

AssetOps

Asset + Field Operations

Build field-service, facilities, fleet, property, equipment, inspection, or maintenance software that connects customers and locations to assets, inspections, work orders, technicians, parts, photos, documents, invoices, and complete service history.

ASSETOPS - WALKTHROUGH SCENARIO

A recurring temperature anomaly across inspections is connected to the asset history, customer site, prior work orders, parts, warranty, and technician schedule to prepare preventive service.

[View evidence](#)[Start workflow](#)

1

GOVERNED PRODUCT MODEL

6

CORE CAPABILITIES

5

WALKTHROUGH STAGES

4

SEPARATED DATA / CODE PLANES

What AssetOps is

AssetOps packages reusable BuildWithHQ primitives into a focused asset + field operations product experience.

01. Product layer

A branded application experience shaped around a specific operating job.

02. Governed intelligence

AI can analyze, retrieve, recommend, and prepare actions inside the same permission envelope as the application.

03. Operational actions

Workflows, approvals, APIs, webhooks, MCP tools, and custom services move from insight to controlled execution.

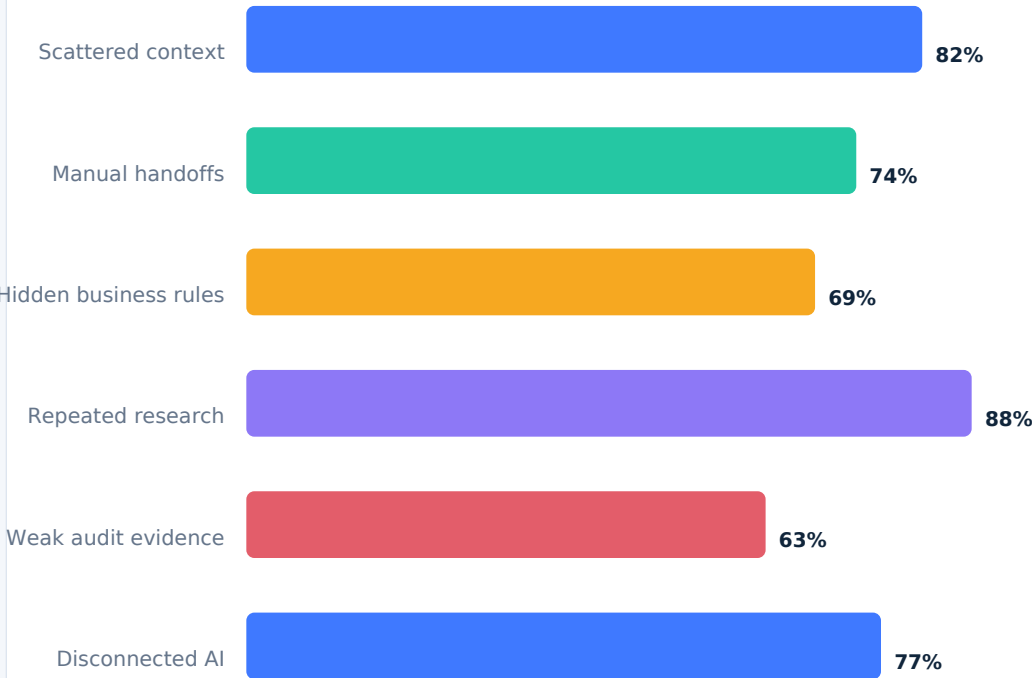
04. Platform leverage

Dedicated data planes, audit evidence, exports, and optional container services stay reusable under the product.

The operational problem

AssetOps is designed for work that becomes expensive when context, decisions, and actions are fragmented across tools and people.

Illustrative sources of operating friction



What changes

Instead of treating AI, records, workflows, permissions, and integrations as separate projects, the product keeps them inside one governed operating model.

Design goal

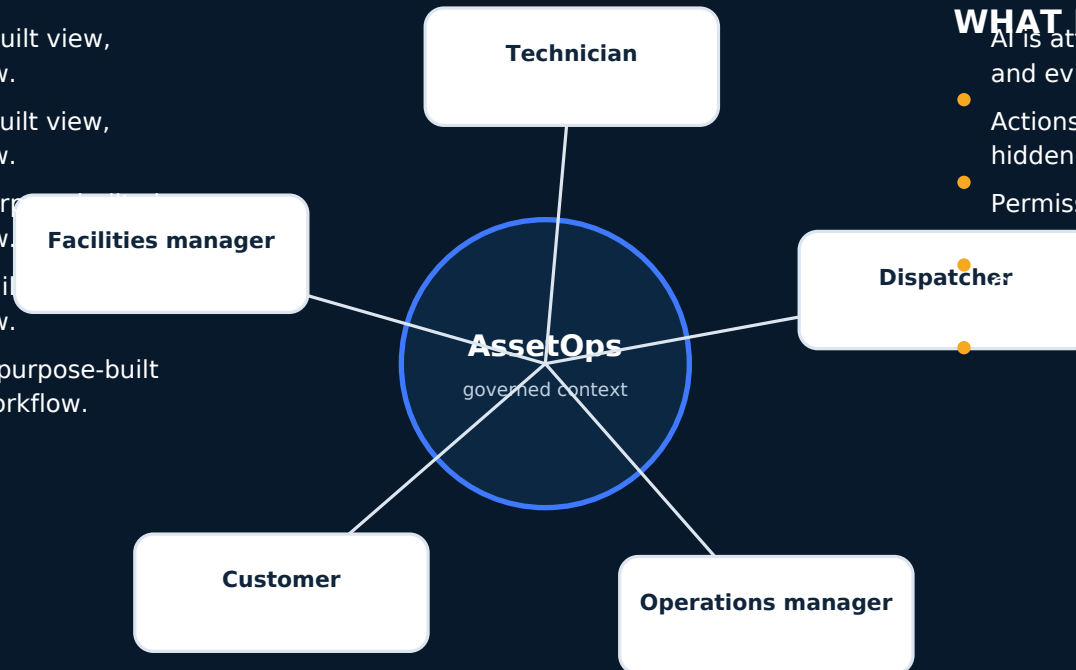
Reduce the distance between context, decision, and controlled action without flattening the business rules that make the workflow safe.

A complete operating product

Connect every asset to the work required to keep it running.

WHO IT SERVES

- Dispatcher gets a purpose-built view, authority level, and workflow.
- Technician gets a purpose-built view, authority level, and workflow.
- Facilities manager gets a purpose-built view, authority level, and workflow.
- Customer gets a purpose-built view, authority level, and workflow.
- Operations manager gets a purpose-built view, authority level, and workflow.
-



WHAT MAKES IT DIFFERENT

- AI is attached to governed records and evidence.
- Actions are explicit contracts, not hidden autonomous behavior.
- Permissions remain server-side screens, APIs, AI and exports.
- Code is optional - not the only match.

The six building blocks of AssetOps

Build field-service, facilities, fleet, property, equipment, inspection, or maintenance software that connects customers and locations to assets, inspections, work orders, technicians, parts, photos, documents, invoices, and complete service history.

