and flags what has stalled.

Where the human stays

Anything that writes to a system of record or triggers something irreversible gets an approval step. The pattern that works, every time, is that agents draft and stage, and humans authorize.

Marketing and content

> *A non-coder ran growth marketing on Claude Code. That is the flagship case.*

The proof is in-house. Anthropic's own marketing team is the headline example. A non-coder ran growth marketing on Claude Code, and the production numbers moved hard.

Ad creative
From 30 minutes to 30 seconds.

Case studies
From 2.5 hours to 30 minutes, about 10 hours saved a week.

On voice by construction

A brand-voice command checks every output against the brand rules before it ships. The work comes back on voice by construction, not by hope and a second read.

Then it multiplies. Hand it one long-form asset and it repurposes that into a campaign's worth of derivatives, with the voice held constant across all of them. It runs research and competitive monitoring before a single word is written.

Where the human stays

The final creative call and the brand's taste stay with a person. The agent removes the production grind. It does not remove the editorial judgment.

Sales and CRM

> *The work runs ahead of the rep. The send waits for a person.*

Build the list, then warm it. Claude can assemble a prospect list filtered to your ideal customer, enrich each record, and ground every detail in something real about the company.

Collapse meeting prep. Pre-call briefs pull account context and materials into one read. ServiceNow saw up to a 95% reduction in meeting-prep time across 29,000 people doing exactly this.

Outreach: personalized openers in under 30 seconds an email, each grounded in a real detail.	**Pipeline:** surface stalled deals and missing next steps.
Hygiene: drafted call summaries and logged activity keep the CRM clean.	**Proposals:** first drafts built on proven templates.

Where the human stays

The bright line is the send. Draft and stage outreach automatically. A person authorizes the actual send to a real prospect or customer. This is a documented failure mode, so name it plainly and hold the gate.

Customer support

> *Answers are only as good as the knowledge behind them.*

Deflect the documented questions. Agentic support reliably resolves 55 to 70% of volume. These are the order-status, returns, billing, and password-reset tickets that follow a known path.

Help the agents who handle the rest. Claude drafts replies and pulls the right knowledge in real time. SNCF, the French national railway, assists 150 customer-service agents this way. It also summarizes long threads so a handoff starts informed.

Reach inside the building too. The same pattern powers HR and IT helpdesks. Newfront's HR assistant reclaimed more than a month a year of admin time.

Keep the knowledge fresh. Claude can spot gaps in the knowledge base and draft new articles from resolved tickets, so the answers improve as the volume flows through.

Where the human stays

Complex, emotional, or exception cases route to a person. A review path catches the confident-but-wrong answer before it reaches the customer.

Finance and admin

> *Proven on real books, including Anthropic's own.*

This is not theory. Anthropic's own finance team runs more than 150 skills against live numbers.

Reconciliation with a learning curve. Roughly 60% auto-matched in the first month, 85% or more by the second, with a controller-ready exception report at the end.

Reporting: drafted variance commentary and management reporting from the raw numbers.

Month-end: prepaid schedules, consolidations, and the repeatable close steps.

Invoicing: draft invoices and records through an API, stopping at a draft, never auto-sending.

Context: 59% of senior finance leaders already report using AI in finance.

Where the human stays

Anything that posts to the ledger, files with an authority, or moves money needs controller sign-off. The drafts come fast. The approval stays human.

Legal and compliance

> *High value, and the loudest caution in the book.*

First-pass review, faster. Contract review shows about a 67% time cut, roughly four hours down to 55 minutes per contract.

At deal scale. On due diligence, a mid-size firm ran the contract review for a $45M acquisition, a three-associate, two-week job, in three days.

Research and policy. Claude synthesizes legal research and drafts policy, with every citation and conclusion verified by a lawyer. A compliance gap analysis lands as a structured first cut.

Built for regulated work. It handles data-processing agreements and business-associate agreements, offers a zero-data-retention option, and does not train on your data.

