



AI platform and data foundations

Choosing the data, integration, identity, observability, and vendor foundations for scale.

WORKS WITH THE ASSISTANTS TEAMS ALREADY USE



EDITION 16 / EMBER & AMBER

EXECUTIVE WORKING EDITION - 08 OCT 2026

16

AT A GLANCE

Choosing the data, integration, identity, observability, and vendor foundations for scale.



Most AI pilots work because a few people handled the plumbing by hand. Scale fails when data is scattered, access is unclear, integrations are brittle and costs are invisible. This playbook describes the foundation layers an organization needs: data readiness, identity and access, integration, a model access layer, observability, environments and cost control. It explains build versus buy choices, how to avoid lock-in, and how to sequence investment so each use case makes the next one cheaper.

WHO IT IS FOR

CIOs, CTOs, data and platform leaders, enterprise architects and procurement leads who must decide what to standardize before AI use multiplies.

TIME TO USE

80 minutes to read; two working sessions to produce a foundation roadmap



What you will have at the end

- A clear picture of the foundation layers and which ones your next three use cases actually need
- A data readiness assessment method for the sources that matter
- A build, buy or configure decision frame with lock-in and exit tests
- An observability and cost control design that makes AI spend and quality visible
- A sequenced 90-day plan that avoids a big-bang platform program



Contents

01	Foundations are what turn pilots into a capability	03
02	How much foundation you need, and when	04
03	The foundation layers in plain language	05
04	Who runs the platform and who builds on it	07
05	Build, buy or configure, and how to avoid lock-in	08
06	Seeing what AI is doing and what it costs	10
07	Foundation risks: reliability, security, continuity and compliance	11
08	A sequenced path from pilot to platform	13
09	Common failures and an illustrative scenario	14
10	A first quarter that produces evidence	15
11	Terms used in this publication	16



EXECUTIVE BRIEF

Foundations are what turn pilots into a capability

01



A foundation is the shared set of data, access, integration and monitoring that lets every new AI use case start most of the way there instead of from zero.

The first AI use case in a company often succeeds through heroics: someone exports a spreadsheet, a developer wires an API key into a script, a manager approves access informally. The second and third use cases repeat that effort, each with a different tool, different data copy and different risks. By the fifth, the organization has five fragile systems, unclear spend and no way to answer basic questions about who can see what.



Foundations break that cycle. They are not a giant platform program. They are a short list of shared capabilities, built in the order your use cases need them: reliable access to the right data, a consistent way to authenticate users and agents, a standard way to call models and tools, logging that shows what happened, separate environments for testing and production, and cost visibility. Each capability is boring on its own and valuable in combination.

- Data is usually the binding constraint, not the model. Quality, ownership and access of source data decide what AI can do.
- Standardize the interfaces, not the ideas. Teams should be free to experiment on top of common access, logging and security.
- Model choice will change. Design so that swapping a model or vendor is a configuration and testing task, not a rebuild.
- Spend the first money on the foundation layers that your next three use cases will share.



SCOPE NOTE

This playbook is general guidance. It does not recommend specific vendors or products, and it makes no claims about particular model versions or prices. Check current offerings, terms and your own requirements before selecting any supplier.



WORKING QUESTION

If your best AI pilot had to double its users next quarter, which hand-built step would break first?

WHERE IT FITS

How much foundation you need, and when

02

Build foundations in proportion to the number and criticality of use cases, and not before there is demand.

Over-investment is as real as under-investment. A company with one low-risk use case does not need a multi-year data platform. A company running customer-facing AI, several agents and regulated data does. Think in stages and let evidence of repeated need justify each next layer.

STAGE	TYPICAL SITUATION	FOUNDATION FOCUS
Explore	A few teams trial approved tools	Identity, approved tools, usage rules, a small inventory
Pilot	One or two use cases connect to company data	Clean source data, access control, basic logging, test environment
Scale	Many use cases across functions	Shared model access layer, retrieval services, observability, cost allocation
Embed	AI in core processes and customer products	Service levels, failover, evaluation pipelines, vendor management, audit

Signals you need more foundation

- Teams keep copying the same data into separate tools.
- Nobody can say how much was spent on AI services last month, or by whom.

- Each new project negotiates its own security review from scratch.
- A vendor change would force several teams to rebuild.
- Outputs cannot be traced back to the model, prompt and source version that produced them.



