delvr.ai

AI setup basics for business.

Workspaces, agentic tools and the data underneath. What to build, in what order, and who owns it. Written from the buyer's side, not the vendor's.

PAUL SALVAGE · FRACTIONAL CHIEF AI OFFICER · DELVR.AI · JULY 2026

01

Where this guide sits

This is a guide to setting up AI tools, and tools are the smallest layer of the work. Getting workspaces, agents and knowledge organised is necessary, mostly mechanical, and increasingly commodity. It will not change what your business is.

The larger layers sit above it: which products get reimagined now that this capability exists, which processes get rebuilt rather than assisted, and what the operating model looks like when some of the work no longer needs a person. That is strategy work, it is specific to each business, and no generic guide can do it.

This guide covers the layer a generic guide can cover, properly. Get it right and the strategic work has something to stand on. Get it wrong and even good strategy dies in tool sprawl and unowned knowledge.

02

Start with the problem, not the product

Every major AI vendor now sells the same four surfaces under different names:

1. **Shared chat workspaces**: chat with persistent context and a knowledge base attached (Claude Projects, ChatGPT Projects, Gemini Gems)
2. **Supervised agentic work**: an environment where the AI executes multi-step tasks across your files and tools while you watch (Claude Cowork, ChatGPT with connectors and computer use, Copilot)
3. **Agentic work against files and repositories**: the same capability pointed at version-controlled files (Claude Code, Codex, Gemini CLI)
4. **Unattended agents**: scheduled or triggered runs of the above

The marketing will happily sell you all four. Most companies need one or two, chosen deliberately.

The real question underneath all of them is the same: **where does your context live, and who does the work?**

Every setup decision reduces to those two variables.

- Context can live in people's heads, in chat threads, in a workspace knowledge base, or in version-controlled files.
- Work can be done by a person asking questions, by a person supervising an agent, or by an agent running unsupervised.

The tools are just different positions on that grid. Pick positions based on the problem, then map to whichever vendor's tool fits.

03

The four surfaces, honestly described

Shared chat workspaces: context for conversation

What they actually are. A folder of documents plus a standing instruction, shared across chats and optionally across a team. Every conversation inside can draw on that knowledge.

The problem they solve. People asking the same category of question repeatedly against the same body of knowledge. Sales teams querying the pricing playbook. Ops asking about SOPs. A founder drafting in a consistent voice.

Where they earn their keep. Low skill floor. Anyone who can use chat can use one. Setup is an afternoon. For a 20 to 200 person business this is usually the right first move because adoption cost is near zero and the failure mode is cheap.

The honest limitations. Workspace knowledge is a silo. It lives inside the vendor's product, is awkward to audit, drifts out of date silently, and does not version. Nobody knows which copy of the SOP is in there or who changed the instructions. It answers questions well; it does not do work. And knowledge maintenance is a job someone has to own or the workspace quietly rots.

Products, runtime, ownership. - Examples: Claude Projects, ChatGPT Projects, Gemini Gems, Copilot agents built in Copilot Studio - Runs: in the vendor's cloud, accessed via browser or mobile app. Nothing on your infrastructure. - Owned by: the relevant function head. The sales workspace belongs to the sales lead, the ops workspace to the ops lead. IT administers seats and access; it should not own content, because IT cannot tell when the pricing playbook is out of date.

Supervised agentic work: task execution with a person watching

What it actually is. An environment where the AI works across files, connected tools (email, CRM, calendar, drives) and the web, over multiple steps, under supervision.

The problem it solves. Tasks, not questions. Produce the board pack. Reconcile these two spreadsheets. Research these ten prospects and draft outreach. Anything a capable analyst would take half a day to do with the materials in front of them.

Where it earns its keep. The value is in delegation of multi-step work that touches several systems. If your team's bottleneck is production of deliverables rather than access to answers, this is where the hours come back.

The honest limitations. It is per-person, per-session. The output is good but the process is not captured anywhere. Two people delegating the same task will get it done two different ways. It is a productivity tool, not an operating system.

Products, runtime, ownership. - Examples: Claude Cowork (desktop app, works against local files), ChatGPT Agent (runs in a cloud VM, no local file access), Microsoft Copilot Cowork (web, deepest Microsoft 365 integration), Manus. The local-versus-cloud distinction matters: desktop tools can work directly on the files your team already has; cloud agents need everything uploaded or connected first. - Runs: on the individual's machine (desktop apps) or in the vendor's cloud, with connectors reaching into your email, CRM and drives. - Owned by: the individual producer using it, day to day. IT or ops owns

connector approval and data access policy, because a connected agent inherits whatever the person can see. This is the surface where governance actually matters: the tool acts, it does not just answer.