Where the human stays (non-negotiable)

Every legal conclusion, and anything client-facing or filed, gets lawyer review. Claude is a first-draft assistant here, not a final authority.

PART 5 · THE OPERATING LAYER

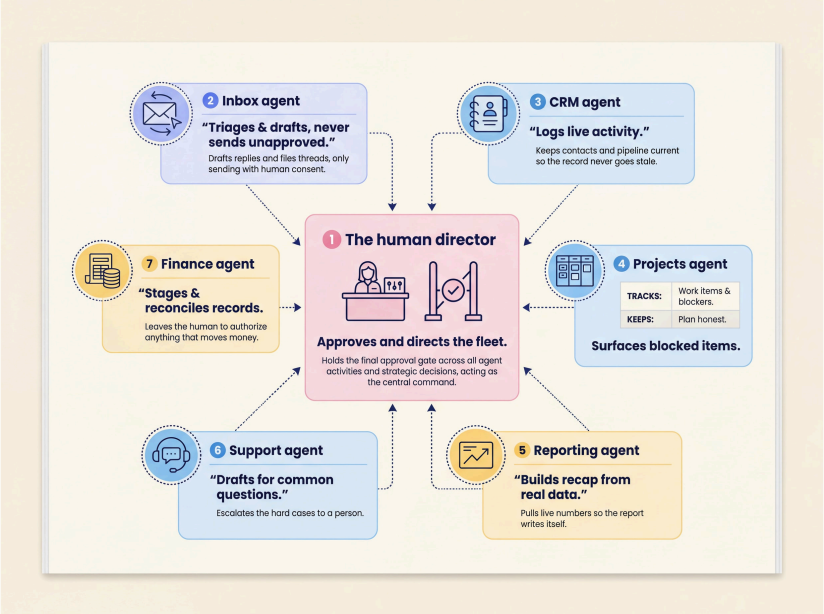
From scattered agents to a command center

> *One human. A fleet. One surface.*

No single model runs a whole company. The destination is not one giant agent. It is many specialized ones, coordinated.

Specialized, not general. You end up with one agent wired into the inbox, another into the CRM, another into projects, another into reporting. Each does the production work of one function, and does it well, because its job is narrow and its context is deep.

Coordinated through one surface. A fleet of agents working in isolation is just scattered automation. The value comes from running them through a single control surface, with one human holding the approval gate across all of them. That is the difference between a pile of tools and a command center.



Where this is heading. Analysts now describe AI becoming the primary user of enterprise software, with people supervising rather than clicking. The agent market is projected to grow from roughly 7.8 billion dollars in 2025 to over 50 billion by 2030. As that happens, the edge moves from which model you have to the orchestration layer that keeps the fleet coherent.

The line that keeps it safe

> *Let the agent do the volume. Keep the judgment and the irreversible step with a person.*

One principle runs through every function in this guide. It is worth stating once, plainly, because it governs all the rest.

The line. Agents draft, stage, and propose. Humans authorize, judge, and own anything that cannot be undone. Give the agent full autonomy on the safe majority of the work. Put a human gate on the part that carries real risk.

Why this one line resolves both fears

Over-trust. The fear that an agent will do something costly and irreversible on its own. The line removes it by construction. Nothing that cannot be undone happens without a person approving it first. The blast radius of a mistake stays small.

Skill atrophy. The fear that people stop being able to do the work. The line removes it too, because the human never leaves the judgment seat. You are not watching the agent think. You are deciding what ships.

How it works in practice

Plan mode. Before the agent changes anything, it shows you exactly what it intends to touch. Nothing moves until you approve. **Read the change.** Treat the agent's output the way you would read a colleague's work before it goes out. Skim what is routine. Read closely where it counts.