01 Assets + location hierarchy

Model customers, properties, facilities, rooms, vehicles, equipment, components, serial numbers, warranties, and any hierarchy your operation needs.

02 Recursive operational relationships

Relate an asset to the site, customer, inspections, work orders, technicians, parts, photos, documents, invoices, and service history without trapping context in one module.

03 Inspections + field forms

Build mobile-friendly inspection checklists, readings, signatures, photos, issue capture, pass/fail logic, and required corrective actions.

04 Work orders + scheduling

Generate work from inspections or alerts, assign technicians, track status, schedule visits, record labor and parts, and close the loop back to the asset.

05 Complete service evidence

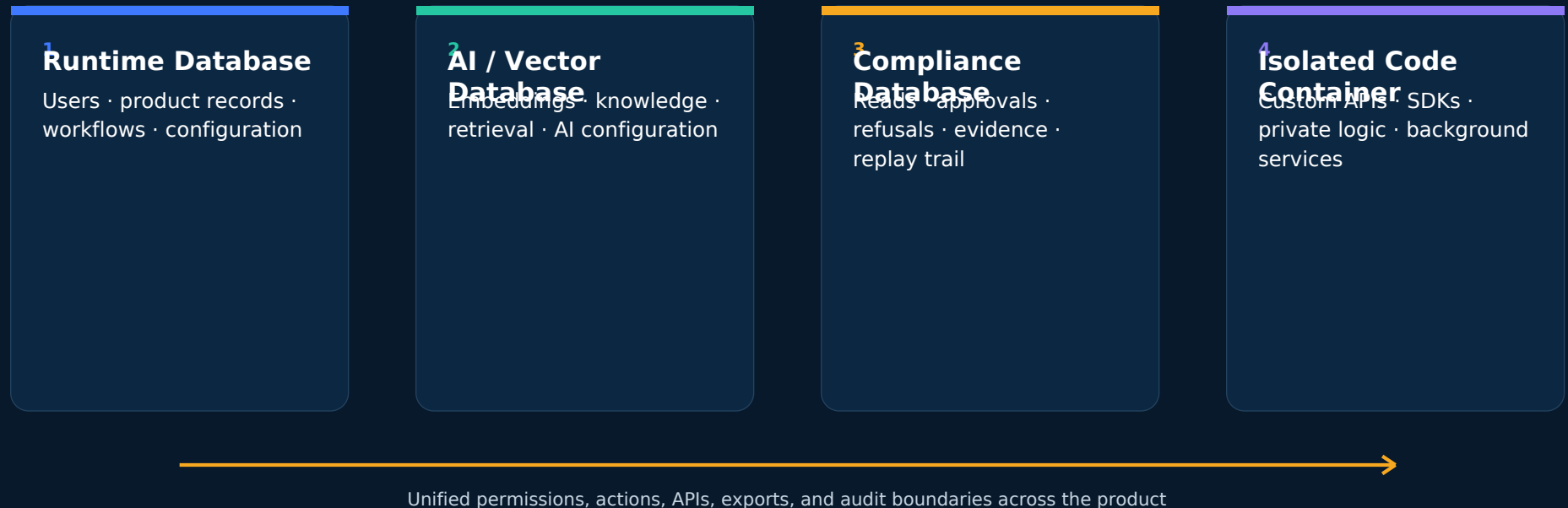
Preserve manuals, images, invoices, warranties, technician notes, measurements, historical failures, and customer communications alongside the operational record.

06 AI maintenance intelligence

Use permitted history and inspection data to summarize asset condition, detect recurring issues, prepare work orders, and surface maintenance opportunities for human review.

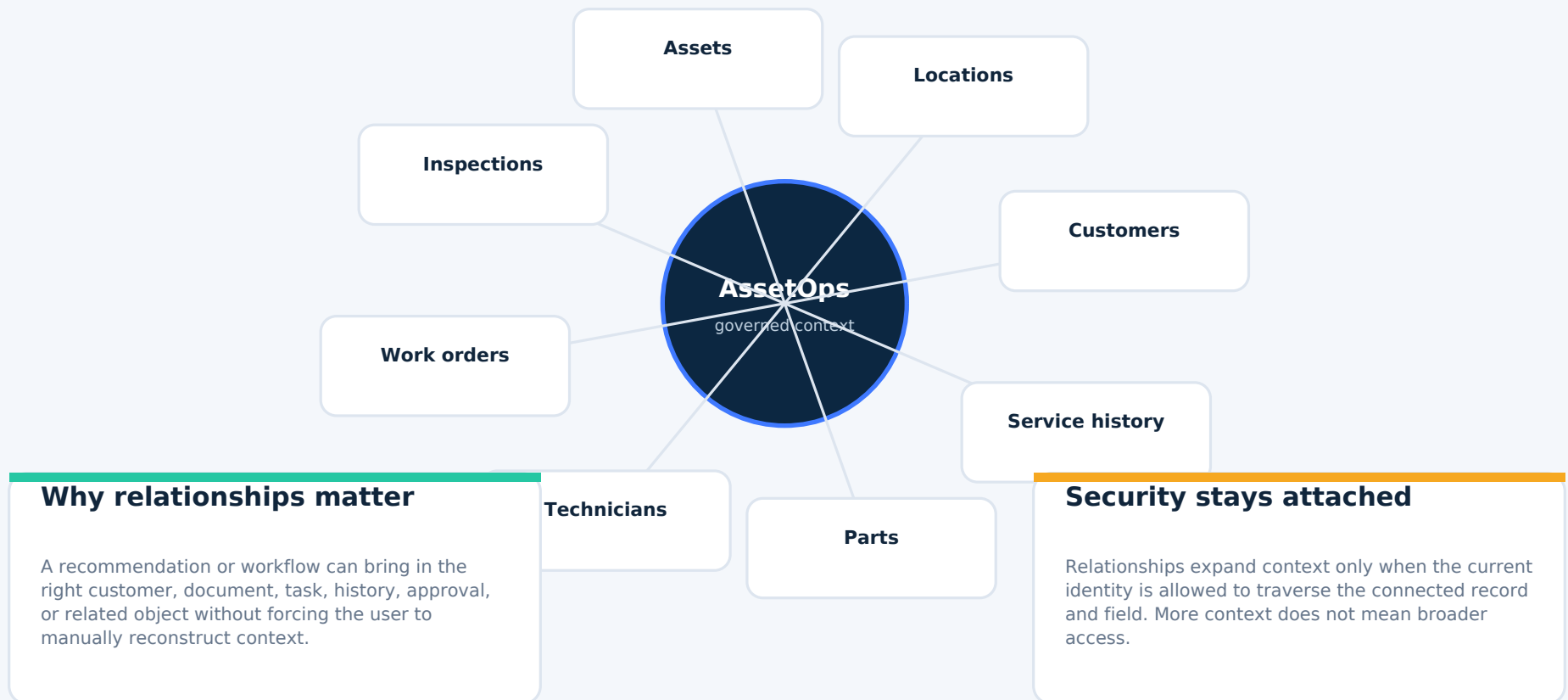
Separated by responsibility. One product to the user.