Files and repositories: durable, versioned, auditable

What it actually is. The same agentic capability, but pointed at a folder of files under version control. Despite the developer branding on these tools, the pattern is not only for software. Your SOPs, playbooks, client records, brand voice, decision logs and prompts can all live as plain text files in a repo that the AI reads and edits.

The problem it solves. Knowledge and process that must be durable, reviewable and owned by the company rather than trapped in a chat product. Every change is a diff. You can see who changed what, roll back, and take the whole thing to a different vendor tomorrow.

Where it earns its keep. This is where compounding happens. When your processes are files, agents can execute them consistently, improvements persist, and onboarding becomes "read the repo." Companies that treat their operating knowledge as an engineered asset get more out of every subsequent AI capability that ships, from any vendor.

The honest limitations. The skill floor is real. Someone needs to be comfortable with files, folders and version control, and most mid-market teams have two or three such people at best. Forcing this pattern on a team that lives in Word and Outlook fails. It is the right destination and usually the wrong starting point.

Products, runtime, ownership. - Examples: Claude Code, OpenAI Codex, Gemini CLI. All run in a terminal or IDE; some also ship in desktop apps. The repo itself lives on GitHub, GitLab or Bitbucket, which is your infrastructure decision, not the AI vendor's. - Runs: locally on the operator's machine, reading and editing files in the repo. The repo host provides the versioning, review and access control. - Owned by: split ownership, and this split is the point. A technical lead owns the tooling and repo structure. Content ownership is distributed: each playbook, SOP or policy file has a named owner from the relevant function, visible in the repo history. The company, not any vendor, owns the asset itself.

Unattended agents: work without a person in the loop

What they actually are. Scheduled or triggered runs of the above. Nightly pipeline reviews, inbound lead triage, weekly reporting.

The problem they solve. Recurring work where the process is stable and the cost of an occasional error is tolerable or catchable.

The honest position. Automate last. An agent is only as good as the codified process behind it, which is why this sits downstream of the repo pattern. Automating an undocumented process produces fast, consistent versions of your existing mistakes. Every unsupervised agent needs a defined failure mode: what happens when it is wrong, who notices, and how fast.

Products, runtime, ownership. - Examples, in ascending order of capability and cost to maintain: scheduled tasks inside the chat platforms (ChatGPT Tasks, Claude scheduled tasks), workflow platforms with AI steps (Zapier, Make, n8n), agent platforms (Copilot Studio, Gemini Enterprise workflows), and custom builds on agent SDKs for anything genuinely bespoke. Most mid-market automation belongs in the first two tiers. - Runs: in the vendor's cloud on a schedule or trigger. Custom builds run wherever you host them. - Owned by: the owner of the process being automated, full stop. If the weekly pipeline report agent

breaks or drifts, the sales lead notices and answers for it, not IT. IT owns the platform, credentials and kill switch. An agent with no process owner is an incident waiting for a discovery date.

04

The data layer: integrate or start fresh?

Everything above runs on context, so the underlying data question decides how well any of it works. The answer is: integrate to systems of record, curate everything else. And you almost certainly do not need a vector database.

Two kinds of company data, two different answers

Live operational data (CRM, email, calendar, finance system, support tickets) should never be copied into a new layer. It changes constantly, the system of record already has permissions and audit built in, and any copy is stale within hours. Connect the AI to the source directly. Every major platform now supports this through connector standards (MCP has become the common protocol), and letting the model query the live system is settled practice, not a vendor pitch.

Reference knowledge (SOPs, playbooks, policies, pricing logic, past decisions) is where "integrate to whatever exists" quietly fails. In most companies this material is scattered across SharePoint, Drive, someone's desktop and four versions of the same document, none marked canonical. Pointing AI at that mess produces confident answers from the wrong version. Here you do build fresh, but the fresh thing is not a database. It is a curated corpus: the repo of plain text files described above. Small, canonical, owned, versioned. This is editorial work, not a data project.

The architecture that works at mid-market scale

1. **Connectors to systems of record** for anything live. No copies.
2. **A curated file corpus** for reference knowledge. Markdown in a repo, or workspace knowledge if the team cannot handle a repo yet. Deliberately small; a hundred good documents beats ten thousand unsorted ones.
3. **Nothing in between.** No middleware knowledge platform, no enterprise search product, no vector infrastructure, until a specific problem demands it and you can name the problem.

The uncomfortable truth: the bottleneck is never retrieval technology. It is that nobody can say which document is authoritative. That is an ownership and hygiene problem, and buying infrastructure to route around it just retrieves the mess faster. The highest-value work in the whole stack is someone senior spending two weeks deciding what is canonical and deleting the rest.