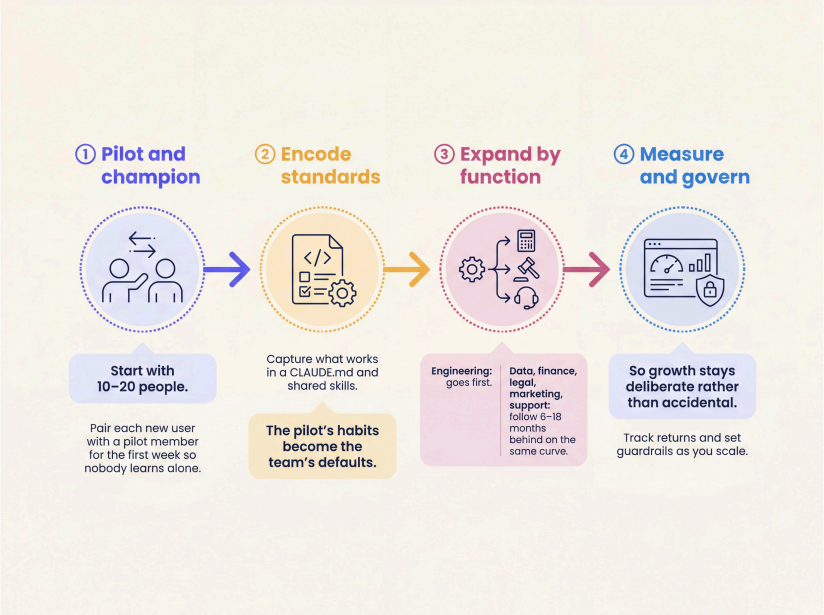
PART 6 · ADOPTING IT WELL

Rolling it out

Adoption does not happen by mandate. It happens by example. The teams that win do not announce a tool. They grow a habit, one person at a time.

Start with a champion, not a memo. One person uses it well. They share wins where the team already reads, the standup channel, the weekly review. They become the person others ask. Then they grow the circle. This takes about forty minutes a week, and it is the highest-leverage forty minutes in a rollout.

Crawl, walk, run. Begin with a pilot of ten to twenty people. Pair each new user with a pilot member for the first week, so nobody learns alone. When the pilot is fluent, expand. Skipping the pilot is the most common way a rollout stalls.



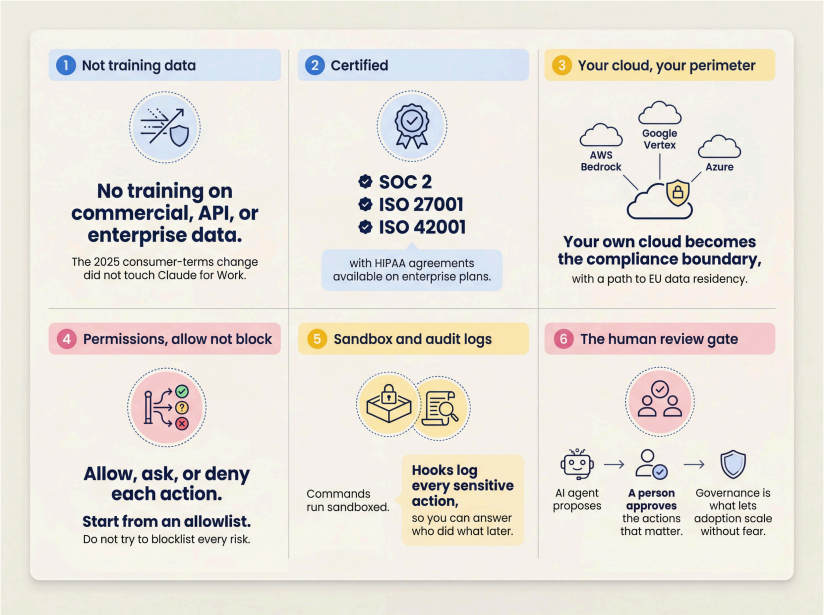
Name the lag out loud. The gains land in engineering first. Data, finance, legal, marketing, and support follow six to eighteen months behind, on the same curve. That is not a failure. It is the shape of the curve.

Governance and security

Before a security team will say yes, they need one fact. Start there, then give them the rest.