COMMON MISTAKE

Starting with a platform selection exercise before any use case exists. Begin with the data and access needs of two real use cases, and let those requirements drive the choices.

WORKING QUESTION

Which stage best describes your organization today, and what is the cheapest foundation upgrade that would help the next use case?



HOW IT WORKS

The foundation layers in plain language

03



Seven layers cover most needs: data, identity, integration, model access, retrieval, observability and operations.

LAYER	WHAT IT DOES	FAILURE IF MISSING
Data foundation	Makes trusted data findable, owned, documented and accessible	Wrong or stale answers; endless data wrangling
Identity and access	Ensures users and agents act with the right permissions	Data leaks; no audit trail
Integration	Connects AI to systems of record through stable interfaces	Brittle scripts; manual copy and paste



LAYER	WHAT IT DOES	FAILURE IF MISSING
Model access layer	One governed route to models with keys, routing and limits	Scattered keys; lock-in; surprise bills
Retrieval and knowledge	Indexes approved content so answers can be grounded	Unverifiable answers; duplicate indexes
Observability	Logs, traces, quality and cost metrics	Blind operation; slow incident response
Environments and release	Separate dev, test, production with change control	Untested changes reaching users

Data readiness in practice

Assess each source a use case needs against a short list of questions. Who owns it and can answer for it? Is it current, and is the update path clear? Are there duplicates or conflicting versions? Is it labelled by sensitivity and permission? Can it be accessed through an interface, rather than by exporting files? Is its structure documented so a person new to it could use it? Sources that fail these tests should be fixed first or excluded, because AI makes bad data look authoritative.

01 List the sources

For each use case, list documents, tables and systems it will read or write, and the people who own them.

02 Score readiness

Rate each source on ownership, currency, quality, permissions and accessibility. A simple red, amber, green scale works.

03 Fix or exclude

Assign an owner to repair red sources, or remove them from scope until they are fixed.

04 Publish a catalog entry

Record the source, owner, sensitivity and access route so other teams can reuse it with confidence.

WORKING QUESTION

For your top use case, which source would you least trust to be correct today, and who owns it?

OPERATING MODEL

Who runs the platform and who builds on it

04



A small central platform team plus federated use-case teams works better than either full centralization or free-for-all.

A common effective pattern is a small platform group that owns shared services and standards, with business-aligned teams that build use cases on top. The platform group acts like an internal product team: it has users, a roadmap and service levels. It should say yes by default within guardrails, and publish templates so that a team can start a compliant project in days, not months.

ROLE	OWNS	DOES NOT OWN
Platform team	Model access layer, identity integration, logging, shared retrieval, environments, cost reporting	Business logic of individual use cases
Data owners	Quality, currency, access rules and documentation of their sources	Platform tooling
Use-case teams	Application logic, prompts, evaluation, user adoption	Core security and logging services
Security and privacy	Standards, reviews, testing of the shared layers	Day-to-day operation
Finance partner	Budget, allocation of AI cost to business units	Technical design



- Publish a service catalog: what the platform offers, how to request it and what it costs internally.
- Use a standard intake for new use cases, so review is quick and reuse is encouraged.
- Hold a monthly forum where use-case teams share what they built so others can reuse it.
- Treat the platform roadmap as a backlog driven by use-case demand, not by technology interest.




AVOID

A platform team that builds for two years before serving a user. Ship a minimal shared service within the first quarter and add to it as real projects need more.

WORKING QUESTION

Who in your organization could today give a new AI project a compliant starting kit within a week?


BUILD OR BUY

Build, buy or configure, and how to avoid lock-in

05



Buy what is commodity, build what differentiates you, and keep your data and evaluation assets portable.

The model itself is rarely worth building for a typical business. The choices that matter are above and around it: which data you expose, how you wire it into workflows, how you evaluate quality and how you control cost. Use the build, buy or configure frame for each layer. Buy or configure where the capability is common and well served. Build where the logic encodes your own process and advantage.

QUESTION	POINTS TO BUY OR CONFIGURE	POINTS TO BUILD
Is this a commodity capability?	Yes: login, logging, standard connectors	No: unique process logic
Does it hold our competitive knowledge?	No	Yes: keep control of prompts, evaluations, data
How fast do we need it?	Fast: use existing services	Not urgent: invest for fit
Can we maintain it?	No spare engineering capacity	Yes: stable team with skills
Is exit easy?	Standard formats, export rights	You own the code and data



Lock-in tests

Ask before committing

- Can we export our prompts, indexes, logs, test sets and fine-tuning data in open format?
- Could we switch to another model provider by changing configuration and rerunning our evaluation set?
- Do contract terms cover price changes, model retirement and notice periods?
- Where is data stored and processed, and which sub-processors are involved?
- What are the service levels, rate limits and support routes?
- Are our data and outputs excluded from vendor training unless we opt in?
- What happens to our data and access on termination?