BuildWithHQ keeps runtime records, AI context, compliance evidence, and custom execution separate while the application presents one coherent experience.



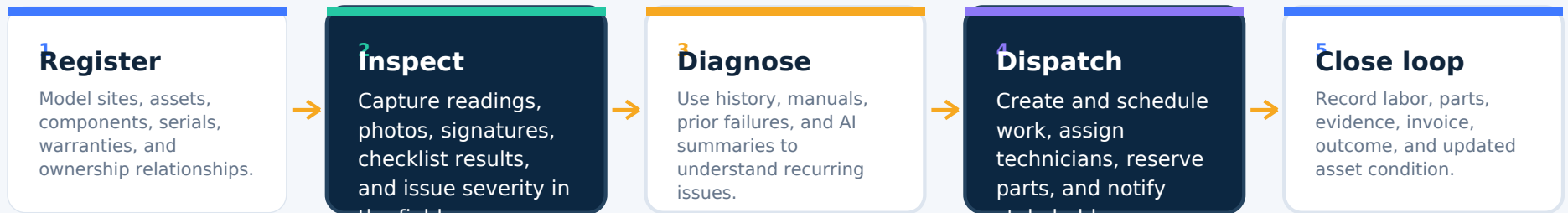
Everything important becomes connected context

The product data model is not a set of isolated modules. Records can relate recursively so users and AI can traverse the real operating context.



From input to governed outcome

A recurring temperature anomaly across inspections is connected to the asset history, customer site, prior work orders, parts, warranty, and technician schedule to prepare preventive service.



The five stages are illustrative - each BuildWithHQ implementation can add branches, approvals, retries, SLAs, external systems, and custom services.

1. Register

Model sites, assets, components, serials, warranties, and ownership relationships.

INPUT

What enters this stage

Identity, permitted records, related context, workflow state, and any source-specific inputs required for register.

EVIDENCE

Evidence produced

Store the records consulted, source versions, relevant model output, workflow transition, approvals/denials, and mutation result so the operation can be explained later.

ENGINE

What BuildWithHQ does

Model sites, assets, components, serials, warranties, and ownership relationships. The runtime enforces tenant, role, record, field, rate-limit, and action boundaries before the stage can progress.

EXPERIENCE

What the user sees

A focused AssetOps screen should expose the result, why it happened, what remains unresolved, and the next safe action - without forcing the user to inspect raw infrastructure.

2. Inspect

Capture readings, photos, signatures, checklist results, and issue severity in the field.

INPUT

What enters this stage

Identity, permitted records, related context, workflow state, and any source-specific inputs required for inspect.

EVIDENCE

Evidence produced

Store the records consulted, source versions, relevant model output, workflow transition, approvals/denials, and mutation result so the operation can be explained later.

ENGINE

What BuildWithHQ does

Capture readings, photos, signatures, checklist results, and issue severity in the field. The runtime enforces tenant, role, record, field, rate-limit, and action boundaries before the stage can progress.

EXPERIENCE

What the user sees

A focused AssetOps screen should expose the result, why it happened, what remains unresolved, and the next safe action - without forcing the user to inspect raw infrastructure.

3. Diagnose

Use history, manuals, prior failures, and AI summaries to understand recurring issues.

INPUT

What enters this stage

Identity, permitted records, related context, workflow state, and any source-specific inputs required for diagnose.

EVIDENCE

Evidence produced

Store the records consulted, source versions, relevant model output, workflow transition, approvals/denials, and mutation result so the operation can be explained later.

ENGINE

What BuildWithHQ does

Use history, manuals, prior failures, and AI summaries to understand recurring issues. The runtime enforces tenant, role, record, field, rate-limit, and action boundaries before the stage can progress.

EXPERIENCE

What the user sees

A focused AssetOps screen should expose the result, why it happened, what remains unresolved, and the next safe action - without forcing the user to inspect raw infrastructure.

4. Dispatch

Create and schedule work, assign technicians, reserve parts, and notify stakeholders.

INPUT

What enters this stage

Identity, permitted records, related context, workflow state, and any source-specific inputs required for dispatch.

EVIDENCE

Evidence produced

Store the records consulted, source versions, relevant model output, workflow transition, approvals/denials, and mutation result so the operation can be explained later.

ENGINE

What BuildWithHQ does

Create and schedule work, assign technicians, reserve parts, and notify stakeholders. The runtime enforces tenant, role, record, field, rate-limit, and action boundaries before the stage can progress.

EXPERIENCE

What the user sees

A focused AssetOps screen should expose the result, why it happened, what remains unresolved, and the next safe action - without forcing the user to inspect raw infrastructure.

5. Close loop

Record labor, parts, evidence, invoice, outcome, and updated asset condition.

INPUT

What enters this stage

Identity, permitted records, related context, workflow state, and any source-specific inputs required for close loop.

EVIDENCE

Evidence produced

Store the records consulted, source versions, relevant model output, workflow transition, approvals/denials, and mutation result so the operation can be explained later.

ENGINE

What BuildWithHQ does

Record labor, parts, evidence, invoice, outcome, and updated asset condition. The runtime enforces tenant, role, record, field, rate-limit, and action boundaries before the stage can progress.

EXPERIENCE

What the user sees

A focused AssetOps screen should expose the result, why it happened, what remains unresolved, and the next safe action - without forcing the user to inspect raw infrastructure.

Permissions follow the user into AI, APIs and actions

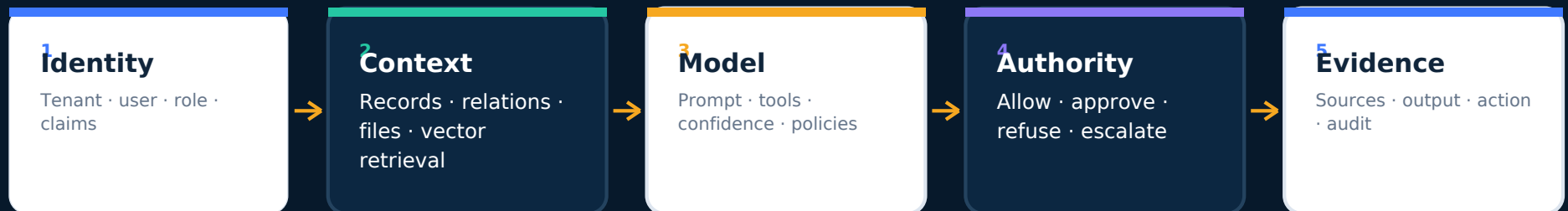
One authorization model should determine what each role can read, retrieve, export, approve, or mutate.