Your data is not training data. By default, Anthropic does not train on commercial, API, or enterprise data. The 2025 consumer-terms change that made headlines applied to consumer accounts. It did not apply to Claude for Work, the API, Bedrock, or Vertex. That is the sentence that unblocks most reviews.

The certifications are in place. SOC 2, ISO 27001, and ISO 42001, with HIPAA agreements available on enterprise plans. You can run Claude inside AWS Bedrock, Google Vertex, or Azure, which makes your own cloud the compliance perimeter, with a path to EU data residency.



Claude Code's own controls

Managed settings: IT sets them centrally and no one can override them.

Permissions: allow, ask, or deny each action. Start from an allowlist.

Sandboxing: commands run contained, not loose on the machine.

Hooks: every sensitive action is logged for audit.

The pitfalls, and how to avoid them

This is the honest chapter. Four failure modes show up again and again. Each has a fix, and none of them is "use it less."

Shadow AI. Most data leaks come from unapproved tools, not approved ones. Banning the technology pushes people onto personal accounts you cannot see, which is the bigger risk. Formalize use and monitor it instead.

Over-trust. Automation bias makes people accept confident-but-wrong output. Build small friction back in. Ask the agent for the counterargument. Cross-check the things that actually matter before they ship.

Skill atrophy. A Microsoft and Carnegie Mellon study found heavier reliance correlated with less critical thinking. Protect the doing that keeps judgment sharp.

Cost blowups. Microsoft rolled Claude Code to thousands of engineers, then pulled most of it back. Not because it was too expensive, but because they ran a metered tool with no spend controls. Put spend limits on from day one.

> *Agents run, humans direct.*

That one line resolves all four. The bottleneck is not access to the technology. It is the quality of adoption around it.

Measuring the return

Measure it honestly, or you will either oversell it and lose trust, or undersell it and lose the budget.

The real signal. Anthropic's own engineers merged 67% more pull requests per day after adopting Claude Code. They reported productivity rising from about 20% to 50% over a year. And 27% of the assisted work was tasks that would not have been done at all, work that simply did not happen before.

A frame leaders can borrow

Take fifty developers on the top plan, roughly $120K a year. If that lifts merged pull requests from 5,200 to 8,400, that is about $37.50 per extra pull request. If each one saves a couple of hours, that is a four-to-one return. The arithmetic is simple, which is the point.

Time-to-first-PR: how fast a new hire ships their first change.	**Hours returned:** time the team gets back for higher work.
The hire avoided: the role you no longer need to fill.	**The tools dropped:** the subscriptions you stop renewing.

Keep it honest. Anthropic's own study stresses it does not count the time spent checking the agent's work. The median company still trails its power users. The bottleneck is adoption quality, not access.

PART 7 · WHERE WE FIT

Where Enterprise DNA fits

> *We run our own company on exactly this.*

Everything in this guide is something we do ourselves. Enterprise DNA serves customers across three continents with a small team, because the work around the work is run by a command center of Claude agents, with us holding the gate. If it stops working for us, we feel it before you do. If we improve it, you get the improvement.

There are three ways we help, and you can start with any one of them.

Learn it
Online learning and live training to get your team fluent with Claude and Claude Code.

Build it
We install your operating layer. The agents, the connections, the guardrails, the review surface.

Run it
We operate the layer alongside your team every month, and keep improving it.

Start with the audit

Thirty minutes, free. We find the first three places Claude can remove work in your business this quarter. You leave with the map whether or not we build it together.

Email: sam.mckay@enterprisedna.co.nz	**Web:** enterprisedna.co
Book: calendly.com/sam-mckay/discovery-call	**The decision:** yours. No pressure.

ENTERPRISE DNA

A NOTE IN CLOSING

You no longer ask software a question. You hand an agent a job, and you supervise. The companies that learn to direct a fleet of agents, with a person holding the gate, will run circles around the ones that did not.

FROM INSIDE THIS GUIDE.

Put Claude to work across your business.

ENTERPRISE DNA · AUCKLAND · 2026 · ENTERPRISEDNA.CO