05

The generic problems, mapped

Problem you are actually solving	Right surface
"People keep asking the same questions and the answers live in documents"	Shared workspace
"Everyone gets a different quality of answer depending on how they prompt"	Shared workspace with a firm instruction set
"Our team spends hours producing recurring deliverables"	Supervised agentic work
"Work touches email, CRM and spreadsheets at once"	Supervised agentic work with connectors
"Our operating knowledge has no source of truth"	Curated file corpus, before any AI question
"We want AI improvements to persist rather than reset each session"	Repo plus an agentic coding tool
"The same task runs weekly and the process is stable"	Unattended agent, only after the process is codified

06

Best practice sequence for a company starting cold

Phase 1: Shared workspaces (weeks 1 to 4). Pick two or three high-question-volume domains. Build a workspace for each with a tight instruction and current documents. Assign an owner per workspace whose job includes keeping the knowledge current. Measure whether people stop interrupting each other for answers.

Phase 2: Agentic work for the producers (months 2 to 3). Identify the five people whose job is producing deliverables rather than answering questions. Give them a supervised agentic environment and connectors to the systems they already use. Have each one document their best delegated task so wins spread.

Phase 3: Repo as source of truth (months 3 to 6). Take the knowledge that proved valuable in Phase 1 and move it into version-controlled files. This is a knowledge management project wearing an AI costume, and it is the single highest-leverage move on this page. From here, workspaces and agentic tools both draw from the repo rather than from stale copies. This is also the phase where the data layer decisions above get made: connectors for live systems, curation for reference knowledge.

Phase 4: Unattended agents (month 6 onward). Automate only processes that are documented in the repo, have run supervised at least a dozen times, and have a defined failure path.

Most companies try to run this sequence backwards. They want the agent first because it demos well, and they skip the knowledge work because it is boring. The result is impressive pilots and no compounding.

07

A worked example: 60-person professional services firm

Fictional but representative. Sixty staff, Microsoft 365, a CRM, a finance system, operating knowledge spread across SharePoint and people's heads.

Month 1. Two workspaces: one for delivery teams holding methodology and templates, one for sales holding pricing and proposal guidance. The ops manager and sales lead own one each. Cost: seats for 25 people, roughly a mid four-figure annual spend, plus about two days per owner to assemble and prune the documents.

Months 2 to 3. Five people identified as deliverable producers: two who build client reports, one who does monthly reconciliations, one running proposals, one managing board packs. They get a supervised agentic environment connected to email, files and the CRM. Each documents their best delegated task. Realistic result at this stage: four to eight hours per week back per person, concentrated in the reporting and reconciliation roles.

Months 3 to 5. The methodology and pricing knowledge that proved valuable moves into a version-controlled repo. The operations manager and one technically comfortable senior consultant run it. This is the phase that costs owner-hours rather than licence dollars, roughly two weeks of a senior person's time spread over two months, and it is the phase most firms skip.

Month 6. One agent: the weekly pipeline summary, automated from the now-documented process, owned by the sales lead. Nothing else is automated because nothing else has run supervised long enough.

Total first-year cash cost lands in the low tens of thousands. The larger investment is roughly six weeks of senior people's time across the year, which is also where the return comes from.

08

What it costs

Licence costs are the small number. Standard team seats across the major vendors run roughly $30 to $70 per user per month depending on tier, with premium agentic tiers running several times that for the people who need them. For a mid-market firm this is a rounding error against payroll. Verify current pricing directly with vendors; it changes quarterly.

The real costs are three:

- **Owner time.** Every workspace, repo and agent needs someone maintaining it. Budget a few hours per month per owned asset, and two or more weeks of senior time for the Phase 3 curation work.
- **The canonical-knowledge exercise.** Deciding what is authoritative and deleting the rest. Unavoidable, unglamorous, and the highest-return line item on this page.
- **Opportunity cost of sequence errors.** Buying agents before codifying process, or seats before assigning owners, produces spend with no compounding. This is the most common and most expensive cost in practice.

09

What can go wrong

Plain-English risk, because a connected AI is a different proposition from a chatbot:

- **Access inheritance.** A connected agent sees everything the person connecting it can see. If your file permissions are a mess, the agent inherits the mess. Connector approval is a security decision, not an IT formality, and it is worth an hour of deliberate thought per connector.
- **Confident wrongness.** AI answers fluently from stale or wrong documents. This is why unowned knowledge is worse than no knowledge, and why every workspace needs a named owner.
- **Silent drift.** Unattended agents fail quietly. A weekly report agent producing subtly wrong numbers for a month is worse than one that crashes. Hence the rules: dozen supervised runs first, defined failure path, named process owner, kill switch.
- **Data leaving the building.** On business and enterprise tiers, the major vendors do not train on your data and this is contractual. On consumer tiers the position is weaker. If staff are using personal accounts for company work, that is your actual exposure today, and moving them onto managed seats is partly a containment exercise.