Illustrative authority matrix - 1 restricted to 5 broad

	Read	AI context	Recommend	Approve	Execute
Dispatcher	5	4	2	5	1
Technician	5	4	4	5	4
Facilities manager	5	4	4	2	4
Customer	5	4	4	2	1
Operations manager	5	3	2	5	1

AI is a participant in the workflow - not a bypass around

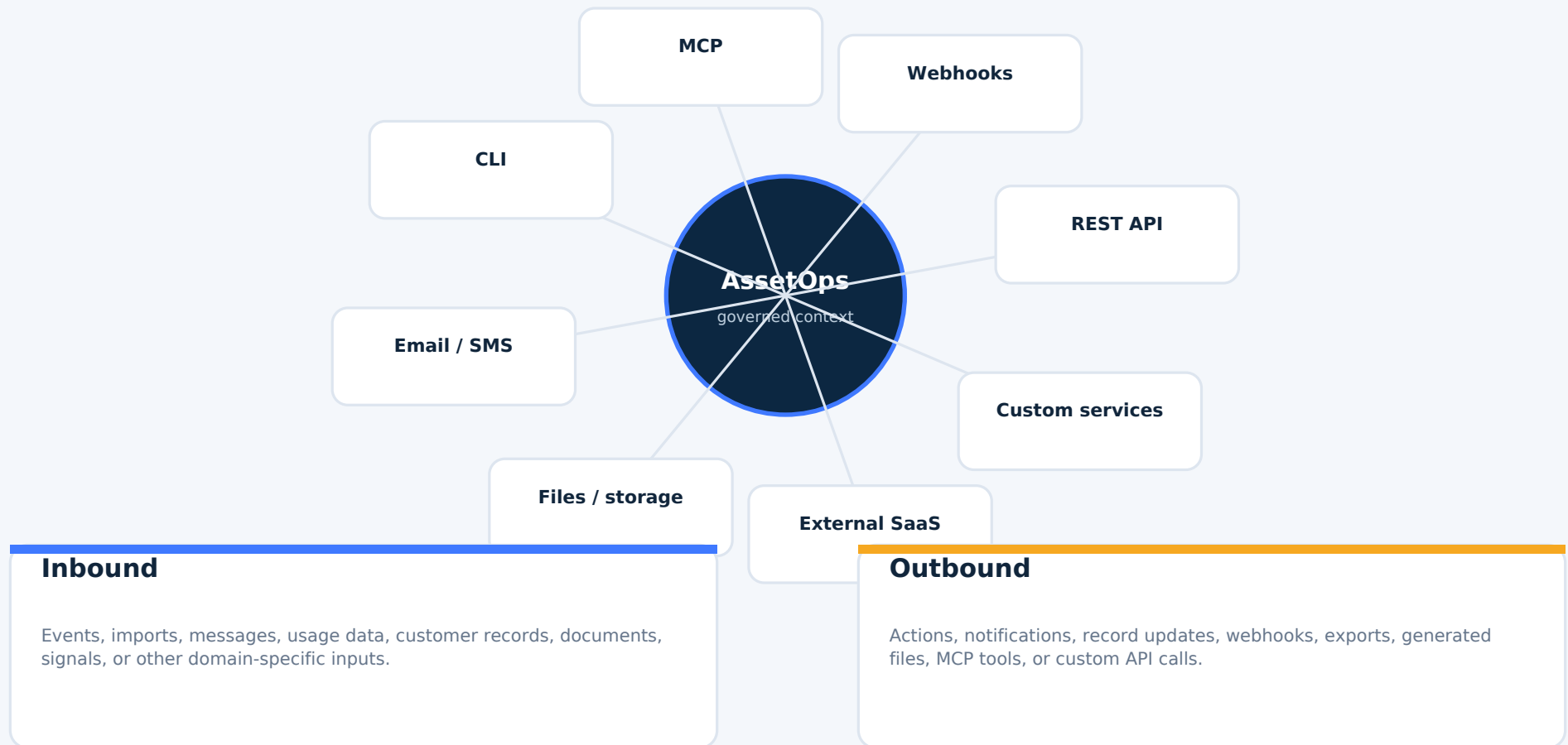
The AI layer can retrieve, synthesize, rank, draft, and recommend. Server-side policies still decide what data and actions are actually available.



This pattern lets the same intelligence layer support employees, end customers, scheduled agents, MCP clients, and API-driven experiences.

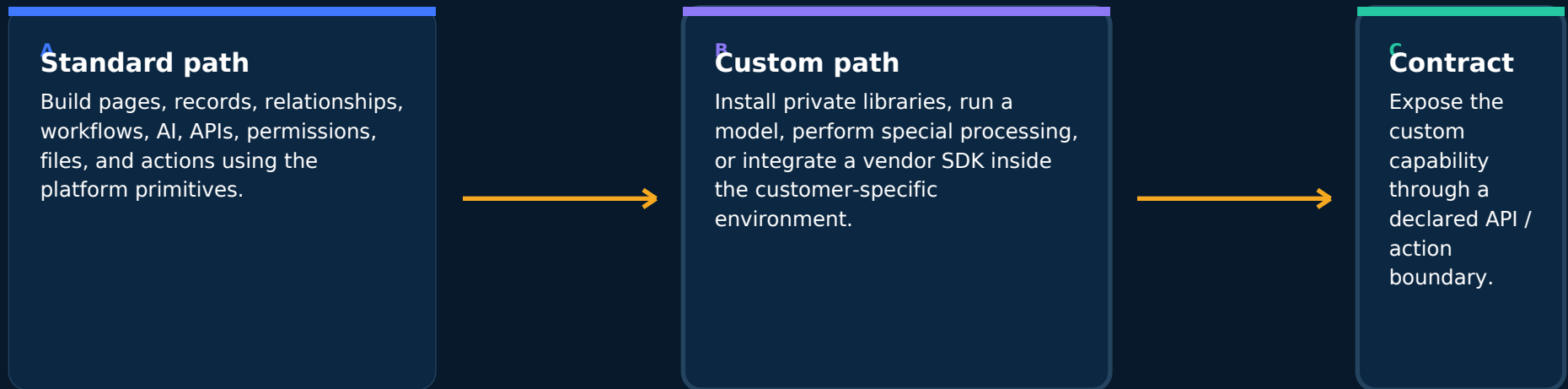
Connect to the systems already around the business

API, webhooks, MCP, CLI, file import/export, and optional private services let the product fit into an existing stack instead of becoming another isolated island.