Put a thin abstraction between your applications and any model provider. It need not be elaborate: a single internal interface for requests, a place to store keys, and consistent logging. This is the practical meaning of avoiding lock-in. Keep an evaluation set so that you can test a replacement quickly, since the real cost of switching is the uncertainty about whether quality holds, not the code.

WORKING QUESTION

If your main model provider doubled its price or retired a model with ninety days notice, how long would it take you to move?

OBSERVABILITY AND COST

Seeing what AI is doing and what it costs

06

What you cannot trace you cannot improve, secure or budget.

AI workloads have a cost model unlike most software. Spend varies with usage, length of prompts and outputs, the model chosen and the number of tool calls in an agent loop. Without per-use-case tracking, finance sees one rising bill with no owner. Observability should therefore cover three things together: behavior, quality and cost.

SIGNAL	WHY IT MATTERS	HOW TO CAPTURE
Request traces	Reconstruct what happened in a flow	Inputs, retrieved sources, tool calls, outputs, versions
Latency and errors	User experience and reliability	Time per step, failure rates, retries
Quality indicators	Detect drift and regressions	Sampled scoring, user flags, override rates
Token and call usage	Understand cost drivers	Usage per use case, user group and model
Cost per outcome	Tie spend to value	Spend divided by tasks completed or resolved
Security events	Detect abuse and leakage	Anomalous volume, blocked requests, permission denials

Cost control levers

- Tag every request with a use case and cost center from day one so allocation is possible.
- Set budgets and alerts per use case, and rate limits per user and per agent.
- Match model size to task: use a smaller, cheaper model for routine classification and a larger one for hard reasoning, validated by your evaluation set.
- Cache repeated requests where answers are stable, and trim prompts that carry unneeded context.
- Review top spenders monthly and ask what outcome each bought.

LOG WITH CARE

Traces can contain personal or confidential data. Apply the same retention, access and masking rules to logs as to the source data, and decide who may read them.



WORKING QUESTION

Can you say today what your three AI use cases each cost last month, and what each delivered?

RISKS AND CONTROLS

Foundation risks: reliability, security, continuity and compliance

07



Shared services concentrate both value and risk, so design for failure and for scrutiny from the start.



When many use cases depend on one access layer or one index, an outage or misconfiguration affects all of them. Treat the foundation as production infrastructure. That means defined service levels, on-call ownership, tested recovery and a plan for what each use case does when a model provider is unavailable. Some tasks can fall back to a second provider, a cached answer or a manual process. Decide which in advance, and rehearse it.

RISK	CONTROL	EVIDENCE
Provider outage or rate limiting	Fallback model or manual procedure, retry and queue design	Failover test result
Silent model change by a vendor	Version pinning where offered, regression run on every change	Test results by version
Overbroad data access	Per-user permission checks at retrieval, least-privilege service accounts	Permission scan and access logs



RISK	CONTROL	EVIDENCE
Sensitive data in logs and indexes	Masking, retention limits, restricted log access	Retention configuration
Stale or conflicting content	Source owners, update schedules, version flags	Freshness reports
Data residency or contract breach	Region settings, approved suppliers, list, contract review	Configuration and contract file

- Keep secrets such as API keys in a managed vault, rotate them and never place them in prompts or code repositories.
- Make every index traceable to its source documents and owners so removal requests and corrections can be honored.
- Document data flows for each use case; privacy and security reviews need them, and so do customers who ask.
- Check applicable data protection, sector and AI rules for your jurisdictions, and involve legal early for personal or regulated data.

CONTINUITY TEST

Pick your most important AI use case and ask what happens if its model provider is unreachable for a day. If the answer is that nobody knows, you have found your first resilience task.

WORKING QUESTION

Which shared component, if it failed tomorrow morning, would stop the most AI use at once?

IMPLEMENTATION GUIDE

A sequenced path from pilot to platform

08



Build the foundation in thin slices driven by use cases, and prove each slice in production before adding the next.