None of these are reasons not to proceed. All of them are reasons the sequence and the ownership model matter more than the tool choice.

10

How you know it is working

Three numbers, reviewed monthly, and none of them is "usage":

1. **Hours removed from named processes.** The only number that matters. If you cannot name the process, the hours are not real.
2. **Questions that stopped.** Are the function heads being interrupted less for answers that live in a workspace? Ask them; they know.
3. **Owned assets versus orphaned ones.** Count the workspaces, connectors and agents with a named, current owner. Anything unowned gets an owner or gets deleted this month.

11

Signs you are doing it wrong

- You are paying for more seats than you have named processes.
- Someone demos an agent and nobody can say who owns the process it automates.
- The same document exists in the AI workspace and on SharePoint and they disagree.
- Your measure of success is logins or messages sent.
- IT owns the content of a sales or ops knowledge base.
- You are evaluating a second platform before anyone can name what the first one saved.

Any two of these together mean stop buying and start owning.

12

Who owns what: the one-page ownership model

Surface	Example products	Where it runs	Day-to-day owner	Governance owner
Shared workspaces	Claude Projects, ChatGPT Projects, Gemini Gems	Vendor cloud, via browser	Function head (content and instructions)	IT or ops (seats, access)
Supervised agentic work	Claude Cowork, ChatGPT Agent, Copilot Cowork	Desktop app or vendor cloud, connected to your systems	Individual producer	IT or ops (connectors, data access policy)
Files and repos	Claude Code, Codex, Gemini CLI on GitHub or GitLab	Locally, against a repo you host	Named owner per document	Technical lead (tooling, repo structure)
Unattended agents	Platform scheduled tasks, Zapier, Make, n8n, Copilot Studio	Vendor cloud, on schedule or trigger	Process owner in the business	IT (platform, credentials, kill switch)
Data layer	Connectors to systems of record plus curated corpus	Existing systems plus repo	Same owners as the source systems and documents	IT or security (connector approval)

Two roles sit above the table. An executive sponsor owns the sequence and the budget, and decides what does not get built. And someone, in a mid-market business often a fractional or part-time senior hire, owns the whole architecture: which surfaces exist, why, and whether they are earning their keep. Without that role the table above degrades into whatever each vendor's account manager proposed last quarter.

13

Where the real value is

Not in the tool choice, and not in the vendor choice. Three places:

1. Codified context. The gap between a mediocre and an excellent AI setup is almost entirely the quality of written context: instructions, examples, current documents, decision criteria. This is durable, portable and vendor-neutral. Everything else depreciates as models improve; context appreciates.

2. Process ownership. Every workspace, environment and agent needs a named owner. Shared context with no owner becomes wrong context within a quarter, and wrong context is worse than none because people trust it.

3. Selective automation. The economics are in removing recurring hours from stable processes. One well-built agent on a weekly reporting task beats twenty licenses used for spellchecking.

And one anti-pattern to name plainly: buying seats is not adoption. Utilisation of a chat tool measures curiosity. Value shows up as hours removed from named processes, and if you cannot name the process, you have bought a subscription, not a capability.

14

Quick reference

- **Questions against stable knowledge** → shared workspace
- **Supervised multi-step tasks** → agentic work environment
- **Durable, auditable, company-owned knowledge and process** → repo plus agentic tooling
- **Unsupervised recurring work** → agents, last, on codified process only
- **Live operational data** → connect to the system of record, never copy
- **Reference knowledge** → curate a small canonical corpus, delete the rest
- **The compounding asset** → written context under version control, owned by someone

Product names and plan features shift constantly across all vendors. Verify specifics against current documentation rather than any static guide, this one included.

About this guide

I'm Paul Salvage, a Fractional Chief AI Officer (CAIO) for mid-market businesses. I've been the CIO of a major telco group, CTO of a Silicon Valley scale-up, CEO of a product development agency, and founder of a wealth management AI startup. I still ship code.

This guide covers the tools layer. The work above it, which products get reimagined, which processes get rebuilt, what the operating model becomes, is specific to each business. That is what I do. The starting point is the 90-day AI Strategy Jump Start: every meaningful system gets an integrate, extend, replace or leave-alone call, and you get a prioritised roadmap with owners and sequence.

delvr.ai/jumpstart · paul@delvr.ai