Optional isolated container when the product needs

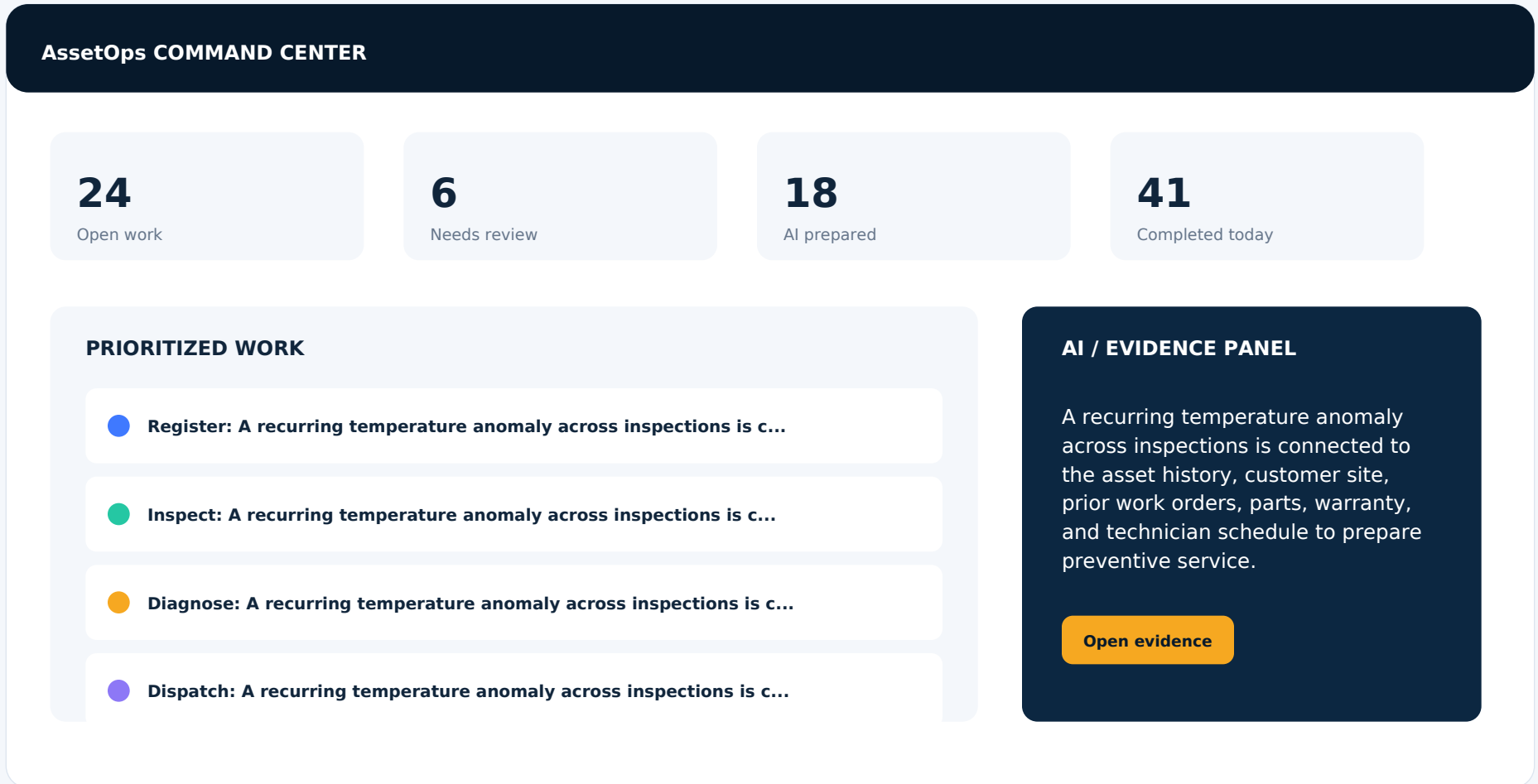
Most of the application can remain schema-, workflow-, and API-driven. When a developer needs a special SDK, proprietary algorithm, or long-running service, the isolated container provides a controlled extension point.



The central execution layer can call the declared service while the custom code remains isolated from the core platform and from other customers.

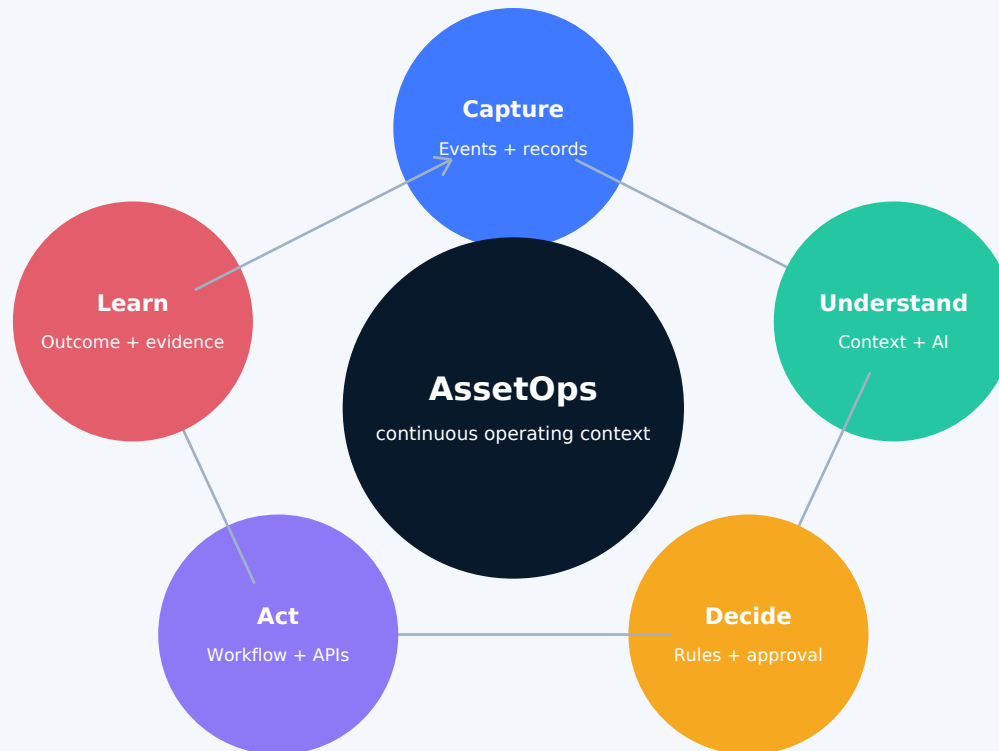
What a AssetOps operating screen can look like

The point of the blueprint is to turn deep infrastructure into a focused daily workflow.