01 Pick two anchor use cases

Choose use cases with different needs, such as a document assistant and a workflow agent, so shared requirements are real rather than guessed.



02 Map data and access needs

Score the sources, assign owners and define permissions. Fix the reds that block the anchors.

03 Stand up the minimum shared services

Provide single sign-on, a model access layer with keys and logging, separate test and production environments and cost tagging.

04 Add retrieval and integration

Create one governed index for approved content and stable API connections to the systems of record the anchors need.



05 Instrument and evaluate

Turn on tracing, quality sampling and cost dashboards. Build the test sets for each anchor.

06 Productize and reuse

Package what worked as templates and documentation, then onboard the next use case through them.

Foundation readiness for a new use case

- Data sources named, owned, rated and permissioned
- Users and agents authenticate through the standard identity route
- Model calls go through the governed access layer
- Logs, cost tags and quality sampling are enabled



- Test and production environments are separate, with change control
- Exit plan documented for each external dependency

Resist the urge to unify every dataset first. A data lake project that tries to clean everything before delivering any value usually stalls. Cleaning the specific sources for two to three use cases gives results within months and builds a catalog of reusable assets. Over time, repeated needs show which data deserves a durable shared product.

WORKING QUESTION

Which two use cases would make the best anchors for your foundation, and what do they share?

PITFALLS

Common failures and an illustrative scenario

09

Foundation programs fail by being too early, too big or too disconnected from real use cases.

- A platform purchase made before a use case defines requirements.
- Cleaning all data instead of the data that matters.
- Embedding one vendor so deeply that exit becomes impractical.
- No cost tagging, so finance cannot attribute spend.
- Production and test sharing one environment and one set of keys.
- Treating logs as a free resource and creating a new store of sensitive data.

ILLUSTRATIVE SCENARIO (HYPOTHETICAL)

A mid-sized insurer launches three AI pilots in different departments. Each uses a different tool, copies policy documents separately and stores its own keys. When a model provider changes its terms, all three need rework, and the monthly bill cannot be split by team. The insurer consolidates on one access layer, one governed index and tagged costs, then rebuilds the pilots on it.



The takeaway is that the third pilot should cost less than the first. If it does not, you are building projects rather than a capability. Track that as a leading measure: time and cost to launch each new use case. A falling number shows the foundation is paying back, while a flat or rising number says the shared work is not being reused.

WORKING QUESTION

Is the second AI project in your organization measurably cheaper and faster to launch than the first?



30 / 60 / 90

A first quarter that produces evidence

10



The aim of the first quarter is a working shared core and two reference use cases, not a finished platform.

First 30 days: ground it

- Choose two anchor use cases and list their data sources
- Score source readiness and assign owners to fix the reds
- Inventory current AI tools, keys and spend
- Agree the build, buy or configure frame for each layer



Days 31 to 60: test it

- Stand up single sign-on integration and a governed model access layer
- Separate test and production environments
- Tag requests by use case and cost center
- Create the first governed index of approved content

Days 61 to 90: decide

- Turn on tracing, quality sampling and cost dashboards
- Launch both anchors on the shared services
- Publish templates and a service catalog for the next teams
- Measure time and cost to launch use case three against use case one

GLOSSARY

Terms used in this publication

11

TERM	MEANING
Data foundation	Trusted, owned, documented and accessible data that AI use cases can rely on.
Data catalog	A searchable record of data sources with owners, sensitivity and access routes.
Model access layer	A governed internal route for calling models, handling keys, routing, limits and logging.
Retrieval index	A searchable store of approved content that lets an assistant find relevant passages.
API	A defined interface that lets one system request data or actions from another.
Observability	Logs, traces and metrics that show how a system behaves, performs and costs.
Lock-in	A dependency on one supplier that makes switching slow, costly or risky.

TERM	MEANING
Token	A unit of text processed by a model; usage and cost are often measured in tokens.
Environment	A separate setup, such as development, test or production, with its own data and controls.
Cost allocation	Assigning shared spend to the teams or use cases that caused it.
Single sign-on	A central login that lets users and agents authenticate once with managed permissions.
Service level	A stated commitment on availability, response time or support.

A note on use

This playbook is general guidance and does not endorse any vendor. Verify current products, terms, pricing and legal requirements before purchase.

TAKE THE NEXT STEP

Take the free Halden assessment at halden.company/assessments, or talk to us at halden.company/contact about turning this into a plan for your organization.