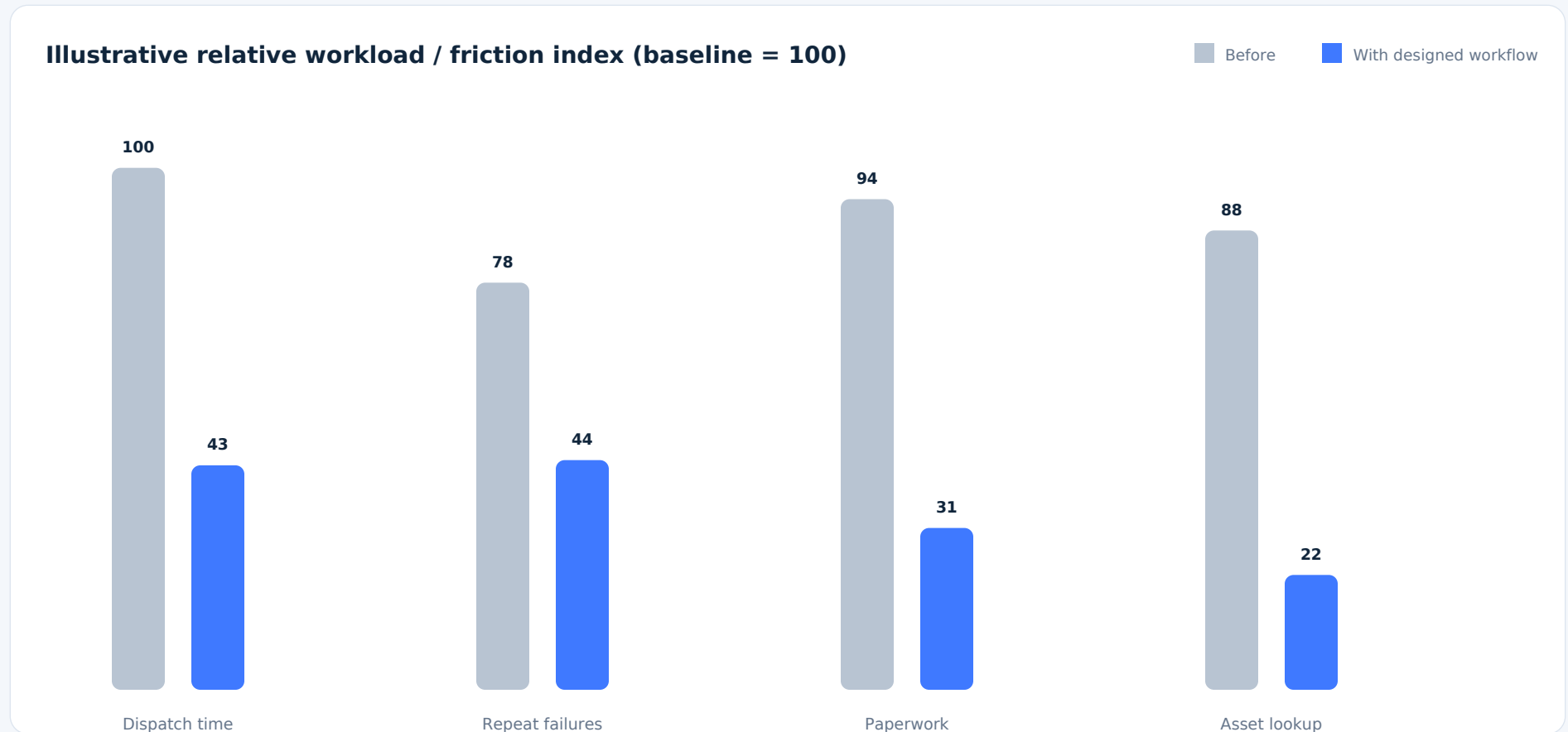
Every completed cycle can improve the next one

The product gets stronger when outcomes, approvals, exceptions, source quality, user corrections, and workflow performance are fed back into the operating model.



Where operating leverage can show up

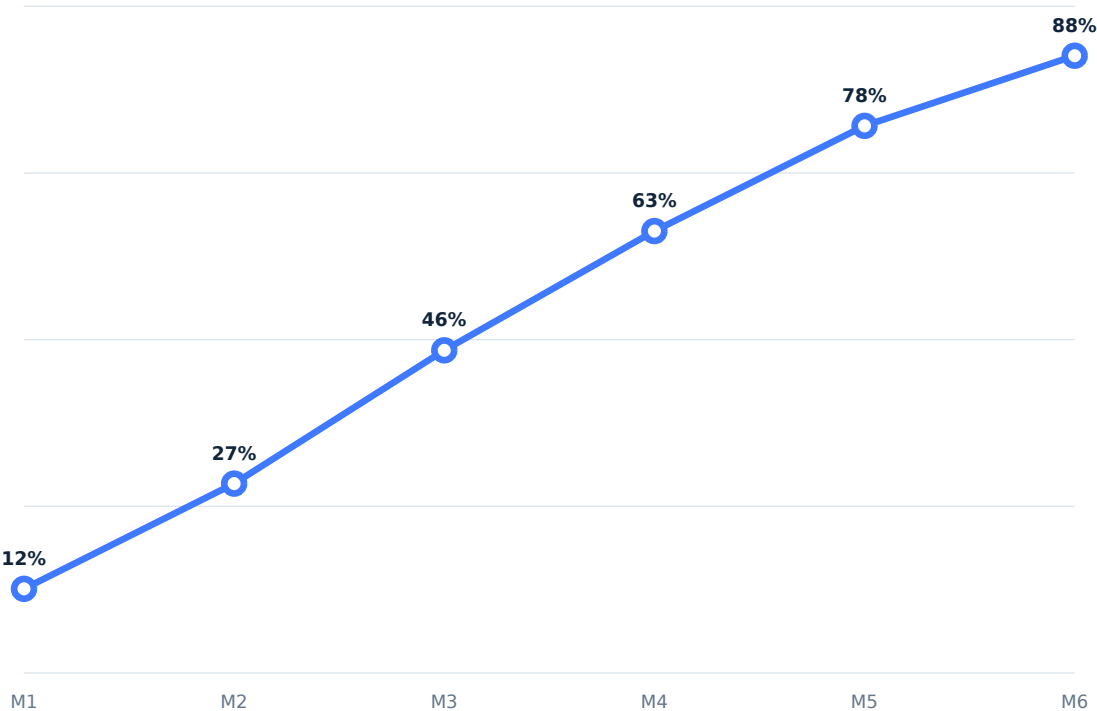
The numbers below are not measured BuildWithHQ customer results. They are an illustrative model showing the kinds of friction the product is designed to reduce.



Value compounds as more work enters the governed

A staged deployment lets the team prove retrieval, routing, approvals, and actions before widening automation authority.

Illustrative percent of target workflow running through the product



Phase 1 - Assist

Read-only context, search, summaries, classification, and drafts. Human users make every consequential decision.

Phase 2 - Governed action

Add explicit tools, approval thresholds, scheduled runs, and carefully bounded auto-actions where the evidence is strong.

Six ways to package AssetOps

One BuildWithHQ blueprint can be narrowed to a vertical, customer segment, operating team, or recurring workflow and then branded as its own SaaS product.

01 HVAC service

Use the same core AssetOps architecture, then customize terminology, records, permissions, integrations, AI behavior, and workflows for this market.

02 Fleet maintenance

Use the same core AssetOps architecture, then customize terminology, records, permissions, integrations, AI behavior, and workflows for this market.

03 Property inspections

Use the same core AssetOps architecture, then customize terminology, records, permissions, integrations, AI behavior, and workflows for this market.

04 Equipment service

Use the same core AssetOps architecture, then customize terminology, records, permissions, integrations, AI behavior, and workflows for this market.

05 Facilities operations

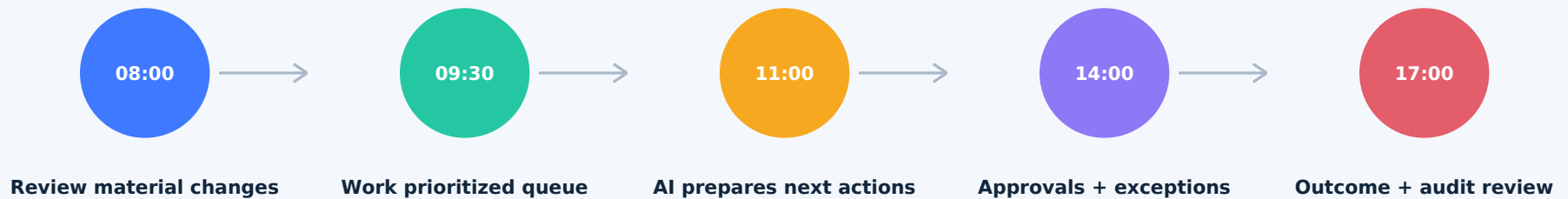
Use the same core AssetOps architecture, then customize terminology, records, permissions, integrations, AI behavior, and workflows for this market.

06 Field compliance

Use the same core AssetOps architecture, then customize terminology, records, permissions, integrations, AI behavior, and workflows for this market.

A role-aware experience from morning to close

Different users can work on the same underlying records while seeing different queues, context, evidence, and action rights.



The important part

The application is not forcing every role into the same dashboard. The runtime can expose the same record with different fields, related context, actions, AI tools, and approval rights based on who is looking at it.

Ways a builder could monetize the product

These are product packaging ideas, not fixed BuildWithHQ pricing. A builder can mix software subscriptions, usage, premium modules, infrastructure, and services.

Per-asset tier

Technician seats

Inspection packs

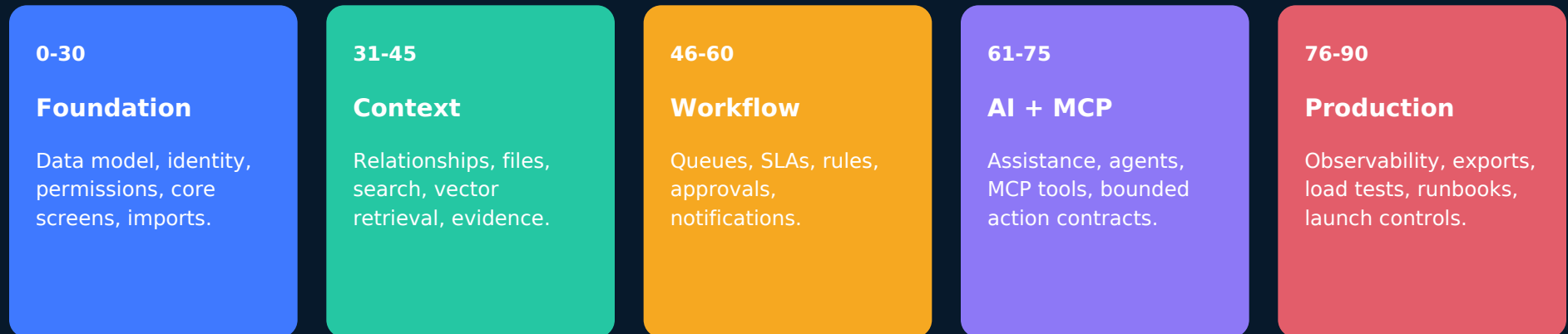
AI maintenance intelligence

Vertical template packages

Illustrative packaging ladder: start simple, then monetize the capabilities that scale with customer value or infrastructure consumption.

A practical 90-day build sequence

The exact timeline depends on scope, but the safest pattern is to establish data + identity first, then intelligence, then actions, then expansion.

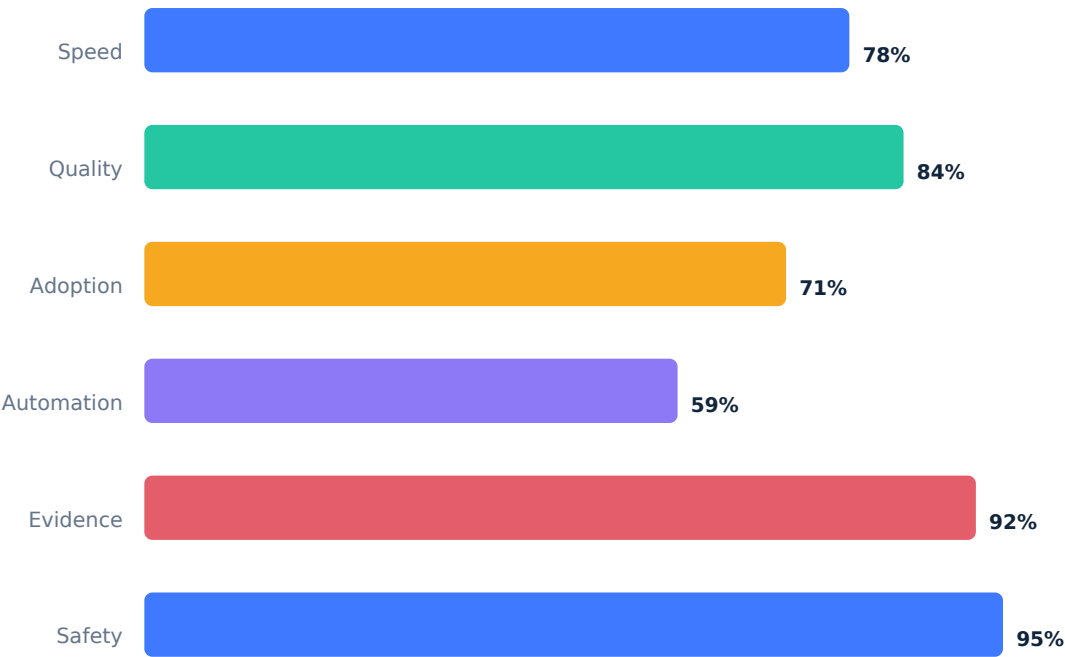


Launch authority should expand only after the evidence trail, failure handling, permission checks, and rollback path are tested.

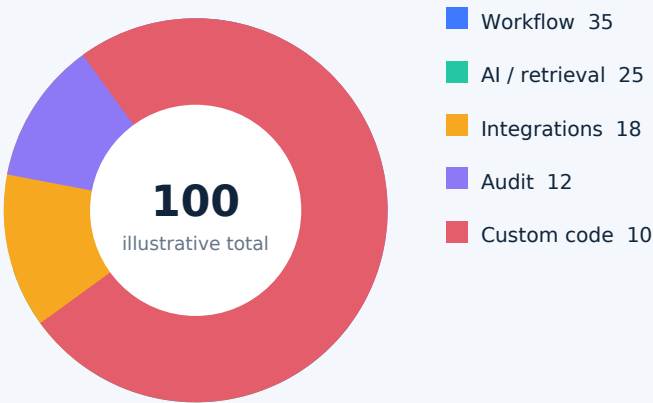
What to measure after launch

The product should be evaluated on operating outcomes, not just model usage. Track speed, quality, safety, adoption, and economic leverage together.

Illustrative maturity scorecard



Illustrative implementation effort mix



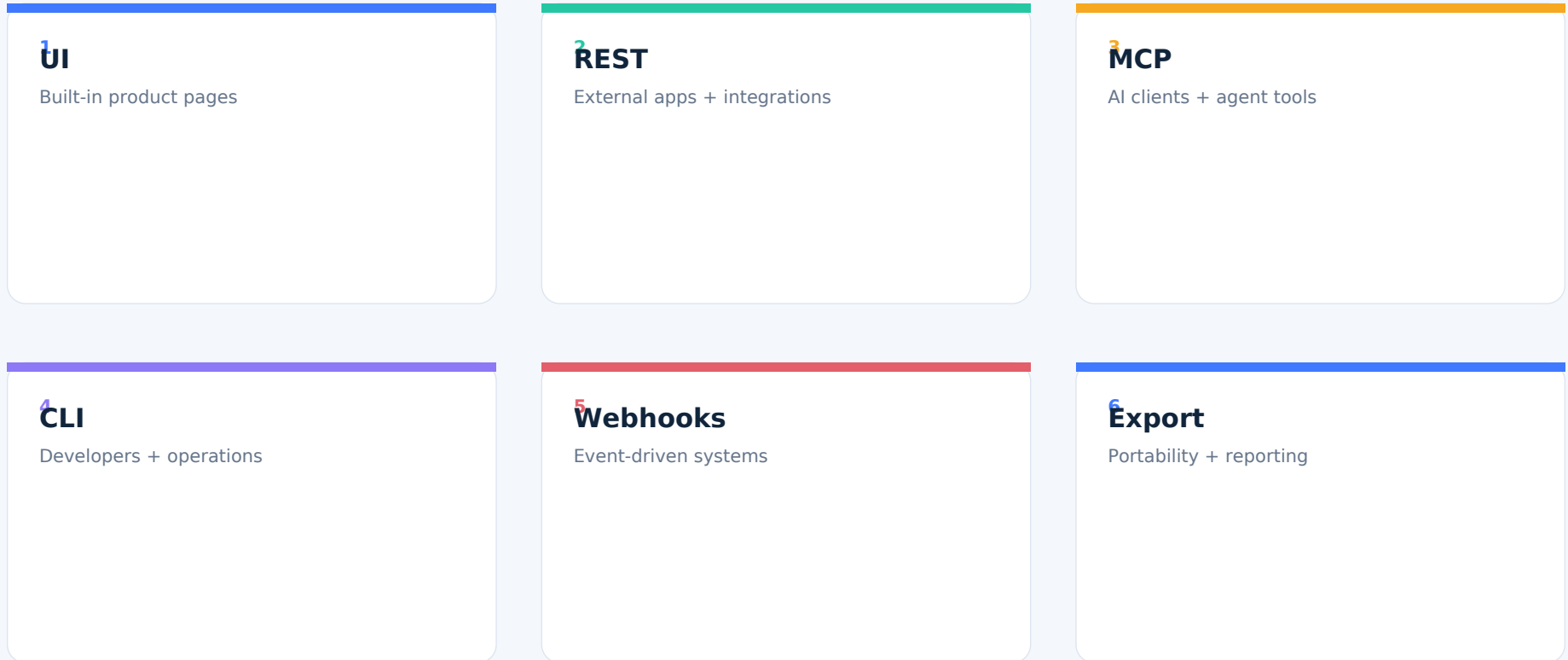
Make the dangerous path harder than the safe path

The goal is not to promise that AI or automation cannot fail. The goal is to constrain what failure can affect, surface evidence, and preserve a deterministic way to stop or reverse the operation.



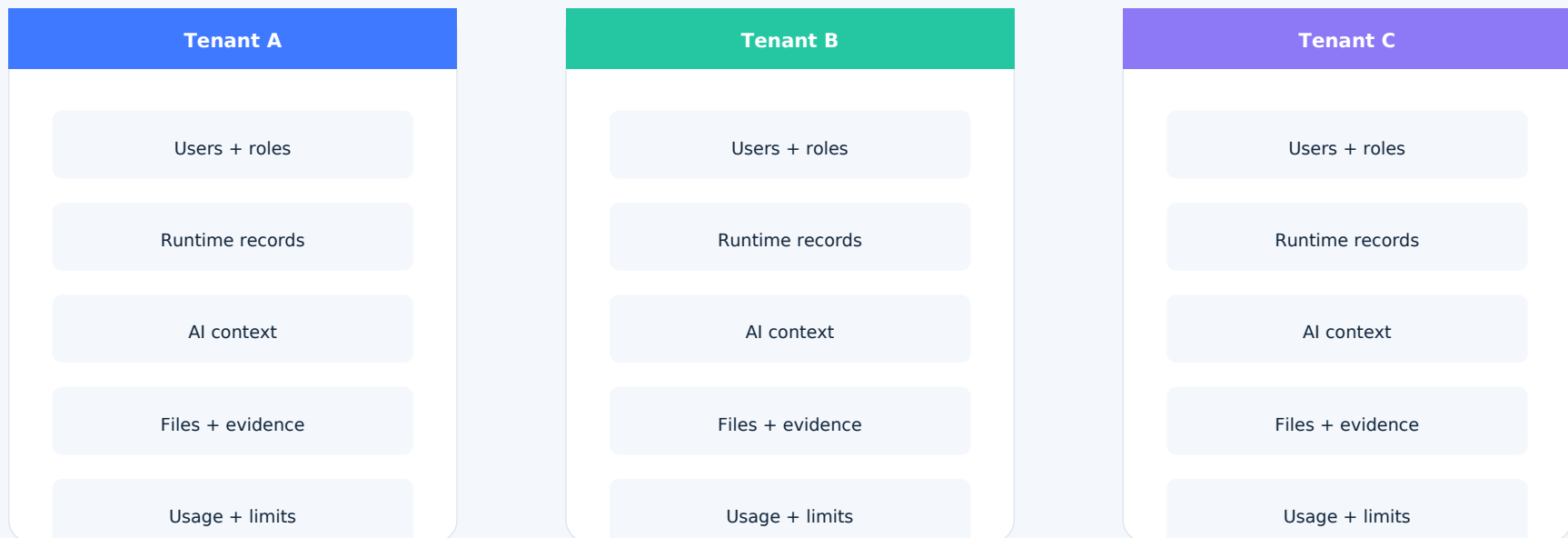
One backend, multiple ways to use it

The same product can support a visual SaaS interface, an external frontend, internal automation, developer tooling, and agent access without creating five disconnected permission models.



Productize once. Isolate customers by design.

BuildWithHQ is intended to let a builder define the product experience while tenant, user, permission, data, AI, and optional service boundaries keep each customer context separated.



Shared platform primitives - isolated customer context - configurable product experience

BUILD THIS

Start with AssetOps. Make it yours.

AssetOps is a product direction, not a fixed template. Use the blueprint as a starting architecture, then shape the records, screens, AI behavior, integrations, permissions, workflows, container services, and commercial model around your market.

1. Define the job

Choose the recurring operating problem and the user roles that feel it most.

2. Map the system

Define records, relationships, evidence, permissions, workflows, and required integrations.

3. Add intelligence

Layer AI and agents into the governed operating loop instead of around it.

BuildWithHQ turns reusable platform primitives into products you can brand, extend, and sell

buildwithhq.com / Build This